# Watermarking PRFs from Lattices: Stronger Security via Extractable PRFs

Sam Kim[*] David J. Wu[†]

**Abstract**

A software watermarking scheme enables one to embed a "mark" (i.e., a message) within a program while preserving the program's functionality. Moreover, there is an extraction algorithm that recovers an embedded message from a program. The main security goal is that it should be difficult to remove the watermark without destroying the functionality of the program. Existing constructions of watermarking focus on watermarking cryptographic functions like pseudorandom functions (PRFs); even in this setting, realizing watermarking from standard assumptions remains difficult. The first lattice-based construction of secret-key watermarking due to Kim and Wu (CRYPTO 2017) only ensures mark-unremovability against an adversary who does not have access to the mark-extraction oracle. The construction of Quach et al. (TCC 2018) achieves the stronger notion of mark-unremovability even if the adversary can make extraction queries, but has the drawback that the watermarking authority (who holds the watermarking secret key) can break pseudorandomness of all PRF keys in the family (including *unmarked* keys).

In this work, we construct new lattice-based secret-key watermarking schemes for PRFs that both provide unremovability against adversaries that have access to the mark-extraction oracle and offer a strong and meaningful notion of pseudorandomness even against the watermarking authority (i.e., the outputs of unmarked keys are pseudorandom almost everywhere). Moreover, security of several of our schemes can be based on the hardness of computing *quasi-polynomial* approximations to worst-case lattice problems. This is a *qualitatively weaker* assumption than that needed for existing lattice-based constructions of watermarking (that support message-embedding), all of which require sub-exponential approximation factors. Our constructions rely on a new cryptographic primitive called an *extractable PRF*, which is of independent interest.

# 1 Introduction

A software watermarking scheme enables a user or an authority to embed a "mark" within a program in a way that the marked program behaves almost identically to the original program. It should be difficult to remove the watermark from a marked program without significantly altering the program's behavior, and moreover, it should be difficult to create new (or malformed) programs that are considered to be watermarked. The first property of *unremovability* is useful for proving ownership of software (e.g., in applications to digital rights management) while the second property of *unforgeability* is useful for authenticating software (e.g., for proving that the software comes from a trusted distributor).

## 1.1 Background and Motivation

Barak et al. [BGI$^+$01, BGI$^+$12] and Hopper et al. [HMW07] introduced the first rigorous mathematical framework for software watermarking. Realizing the strong security requirements put forth

[*] Stanford University. Email: skim13@cs.stanford.edu.

[†] Stanford University and University of Virginia. Email: dwu4@virginia.edu.

in these works has, however, remained difficult. Several early works [NSS99, YF11, Nis13] made partial progress by considering weaker security models and imposing restrictions on the adversary's capabilities. This changed with the seminal work of Cohen et al. [CHN$^+$16], who gave the first positive construction of software watermarking (for classes of cryptographic functionalities) that achieved unremovability against *arbitrary* adversarial strategies from indistinguishability obfuscation.

More formally, a software watermarking scheme consists of two main algorithms. First, the marking algorithm takes a circuit $C$ and outputs a "marked" circuit $C'$ with the property that $C'$ and $C$ agree almost everywhere. Second, a verification algorithm takes a circuit $C$ and outputs MARKED or UNMARKED. In the message-embedding setting, the marking algorithm also takes a message $m$ in addition to the circuit and embeds the message $m$ within the circuit as the watermark. In this case, we replace the verification algorithm with a mark-extraction algorithm that takes a circuit as input and which outputs either the embedded message or UNMARKED. Now, a watermarking scheme is robust against *arbitrary* removal strategies if the the adversary is given complete flexibility in crafting a circuit $\tilde{C}'$ that mimics the behavior of a marked circuit $C'$, and yet does not contain the watermark (from the perspective of the verification or the extraction algorithm). This most directly captures our intuitive notions of unremovability and is the setting that we focus on in this work.

Since the work of Cohen et al., there has been a flurry of works on both building stronger variants of software watermarking [YAL$^+$17, YAL$^+$18] as well as constructing watermarking (and variants) from simpler assumptions and primitives [BLW17, KW17, BKS17, QWZ18]. While this latter line of work has made tremendous progress and has yielded constructions of watermarking from standard lattice assumptions [KW17], CCA-secure encryption [QWZ18], and even public-key encryption (in the stateful setting) [BKS17], these gains have come at the price of relaxing the watermarking security requirements. As such, there is still a significant gap between the security and capabilities of the Cohen et al. construction [CHN$^+$16] from indistinguishability obfuscation and the best schemes we have from standard assumptions. In this work, we narrow this gap and introduce a new lattice-based software watermarking scheme for pseudorandom functions (PRFs) that satisfies stronger security and provides more functionality than the previous constructions from standard assumptions. Along the way, we introduce the notion of an "extractable PRF" that may be useful beyond its applications to cryptographic watermarking.

**Watermarking PRFs.** While the notion of software watermarking is well-defined for general functionalities, Cohen et al. [CHN$^+$16] showed that watermarking is impossible for any class of learnable functions. Consequently, research on watermarking has focused on cryptographic functions like PRFs, digital signatures, and public-key encryption. In their work, Cohen et al. gave the first constructions of watermarking for PRFs (as well as several public-key primitives) from indistinguishability obfuscation. The Cohen et al. watermarking construction has the appealing property in that the scheme supports *public mark-extraction* (i.e., anyone is able to extract the embedded message from a watermarked program). The main drawback though is their reliance on strong (and non-standard) assumptions. Subsequently, Boneh et al. [BLW17] introduced the concept of a private puncturable PRF and showed how to construct *secretly-extractable* watermarking schemes from a variant of private puncturable PRFs (called private programmable PRFs). Building on the Boneh et al. framework, Kim and Wu [KW17] showed that a relaxation of private programmable PRFs also sufficed for watermarking, and they gave the first construction of watermarking from standard lattice assumptions. Neither of these constructions support public extraction, and constructing watermarking schemes that support public extraction from standard assumptions remains a major open problem.

**Towards publicly-extractable watermarking.** Not only did the schemes in [BLW17, KW17] not support public extraction, they had the additional drawback that an adversary who only has access to the extraction oracle for the watermarking scheme can easily remove the watermark from a marked program (using the binary-search-based algorithm described in [CHN$^+$16, §2.4]). Thus, it is very unclear whether these schemes bring us any closer to a watermarking scheme with public extraction. A natural stepping stone towards a publicly-extractable watermarking scheme is to construct a secretly-extractable watermarking scheme, except we give the adversary access to the extraction oracle. The difficulty in handling extraction queries closely resembles the *verifier rejection* problem that arises in the study of designated-verifier argument systems. In those settings, a malicious prover can submit different statements and proofs to the verifier and then from the verifier responses, learn information about the verifier's secret verification state. Correspondingly, in the case of watermarking, the adversary can carefully craft the circuits it submits to the extraction oracle and from the verifier's responses, learn information about the secret watermarking key.

Recently, Quach et al. [QWZ18] gave a very elegant and conceptually-simple construction of secretly-extractable watermarking that provided unremovability in this stronger model where the adversary has access to the extraction oracle. Moreover, their scheme also supports *public marking*: namely, anyone is able to take a PRF and embed a watermark within it. The basic version of their scheme is mark-embedding (i.e., programs are either marked or unmarked) and can be instantiated from any CCA-secure public-key encryption scheme. To support full message-embedding, their construction additionally requires private puncturable PRFs (and thus, the only standard-model instantiation today relies on lattices). In both cases, however, their scheme has the drawback in that the holder of the watermarking secret key completely compromises pseudorandomness of all PRF keys in the family (including *unmarked* keys). In particular, given even two evaluations of a PRF (on distinct points), the watermarking authority in the scheme of [QWZ18] can already distinguish the evaluations from random. While it might be reasonable to trust the watermarking authority, we note here that users must *fully* trust the authority (even if they generate a PRF key only for themselves and never interact with the watermarking authority). Even if the authority *passively* observes PRF evaluations (generated by honest users), it is able to tell those evaluations apart from truly random values. In our view, this is a significant drawback of their construction and limits its applicability. Previous constructions [CHN$^+$16, BLW17, KW17] did not have this drawback.

## 1.2 Our Contributions

In this work and similar to [QWZ18], we study secretly-verifiable watermarking schemes for PRFs that provide unremovability (and unforgeability) against adversaries that have access to both the marking and the extraction oracles. Our goal is to achieve these security requirements while maintaining security even against the watermarking authority. We provide several new constructions of secretly-verifiable watermarking schemes for PRFs from standard lattice assumptions where the adversary has access to the extraction oracle. Moreover, we show that all of our constructions achieve a relaxed (but still meaningful) notion of pseudorandomness for unmarked keys even in the presence of the watermarking authority. Our constructions also simultaneously achieve unremovability and unforgeability (with parameters that match the lower bounds in Cohen et al. [CHN$^+$16]). In fact, we show that meaningful notions of unforgeability (that capture the spirit of unforgeability and software authentication as discussed in [HMW07, CHN$^+$16, YAL$^+$18]) are even possible for schemes that support public marking. Our constructions are the first to simultaneously provide all of these features, and Moreover, we are able to realize these new features while relying on *qualitatively weaker* lattice-based assumptions compared to all previous watermarking constructions from standard assumptions. We summarize these results below.

**Extractable PRFs.** The key cryptographic building block we introduce in this work is the notion of an *extractable PRF*. An extractable PRF is a standard PRF family $\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ outfitted with an extraction trapdoor $\mathsf{td}$. The extractability property says that given oracle access to the PRF $\mathsf{F}(k, \cdot)$, the holder of the trapdoor $\mathsf{td}$ can recover the PRF key $k$ (with overwhelming probability). In fact, the extraction process is *robust* in the following sense: given any circuit $C \colon \mathcal{X} \to \mathcal{Y}$ whose behavior is "close" to $\mathsf{F}(k, \cdot)$, the extraction algorithm still extracts the PRF key $k$. The notion of closeness that we use is $\varepsilon$-closeness: we say that two circuits $C_0$ and $C_1$ are $\varepsilon$-close if $C_0$ and $C_1$ only differ on at most an $\varepsilon$-fraction of the domain. Of course, for extraction to be well-defined, it must be the case that for any pair of distinct keys $k_1, k_2$, the functions $\mathsf{F}(k_1, \cdot)$ and $\mathsf{F}(k_2, \cdot)$ are far apart. We capture this by imposing a statistical requirement on the PRF family called key-injectivity,[1] which requires that $\mathsf{F}(k_1, \cdot)$ and $\mathsf{F}(k_2, \cdot)$ differ on at least an $\varepsilon'$-fraction of points where $\varepsilon' \gg \varepsilon$. This ensures that if $C$ is $\varepsilon$-close to some PRF $\mathsf{F}(k, \cdot)$, then $k$ is unique (and extraction recovers $k$). In Section 2, we provide a detailed technical overview of how to construct extractable PRFs from standard lattice assumptions. We give the formal definition, construction, and security analysis in Section 4.

**From extractable PRFs to watermarking.** The combination of extractability and key-injectivity gives a natural path for constructing a secret-key watermarking scheme for PRFs. We begin with a high-level description of our basic mark-embedding construction which illustrates the main principles. First, we will need to extend our extractable PRF family to additionally support puncturing. In a puncturable PRF [BW13, KPTZ13, BGI14], the holder of a PRF key $k$ can puncture $k$ at a point $x^*$ to derive a "punctured key" $k_{x^*}$ with the property that $k_{x^*}$ can be used to evaluate the PRF on all points $x \neq x^*$. Moreover, given the punctured key $k_{x^*}$, the value of the PRF $\mathsf{F}(k, x^*)$ at $x^*$ is still indistinguishable from a uniformly random value.

Suppose now that we have an extractable PRF where the PRF keys can be punctured. To construct a mark-embedding watermarkable family of PRFs $\mathsf{F} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, we take the watermarking secret key to be the trapdoor for the extractable PRF family. To mark a PRF key $k \in \mathcal{K}$, the watermarking authority derives a special point $x^{(k)} \in \mathcal{X}$ from $k$ (using a PRF key that is also part of the watermarking secret key), and punctures $k$ at $x^{(k)}$ to obtain the punctured key $k_{x^{(k)}}$. The watermarked program just implements PRF evaluation using the punctured key $k_{x^{(k)}}$. To check whether a circuit $C \colon \mathcal{X} \to \mathcal{Y}$ is marked, the watermarking authority applies the extraction algorithm to $C$ to obtain a key $k \in \mathcal{K}$ (or $\perp$ if extraction does not output a key). If the extraction algorithm outputs a key $k \in \mathcal{K}$, the verification algorithm computes the special point $x^{(k)}$ from $k$ and outputs MARKED if $C(x^{(k)}) \neq \mathsf{F}(k, x^{(k)})$ and UNMARKED otherwise. If the extraction algorithm outputs $\perp$, the algorithm outputs UNMARKED.

Unremovability of this construction essentially reduces to puncturing security. By robust extractability (and key-injectivity), if the adversary only corrupts a small number of points in a marked key (within the unremovability threshold), then the extraction algorithm successfully recovers $k$ (with overwhelming probability). To remove the watermark, the adversary's task is to "fix" the value of the PRF at the punctured point $x^{(k)}$. Any adversary that succeeds to do so breaks puncturing security (in particular, the adversary must be able to recover the real value of the PRF at the punctured point given only the punctured key). Note that here, we do require that the range $\mathcal{Y}$ of the PRF to be super-polynomial (if the range was polynomial, then the adversary can guess the correct value of the PRF at $x^{(k)}$ with noticeable probability and remove the watermark). Note that this basic scheme neither provides unforgeability (i.e., it is easy to construct circuits that are considered MARKED even without the watermarking key) nor supports message-embedding. As we discuss in greater detail below, both of these properties can be achieved with additional work.

**Handling extraction queries.** A primary objective of this work is to construct a watermarking

[1] Key-injectivity also played a role in previous watermarking constructions, though in a different context [CHN$^+$16, KW17].

scheme for PRFs where unremovability holds even against an adversary that has access to the extraction oracle. At first glance, our marking algorithm may appear very similar to that in [BLW17, KW17], since all of these constructions rely on some form of puncturable PRFs. These previous constructions do not satisfy unremovability in the presence of an extraction oracle because they critically rely on the adversary not being able to identify the special point $x^{(k)}$. Namely, in these constructions, to check whether a circuit $C$ is marked or not, the authority derives the special point $x^{(k)}$ from the input/output behavior of $C$ and then checks whether $C(x^{(k)})$ has a *specific* structure. If the adversary is able to learn the point $x^{(k)}$, then it can tweak the value of the marked circuit at $x^{(k)}$ and remove the watermark. In fact, even if the puncturable PRF completely hides the special point $x^{(k)}$, the binary search attack from Cohen et al. [CHN$^{+}$16] allows the adversary to use the extraction oracle to recover $x^{(k)}$, and thus, defeat the watermarking scheme.

In our construction, to decide whether a circuit $C$ is marked or not, the authority first extracts a key $k$ and checks whether $C(x^{(k)}) = \mathsf{F}(k, x^{(k)})$. Therefore, in order to remove the watermark, it is not enough for the adversary to just recover the special point $x^{(k)}$ via the extraction oracle (in fact, the special point $x^{(k)}$ is public). To succeed, the adversary has to recover the original *value* of the PRF at $x^{(k)}$, which is hard when the PRF has a super-polynomial range and the PRF satisfies puncturing security. The fact that we do *not* rely on the unpredictability of the special point for security is a subtle but important distinction in our construction. In Section 5, we show that assuming the underlying PRF provides robust extractability (and key-injectivity), the adversary can simulate for itself the behavior of the extraction oracle. Thus, the presence of the extraction oracle cannot help the adversary break unremovability.

**Unforgeability and message-embedding via multi-puncturing.** While the basic construction above provides unremovability, it is easy to forge watermarked programs. Namely, an adversary can simply take a circuit that implements a PRF $\mathsf{F}(k, \cdot)$ and randomly corrupt a $(1/\mathsf{poly}(\lambda))$-fraction of the output (where $\lambda$ is a security parameter). Then, with noticeable probability, the adversary will corrupt the PRF at the special point $x^{(k)}$ associated with $k$, thereby causing the verification algorithm to conclude that the circuit is marked. This is easily prevented by puncturing $k$ at $\lambda$ points $x_1^{(k)}, \ldots, x_\lambda^{(k)}$. We now say that a circuit $C$ is marked only if $C(x_i^{(k)}) \neq \mathsf{F}(k, x_i^{(k)})$ for all $i \in [\lambda]$. Of course, this modification does not affect unremovability. Now, to forge a watermarked program, the adversary has to construct a circuit $C$ whose behavior closely resembles $\mathsf{F}(k, \cdot)$, and yet, $C$ and $\mathsf{F}(k, \cdot)$ disagree on *all* of the special points $x_1^{(k)}, \ldots, x_\lambda^{(k)}$, which are derived pseudorandomly from the key $k$. This means that unless the adversary previously made a request to mark $k$ (in which case its circuit $C$ would no longer be considered a forgery), the points $x_1^{(k)}, \ldots, x_\lambda^{(k)}$ associated with $k$ look uniformly random to $\mathcal{A}$. But now, if $C$ and $\mathsf{F}(k, \cdot)$ are close, they will not differ on $\lambda$ random points, except with negligible probability. We formalize this argument in Section 5.2.

The same technique of puncturing at multiple points also enables us to extend our basic mark-embedding watermarking scheme into a scheme that supports message-embedding. We take a basic bit-by-bit approach similar in spirit to the ideas taken in [Nis13, KW17, QWZ18]. Specifically, to support embedding messages of length $t$ in a PRF key $k$, we first derive from $k$ a collection of $\lambda$ pseudorandom points for each index and each possible bit: $S_{i,b}^{(k)} = \{x_{i,b,1}^{(k)}, \ldots, x_{i,b,\lambda}^{(k)}\}$ for all $i \in [t]$ and $b \in \{0, 1\}$. To embed a message $m \in \{0, 1\}^t$ in the key $k$, the marking algorithm punctures $k$ at all of the points in the sets $S_{i,m_i}^{(t)}$ for $i \in [t]$. To recover the watermark, the extraction algorithm proceeds very similarly as before. Specifically, on input a circuit $C$, the extraction algorithm uses the trapdoor for the underlying extractable PRF to obtain a candidate key $k$ (or outputs UNMARKED if no key is extracted). Given a candidate key $k$, the extraction algorithm derives the sets $S_{i,b}^{(k)}$ for each index $i \in [t]$ and bit $b \in \{0, 1\}$. For each index, the algorithm counts the number of points in $S_{i,0}^{(k)}$ and $S_{i,1}^{(k)}$ on which $C$ and $\mathsf{F}(k, \cdot)$ disagree. For correctly-watermarked keys, $C$ and $\mathsf{F}(k, \cdot)$ will disagree on all of the points in one of the sets and none of the points in the other set. This difference in behavior allows the extraction algorithm to recover the bit at index $i$. We provide the full description and analysis in Section 5.3.

**Public marking in the random oracle model.** In the mark-embedding and message-embedding watermarking constructions we have described so far, both marking and extraction require knowledge of the watermarking secret key. If we look more closely at the marking algorithm, however, we see that the only time the watermarking key is used during marking is to derive the set of points to be punctured (specifically, the set of points to be punctured is derived by evaluating a PRF on the key $k$). Critically, we do *not* require that the set of punctured points be hidden from the adversary (and indeed, the watermarked key completely reveals the set of punctured points), but only that they are unpredictable (without knowledge of $k$). Thus, instead of using a PRF to derive the points to be punctured, we can use a random oracle. This gives a construction of a message-embedding watermarking scheme that supports *public marking*. We provide the full description and analysis of this scheme in Section 5.4. We note that Quach et al. [QWZ18] were the first to give a watermarking scheme that supported public marking *without* random oracles (for mark-embedding, they only needed CCA-secure public-key encryption while for full message-embedding, they relied on lattices). However, as noted before, their scheme does not provide any security against a malicious watermarking authority (or provide unforgeability, which we discuss below).

**Unforgeability and public marking.** Recall that unforgeability for a watermarking scheme says that no efficient adversary should be able to construct a marked circuit that is significantly different from marked circuits it already received. This property seems at odds with the semantics of a watermarking scheme that supports public marking, since in the latter, anyone can mark programs of their choosing. However, we can still capture the following spirit of unforgeability by requiring that the only marked circuits that an adversary can construct are those that are close to circuits that are contained in the function class. In the case of watermarking PRFs, this means that the only circuits that would be considered to be watermarked are those that are functionally close to a legitimate PRF. This property is useful in scenarios where the presence of a watermark is used to argue *authenticity* of software (e.g., to prove to someone that the software implements a specific type of computation). In this work, we introduce a weaker notion of unforgeability that precisely captures this authenticity property (Definition 5.13). We then show that our watermarking construction supports public-marking while still achieving this form of weak unforgeability. The only previous candidate of software watermarking that supports public marking [QWZ18] does not satisfy this property, and indeed, in their scheme, it is easy to construct functions that are constant everywhere (which are decidedly *not* pseudorandom), but nonetheless would be considered to be marked.

**Optimal bounds for unremovability and unforgeability.** We say that a watermarking scheme is $\varepsilon$-unremovable if an adversary who only changes an $\varepsilon$-fraction of the values of a marked circuit cannot remove the watermark,[2] and that it is $\delta$-unforgeable if an adversary cannot create a new marked program that differs on at least a $\delta$-fraction of points from any marked circuits it was given. Conceptually, larger values of $\varepsilon$ means that the watermark remains intact even if the adversary can corrupt the behavior of the marked program on a larger fraction of inputs, while smaller values of $\delta$ means that the adversary's forgery is allowed to agree on a larger fraction of the inputs of a marked program. Previously, Cohen et al. [CHN$^+$16] showed that any message-embedding watermarking scheme can at best achieve $\varepsilon = 1/2 - 1/\mathsf{poly}(\lambda)$ and $\delta = \varepsilon + 1/\mathsf{poly}(\lambda)$. Our constructions in

[2] This definition is the complement of the definition from previous works on watermarking [CHN$^+$16, BLW17, KW17, YAL$^+$17, QWZ18, YAL$^+$18], but we adopt this to maintain consistency with our definition for robust extractability.

this work achieve both of these bounds (for any choice of $\mathsf{poly}(\lambda)$ factors). Previous constructions like [CHN+16, QWZ18] did not provide unforgeability while [BLW17, KW17] could only tolerate $\varepsilon = \mathsf{negl}(\lambda)$ (and any $\delta = 1/\mathsf{poly}(\lambda)$).

**Security against the watermarking authority.** The key property of extractable PRFs that underlies our watermarking constructions is that there is an extraction trapdoor $\mathsf{td}$ that can be used to extract the original PRF key $k$ from any circuit whose behavior is sufficiently similar to that of $\mathsf{F}(k, \cdot)$. In the case of watermarking, the watermarking authority must hold the trapdoor to use it to extract watermarks from marked programs. This raises a new security concern as the watermarking authority can now break security of *all* PRFs in the family, including *unmarked* ones. As discussed in Section 1.1, this was the main drawback of the Quach et al. [QWZ18] watermarking construction.

Due to our reliance on extractable PRFs, our watermarkable family of PRFs also cannot satisfy full pseudorandomness against the watermarking authority. However, we can show a weaker property against the watermarking authority we call $T$-restricted pseudorandomness. Namely, we can associate a set $S \subseteq \mathcal{X}$ of size at most $T$ with our watermarkable family of PRFs such that any adversary (even if they have the extraction trapdoor) is unable to break pseudorandomness of any (unmarked) PRF, provided that they do not query the function on points in $S$. The distinguisher is also provided the set $S$. In other words, our family of PRFs still provides pseudorandomness everywhere except $S$. In our concrete constructions (Construction 4.19), the restricted set $S$ consists of $\lambda$ *randomly-chosen* points in $\mathcal{X}$. This means that if the domain of the PRF is super-polynomial, our notion of $T$-restricted pseudorandomness strictly interpolates between weak pseudorandomness (or even non-adaptive pseudorandomness)[3] and strong pseudorandomness. It is also worth noting that from the perspective of a user who does not hold the watermarking secret key, the points in $S$ are *statistically hidden.* This means that in any standard usage of the PRF between honest users, with overwhelming probability, the PRF would never be evaluated on one of the restricted points. Equivalently, if the watermarking authority only sees passive evaluations of the PRF, then it will not be able to break pseudorandomness of the underlying PRF. This notion of "passive" security against the watermarking authority strictly improves upon the lattice-based message-embedding watermarking construction in [QWZ18]. In their setting, the watermarking authority is able to break pseudorandomness given *any* two (distinct) evaluations of the PRF; that is, their scheme does not even satisfy weak pseudorandomness against the watermarking authority. It is an interesting and important question to obtain watermarking with security in the presence of an extraction oracle and which retrains full pseudorandomness even against the watermarking authority. The only constructions that satisfy this notion rely on obfuscation.

**Watermarking *without* private puncturing.** All existing constructions of message-embedding watermarking from standard assumptions have relied on *private* puncturable PRFs[4] in some form [KW17, QWZ18]. Our message-embedding watermarking construction is the first that does *not* rely on private puncturing; standard puncturing in conjunction with key-extractability suffices. While this might seem like a minor distinction, we note that constrained PRFs can be constructed from weaker assumptions. For instance, puncturable PRFs can be built from one-way functions [GGM84, BW13, KPTZ13, BGI14] while the simplest constructions of private puncturable PRFs rely on lattice-based assumptions [BKM17, CC17, BTVW17, PS18, CVW18]. If we just con-

[3] In the weak pseudorandomness game, the adversary is given outputs of the PRF on random inputs, while in the non-adaptive pseudorandomness game, the adversary must declare *all* of its evaluation queries before seeing any evaluations of the PRF or the public parameters.

[4] A private puncturable PRF [BLW17] is a puncturable PRF where the punctured key also *hides* the punctured point. There are several lattice-based constructions of private puncturable PRFs (and more generally, private constrained PRFs) [CC17, BKM17, BTVW17, PS18, CVW18].

sider lattice-based constrained PRFs, the Brakerski-Vaikuntanathan puncturable PRF [BV15] can be based on the (polynomial) hardness of solving worst-case lattice problems with a *quasi-polynomial* approximation factor (e.g., $2^{\log^c(n)}$ for $c > 1$),[5] while constructions of private puncturable PRFs from lattices [BKM17, CC17, BTVW17, PS18, CVW18] can only be based on the hardness of solving worst-case lattice problems with a *sub-exponential* approximation factor (e.g., $2^{n^c}$ for $c > 0$). Since all of the existing constructions of message-embedding watermarking from standard assumptions rely on private puncturing in some form, they can only be reduced to worst-case lattice problems with sub-exponential approximation factors. In this work, we show that a variant of our construction (satisfying a relaxed notion of unforgeability as in [KW17]) can be based solely on worst-case lattice problems with a quasi-polynomial approximation factor (Remark 4.27). At a conceptual level, our construction gives the first message-embedding watermarkable PRF family that does not rely on fully homomorphic encryption (FHE) [Gen09] or the GGH15 multilinear encodings [GGH15]—in contrast to existing constructions of watermarking based on private puncturable PRFs. Concretely, we give the first (message-embedding) watermarking scheme whose security can be based on computing quasi-polynomial approximations to worst-case lattice problems (Corollaries 5.39 and 5.40).[6]

**Additional properties.** Finally, in Section 5.5, we briefly describe some additional features of our new watermarking scheme. For instance, we show a new transferability property supported by our scheme (i.e., the watermarking authority can remove the watermark from a marked key and then re-watermark it under a different user's identity).

### 1.3 Additional Related Work

We now survey some additional works that use similar techniques as those in our construction.

**Lattice-based PRFs.** The study of lattice-based PRFs started with the seminal work of Banerjee et al. [BPR12]. Subsequently, [BLMR13, BP14] constructed the first lattice-based key-homomorphic PRFs. The first circuit-constrained PRFs were constructed in [BV15, BFP$^+$15] and were later extended to private constrained PRFs in [BKM17, CC17, BTVW17, PS18, CVW18].

**Matrix embeddings.** The matrix embedding techniques used in this work build on a series of works in the areas of attribute-based encryption [SW05] and predicate encryption [BW07, KSW08] from LWE. These include the attributed-based encryption constructions of [ABB10, GVW13, BGG$^+$14, GV15, BV16, BCTW16] and the (one-sided) predicate encryption constructions of [AFV11, GMW15, GVW15, BTVW17, GKW17, WZ17].

## 2 Technical Overview

In this section, we provide a technical overview of our construction of extractable PRFs from standard lattice assumptions. As described in Section 1.2, this is the key cryptographic primitive we rely on in our watermarking constructions (described formally in Section 5). We believe that the algebraic techniques we develop for constructing our extractable PRF are general and will find applications beyond the study of PRFs and watermarking. We highlight the core principles and techniques here, but defer the formal definitions, constructions, and analysis to Section 4.

[5] While the general construction described in [BV15] relies on worst-case lattice problems with sub-exponential approximation factors, we can specialize it to just puncturing constraints. Combined with the techniques from [BV14], this yields a puncturable PRF from worst-case lattice problems with a quasi-polynomial approximation factor.

[6] Obtaining a lattice-based PRF (let alone a watermarkable PRF) from a worst-case lattice problem with a *polynomial* approximation factor is a major open problem (without going through the generic transformation of [GGM84]).

**The LWE assumption.** The learning with errors (LWE) assumption [Reg05], parameterized by $n$, $m$, $q$, $\chi$, states that for a uniformly random vector $\mathbf{s} \in \mathbb{Z}_q^n$, a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and a noise vector $\mathbf{e}$ sampled from a (low-norm) error distribution $\chi$, the distribution $(\mathbf{A}, \mathbf{s} \cdot \mathbf{A} + \mathbf{e})$[7] is computationally indistinguishable from the uniform distribution over $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m$. Equivalently, rather than explicitly adding noise, the LWE assumption can instead be defined with respect to a rounding modulus $p < q$ and the component-wise rounding operation $\lfloor \cdot \rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$ [BPR12]. This variant of the LWE assumption states that the distribution $(\mathbf{A}, \lfloor \mathbf{s} \cdot \mathbf{A} \rceil_p)$ is computationally indistinguishable from the uniform distribution over $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_p^m$; this is also known as the *learning with rounding* (LWR) assumption [BPR12]. For the setting of parameters that we consider in this work, the LWR assumption is equivalent to the LWE assumption [BPR12].

**Lattice-based PRFs.** A natural way to construct a pseudorandom function $\mathsf{F} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ from the LWE assumption is to take the PRF key $k \in \mathcal{K}$ to be the LWE secret $\mathbf{s} \in \mathbb{Z}_q^n$ and define $\mathsf{F}(\mathbf{s}, x)$ to output an LWE sample $\lfloor \mathbf{s} \cdot \mathbf{A}_x \rceil_p$ for a matrix $\mathbf{A}_x$ that is uniquely determined by the input $x \in \mathcal{X}$. Note that when the domain $\mathcal{X}$ is super-polynomial, the matrix $\mathbf{A}_x$ cannot be a uniformly random matrix as required by the LWE assumption since $\mathsf{F}(\mathbf{s}, \cdot)$ must be an (efficiently-computable) deterministic function. Constructing a PRF from LWE thus amounts to designing a suitable mapping $x \mapsto \mathbf{A}_x$ such that the vector $\lfloor \mathbf{s} \cdot \mathbf{A}_x \rceil_p$ is still pseudorandom under LWE.

Nearly all existing LWE-based PRF constructions follow this general blueprint; we refer to Section 1.3 for a more comprehensive discussion of related work. Specifically, these PRF families are defined with respect to a set of public matrices $\mathsf{pp} = (\mathbf{A}_1, \ldots, \mathbf{A}_\rho)$ and an input-to-matrix mapping $\mathsf{Eval}_{\mathsf{pp}} \colon \mathcal{X} \to \mathbb{Z}_q^{n \times m}$ (that implements the mapping $x \mapsto \mathbf{A}_x$) such that the outputs of

$$\mathsf{F}(\mathbf{s}, x) := \lfloor \mathbf{s} \cdot \mathbf{A}_x \rceil_p \text{ where } \mathbf{A}_x \leftarrow \mathsf{Eval}_{\mathsf{pp}}(x) \tag{2.1}$$

are computationally indistinguishable from uniform vectors over $\mathbb{Z}_p^n$ under the LWE assumption. In this overview, rather than focusing on a particular PRF construction, we show how to obtain an extractable PRF from any lattice-based PRF family that follows this blueprint.

**Key extraction via lattice trapdoors.** Recall from Section 1 that in an extractable PRF family, the holder of a trapdoor $\mathsf{td}$ (for the PRF family) can recover the PRF key $k \in \mathcal{K}$ given only oracle access to the PRF $\mathsf{F}(k, \cdot)$. Using the basic structure of lattice-based PRF candidates from Eq. (2.1), a natural starting point is to design the mapping $\mathsf{Eval}_{\mathsf{pp}} \colon \mathcal{X} \to \mathbb{Z}_q^{n \times m}$ such that for a special input $x^* \in \mathcal{X}$, the matrix $\mathbf{D} \leftarrow \mathsf{Eval}_{\mathsf{pp}}(x^*)$ has a known (lattice) trapdoor $\mathsf{td}_{\mathbf{D}}$, which can be included as part of the trapdoor for the extractable PRF family (together with the special input $x^*$).

A lattice trapdoor $\mathsf{td}_{\mathbf{D}}$ for a matrix $\mathbf{D} \in \mathbb{Z}_q^{n \times m}$ enables sampling short preimages under the matrix $\mathbf{D}$ [Ajt99, GPV08, AP09, MP12, LW15]. Specifically, given an arbitrary target matrix $\mathbf{T} \in \mathbb{Z}_q^{n \times m}$, the trapdoor $\mathsf{td}_{\mathbf{D}}$ enables sampling a short matrix $\mathbf{R}_{\mathbf{T}} \in \mathbb{Z}_q^{m \times m}$ such that $\mathbf{D} \cdot \mathbf{R}_{\mathbf{T}} = \mathbf{T}$. Additionally, the trapdoor for $\mathbf{D}$ can be used to solve the search version of the LWE problem: given an LWE instance $(\mathbf{D}, \lfloor \mathbf{s} \cdot \mathbf{D} \rceil_p)$, one first computes a short matrix $\mathbf{R}_{\mathbf{G}}$ using the trapdoor $\mathbf{D}$ and then derive the vector

$$\lfloor \mathbf{s} \cdot \mathbf{D} \rceil_p \cdot \mathbf{R}_{\mathbf{G}} = \lfloor \mathbf{s} \cdot \mathbf{D} \cdot \mathbf{R}_{\mathbf{G}} \rceil_p + \mathsf{noise} = \lfloor \mathbf{s} \cdot \mathbf{G} \rceil_p + \mathsf{noise} \in \mathbb{Z}_p^m,$$

where $\mathsf{noise}$ is a small error vector that occurs from the modular rounding and $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ is the standard powers-of-two gadget matrix [MP12]. Since $\mathbf{G}^T$ is the generator matrix for a linear error-correcting code, recovering $\mathbf{s}$ form $\lfloor \mathbf{s} \cdot \mathbf{G} \rceil_p + \mathsf{noise}$ is straightforward (c.f., [MP12]).

[7] For notational simplicity, we drop the transpose notation when it is clear from context.

Given the trapdoor $\mathsf{td}_{\mathbf{D}}$, it is straightforward to implement the $\mathsf{Extract}$ algorithm. Namely, $\mathsf{Extract}$ first queries $\mathsf{F}(\mathbf{s}, \cdot)$ on the special point $x^* \in \mathcal{X}$ to obtain the output $\lfloor \mathbf{s} \cdot \mathbf{D} \rceil_p$. It then uses the trapdoor information $\mathsf{td}_{\mathbf{D}}$ to recover the secret key $\mathbf{s}$.

**Programming the trapdoor.** The problem of constructing an extractable PRF family now boils down to generating a set of public parameters $\mathsf{pp}$ and a suitable mapping $\mathsf{Eval}_{\mathsf{pp}} \colon \mathcal{X} \to \mathbb{Z}_q^{n \times m}$ such that the matrix $\mathbf{A}_{x^*} \leftarrow \mathsf{Eval}_{\mathsf{pp}}(x^*)$ can be programmed to be a trapdoor matrix $\mathbf{D}$. At the same time, $\mathsf{Eval}_{\mathsf{pp}}$ must be designed so that the basic blueprint from Eq. (2.1) still satisfies pseudorandomness. The concept of programming the output of a PRF was previously explored in the context of constrained PRFs [BLW17, KW17, CC17, PS18]. These works study the notion of a *private programmable PRF* where *constrained keys* can be programmed to a specific value at a particular point (or set of points). However, the techniques used in these works do not directly apply to our setting as our goal is fundamentally different. To construct an extractable PRF, we need a PRF family such that the evaluation of *every* PRF key from the family is programmed to a trapdoor matrix. In fact, our notion is completely independent of constraining, and an extractable PRF family need not even support constraining. In other words, we want programmability with respect to the public parameters of the PRF family rather than just an individual PRF key.

The way we construct the function $\mathsf{Eval}_{\mathsf{pp}}$ is quite simple and general. We take any existing PRF construction $\mathsf{F}' \colon \mathbb{Z}_q^m \times \mathcal{X} \to \mathbb{Z}_p^m$ following the blueprint from Eq. (2.1) that is defined respect to a set of matrices $\mathsf{pp}' = (\mathbf{A}_1, \ldots, \mathbf{A}_\rho)$ and mapping $\mathsf{Eval}'_{\mathsf{pp}'} \colon \mathcal{X} \to \mathbb{Z}_q^{n \times m}$, and define a new *shifted* mapping

$$\mathsf{Eval}_{\mathsf{pp}}(x) := \mathsf{Eval}'_{\mathsf{pp}'}(x) + \mathbf{W} = \mathbf{A}_x + \mathbf{W},$$

for some shift matrix $\mathbf{W} \in \mathbb{Z}_q^{n \times m}$ and a new set of public matrices $\mathsf{pp} = (\mathbf{A}_1, \ldots, \mathbf{A}_\rho, \mathbf{W})$. First, observe that given a point $x^* \in \mathcal{X}$ and a trapdoor matrix $\mathbf{D}$, it is easy to generate a programmed set of public parameters:

1. Generate the matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\rho \in \mathbb{Z}_q^{n \times m}$ as in the original PRF family.
2. Set $\mathbf{W} = \mathbf{D} - \mathbf{A}_{x^*}$ where $\mathbf{A}_{x^*} \leftarrow \mathsf{Eval}'_{\mathsf{pp}'}(x^*)$.

It is easy to see that security of the original PRF family is preserved. Specifically, we now have

$$\mathsf{F}(\mathbf{s}, x) = \lfloor \mathbf{s} \cdot (\mathbf{A}_x + \mathbf{W}) \rceil_p \approx \lfloor \mathbf{s} \cdot \mathbf{A}_x \rceil_p + \lfloor \mathbf{s} \cdot \mathbf{W} \rceil_p = \mathsf{F}'(\mathbf{s}, x) + \lfloor \mathbf{s} \cdot \mathbf{W} \rceil_p, \tag{2.2}$$

Since a randomly sampled trapdoor matrix $\mathbf{D}$ is statistically close to uniform, the matrix $\mathbf{W}$ is also statistically close to uniform. This means that the additional vector offset $\mathbf{w} = \lfloor \mathbf{s} \cdot \mathbf{W} \rceil_p$ introduced by $\mathbf{W}$ looks indistinguishable from a uniformly random vector under LWE. Moreover,

$$\mathsf{F}(\mathbf{s}, x^*) = \lfloor \mathbf{s} \cdot (\mathbf{A}_{x^*} + \mathbf{W}) \rceil_p = \lfloor \mathbf{s} \cdot \mathbf{D} \rceil_p,$$

so given the trapdoor $\mathsf{td}_{\mathbf{D}}$, it is easy to recover the key $\mathbf{s}$.

## 2.1 Robust Extractability

The PRF family $\mathsf{F} \colon \mathbb{Z}_q^n \times \mathcal{X} \to \mathbb{Z}_p^n$ defined in Eq. (2.2) already satisfies a basic notion of key-extractability. Namely, any authority who holds the trapdoor information $(x^*, \mathsf{td}_{\mathbf{D}})$ is able to extract the PRF key given just oracle access to the function $\mathsf{F}(\mathbf{s}, \cdot)$; moreover, $\mathsf{F}(\mathbf{s}, \cdot)$ remains pseudorandom to anyone who does not possess the trapdoor. To support watermarking, however, we require a stronger security property called *robust extractability* (Definition 4.9).

**Robustness and key-injectivity.** At a high level, robust extractability says that the $\mathsf{Extract}$ algorithm should successfully recover the PRF key even if it is just given access to a function (modeled as a circuit) whose behavior is "close" to $\mathsf{F}(\mathbf{s}, \cdot)$. In fact, even if the adversary has oracle access to $\mathsf{Extract}$, it should not be able to produce a circuit $C$ whose behavior is sufficiently "close" to $\mathsf{F}(\mathbf{s}, \cdot)$ for some key $\mathbf{s} \in \mathbb{Z}_q^n$, and for which, the extraction algorithm fails to extract $\mathbf{s}$ from $C$. The closeness metric that we use in this work is $\varepsilon$-closeness; namely, we say that two circuits $C$ and $C'$ are $\varepsilon$-close if they agree on all but an $\varepsilon$-fraction of elements in the domain. In all of our constructions, $\varepsilon = 1/\mathsf{poly}(\lambda)$. Of course, for the extractability property to be well-defined, it should be the case that for distinct keys $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{Z}_q^n$, $\mathsf{F}(\mathbf{s}_1, \cdot)$ and $\mathsf{F}(\mathbf{s}_2, \cdot)$ should be "far" apart. As discussed in Section 1.2, we capture this by defining a notion of *key-injectivity* (Definition 4.11) similar in flavor to previous definitions from [CHN+16, KW17], and then show (Theorem 4.22) that over the randomness used to sample the public parameters, the basic construction in Eq. (2.2) satisfies our key-injectivity property. Thus, in the subsequent discussion, we assume without loss of generality that if a circuit $C$ is $\varepsilon$-close to $\mathsf{F}(\mathbf{s}, \cdot)$ for any $\varepsilon = 1/\mathsf{poly}(\lambda)$, then $\mathbf{s}$ is unique.

The basic PRF construction from Eq. (2.2) does *not* satisfy robust extractability (for *any* closeness parameter $\varepsilon = 1/\mathsf{poly}(\lambda)$). Specifically, the adversary can recover the special point $x^* \in \mathcal{X}$ using binary search. To mount the attack, the adversary first chooses a key $\mathbf{s} \in \mathbb{Z}_q^n$, and constructs the circuit $\mathsf{F}(\mathbf{s}, \cdot)$. The adversary then (arbitrarily) partitions the domain into two halves $\mathcal{X}_1$ and $\mathcal{X}_2$ and queries the extraction oracle on a circuit $C$ that agrees with $\mathsf{F}(\mathbf{s}, \cdot)$ on $\mathcal{X}_1$ and outputs $\perp$ on $\mathcal{X}_2$. Depending on whether the extraction algorithm succeeds in recovering $\mathbf{s}$ or not, the adversary learns which of $\mathcal{X}_1$ or $\mathcal{X}_2$ contains the special point $x^*$. After a polynomial number of queries, the adversary learns $x^*$. Once the adversary learns the special point $x^*$, it can always cause extraction to fail on a circuit by simply having the circuit output $\perp$ on $x^*$ (and $\mathsf{F}(\mathbf{s}, x)$ on all $x \neq x^*$). Moreover, this circuit agrees with $\mathsf{F}(\mathbf{s}, \cdot)$ on all but a single point (i.e., they agree on all but a negligible fraction of the domain when $|\mathcal{X}|$ is super-polynomial), which breaks robust extractability.

**Defending against binary search.** Effectively, the binary search attack relies on the fact that the adversary can easily construct circuits $C$ such that the behavior of $\mathsf{Extract}$ on $C$ (specifically, whether $\mathsf{Extract}$ succeeds or not) is correlated with the secret extraction trapdoor (specifically, the point $x^*$). To defend against this, we develop a way to ensure that the behavior of $\mathsf{Extract}$ on a circuit $C$ depends *only* on properties of the circuit $C$ (and *not* on the extraction trapdoor). If this is the case, then the extraction oracle does not leak information about the extraction trapdoor, and in turn, robust extractability holds. We note that this type of approach is conceptually very similar to the notion of *strong soundness* in the context of constructing *multi-theorem* argument systems in the designated-verifier setting [BCI+13, BISW17].[8] To achieve this, we proceed in two steps. First, we modify the $\mathsf{Extract}$ algorithm to force the adversary to only submit circuits that are very close to an actual PRF circuit $\mathsf{F}(\mathbf{s}, \cdot)$. Then, we tweak the construction to ensure that extraction queries on circuits $C$ that are too close to a real PRF circuit are not helpful to the adversary. We describe this below.

- **Testing for closeness.** After the $\mathsf{Extract}$ algorithm recovers a candidate key $\mathbf{s} \in \mathbb{Z}_q^n$ from a circuit $C$, it additionally checks whether the behavior of the circuit $C$ and $\mathsf{F}(\mathbf{s}, \cdot)$ are "similar." While computing the exact distance between $C$ and $\mathsf{F}(\mathbf{s}, \cdot)$ cannot be done in polynomial time, it is straightforward to construct a randomized algorithm that accepts (with

[8] In designated-verifier argument systems, an adversary who has oracle access to the verifier can observe the verifier's behavior on different statements and proof strings. When the verifier's responses are correlated with its secret verification state, the prover can potentially leverage the leakage and compromise soundness. This is the so-called "verifier rejection" problem. Strong soundness is a property that says that the responses of the verifier depend only on the statement or proof string, and *not* on the secret verification state (the analog in our setting is that the behavior of the extraction oracle only depends on the input circuit and not the extraction trapdoor). This property is very useful for arguing soundness in the presence of a verification oracle for designated-verifier argument systems.

overwhelming probability) whenever $C$ and $\mathsf{F}(\mathbf{s}, \cdot)$ are $\varepsilon_1$-close and rejects (with overwhelming probability) whenever $C$ and $\mathsf{F}(\mathbf{s}, \cdot)$ are $\varepsilon_2$-far, for any choice of $\varepsilon_2 > \varepsilon_1 + 1/\mathsf{poly}(\lambda)$. This can be done by sampling random points $x_1, \ldots, x_\xi \xleftarrow{\mathrm{R}} \mathcal{X}$ and counting the number of inputs where $C(x_i) = \mathsf{F}(\mathbf{s}, x_i)$. If the number of points on which the two circuits differ is greater than $\xi \cdot (\varepsilon_1 + \varepsilon_2)/2$, then the $\mathsf{Extract}$ algorithm outputs $\perp$. By choosing $\xi = \mathsf{poly}(\lambda)$ accordingly, we can appeal to standard concentration bounds and show that $\mathsf{Extract}$ will only output a candidate key when $C$ and $\mathsf{F}(\mathbf{s}, \cdot)$ are at least $\varepsilon_2$-close. When applied to watermarking, the parameter $\varepsilon_1$ corresponds to the unremovability threshold while the parameter $\varepsilon_2$ corresponds to the unforgeability threshold.

- **Embedding multiple trapdoors.** The closeness test prevents the adversary from querying the extraction oracle on circuits that are more than $\varepsilon_2$-far from valid PRF circuits $\mathsf{F}(\mathbf{s}, \cdot)$, since the output of $\mathsf{Extract}$ on these queries is $\perp$ with overwhelming probability. However, since $\varepsilon_2 = 1/\mathsf{poly}(\lambda)$, the adversary can still query the extraction oracle on circuits that are $\varepsilon_2$-close to the real PRF circuit $\mathsf{F}(\mathbf{s}, \cdot)$. In this case, each query still (roughly) allows the adversary to rule out at least an $\varepsilon_2$-fraction of the domain, and so, in time $\mathsf{poly}(1/\varepsilon_2) = \mathsf{poly}(\lambda)$, the adversary is again able to extract the special point $x^*$ for the PRF family.

  The second ingredient in our construction is to embed *multiple* trapdoors. Specifically, instead on just embedding a single lattice trapdoor at $x^*$, we instead embed $\lambda$ distinct trapdoors at $\lambda$ special points $x_1^*, \ldots, x_\lambda^* \xleftarrow{\mathrm{R}} \mathcal{X}$. Now, on input a circuit $C$, the $\mathsf{Extract}$ algorithm evaluates $C$ at each special point $x_i^*$, and use the lattice trapdoor inversion algorithm to obtain candidate keys $\mathbf{s}_i$. It performs the closeness test described above on each candidate key $\mathbf{s}_i$ and outputs $\mathbf{s}_i$ if the closeness test succeeds, and $\perp$ if none succeed. By key-injectivity, there can only be one key $\mathbf{s}$ where $\mathsf{F}(\mathbf{s}, \cdot)$ is $\varepsilon_2$-close to $C$ whenever $\varepsilon_2 < 1/2$. At a very high level, the benefit of having multiple trapdoors is that the adversary has to corrupt the value at all of the trapdoors in order to cause the output of the $\mathsf{Extract}$ algorithm to differ (in a manner that is correlated with the secret extraction state). Since the special points $x_1^*, \ldots, x_\lambda^*$ are independently and uniformly distributed, and the adversary is effectively constrained to choosing circuits $C$ which are $\varepsilon_2$-close to some $\mathsf{F}(\mathbf{s}, \cdot)$, the probability that the adversary succeeds in constructing such a circuit is $\varepsilon_2^\lambda = \mathsf{negl}(\lambda)$. We refer to Section 4.2 and Theorem 4.26 for the formal analysis.

## 2.2 Puncturing and Pseudorandomness Given the Trapdoor

Recall from Section 1.2 that to obtain a watermarking scheme from an extractable PRF, we additionally require that the extractable PRFs support puncturing constraints. Since our techniques for building extractable PRFs are broadly applicable to many lattice-based PRFs, we can take an existing candidate with the structure from Eq. (2.1) and derive from it an extractable PRF. In particular, we can apply our general construction to the Brakerski-Vaikuntanathan constrained PRF [BV15], and obtain a puncturable extractable PRF. To achieve the stronger security notion of ($T$-restricted) pseudorandomness against an authority that holds the extraction trapdoor, we have to develop new techniques. We discuss the challenges below.

**Security against the authority.** As discussed in Section 1.2, a key contribution of our work is showing that the keys in our watermarkable PRF family still provide a relaxed form of pseudorandomness even against the holder of the watermarking secret key. This property amounts to showing that the underlying extractable PRF satisfies $T$-restricted pseudorandomness against an adversary who is given the extraction trapdoor. Specifically, we show that as long as the adversary (who has the trapdoor) is not allowed to query the PRF on the special points $x_1^*, \ldots, x_\lambda^*$, then pseudorandomness holds. This set of special points constitute the *restricted set* in the $T$-restricted pseudorandomness experiment. First, recall from Eq. (2.2) that

$$\mathsf{F}(\mathbf{s}, x) = \lfloor \mathbf{s} \cdot (\mathbf{A}_x + \mathbf{W}) \rceil_p \approx \mathsf{F}'(\mathbf{s}, x) + \lfloor \mathbf{s} \cdot \mathbf{W} \rceil_p \,,$$

where $\mathsf{F}'(\mathbf{s}, x)$ is the existing PRF (specifically, the Brakerski-Vaikuntanathan PRF [BV15]). At first glance, one might be tempted to believe that $T$-restricted pseudorandomness against the authority follows immediately from the security of $\mathsf{F}'$ since the value $\mathsf{F}(\mathbf{s}, x)$ is just $\mathsf{F}'(\mathbf{s}, x)$ shifted by $\lfloor \mathbf{s} \cdot \mathbf{W} \rceil_p$ where $\mathbf{W} = \mathbf{D} - \mathbf{A}_{x^*}$. Without the extraction trapdoor, $\mathbf{D}$ is statistically close to uniform, so we can appeal to LWE to argue that the shift $\lfloor \mathbf{s} \cdot \mathbf{W} \rceil_p$ is uniformly random (and looks independent of $\mathsf{F}'(\mathbf{s}, x)$). But given the trapdoor matrix $\mathbf{D}$, this is no longer the case; the shift $\lfloor \mathbf{s} \cdot \mathbf{W} \rceil_p$ is *correlated* with the PRF key $\mathbf{s}$, and not easily simulated without knowing $\mathbf{s}$ itself. Thus, it is unclear how to directly reduce security of $\mathsf{F}$ to security of the underlying PRF $\mathsf{F}'$.

Consider a potential reduction algorithm $\mathcal{B}$ that uses an adversary for $\mathsf{F}$ in the $T$-restricted pseudorandomness security game to break the security of $\mathsf{F}'$. In this case, $\mathcal{B}$ is given the extraction trapdoor. If the reduction algorithm $\mathcal{B}$ is able to correctly simulate the evaluation $\mathsf{F}(\mathbf{s}, x)$ on all points $x \in \mathcal{X}$, then it can use its trapdoor information $\mathsf{td}_{\mathbf{D}}$ to extract $\mathbf{s}$ and break security of $\mathsf{F}'$ itself. Thus, for the proof to go through, we minimally need to rely on some type of "puncturing" argument (c.f., [BV15]). A possible starting point is to give the reduction algorithm $\mathcal{B}$ a punctured key $k'_S$ for $\mathsf{F}'$ that enables evaluation of $\mathsf{F}'$ at all points except the restricted points $S = (x_1^*, \ldots, x_\lambda^*)$. Then, $\mathcal{B}$ can simulate the correct PRF evaluations at all non-restricted points, but it is unable compute the evaluations at the special points for itself.

Unfortunately, this basic puncturing approach is still insufficient to prove security. Namely, even if the reduction algorithm can simulate the non-shifted PRF evaluation $\mathsf{F}'(\mathbf{s}, x)$ at all of the non-restricted points, it must still simulate the shift $\lfloor \mathbf{s} \cdot \mathbf{W} \rceil_p$ *without* knowledge of the key $\mathbf{s}$. To address this, we additionally need to "program" the evaluations of the punctured key $k_S$ at the non-punctured points. Specifically, we program the key $k_S$ to introduce a shift by the key-dependent vector $\lfloor \mathbf{s} \cdot \mathbf{W} \rceil_p$ at all of the non-punctured points. This latter step relies on an adaptation of the technique of programmable matrix embeddings from [KW17]. This enables $\mathcal{B}$ to simulate the full PRF evaluation $\mathsf{F}(\mathbf{s}, x) = \mathsf{F}'(\mathbf{s}, x) + \lfloor \mathbf{s} \cdot \mathbf{W} \rceil_p$ for the adversary. We refer to Section 4.2 for the full details of the construction and security analysis.

# 3 Preliminaries

We begin by introducing some of the notation we use in this work. For an integer $n \geq 1$, we write $[n]$ to denote the set of integers $\{1, \ldots, n\}$. For a distribution $\mathcal{D}$, we write $x \leftarrow \mathcal{D}$ to denote that $x$ is sampled from $\mathcal{D}$; for a finite set $S$, we write $x \xleftarrow{\text{R}} S$ to denote that $x$ is sampled uniformly from $S$. We write $\mathsf{Funs}[\mathcal{X}, \mathcal{Y}]$ to denote the set of all functions mapping from a domain $\mathcal{X}$ to a range $\mathcal{Y}$.

Unless specified otherwise, we use $\lambda$ to denote the security parameter. We say a function $f(\lambda)$ is negligible in $\lambda$, denoted by $\mathsf{negl}(\lambda)$, if $f(\lambda) = o(1/\lambda^c)$ for all $c \in \mathbb{N}$. We say that an event happens with overwhelming probability if its complement happens with negligible probability. We say an algorithm is efficient if it runs in probabilistic polynomial time in the length of its input. We use $\mathsf{poly}(\lambda)$ to denote a quantity whose value is bounded by a fixed polynomial in $\lambda$. We say that a family of distributions $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda \in \mathbb{N}}$ is *$B$-bounded* if the support of $\mathcal{D}$ is $\{-B, \ldots, B-1, B\}$ with probability 1. For two families of distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, we write $\mathcal{D}_1 \overset{c}{\approx} \mathcal{D}_2$ if the two distributions are computationally indistinguishable (i.e., no efficient algorithm can distinguish $\mathcal{D}_1$ from $\mathcal{D}_2$, except with negligible probability). We write $\mathcal{D}_1 \overset{s}{\approx} \mathcal{D}_2$ if the two distributions are statistically indistinguishable (i.e., the statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is negligible). We now define the circuit-similarity metric we use in this work.

**Definition 3.1** (Circuit Similarity)**.** Fix a circuit class $\mathcal{C}$ on $\rho$-bit inputs. For two circuits $C, C' \in \mathcal{C}$ and for a non-decreasing function $\varepsilon \colon \mathbb{N} \to \mathbb{N}$, we write say that $C$ is $\varepsilon$-close to $C'$, denoted $C \sim_\varepsilon C'$, if $C$ and $C'$ agree on all but an $\varepsilon$-fraction of inputs. More precisely, we write

$$C \sim_\varepsilon C' \iff \Pr[x \overset{\text{R}}{\leftarrow} \{0,1\}^\rho : C(x) \neq C'(x)] \leq \varepsilon.$$

Similarly, we write $C \not\sim_\varepsilon C'$ to denote that $C$ and $C'$ differ on at least an $\varepsilon$-fraction of inputs.

**Vectors and matrices.** We use bold lowercase letters (*e.g.*, $\mathbf{v}, \mathbf{w}$) to denote vectors and bold uppercase letter (*e.g.*, $\mathbf{A}, \mathbf{B}$) to denote matrices. To simplify notation, we often omit the vector transpose notation. In particular, for a vector $\mathbf{s}$ and a matrix $\mathbf{A}$, we simply write $\mathbf{sA}$ to denote their vector-matrix product. Throughout this work, we always use the infinity norm for vectors and matrices. For a vector $\mathbf{x}$, we write $\|\mathbf{x}\|$ to denote $\max_i |x_i|$. Similarly, for a matrix $\mathbf{A}$, we write $\|\mathbf{A}\|$ to denote $\max_{i,j} |A_{i,j}|$. If $\mathbf{x} \in \mathbb{Z}^n$ and $\mathbf{A} \in \mathbb{Z}^{n \times m}$, then $\|\mathbf{xA}\| \leq n \cdot \|\mathbf{x}\| \cdot \|\mathbf{A}\|$.

**Modular rounding.** For an integer $p \leq q$, we define the modular "rounding" function

$$\lfloor \cdot \rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p \text{ that maps } x \to \lfloor (p/q) \cdot x \rceil$$

and extend it coordinate-wise to matrices and vectors over $\mathbb{Z}_q$. Here, the operation $\lfloor \cdot \rceil$ is the rounding operation over the real numbers.

**The gadget matrix.** We define the "gadget matrix" $\mathbf{G} = \mathbf{g} \otimes \mathbf{I}_n \in \mathbb{Z}_q^{n \times n \cdot \lceil \log q \rceil}$ where $\mathbf{g} = (1, 2, 4, \ldots, 2^{\lceil \log q \rceil - 1})$. We define the inverse function $\mathbf{G}^{-1} : \mathbb{Z}_q^{n \times m} \to \mathbb{Z}_q^{n \lceil \log q \rceil \times m}$ which expands each entry $x \in \mathbb{Z}_q$ in the input matrix into a column of size $\lceil \log q \rceil$ consisting of the bits of the binary representation of $x$. To simplify the notation, we always assume that $\mathbf{G}$ has width $m$ (in our construction, $m = \Theta(n \log q)$). Note that this is without loss of generality since we can always extend $\mathbf{G}$ by appending all-zero columns. We have the property that for any matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we have that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$.

## 3.1 Lattice Preliminaries

In this section, we provide some background on the computational lattice problems and lattice-based techniques that we use in this work.

**The 1D-SIS problem.** Following [BV15, BKM17, KW17], we also use a special case of the well-known short integer solution (SIS) problem that was introduced by Ajtai [Ajt96] and studied in a series of works [Mic04, MR07, MP13, GINX16].

**Definition 3.2** (One-Dimensional Short Integer Solution [Ajt96])**.** Fix a security parameter $\lambda$ and integers $m = m(\lambda)$, $q = q(\lambda)$, and $B = B(\lambda)$. The one-dimensional short integer solution (1D-SIS) problem $\mathsf{1D\text{-}SIS}_{m,q,B}$ is defined as follows:

$$\text{given } \mathbf{v} \overset{\text{R}}{\leftarrow} \mathbb{Z}_q^m, \text{ compute } \mathbf{z} \in \mathbb{Z}^m \text{ such that } \|z\| \leq B \text{ and } \langle \mathbf{v}, \mathbf{z} \rangle = 0 \bmod q.$$

The $\mathsf{1D\text{-}SIS}_{m,q,B}$ assumption states that no efficient adversary is able to solve the $\mathsf{1D\text{-}SIS}_{m,q,B}$ problem except with negligible probability.

The hardness of the $\mathsf{1D\text{-}SIS}_{m,p,q,B}$ problem has been studied in a series of works [Mic04, MR07, MP13, GINX16] for various choices of the modulus $q$. In particular, the work of [GINX16] shows that the 1D-SIS problem is as hard as approximating worst-case various lattice problems (e.g., $\mathsf{GapSVP}$ or $\mathsf{SIVP}$) on any $n$-dimensional lattice where $n = \tilde{O}(\log q)$ to within a factor of of $B \cdot \tilde{O}(\sqrt{n})$.

In this work, we make use of the "rounded" variant of the 1D-SIS assumption, which was first introduced in [BV15] for constructing single-key circuit-constrained PRFs and used in [BKM17, KW17, PS18] for constructing single-key private constrained PRFs.

**Definition 3.3** (One-Dimensional Rounded Short Integer Solution [BV15, BKM17])**.** Fix a security parameter $\lambda$ and integers $m = m(\lambda)$, $p = p(\lambda)$, $q = q(\lambda)$, and $B = B(\lambda)$. The one-dimensional rounded short integer solution (1D-SIS-R) problem $\mathsf{1D\text{-}SIS\text{-}R}_{m,p,q,B}$ problem is defined as follows:

$$\text{given } \mathbf{v} \xleftarrow{\text{R}} \mathbb{Z}_q^m, \text{ compute } \mathbf{z} \in \mathbb{Z}^m \text{ such that } \|z\| \leq B,$$

and one of the following conditions hold:

$$\langle \mathbf{v}, \mathbf{z} \rangle \in [-B, B] + (q/p) \cdot \mathbb{Z} \qquad \text{or} \qquad \langle \mathbf{v}, \mathbf{z} \rangle \in [-B, B] + (q/p)(\mathbb{Z} + 1/2).^{9}$$

The $\mathsf{1D\text{-}SIS\text{-}R}_{m,p,q,B}$ assumption states that no efficient adversary can solve the $\mathsf{1D\text{-}SIS\text{-}R}_{m,p,q,B}$ problem except with negligible probability.

The works of [BV15, BKM17] show that the $\mathsf{1D\text{-}SIS\text{-}R}_{m,p,q,B}$ problem is as hard as the $\mathsf{1D\text{-}SIS}_{m,q/p,B}$ problem and therefore, is as hard as approximating certain worst-case lattice problems (e.g., $\mathsf{GapSVP}$ or $\mathsf{SIVP}$) on any $n$-dimensional lattice where $n = \tilde{O}(\log(q/p))$ to within a factor of $B \cdot O(\sqrt{n})$.

**Learning with errors.** The learning with errors (LWE) assumption was first introduced by Regev [Reg05]. In the same work, Regev showed that solving LWE in the *average case* is as hard as (quantumly) approximating several standard lattice problems in the *worst case*. We state the assumption below.

**Definition 3.4** (Learning with Errors [Reg05])**.** Fix a security parameter $\lambda$ and integers $n = n(\lambda)$, $m = m(\lambda)$, $q = q(\lambda)$, and an error (or noise) distribution $\chi = \chi(\lambda)$ over the integers. Then, the (decisional) learning with errors (LWE) assumption $\mathsf{LWE}_{n,m,q,\chi}$ states that for $\mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, $\mathbf{e} \xleftarrow{\text{R}} \chi^m$, and $\mathbf{u} \xleftarrow{\text{R}} \mathbb{Z}_q^m$, the following two distributions are computationally indistinguishable:

$$(\mathbf{A}, \mathbf{sA} + \mathbf{e}) \quad \text{and} \quad (\mathbf{A}, \mathbf{u})$$

When the error distribution $\chi$ is $B$-bounded (oftentimes, a discrete Gaussian distribution), and under mild conditions on the modulus $q$, the $\mathsf{LWE}_{n,m,q,\chi}$ assumption is true assuming various worst-case lattice problems such as $\mathsf{GapSVP}$ and $\mathsf{SIVP}$ on an $n$-dimensional lattice are hard to approximate within a factor of $\tilde{O}(n \cdot q/B)$ by a quantum algorithm [Reg05]. Similar reductions of LWE to the *classical* hardness of approximating worst-case lattice problems are also known [Pei09, ACPS09, MM11, MP12, BLP$^{+}$13].

**Lattice trapdoors.** Although LWE is believed to be hard, with some auxiliary trapdoor information, the problem becomes easy. Lattice trapdoors have been used in a wide variety of context and are studied extensively in the literature [Ajt99, GPV08, AP09, MP12, LW15]. Since the specific details of the trapdoor constructions are not necessary for this work, we highlight just the properties we require in the following theorem.

[9] Here, we write $(q/p)(\mathbb{Z} + 1/2)$ to denote values of the form $\lfloor q/2p \rceil + (q/p) \cdot \mathbb{Z}$.

**Theorem 3.5** (Lattice Trapdoors [Ajt99, GPV08, AP09, MP12, LW15])**.** *Fix a security parameter $\lambda$, lattice parameters $n$, $m$, $q$, and a rounding modulus $p$ where $m = \Omega(n \log q)$ and $q = \Omega(np\sqrt{\log q})$. Then, there exists a tuple of efficient algorithms* $(\mathsf{TrapGen}, \mathsf{Invert})$ *with the following properties:*

- $\mathsf{TrapGen}(1^\lambda) \to (\mathbf{A}, \mathsf{td})$*: On input the security parameter $\lambda$, the trapdoor generation algorithm outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a trapdoor* $\mathsf{td}$.
- $\mathsf{Invert}(\mathsf{td}, \mathbf{v}) \to \mathbf{s}/\bot$*: On input a trapdoor* $\mathsf{td}$*, and a vector $\mathbf{v} \in \mathbb{Z}_p^m$, the inversion algorithm outputs a vector $\mathbf{s} \in \mathbb{Z}_q^n$ or $\bot$.*
- *For all vectors $\mathbf{s} \in \mathbb{Z}_q^n$, $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^\lambda)$, we have*
$$\Pr\Big[\mathsf{Invert}\big(\mathsf{td}, \lfloor \mathbf{s} \cdot \mathbf{A} \rceil_p\big) = \mathbf{s}\Big] = 1.$$
- *For $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^\lambda)$ and $\mathbf{A}' \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$, we have $\Delta(\mathbf{A}, \mathbf{A}') \le q^{-n}$.*

## 3.2 Embedding Circuits into Matrices

We also use the matrix encoding scheme introduced by Boneh et al. [BGG+14]. A matrix encoding scheme allows one to embed a sequence of input bits $x_1, \ldots, x_\rho \in \{0, 1\}$ into matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\rho \in \mathbb{Z}_q^{n \times m}$. Moreover, it is possible to homomorphically evaluate any circuit (of *a priori* bounded depth) over the encoded input bits. The specific implementation of the homomorphic operations are non-essential to this work, so we just give the basic schematic that we need in the theorem below:

**Theorem 3.6** (Matrix Embeddings [BGG+14])**.** *Fix a security parameter $\lambda$, and lattice parameters $n, m, q$. Then, there exist a pair of efficiently-computable algorithms* $(\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}})$ *with the following syntax:*

- $\mathsf{Eval}_{\mathsf{pk}}\big(C, (\mathbf{A}_i)_{i \in [\rho]}\big) \to \mathbf{A}_C$*: On input a circuit $C\colon \{0,1\}^\rho \to \{0,1\}$, and a set of matrices $(\mathbf{A}_i)_{i \in [\rho]}$, the* $\mathsf{Eval}_{\mathsf{pk}}$ *algorithm outputs a matrix $\mathbf{A}_C \in \mathbb{Z}_q^{n \times m}$.*
- $\mathsf{Eval}_{\mathsf{ct}}\big(C, x, (\mathbf{A}_i)_{i \in [\rho]}, (\mathbf{a}_i)_{i \in [\rho]}\big) \to \mathbf{a}_{C,x}$*: On input a circuit $C\colon \{0,1\}^\rho \to \{0,1\}$, an input $x \in \{0,1\}^\rho$, a set of matrices $(\mathbf{A}_i)_{i \in [\rho]}$, and a set of vectors $(\mathbf{a}_i)_{i \in [\rho]}$, the* $\mathsf{Eval}_{\mathsf{ct}}$ *algorithm outputs a vector $\mathbf{a}_{C,x} \in \mathbb{Z}_q^m$.*

*Moreover, the algorithms* $(\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}})$ *satisfy the following property. Let $(\mathbf{A}_i)_{i \in [\rho]}$ be a set of matrices in $\mathbb{Z}_q^{n \times m}$, let $x \in \{0,1\}^\rho$ be an input, let $C\colon \{0,1\}^\rho \to \{0,1\}$ be a Boolean circuit of depth $d$, and let $(\mathbf{a}_i)_{i \in [\rho]}$ be vectors of the form*
$$\mathbf{a}_i = \mathbf{s}(\mathbf{A}_i + x_i \cdot \mathbf{G}) + \mathbf{e}_i \quad \forall i \in [\rho],$$
*for some vector $\mathbf{s} \in \mathbb{Z}_q^n$ and where $\|\mathbf{e}_i\| \le B$ for all $i \in [\rho]$. Then, if we compute the matrix $\mathbf{A}_C \leftarrow \mathsf{Eval}_{\mathsf{pk}}\big(C, (\mathbf{A}_i)_{i \in [\rho]}\big)$ and the vector $\mathbf{a}_{C,x} \leftarrow \mathsf{Eval}_{\mathsf{ct}}\big(C, x, (\mathbf{A}_i)_{i \in [\rho]}, (\mathbf{a}_i)_{i \in [\rho]}\big)$, we have that*
$$\mathbf{a}_{C,x} = \mathbf{s}(\mathbf{A}_C + C(x) \cdot \mathbf{G}) + \mathbf{e}_{C,x}, \tag{3.1}$$
*for an error vector $\|\mathbf{e}_{C,x}\| \le B \cdot m^{O(d)}$.*

The private translucent PRF construction from [KW17] introduced a way to the "program" the matrix in Eq. (3.1) so that when $C(x) = 1$, the evaluation procedure outputs $\mathbf{s}(\mathbf{A}_C + \mathbf{D}) + \mathbf{e}_{C,x}$, for an arbitrary matrix $\mathbf{D}$ (instead of the gadget matrix $\mathbf{G}$). The ability to program the target matrix will be useful in the security analysis of our puncturable extractable PRF construction (Section 4). Again, we do not need the specific details of how the homomorphic evaluation and programming is implemented, so we abstract out the core construction from [KW17] in the following theorem:

**Theorem 3.7** (Matrix Embeddings with Programming [KW17])**.** *Fix a security parameter $\lambda$ and lattice parameters $n, m, q$. Then, there exists a pair of efficiently-computable algorithms* $(\mathsf{EvalP_{pk}}, \mathsf{EvalP_{ct}})$ *with the following syntax:*

- $\mathsf{EvalP_{pk}}\big(C, (\mathbf{A}_i)_{i\in[\rho]}, (\widetilde{\mathbf{A}}_{\alpha,\beta})_{\alpha\in[n],\beta\in[m]}\big) \to \mathbf{A}_C$*: On input a circuit $C\colon \{0,1\}^\rho \to \{0,1\}$, and two sets of matrices $(\mathbf{A}_i)_{i\in[\rho]}$, $(\widetilde{\mathbf{A}}_{\alpha,\beta})_{\alpha\in[n],\beta\in[m]} \in \mathbb{Z}_q^{n\times m}$, the* $\mathsf{EvalP_{pk}}$ *algorithm outputs a matrix $\mathbf{A}_C \in \mathbb{Z}_q^{n\times m}$.*

- $\mathsf{EvalP_{ct}}\big(C, x, (\mathbf{A}_i)_{i\in[\rho]}, (\widetilde{\mathbf{A}}_{\alpha,\beta})_{\alpha\in[n],\beta\in[m]}, (\mathbf{a}_i)_{i\in[\rho]}, (\widetilde{\mathbf{a}}_{\alpha,\beta})_{\alpha\in[n],\beta\in[m]}\big) \to \mathbf{a}_{x,C}$*: On input a circuit $C\colon \{0,1\}^\rho \to \{0,1\}$, an input $x \in \{0,1\}^\rho$, sets of matrices $(\mathbf{A}_i)_{i\in[\rho]}$, $(\widetilde{\mathbf{A}}_{\alpha,\beta})_{\alpha\in[n],\beta\in[m]} \in \mathbb{Z}_q^{n\times m}$, and sets of vectors $(\mathbf{a}_i)_{i\in[\rho]}$, $(\widetilde{\mathbf{a}}_{\alpha,\beta})_{\alpha\in[n],\beta\in[m]} \in \mathbb{Z}_q^m$, the* $\mathsf{EvalP_{ct}}$ *algorithm outputs a vector $\mathbf{a}_{C,x} \in \mathbb{Z}_q^m$.*

*Moreover, the algorithms* $(\mathsf{EvalP_{pk}}, \mathsf{EvalP_{ct}})$ *satisfy the following property. First, define the following quantities:*

- *Let $(\mathbf{A}_i)_{i\in[\rho]}$ and $(\widetilde{\mathbf{A}}_{\alpha,\beta})_{\alpha\in[n],\beta\in[m]}$ be collections of matrices in $\mathbb{Z}_q^{n\times m}$.*
- *Let $\mathbf{D} \in \mathbb{Z}_q^{n\times m}$ be a matrix where the $(\alpha,\beta)^{th}$ entry is $d_{\alpha,\beta}$.*
- *Let $x \in \{0,1\}^\rho$ and let $C\colon \{0,1\}^\rho \to \{0,1\}$ be a Boolean circuit of depth $d$.*
- *Let $(\mathbf{a}_i)_{i\in[\rho]}$, $(\widetilde{\mathbf{a}}_{\alpha,\beta})_{\alpha\in[n],\beta\in[m]}$ be vectors of the form*

$$\begin{aligned} \mathbf{a}_i &= \mathbf{s}(\mathbf{A}_i + x_i \cdot \mathbf{G}) + \mathbf{e}_i \quad \forall i \in [\rho], \\ \widetilde{\mathbf{a}}_{\alpha,\beta} &= \mathbf{s}(\mathbf{A}_{\alpha,\beta} + d_{\alpha,\beta} \cdot \mathbf{G}) + \mathbf{e}_{\alpha,\beta} \quad \forall \alpha \in [n], \beta \in [m], \end{aligned}$$

  *for some vector $\mathbf{s} \in \mathbb{Z}_q^n$ and $\|\mathbf{e}_i\|, \|\widetilde{\mathbf{e}}_{j,k}\| \le B$ for all $i \in [\rho]$, $\alpha \in [n]$, and $\beta \in [m]$.*

*Then, if we compute the matrix $\mathbf{A}_C \leftarrow$* $\mathsf{EvalP_{pk}}\big(C, (\mathbf{A}_i)_{i\in[\rho]}, (\widetilde{\mathbf{A}}_{\alpha,\beta})_{\alpha\in[n],\beta\in[m]}\big)$ *and the vector $\mathbf{a}_{x,C} \leftarrow$* $\mathsf{EvalP_{ct}}\big(x, C, (\mathbf{A}_i)_{i\in[\rho]}, (\widetilde{\mathbf{A}}_{\alpha,\beta})_{\alpha\in[n],\beta\in[m]}, (\mathbf{a}_i)_{i\in[\rho]}, (\widetilde{\mathbf{a}}_{\alpha,\beta})_{\alpha\in[n],\beta\in[m]}\big)$*, we have that*

$$\mathbf{a}_{x,C} = \mathbf{s}(\mathbf{A}_C + C(x) \cdot \mathbf{D}) + \mathbf{e}_{x,C},$$

*where $\|\mathbf{e}_{x,C}\| \le B \cdot m^{O(d)}$.*

# 4 Extractable PRF

In this section, we introduce the core notion of an extractable PRF that we use throughout this work. We begin by recalling the definition of a PRF and then introduce our relaxed notion of $T$-restricted pseudorandomness (i.e., pseudorandom everywhere except on a fixed set of $T$ points).

**Definition 4.1** (Pseudorandom Function [GGM84])**.** A pseudorandom function with key-space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ is a function $\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ that can be computed by a deterministic polynomial-time algorithm. A PRF can also include a separate key-generation algorithm $\mathsf{F.KeyGen}(1^\lambda)$ that on input the security parameter $\lambda$, outputs a key $k \in \mathcal{K}$. If no explicit key-generation algorithm is provided, then the default behavior of $\mathsf{F.KeyGen}$ is to sample $k \xleftarrow{\text{R}} \mathcal{K}$. A function function $\mathsf{F}$ is a secure PRF if for all efficient adversaries $\mathcal{A}$ and sampling $k \leftarrow \mathsf{F.KeyGen}(1^\lambda)$, $f \xleftarrow{\text{R}} \mathsf{Funs}[\mathcal{X}, \mathcal{Y}]$, we have that

$$\left| \Pr\left[\mathcal{A}^{\mathsf{F}(k,\cdot)}(1^\lambda) = 1\right] - \Pr\left[\mathcal{A}^{f(\cdot)}(1^\lambda) = 1\right] \right| = \mathsf{negl}(\lambda).$$

**Definition 4.2** ($T$-Restricted Pseudorandomness)**.** Let $\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a function with key-space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ (and optionally, a key-generation algorithm $\mathsf{F.KeyGen}(1^\lambda)$ that takes as input a security parameter $\lambda$ and outputs a key $k \in \mathcal{K}$). We say that $\mathsf{F}$ satisfies $T$-restricted pseudorandomness if there exists a set $S \subseteq \mathcal{X}$ where $|S| \le T$, such that for all efficient adversaries $\mathcal{A}$, and sampling $k \leftarrow \mathsf{F.KeyGen}(1^\lambda)$, $f \xleftarrow{\mathrm{R}} \mathsf{Funs}[\mathcal{X}, \mathcal{Y}]$, we have that

$$\left| \Pr\left[\mathcal{A}^{\mathcal{O}_0(k,\cdot)}(1^\lambda, S) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{O}_1(f,\cdot)}(1^\lambda, S) = 1\right] \right| = \mathsf{negl}(\lambda),$$

where the oracles $\mathcal{O}_0$ and $\mathcal{O}_1$ are defined as follows:

- The $S$-restricted (real) evaluation oracle $\mathcal{O}_0$ takes as input a key $k \in \mathcal{K}$ and a point $x \in \mathcal{X}$ and outputs $\mathsf{F}(k, x)$ if $x \notin S$ and $\perp$ otherwise.
- The $S$-restricted (ideal) evaluation oracle $\mathcal{O}_1$ takes as input a function $f\colon \mathcal{X} \to \mathcal{Y}$ and a point $x \in \mathcal{X}$ and outputs $f(x)$ if $x \notin S$ and $\perp$ otherwise.

In other words, the outputs of $\mathsf{F}$ look pseudorandom to any adversary that cannot query the PRF on the restricted set $S$.

**Remark 4.3** (Pseudorandomness Implies $T$-Restricted Pseudorandomness)**.** It is easy to see that if $\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is pseudorandom, then it also satisfies $T$-restricted pseudorandomness for all $T$. Specifically, we take the restricted set $S$ to be the empty set.

**Remark 4.4** (Selective Pseudorandomness)**.** In our security analysis, it will often be easier to work with a weaker notion of pseudorandomness called *selective pseudorandomness.* We give a more precise definition in Definition A.1. Using a technique called complexity leveraging [BB04a], selective pseudorandomness implies full pseudorandomness (as defined in Definition 4.1) at the expense of a super-polynomial loss in the security reduction. We define a notion of selective $T$-restricted pseudorandomness in the same manner (Remark A.3). In Remark A.17, we describe another approach to obtain full security with only a polynomial loss (based on a "partitioning" argument [BB04b, Wat05]).

**Extractable PRFs.** We now define the syntax and security requirements of an extractable PRF family. We refer to Sections 1.2 and 2.1 for an overview of the definitions.

**Definition 4.5** (Extractable PRF)**.** An *extractable PRF* with key-space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ consists of a tuple of efficient algorithms $\Pi_{\mathsf{EPRF}} = (\mathsf{PrmsGen}, \mathsf{SampleKey}, \mathsf{Eval}, \mathsf{Extract})$ with the following syntax:

- $\mathsf{PrmsGen}(1^\lambda) \to (\mathsf{pp}, \mathsf{td})$: On input the security parameter $\lambda$, the parameter-generation algorithm outputs a set of public parameters $\mathsf{pp}$ and a trapdoor $\mathsf{td}$.
- $\mathsf{SampleKey}(\mathsf{pp}) \to k$: On input the public parameters $\mathsf{pp}$, the key-generation algorithm outputs a PRF key $k \in \mathcal{K}$.
- $\mathsf{Eval}(\mathsf{pp}, k, x) \to y$: On input the public parameters $\mathsf{pp}$, a PRF key $k \in \mathcal{K}$, and input $x \in \mathcal{X}$, the PRF evaluation algorithm outputs a value $y \in \mathcal{Y}$.
- $\mathsf{Extract}(\mathsf{pp}, \mathsf{td}, C) \to k/\perp$: On input the public parameters $\mathsf{pp}$, the trapdoor $\mathsf{td}$, and a circuit $C\colon \mathcal{X} \to \mathcal{Y}$, the extraction algorithm outputs a key $k \in \mathcal{K} \cup \{\perp\}$. Without loss of generality, the $\mathsf{Extract}$ algorithm can also be defined to take a circuit whose domain is any *superset* of the PRF domain $\mathcal{X}$.

The public parameters $\mathsf{pp}$ of an extractable PRF induces a PRF family $\mathsf{F}_{\mathsf{pp}} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ where $\mathsf{F}_{\mathsf{pp}}(k, x) := \mathsf{Eval}(\mathsf{pp}, k, x)$ and $\mathsf{F}_{\mathsf{pp}}.\mathsf{KeyGen}(1^\lambda)$ computes and returns $k \leftarrow \mathsf{SampleKey}(\mathsf{pp})$. Note that the description of the induced PRF family $\mathsf{F}$ does *not* include the trapdoor $\mathsf{td}$.

**Definition 4.6** (Pseudorandomness)**.** Fix a security parameter $\lambda$, and let $\Pi_{\mathsf{PRF}} = (\mathsf{PrmsGen}, \mathsf{SampleKey}, \mathsf{Eval}, \mathsf{Extract})$ be an extractable PRF with key-space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$. We say that $\Pi_{\mathsf{PRF}}$ satisfies pseudorandomness if for $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$, the induced PRF family $\mathsf{F}_{\mathsf{pp}} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a secure PRF.

**Definition 4.7** ($T$-Restricted Pseudorandomness Given the Trapdoor)**.** Fix a security parameter $\lambda$, and let $\Pi_{\mathsf{PRF}} = (\mathsf{PrmsGen}, \mathsf{SampleKey}, \mathsf{Eval}, \mathsf{Extract})$ be an extractable PRF with key-space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$. For a parameter $T \in \mathbb{N}$, we say that $\Pi_{\mathsf{PRF}}$ satisfies $T$-restricted pseudorandomness given the trapdoor if for $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$, the induced PRF family $\mathsf{F}_{\mathsf{pp}} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ satisfies $T$-restricted pseudorandomness even when the distinguisher (i.e., the algorithm $\mathcal{A}$) is given the trapdoor (equivalently, even if the description of $\mathsf{F}_{\mathsf{pp}}$ includes the trapdoor $\mathsf{td}$).

**Definition 4.8** (Extract-and-Test)**.** An extractable PRF $\Pi_{\mathsf{EPRF}} = (\mathsf{PrmsGen}, \mathsf{SampleKey}, \mathsf{Eval}, \mathsf{Extract})$ with key-space $\mathcal{K}$ has an "extract-and-test" extraction algorithm if $\mathsf{Extract}$ can additionally be decomposed into two algorithms $(\mathsf{ExtractCandidates}, \mathsf{TestCandidate})$ with the following properties:

- $\mathsf{ExtractCandidates}(\mathsf{pp}, \mathsf{td}, C) \to S$: On input the public parameters $\mathsf{pp}$, the trapdoor $\mathsf{td}$, and a circuit $C$, the candidate extraction algorithm outputs a (possibly empty) set $S \subseteq \mathcal{K}$ of candidate keys, where $|S| = \mathsf{poly}(\lambda)$.
- $\mathsf{TestCandidate}(\mathsf{pp}, \mathsf{td}, C, k) \to b$: On input the public parameters $\mathsf{pp}$, the trapdoor $\mathsf{td}$, a circuit $C \colon \mathcal{X} \to \mathcal{Y}$, and a candidate key $k \in \mathcal{K}$, the test candidate algorithm outputs a bit $b \in \{0, 1\}$. Note that we allow $\mathsf{TestCandidate}$ to be a *randomized* algorithm.

Moreover, the $\mathsf{Extract}(\mathsf{pp}, \mathsf{td}, C)$ algorithm can be written as follows:

- $\mathsf{Extract}(\mathsf{pp}, \mathsf{td}, C)$: First invoke $\mathsf{ExtractCandidates}(\mathsf{pp}, \mathsf{td}, C)$ to obtain a set $S \subseteq \mathcal{K}$ of candidate keys. For each $k \in S$, compute $b_k \leftarrow \mathsf{TestCandidate}(\mathsf{pp}, \mathsf{td}, C, k)$. Output any $k \in S$ where $b_k = 1$. If $b_k = 0$ for all $k \in S$, output $\bot$.

**Definition 4.9** (Robust Extractability)**.** Fix a security parameter $\lambda$ and closeness parameters $\varepsilon_1, \varepsilon_2$. Let $\Pi_{\mathsf{EPRF}} = (\mathsf{PrmsGen}, \mathsf{SampleKey}, \mathsf{Eval}, \mathsf{Extract})$ be an extractable pseudorandom function with key-space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$. Suppose $\Pi_{\mathsf{EPRF}}$ has an extract-and-test extraction algorithm where for $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$ and $\varepsilon_1 < \varepsilon_2$, the $\mathsf{TestCandidate}$ algorithm satisfies the following two properties:

- For all $k \in \mathcal{K}$ and $C(\cdot) \sim_{\varepsilon_1} \mathsf{Eval}(\mathsf{pp}, k, \cdot)$, $\Pr[\mathsf{TestCandidate}(\mathsf{pp}, \mathsf{td}, C, k) = 1] = 1 - \mathsf{negl}(\lambda)$.
- For all $k \in \mathcal{K}$ and $C(\cdot) \not\sim_{\varepsilon_2} \mathsf{Eval}(\mathsf{pp}, k, \cdot)$, $\Pr[\mathsf{TestCandidate}(\mathsf{pp}, \mathsf{td}, C, k) = 1] = \mathsf{negl}(\lambda)$.

Next, for an adversary $\mathcal{A}$, we define two experiments $\mathsf{ExtReal}_{\mathcal{A}}(\lambda, \varepsilon_1, \varepsilon_2)$ and $\mathsf{ExtIdeal}_{\mathcal{A}}(\lambda, \varepsilon_1, \varepsilon_2)$:

- **Setup phase:** At the start of both experiments, the challenger samples $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$ and gives $\mathsf{pp}$ to $\mathcal{A}$.
- **Query phase:** Adversary $\mathcal{A}$ can issue any (polynomial) number of extraction queries to the challenger. On an extraction oracle query $C \colon \mathcal{X} \to \mathcal{Y}$, the challenger in the two experiments responds as follows:

- $\mathsf{ExtReal}$ : In the real experiment, the challenger replies with $\mathsf{Extract}(\mathsf{pp}, \mathsf{td}, C)$.
- $\mathsf{ExtIdeal}$ : In the ideal experiment, the challenger proceeds as follows:
  - ∗ If there exists a *unique* $k \in \mathcal{K}$ where $C(\cdot) \sim_{\varepsilon_2} \mathsf{Eval}(\mathsf{pp}, k, \cdot)$, the challenger computes $b_k \leftarrow \mathsf{TestCandidate}(\mathsf{pp}, \mathsf{td}, C, k)$. It replies with $k$ if $b_k = 1$ and $\perp$ if $b_k = 0$.
  - ∗ Otherwise, the challenger replies with $\perp$.

- **Output phase:** Once the adversary $\mathcal{A}$ is done making queries, it outputs a bit $b \in \{0, 1\}$. This is the output of the experiment.

We say that $\Pi_{\mathsf{EPRF}}$ satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability if for all (possibly unbounded) adversaries $\mathcal{A}$ making any polynomial number $Q = \mathsf{poly}(\lambda)$ queries, we have that

$$\left|\Pr\left[\mathsf{ExtReal}_{\mathcal{A}}(\lambda, \varepsilon_1, \varepsilon_2) = 1\right] - \Pr\left[\mathsf{ExtIdeal}_{\mathcal{A}}(\lambda, \varepsilon_1, \varepsilon_2) = 1\right]\right| = \mathsf{negl}(\lambda).$$

**Remark 4.10** (Generalized Candidate Testing)**.** In our constructions, we will require a generalized version of $\mathsf{TestCandidate}$ with the following properties:

- The $\mathsf{TestCandidate}$ algorithm is *publicly-computable*; namely, $\mathsf{TestCandidate}$ does *not* depend on the trapdoor $\mathsf{td}$. To make this explicit, in the case where $\mathsf{TestCandidate}$ is publicly-computable, we write the algorithm as $\mathsf{TestCandidate}(\mathsf{pp}, C, k)$.

- If $C_1, C_2$ satisfy $C_1 \sim_\varepsilon C_2$ for some $\varepsilon = \mathsf{negl}(\lambda)$, then for all $\mathsf{pp}$ and all $k \in \mathcal{K}$,

$$\Pr[\mathsf{TestCandidate}(\mathsf{pp}, C_1, k) \neq \mathsf{TestCandidate}(\mathsf{pp}, C_2, k)] = \mathsf{negl}(\lambda).$$

- Instead of taking as input a candidate key $k \in \mathcal{K}$ as input, the $\mathsf{TestCandidate}$ can also take as input an arbitrary circuit $C' \colon \mathcal{X} \to \mathcal{Y}$, with the property that for $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$ and $k \leftarrow \mathsf{SampleKey}(\mathsf{pp})$, and any circuit $C' \colon \mathcal{X} \to \mathcal{Y}$ where $C' \sim_\varepsilon \mathsf{Eval}(\mathsf{pp}, k, \cdot)$ and $\varepsilon = \mathsf{negl}(\lambda)$,

$$\{\mathsf{TestCandidate}(\mathsf{pp}, C, k)\} \overset{s}{\approx} \{\mathsf{TestCandidate}(\mathsf{pp}, C, C')\},$$

  where the randomness is taken over the random coins in $\mathsf{PrmsGen}$, $\mathsf{SampleKey}$, and $\mathsf{TestCandidate}$.

**Key-injectivity.** As discussed in Section 2, a property that is often useful in conjunction with robust extractability is key-injectivity. We define this below.

**Definition 4.11** (Key-Injectivity)**.** Let $\Pi_{\mathsf{EPRF}} = (\mathsf{PrmsGen}, \mathsf{SampleKey}, \mathsf{Eval}, \mathsf{Extract})$ be an extractable PRF with key-space $\mathcal{K}$ and domain $\mathcal{X}$. We say that $\Pi_{\mathsf{EPRF}}$ is key-injective if for $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$, we have

$$\Pr\left[\ \exists\ k_0, k_1 \in \mathcal{K}, x \in \mathcal{X} : \mathsf{Eval}(\mathsf{pp} k_0, x) = \mathsf{Eval}(\mathsf{pp} k_1, x) \wedge k_0 \neq k_1\right] = \mathsf{negl}(\lambda).$$

**Lemma 4.12** (Uniqueness of Keys)**.** *Suppose an extractable PRF* $\Pi_{\mathsf{EPRF}} = (\mathsf{PrmsGen}, \mathsf{SampleKey}, \mathsf{Eval}, \mathsf{Extract})$ *with key-space* $\mathcal{K}$*, domain* $\mathcal{X}$*, and range* $\mathcal{Y}$ *satisfies key-injectivity (Definition 4.11). Then, for all positive constants* $\varepsilon < 1/2$ *and taking* $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$*, it holds with overwhelming probability that for all circuits* $C \colon \mathcal{X} \to \mathcal{Y}$*, if there exists a key* $k \in \mathcal{K}$ *such that* $C(\cdot) \sim_\varepsilon \mathsf{Eval}(\mathsf{pp}, k, \cdot)$*, then* $k$ *is unique.*

*Proof.* Take any circuit $C\colon \mathcal{X} \to \mathcal{Y}$, and suppose there exists a key $k^* \in \mathcal{K}$ where $C(\cdot) \sim_\varepsilon \mathsf{Eval}(\mathsf{pp}, k, \cdot)$. First, for any key $k \in \mathcal{K}$, define the set $S_k$ as

$$S_k = |\{x \in \mathcal{X} : C(x) = \mathsf{Eval}(\mathsf{pp}, k, x)\}| .$$

By assumption, $|S_{k^*}| \geq (1 - \varepsilon) |\mathcal{X}|$. Next, by key-injectivity, with overwhelming probability (over the choice of $\mathsf{pp}$), it holds that for all keys $k \neq k^*$ and all $x \in \mathcal{X}$, $\mathsf{Eval}(\mathsf{pp}, k^*, x) \neq \mathsf{Eval}(\mathsf{pp}, k, x)$. Equivalently, with overwhelming probability, $S_k < \varepsilon |\mathcal{X}|$ for all keys $k \neq k^*$. This means that for all $k \neq k^*$, $\mathsf{Eval}(\mathsf{pp}, k, \cdot)$ and $C_\ell$ differ on *at least* a $(1 - \varepsilon)$-fraction of the domain. Since $1 - \varepsilon > 1/2 > \varepsilon$, this means that $C(\cdot) \not\sim_\varepsilon \mathsf{EX.Eval}(\mathsf{pp}, k, \cdot)$ for all $k \neq k^*$. In other words, $k^*$ is unique. □

## 4.1 Puncturable Extractable PRFs

In a puncturable PRF [BW13, KPTZ13, BGI14], the PRF key $k$ can be used to derive a punctured key $k_{x^*}$ that can be used to evaluate the PRF everywhere *except* the punctured point $x^* \in \mathcal{X}$. Moreover, the actual PRF value $\mathsf{F}(k, x^*)$ remains pseudorandom even given the punctured key. More generally, we can consider puncturing the PRF at a set $S \subseteq \mathcal{X}$. In this case, the punctured key $k_S$ can be used to evaluate the PRF at all points in $\mathcal{X} \setminus S$, while the PRF values at points in $S$ remain pseudorandom. This is also called a *constrained PRF* [BW13]. In our setting, we primarily consider puncturing at sets containing up to $\mathsf{poly}(\lambda)$ elements. We now review the formal definitions.

**Definition 4.13** (Puncturable PRF [BW13, KPTZ13, BGI14, adapted])**.** Let $\Pi_{\mathsf{PRF}} = (\mathsf{PrmsGen}, \mathsf{SampleKey}, \mathsf{Eval}, \mathsf{Extract})$ be an extractable PRF with key-space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ (Definition 4.5). We say that $\Pi_{\mathsf{PRF}}$ is a *puncturable PRF* if there additionally exist two efficient algorithms $(\mathsf{Puncture}, \mathsf{PunctureEval})$ with the following syntax:

- $\mathsf{Puncture}(\mathsf{pp}, k, S) \to k_S$: On input the public parameters $\mathsf{pp}$, a PRF key $k \in \mathcal{K}$, and a puncture set $S \subseteq \mathcal{X}$ (where $|S| = \mathsf{poly}(\lambda)$), the puncturing algorithm outputs a punctured key $k_S$.
- $\mathsf{PunctureEval}(\mathsf{pp}, k_S, x) \to y$: On input the public parameters $\mathsf{pp}$, a punctured key $k_S$, and an input $x \in \mathcal{X}$, the punctured-evaluation algorithm outputs a value $y \in \mathcal{Y}$.

**Correctness.** We introduce two notions of correctness. The standard notion of correctness for puncturable PRFs says that the behavior of the punctured key is identical to that of the real key on all points not in the punctured set. In this work, we consider two relaxation of this notion. First, we show that perfect correctness holds for most keys in the domain (e.g., if a key is sampled honestly using $\mathsf{SampleKey}$ and then punctured, the punctured key and the original key agree on all non-punctured points with overwhelming probability). While this property suffices for applications where we can assume that PRF keys are honestly sampled, in scenarios like watermarking, we additionally give the adversary the ability to choose PRF keys. For this reason, we introduce an additional correctness requirement that says that no efficient adversary is able to find a key $k \in \mathcal{K}$ and a set $S$ such that the punctured key $k_S \leftarrow \mathsf{Puncture}(\mathsf{pp}, k, S)$ disagrees with $k$ on a noticeable fraction of non-punctured points. Essentially, this property says that even if we puncture an adversarially-chosen key, the punctured key is still *almost* functionality-preserving.

**Definition 4.14** (Perfect Correctness for Most Keys)**.** Fix a security parameter $\lambda$, and let $\Pi_{\mathsf{EPRF}} = (\mathsf{PrmsGen}, \mathsf{SampleKey}, \mathsf{Eval}, \mathsf{Extract}, \mathsf{Puncture}, \mathsf{PunctureEval})$ be a puncturable extractable PRF with key-space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$. We say that $\Pi_{\mathsf{EPRF}}$ is statistically correct for honest keys if for all $S \subseteq \mathcal{X}$ where $|S| = \mathsf{poly}(\lambda)$, and setting $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$, $k \leftarrow \mathsf{SampleKey}(\mathsf{pp})$, and $k_S \leftarrow \mathsf{Puncture}(\mathsf{pp}, k, S)$, we have that

$$\Pr\left[\exists x \in \mathcal{X} \setminus S : \mathsf{Eval}(\mathsf{pp}, k, x) \neq \mathsf{PunctureEval}(\mathsf{pp}, k_S, x)\right] = \mathsf{negl}(\lambda).$$

**Definition 4.15** (Almost-Functionality-Preserving for Adversarial Keys)**.** Fix a security parameter $\lambda$, and let $\Pi_{\mathsf{EPRF}} = (\mathsf{PrmsGen}, \mathsf{SampleKey}, \mathsf{Eval}, \mathsf{Extract}, \mathsf{Puncture}, \mathsf{PunctureEval})$ be a puncturable extractable PRF with key-space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$. We say that $\Pi_{\mathsf{EPRF}}$ is almost-functionality-preserving for adversarial keys if for all efficient adversaries $\mathcal{A}$, and taking $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$, $(k, S) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp})$ and $k_S \leftarrow \mathsf{Puncture}(\mathsf{pp}, k, S)$, we have that

$$\Pr\left[x \xleftarrow{\text{R}} \mathcal{X} \setminus S : \mathsf{Eval}(\mathsf{pp}, k, x) \neq \mathsf{PunctureEval}(\mathsf{pp}, k_S, x)\right] = \mathsf{negl}(\lambda).$$

**Definition 4.16** (Puncturing Security)**.** Fix a security parameter $\lambda$, and let $\Pi_{\mathsf{EPRF}} = (\mathsf{PrmsGen}, \mathsf{SampleKey}, \mathsf{Eval}, \mathsf{Extract}, \mathsf{Puncture}, \mathsf{PunctureEval})$ be a puncturable extractable PRF with key-space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$. For an efficient adversary $\mathcal{A}$ and a bit $b$, we define experiment $\mathsf{ExptPunc}_{\Pi_{\mathsf{EPRF}},\mathcal{A}}(\lambda, b)$ as follows:

- **Setup phase**: At the beginning of the experiment, the adversary chooses a set $S \subseteq \mathcal{X}$ with $|S| = \mathsf{poly}(\lambda)$ and a challenge input $x^* \in S$, and sends $(S, x^*)$ to the challenger. Then, the challenger samples the public parameters $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$, a master secret key $k \leftarrow \mathsf{SampleKey}(\mathsf{pp})$, and the punctured key $k_S \leftarrow \mathsf{Puncture}(\mathsf{pp}, k, S)$. Furthermore, the challenger computes the challenge evaluation as follows:

  - If $b = 0$, the challenger sets $\mathbf{y}^* \leftarrow \mathsf{Eval}(\mathsf{pp}, k, x^*)$.
  - If $b = 1$, the challenger sets $\mathbf{y}^* \xleftarrow{\text{R}} \mathcal{Y}$.

  Finally, it provides $\mathsf{pp}$, $k_S$, and $y^*$ to $\mathcal{A}$.

- **Query phase**: Adversary $\mathcal{A}$ can issue any (polynomial) number of evaluation queries $x \in S$ to the challenger. The challenger in the two experiments responds with $y \leftarrow \mathsf{Eval}(\mathsf{pp}, k, x)$.

- **Output phase**: Once the adversary $\mathcal{A}$ is done making queries, it outputs a bit $b \in \{0, 1\}$. This is the output of the experiment.

We say that $\Pi_{\mathsf{EPRF}}$ satisfies *selective* puncturing security if for all efficient adversaries $\mathcal{A}$,

$$\left|\Pr\left[\mathsf{ExptPunc}_{\Pi_{\mathsf{EPRF}},\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{ExptPunc}_{\Pi_{\mathsf{EPRF}},\mathcal{A}}(\lambda, 1) = 1\right]\right| = \mathsf{negl}(\lambda).$$

**Remark 4.17** (Selective vs. Adaptive Security)**.** Definition 4.16 requires that the adversary commit to its challenge point $x^* \in \mathcal{X}$ at the beginning of the security game. We can also define an *adaptive* notion of security where the adversary gets to choose the challenge point *after* seeing the public parameters, the punctured key, and making evaluation queries. Note though this security notion is adaptive in the choice of the challenge point, but still selective in the choice of the constraint set $S$. Similar to the setting of selective vs. adaptive pseudorandomness (Remark 4.4), we can use complexity leveraging to boost selective puncturing security to adaptive puncturing security at the expense of a super-polynomial loss in the security reduction. In Remark A.17, we describe another approach of obtaining full adaptivity (in the choice of the challenge evaluation) with only a polynomial loss in the security reduction. This is the technique taken in [BV15] for showing adaptive security of their lattice-based constrained PRF.

**Remark 4.18** (Puncturing Security Implies Pseudorandomness)**.** It is straightforward to see that selective puncturing security implies selective pseudorandomness (Definition A.1), and similarly, that adaptive puncturing implies adaptive pseudorandomness.

## 4.2 Constructing Extractable PRFs

In this section, we present our extractable PRF family from standard lattice assumptions. Although our construction follows the main ideas that we outlined in Section 2, implementing these ideas algebraically is non-trivial. We begin with a brief algebraic overview of our construction.

**Construction overview.** As discussed in Section 2, our PRF family is defined with respect to a set of public matrices in $\mathbb{Z}_q^{n \times m}$, which we denote by $(\mathbf{A}_j)_{j \in [\rho]}$, $(\widetilde{\mathbf{A}}_{\alpha,\beta})_{\alpha \in [n], \beta \in [m]}$, $(\mathbf{B}_{i,j})_{i \in [t], j \in [\rho]}$, $(\mathbf{C}_j)_{j \in [\rho]}$, $\mathbf{V}$, and $\mathbf{W}$. These matrices can be logically partitioned into three sets of matrices that handle different correctness or security goals.

- The matrices $(\mathbf{A}_j)_{j \in [\rho]}$, $(\widetilde{\mathbf{A}}_{\alpha,\beta})_{\alpha \in [n], \beta \in [m]}$ are used for the $T$-restricted pseudorandomness proof. As discussed in Section 2.2, handling the evaluation queries in $T$-restricted pseudorandomness requires generating a punctured key that is specifically programmed to enable simulation of the key-dependent shift (i.e., the $\lfloor \mathbf{s} \cdot \mathbf{W} \rceil_p$ term in Eq. (2.2)). Specifically, a key step in the proof of $T$-restricted pseudorandomness (Theorem 4.25) will rely on generating a punctured key with respect to the matrices $(\mathbf{A}_j)_{j \in [\rho]}$, $(\widetilde{\mathbf{A}}_{\alpha,\beta})_{\alpha \in [n], \beta \in [m]}$.
- The matrices $(\mathbf{B}_{i,j})_{i \in [t], j \in [\rho]}$, $(\mathbf{C}_j)_{j \in [\rho]}$, $\mathbf{V}$ implement the constrained PRF construction of [BV15]. The punctured key that is generated by the puncturing algorithm will be punctured with respect to these matrices.
- The matrix $\mathbf{W}$ is the shift matrix. As described in Section 2, matrix $\mathbf{W}$ is generated by first evaluating $\mathsf{Eval}_{\mathsf{pp}}$ on the rest of the matrices $(\mathbf{A}_j)_{j \in [\rho]}$, $(\widetilde{\mathbf{A}}_{\alpha,\beta})_{\alpha \in [n], \beta \in [m]}$, $(\mathbf{B}_{i,j})_{i \in [t], j \in [\rho]}$, $(\mathbf{C}_j)_{j \in [\rho]}$, $\mathbf{V}$, and then defining it as a corresponding shifted matrix from a trapdoor matrix $\mathbf{D}$.

As discussed in Section 2.1, to achieve robust extractability, we need to embed multiple trapdoors. We support this by simply concatenating together multiple copies of the PRF, where each copy is associated with one of the trapdoors. We now give the formal construction.

**Construction 4.19** (Puncturable Extractable PRFs)**.** Let $\lambda$ be a security parameter, and $\varepsilon_1, \varepsilon_2$ be distance parameters where $0 < \varepsilon_1 < \varepsilon_2 < 1/2$, and $\varepsilon_2 \geq \varepsilon_1 + 1/\mathsf{poly}(\lambda)$. We define the following scheme parameters:

- $(n, m, q, \chi_B)$ – lattice parameters, where $\chi_B$ is a $B$-bounded distribution,
- $p$ – the rounding modulus,
- $t$ – a bound on the number of points to be punctured (indexed by $i$),
- $\rho$ – the bit-length of the PRF input (indexed by $j$),
- $\eta$ – the number of special points where we embed the extraction trapdoor (indexed by $\ell$)

Throughout this section and in the analysis, we will assume that $n, m, t, \rho, \eta = \mathsf{poly}(\lambda)$. Let $(\mathsf{TrapGen}, \mathsf{Invert})$ be the lattice trapdoor algorithms from Theorem 3.5. For an input $x \in \{0,1\}^\rho$, we define the equality function $f_x^{\mathsf{eq}} \colon \{0,1\}^\rho \to \{0,1\}$ where

$$f_x^{\mathsf{eq}}(x^*) = \begin{cases} 1 & \text{if } x = x^* \\ 0 & \text{otherwise.} \end{cases}$$

More generally, for a set of points $S \subseteq \{0,1\}^\rho$ of size $t$ (represented as a concatenation of the bit-strings in $S$), we define the containment function $f_x^{\mathsf{con}} \colon \{0,1\}^{t\rho} \to \{0,1\}$ where

$$f_x^{\mathsf{con}}(S) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{otherwise.} \end{cases}$$

Note that both the equality circuit $f_x^{\mathsf{eq}}$ and the containment circuit $f_x^{\mathsf{con}}$ for any $x \in \{0,1\}^\rho$ can be computed by a circuit of depth $d = O(\log \rho + \log t) = O(\log \lambda)$. Our (puncturable) extractable PRF $\Pi_{\mathsf{PRF}} = (\mathsf{PrmsGen}, \mathsf{SampleKey}, \mathsf{Eval}, \mathsf{Extract}, \mathsf{Puncture}, \mathsf{PunctureEval})$ with key-space $\mathcal{K} = [-B, B]^n$, domain $\mathcal{X} = \{0,1\}^\rho \backslash \{\mathbf{0}\}$, and range $\mathcal{Y} = \mathbb{Z}_p^{\eta m}$ is defined as follows:

- $\mathsf{PrmsGen}(1^\lambda)$: On input the security parameter $\lambda$, the $\mathsf{PrmsGen}$ algorithm begins by sampling $(\mathbf{A}_j^{(\ell)})_{j \in [\rho]}$, $(\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)})_{\alpha \in [n], \beta \in [m]}$, $(\mathbf{B}_{i,j}^{(\ell)})_{i \in [t], j \in [\rho]}$, $(\mathbf{C}_j^{(\ell)})_{j \in [\rho]}$, $\mathbf{V}^{(\ell)}$ uniformly at random from $\mathbb{Z}_q^{n \times m}$ for every $\ell \in [\eta]$. It also samples a set of $\eta$ special points $h^{(\ell)} \xleftarrow{\text{R}} \{0,1\}^\rho$ along with trapdoor matrices $(\mathbf{D}^{(\ell)}, \mathsf{td}_{\mathbf{D}^{(\ell)}}) \leftarrow \mathsf{TrapGen}(1^\lambda)$ for all $\ell \in [\eta]$. Then, for all $\ell \in [\eta]$, it computes

  - $\mathbf{A}_{h^{(\ell)}}^{(\ell)} \leftarrow \mathsf{EvalP}_{\mathsf{pk}}\Big(f_{h^{(\ell)}}^{\mathsf{eq}}, (\mathbf{A}_j^{(\ell)})_{j \in [\rho]}, (\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)})_{\alpha \in [n], \beta \in [m]}\Big)$,
  - $\mathbf{B}_{h^{(\ell)}}^{(\ell)} \leftarrow \mathsf{Eval}_{\mathsf{pk}}\Big(f_{h^{(\ell)}}^{\mathsf{con}}, (\mathbf{B}_{i,j}^{(\ell)})_{i \in [t], j \in [\rho]}\Big)$,
  - $\mathbf{C}_{h^{(\ell)}}^{(\ell)} \leftarrow \mathsf{Eval}_{\mathsf{pk}}\Big(f_{h^{(\ell)}}^{\mathsf{eq}}, (\mathbf{C}_j^{(\ell)})_{j \in [\rho]}\Big)$,

  and defines the matrix

  $$\mathbf{W}^{(\ell)} = \mathbf{A}_{h^{(\ell)}}^{(\ell)} + \mathbf{B}_{h^{(\ell)}}^{(\ell)} \mathbf{G}^{-1}\big(\mathbf{C}_{h^{(\ell)}}^{(\ell)}\big)\mathbf{G}^{-1}\big(\mathbf{V}^{(\ell)}\big) + \mathbf{D}^{(\ell)} \in \mathbb{Z}_q^{n \times m}. \tag{4.1}$$

  Finally, it outputs

  $$\mathsf{pp} = \Big(\mathbf{W}^{(\ell)}, \big(\mathbf{A}_j^{(\ell)}\big)_{j \in [\rho]}, \big(\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)}\big)_{\alpha \in [n], \beta \in [m]}, \big(\mathbf{B}_{i,j}^{(\ell)}\big)_{i \in [t], j \in [\rho]}, \big(\mathbf{C}_j^{(\ell)}\big)_{j \in [\rho]}, \mathbf{V}^{(\ell)}\Big)_{\ell \in [\eta]}, \tag{4.2}$$

  and $\mathsf{td} = \big(h^{(\ell)}, \mathsf{td}_{\mathbf{D}^{(\ell)}}\big)_{\ell \in [\eta]}$.

- $\mathsf{SampleKey}(\mathsf{pp})$: On input the public parameters $\mathsf{pp}$, the key-generation algorithm samples a key $\mathbf{s} \leftarrow \chi^n$, and outputs the PRF key $k = \mathbf{s}$.

- $\mathsf{Eval}(\mathsf{pp}, k, x)$: On input the public parameters $\mathsf{pp}$ (as specified in Eq. (4.2)), a PRF key $k = \mathbf{s}$, and an input $x \in \{0,1\}^\rho \setminus \{\mathbf{0}\}$, the evaluation algorithm first computes the matrices

  - $\mathbf{A}_x^{(\ell)} \leftarrow \mathsf{EvalP}_{\mathsf{pk}}\Big(f_x^{\mathsf{eq}}, (\mathbf{A}_j^{(\ell)})_{j \in [\rho]}, (\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)})_{\alpha \in [n], \beta \in [m]}\Big)$,
  - $\mathbf{B}_x^{(\ell)} \leftarrow \mathsf{Eval}_{\mathsf{pk}}\Big(f_x^{\mathsf{con}}, (\mathbf{B}_{i,j}^{(\ell)})_{i \in [t], j \in [\rho]}\Big)$,
  - $\mathbf{C}_x^{(\ell)} \leftarrow \mathsf{Eval}_{\mathsf{pk}}\Big(f_x^{\mathsf{eq}}, (\mathbf{C}_j^{(\ell)})_{j \in [\rho]}\Big)$,

  for all $\ell \in [\eta]$. Then, it sets

  $$\mathbf{Z}_x^{(\ell)} = \mathbf{A}_x^{(\ell)} + \mathbf{B}_x^{(\ell)} \mathbf{G}^{-1}(\mathbf{C}_x^{(\ell)})\mathbf{G}^{-1}(\mathbf{V}^{(\ell)}), \tag{4.3}$$

  for all $\ell \in [\eta]$, and computes the vector

  $$\widetilde{\mathbf{y}}_x = \mathbf{s}\big(\mathbf{W}^{(1)} - \mathbf{Z}_x^{(1)} \mid \cdots \mid \mathbf{W}^{(\eta)} - \mathbf{Z}_x^{(\eta)}\big) \in \mathbb{Z}_q^{\eta m}. \tag{4.4}$$

  Finally, it outputs the rounded vector $\mathbf{y}_x = \lfloor \widetilde{\mathbf{y}} \rceil_p \in \mathbb{Z}_p^{\eta m}$.

- $\mathsf{Extract}(\mathsf{pp}, \mathsf{td}, C)$: The extraction algorithm is defined with respect to two sub-algorithms $\mathsf{ExtractCandidates}$ and $\mathsf{TestCandidate}$ (as in Definition 4.8) that are defined as follows:

- $\mathsf{ExtractCandidates}(\mathsf{pp}, \mathsf{td}, C)$: On input the public parameters $\mathsf{pp}$ (as specified in Eq. (4.2)), a trapdoor $\mathsf{td} = \big(h^{(\ell)}, \mathsf{td}_{\mathbf{D}^{(\ell)}}\big)_{\ell \in [\eta]}$, and a circuit $C \colon \{0,1\}^\rho \to \mathbb{Z}_p^{\eta m}$, the candidate extraction algorithm evaluates the circuit on the test points to get $\mathbf{y}^{(\ell)} \leftarrow C\big(h^{(\ell)}\big)$ for all $\ell \in [\eta]$. Then, for all $\ell \in [\eta]$, it parses the vector $\mathbf{y}^{(\ell)} = (\mathbf{y}_1^{(\ell)} \mid \cdots \mid \mathbf{y}_\eta^{(\ell)})$ where each $\mathbf{y}_1^{(\ell)}, \ldots, \mathbf{y}_\eta^{(\ell)} \in \mathbb{Z}_p^m$. Then, it extracts $\mathbf{s}^{(\ell)} \leftarrow \mathsf{Invert}(\mathsf{td}_{\mathbf{D}^{(\ell)}}, \mathbf{y}_\ell^{(\ell)})$ and outputs the set of all $\mathbf{s}^{(\ell)}$ for which $\mathbf{s}^{(\ell)} \neq \perp$ and $\mathbf{s}^{(\ell)} \in [-B, B]^n$.
- $\mathsf{TestCandidate}(\mathsf{pp}, C, k)$: Let $\delta = (\varepsilon_2 - \varepsilon_1)/2 = 1/\mathsf{poly}(\lambda)$, $\varepsilon = \varepsilon_1 + \delta$, and $\xi = \lambda/\delta^2 = \mathsf{poly}(\lambda)$. On input the public parameters $\mathsf{pp}$, a key $k = \mathbf{s}$, and a circuit $C \colon \{0,1\}^\rho \to \mathbb{Z}_p^{\eta m}$, the test candidate algorithm samples $x_1^*, \ldots, x_\xi^* \xleftarrow{\text{R}} \{0,1\}^\rho$ and computes the number $N_{\mathbf{s}}$ of indices $i \in [\xi]$ where $C(x_i^*) \neq \mathsf{Eval}(\mathsf{pp}, \mathbf{s}, x_i^*)$. If $N_{\mathbf{s}} \leq \varepsilon\xi$, then output 1. Otherwise, output 0. Note that $\mathsf{TestCandidate}$ is *publicly-computable* (it does not require a trapdoor).

The full extraction algorithm follows the extract-and-test procedure described in Definition 4.8.

- $\mathsf{Puncture}(\mathsf{pp}, k, S)$: On input the public parameter $\mathsf{pp}$ (as specified in Eq. (4.2)), a PRF key $k = \mathbf{s}$, and a set of points to be punctured $S = \{x_i\}_{i \in [t]}$, the $\mathsf{Puncture}$ algorithm first samples error vectors $\mathbf{e}_{\mathbf{A},j}^{(\ell)}, \mathbf{e}_{\widetilde{\mathbf{A}},\alpha,\beta}^{(\ell)}, \mathbf{e}_{\mathbf{B},i,j}^{(\ell)}, \mathbf{e}_{\mathbf{W}}^{(\ell)} \leftarrow \chi^m$ for all $i \in [t]$, $j \in [\rho]$, $\alpha \in [n]$, $\beta \in [m]$, and $\ell \in [\eta]$. Then, for each $\ell \in [\eta]$ it defines the vectors

  - $\mathbf{a}_j^{(\ell)} = \mathbf{s}\mathbf{A}_j^{(\ell)} + \mathbf{e}_{\mathbf{A},j}^{(\ell)}$ for all $j \in [\rho]$,
  - $\tilde{\mathbf{a}}_{\alpha,\beta}^{(\ell)} = \mathbf{s}\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)} + \mathbf{e}_{\widetilde{\mathbf{A}},\alpha,\beta}^{(\ell)}$ for all $\alpha \in [n]$ and $\beta \in [m]$,
  - $\mathbf{b}_{i,j}^{(\ell)} = \mathbf{s}\big(\mathbf{B}_{i,j}^{(\ell)} + x_{i,j} \cdot \mathbf{G}\big) + \mathbf{e}_{\mathbf{B},i,j}^{(\ell)}$ for all $i \in [t]$ and $j \in [\rho]$,
  - $\mathbf{w}^{(\ell)} = \mathbf{s}\mathbf{W}^{(\ell)} + \mathbf{e}_{\mathbf{W}}^{(\ell)}$.

  Finally, it outputs the punctured key

  $$k_S = \Big(S, \big(\mathbf{w}^{(\ell)}, (\mathbf{a}_j^{(\ell)})_{j \in [\rho]}, (\tilde{\mathbf{a}}_{\alpha,\beta}^{(\ell)})_{\alpha \in [n], \beta \in [m]}, (\mathbf{b}_{i,j}^{(\ell)})_{i \in [t], j \in [\rho]}\big)_{\ell \in [\eta]}\Big). \tag{4.5}$$

- $\mathsf{PunctureEval}(\mathsf{pp}, k_S, x)$: On input the public parameters $\mathsf{pp}$ (as specified in Eq. (4.2)), the punctured key $k_S$ (as specified in Eq. (4.5)), and an input $x \in \{0,1\}^\rho \setminus \{\mathbf{0}\}$, the punctured evaluation algorithm computes the following for each $\ell \in [\eta]$:

  - $\mathbf{a}_x^{(\ell)} \leftarrow \mathsf{EvalP}_{\mathsf{ct}}\big(f_x^{\mathsf{eq}}, \mathbf{0}, (\mathbf{A}_j^{(\ell)})_{j \in [\rho]}, (\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)})_{\alpha \in [n], \beta \in [m]}, (\mathbf{a}_j^{(\ell)})_{j \in [\rho]}, (\tilde{\mathbf{a}}_{\alpha,\beta}^{(\ell)})_{\alpha \in [n], \beta \in [m]}\big)$,
  - $\mathbf{b}_x^{(\ell)} \leftarrow \mathsf{Eval}_{\mathsf{ct}}\big(f_x^{\mathsf{con}}, S, (\mathbf{B}_{i,j}^{(\ell)})_{i \in [t], j \in [\rho]}, (\mathbf{b}_{i,j}^{(\ell)})_{i \in [t], j \in [\rho]}\big)$,
  - $\mathbf{C}_x^{(\ell)} \leftarrow \mathsf{Eval}_{\mathsf{pk}}\big(f_x^{\mathsf{eq}}, (\mathbf{C}_j^{(\ell)})_{j \in [\rho]}\big)$.

  Then, for each $\ell \in [\eta]$, it sets

  $$\mathbf{z}_x^{(\ell)} = \mathbf{a}_x^{(\ell)} + \mathbf{b}_x^{(\ell)} \mathbf{G}^{-1}(\mathbf{C}_x^{(\ell)}) \mathbf{G}^{-1}(\mathbf{V}^{(\ell)}), \tag{4.6}$$

  and computes the vector

  $$\mathbf{y}_x = (\mathbf{w}^{(1)} - \mathbf{z}_x^{(1)} \mid \cdots \mid \mathbf{w}^{(\eta)} - \mathbf{z}_x^{(\eta)}) \in \mathbb{Z}_q^{\eta m}. \tag{4.7}$$

  Finally, it outputs the rounded vector $\mathbf{y}_x = \lfloor \widetilde{\mathbf{y}}_x \rceil_p \in \mathbb{Z}_p^{\eta m}$.

**Security analysis.** We now show that under the LWE and 1D-SIS-R assumptions (with suitable parameters), the puncturable extractable PRF construction from Construction 4.19 satisfies correctness, puncturing security, and robust extractability. We give the formal theorem statements here, but defer the formal proofs to Appendix A. For the theorems that mention selective notions of security, we note that there is a simple variant of our construction (using admissible hash functions) that achieves adaptive security (see Remarks 4.17 and A.17).

**Theorem 4.20** (Perfect Correctness for Most Keys)**.** *Fix a security parameter $\lambda$ and lattice parameters $n, m, q, p, B$. Suppose the conditions in Theorem 3.5 hold and $2^\rho B \cdot m^{O(\log \lambda)} \cdot p/q = \mathsf{negl}(\lambda)$. Then, the extractable PRF $\Pi_{\mathsf{EPRF}}$ from Construction 4.19 satisfies perfect correctness for most keys (Definition 4.14).*

**Theorem 4.21** (Almost-Functionality-Preserving for All Keys)**.** *Fix a security parameter $\lambda$ and lattice parameters $n, m, q, p, B$. Suppose the conditions in Theorem 3.5 hold and $\rho = \omega(\log \lambda)$. Then, under the $\mathsf{1D\text{-}SIS\text{-}R}_{m',p,q,E}$ assumption for $m' = nm\eta$ and $E = B \cdot m^{O(\log \lambda)}$, the extractable PRF $\Pi_{\mathsf{EPRF}}$ from Construction 4.19 is almost-functionality-preserving for all keys (Definition 4.15).*

**Theorem 4.22** (Key-Injectivity)**.** *Fix a security parameter $\lambda$ and lattice parameters $n, m, q, p, B$. Suppose the conditions in Theorem 3.5 hold and $2^\rho (4B+1)^n / p^{\eta m} = \mathsf{negl}(\lambda)$. Then, the extractable PRF $\Pi_{\mathsf{EPRF}}$ from Construction 4.19 satisfies key-injectivity (Definition 4.11).*

**Theorem 4.23** (Puncturing Security)**.** *Fix a security parameter $\lambda$ and lattice parameters $n, m, q, p, B$. Suppose the conditions in Theorem 3.5 hold and $2^\rho B \cdot m^{O(\log \lambda)} \cdot p/q = \mathsf{negl}(\lambda)$. Then, under the $\mathsf{LWE}_{n,m',q,\chi}$ assumption for $m' = \eta m(nm + (t+2)\rho + 1) + \eta m$, the extractable PRF $\Pi_{\mathsf{EPRF}}$ from Construction 4.19 satisfies selective puncturing security (Definition 4.16).*

**Corollary 4.24** (Pseudorandomness)**.** *Fix a security parameter $\lambda$ and lattice parameters $n, m, q, p, B$. Suppose the conditions in Theorem 4.23 hold. Then, the extractable PRF $\Pi_{\mathsf{EPRF}}$ from Construction 4.19 satisfies selective pseudorandomness (Definition 4.6, Remark 4.4).*

*Proof.* Follows from Theorem 4.23 and Remark 4.18 □

**Theorem 4.25** ($T$-Restricted Psueodrandomness)**.** *Fix a security parameter $\lambda$ and lattice parameters $n, m, q, p, B$. Suppose the conditions in Theorem 3.5 hold and $2^\rho B \cdot m^{O(\log \lambda)} \cdot p/q = \mathsf{negl}(\lambda)$. Then, under the $\mathsf{LWE}_{n,m',q,\chi}$ assumption for $m' = \eta m(nm + \rho(t+2)) + \eta m$, the extractable PRF $\Pi_{\mathsf{EPRF}}$ from Construction 4.19 satisfies selective $T$-restricted pseudorandomness (Definition 4.7, Remark 4.4) for $T = \eta$.*

**Theorem 4.26** (Robust Extractability)**.** *Fix a security parameter $\lambda$ and lattice parameters $n, m, q, p, B$. Take any $0 < \varepsilon_1 < \varepsilon_2 < 1/2$ where $\varepsilon_2 - \varepsilon_1 \geq 1/\mathsf{poly}(\lambda)$. Let $\Pi_{\mathsf{EPRF}}$ be the extractable PRF from Construction 4.19. Suppose the conditions in Theorem 3.5 hold, $m \geq 2n \log q$, $\lceil q/p \rceil \leq q/4$, and $\eta = \omega(\log \lambda)$, and that $\Pi_{\mathsf{EPRF}}$ satisfies key-injectivity (Definition 4.11). Then, $\Pi_{\mathsf{EPRF}}$ satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability (Definition 4.9). Moreover, the* TestCandidate *algorithm in Construction 4.19 satisfies the generalized candidate testing properties from Remark 4.10.*

## 4.3 Concrete Parameter Instantiations

In this section, we describe one possible instantiation for the parameters of the extractable PRF scheme in Construction 4.19. We choose our parameters so that the underlying LWE and 1D-SIS assumptions on which we rely reduce to approximating worst-case lattice problems to within a sub-exponential factor $2^{\tilde{O}(n^{1/c})}$ for some constant $c$ where $n$ is the lattice dimension. Let $\lambda$ be a security parameter and take any constant $c > 0$.

- We set the PRF input length to be $\rho = \lambda$ and the number of special points to $\eta = \lambda$. We allow the number of punctured points $t = \mathsf{poly}(\lambda)$ to be any *a priori* bounded polynomial in $\lambda$ (this is set depending on the specific application).
- We define $n_{\mathsf{LWE}} = \lambda^c$ and $n_{\mathsf{1D\text{-}SIS}} = \tilde{O}(\lambda)$. Then, we set the lattice dimension $n = n_{\mathsf{LWE}} = \lambda^c$.
- We set the modulus $q = 2^{2n^{1/c}}$ and take $m = \Theta(n \log q)$ (chosen so that Theorem 3.5 holds). We choose the rounding modulus to be $p = 2^{n^{1/c}}$ and the noise distribution $\chi_B$ to be the discrete Gaussian distribution $D_{\mathbb{Z},\sqrt{n}}$. In this case, $B = \mathsf{poly}(n) = \mathsf{poly}(\lambda)$.

It is easy to check that this setting of the parameters satisfies all of the requirements in Theorems 4.20 through 4.26. Moreover, for this setting of parameters, the underlying LWE and 1D-SIS assumptions that we rely on reduce to the hardness of solving worst-case lattice problems with a sub-exponential approximation factor. We refer to Section 3.1 for background on the worst-case reductions. Specifically, we have the following:

- The LWE assumption needed for Theorems 4.23 and 4.25 reduce to the hardness of solving worst-case lattice problems over a lattice of dimension $n = n_{\mathsf{LWE}}$ with approximation factor $\tilde{O}(n \cdot q/B) = 2^{\tilde{O}(n_{\mathsf{LWE}}^{1/c})}$.
- The 1D-SIS-R assumption for Theorem 4.21 translates to solving worst-case lattice problems over a lattice of dimension $\tilde{O}(\log(q/p)) = \tilde{O}(n^{1/c}) = \tilde{O}(\lambda) = n_{\mathsf{1D\text{-}SIS}}$ with approximation factor $B \cdot \tilde{O}(\sqrt{n_{\mathsf{1D\text{-}SIS}}}) = \mathsf{poly}(\lambda)$.

**Remark 4.27** (Extractable PRFs from Weaker Lattice Assumptions)**.** With the above parameter setting, Construction 4.19 satisfies correctness even against adversarially-chosen keys (Theorem 4.21). If we relax the requirements on the extractable PRF and only require the standard notion of correctness (Definition 4.14), then it is possible to instantiate the parameters such that all of the remaining properties only rely on the hardness of solving worst-case lattice problems with a *quasi-polynomial* approximation factor. Namely, we implement the equality and set-containment functions $f_x^{\mathsf{eq}}$ and $f_x^{\mathsf{con}}$ from Construction 4.19 using matrix branching programs as in [BV14]. As we discuss in Section 5, an extractable PRF satisfying this weaker notion of correctness still suffices to give a watermarkable PRF family (Remark 5.24). Note that our current approach for showing correctness against adversarially-chosen keys relies on hardness of the 1D-SIS assumption. Since the hardness of a 1D-SIS-R instance over $\mathbb{Z}_q^m$ reduces to a worst-case lattice problem in a lattice of dimension $\tilde{O}(\log(q/p))$, we need $q$ to be sub-exponential to have meaningful worst-case security. But when $q$ is sub-exponential (and the noise distribution is polynomially-bounded), the approximation factor obtained from reducing LWE to a worst-case lattice problem is sub-exponential at best. It is interesting whether we can get the stronger notion of correctness from worst-case lattice problems with a quasi-polynomial approximation factors.

# 5 Watermarkable PRFs from Puncturable Extractable PRFs

In this section, we formally introduce the notion of a watermarkable family of PRFs. Our definitions (Section 5.1) are adapted from those of previous work [CHN$^+$16, BLW17, KW17, QWZ18]. Then, in Sections 5.2 through 5.4, we present three constructions of watermarkable families of PRFs. Our first construction (Section 5.2) achieves the weakest notion of *mark-embedding watermarking* in the secret-key setting, but the construction highlights the key connections between extractable PRFs and watermarkable PRFs. Then, building on our basic construction, we show how to extend it to a *message-embedding watermarking* scheme in the secret-key setting (Section 5.3). Finally, we show that we can further extend our message-embedding watermarking scheme to obtain a scheme that supports *public marking* in the random oracle model (Section 5.4).

## 5.1 Watermarking PRFs

We begin by formally introducing the notion of a watermarkable PRF family.

**Definition 5.1** (Watermarkable Family of PRFs)**.** Fix a security parameter $\lambda$ and a message space $\mathcal{M}$. A secretly-extractable, message-embedding watermarkable family of PRFs with key-space $\mathcal{K}$, a domain $\mathcal{X}$, and a range $\mathcal{Y}$ is a tuple of algorithms $\Pi_{\mathsf{WM}} = (\mathsf{Setup}, \mathsf{Mark}, \mathsf{Extract})$ with the following properties:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{wsk})$: On input the security parameter $\lambda$, the setup algorithm outputs public parameters $\mathsf{pp}$ and the watermarking secret key $\mathsf{wsk}$.
- $\mathsf{Mark}(\mathsf{wsk}, k, m) \to C$: On input the watermarking secret key $\mathsf{wsk}$, a PRF key $k \in \mathcal{K}$, and a message $m \in \mathcal{M}$, the mark algorithm outputs a circuit $C \colon \mathcal{X} \to \mathcal{Y}$.
- $\mathsf{Extract}(\mathsf{wsk}, C) \to m$: On input the watermarking secret key $\mathsf{wsk}$ and a circuit $C \colon \mathcal{X} \to \mathcal{Y}$, the extraction algorithm outputs a string $m \in \mathcal{M} \cup \{\perp\}$.

Moreover, $\Pi_{\mathsf{WM}}$ includes the description of a PRF family $\mathsf{F} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. The description of the PRF family may include the public parameters $\mathsf{pp}$ for the watermarkable PRF family, as sampled by the $\mathsf{Setup}$ algorithm. We often refer to $\Pi_{\mathsf{WM}}$ as a watermarking scheme for the PRF family $\mathsf{F} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$.

**Remark 5.2** (Mark-Embedding Watermarking)**.** Definition 5.1 describes a *message-embedding* watermarking scheme that allows the watermarking authority to embed an arbitrary message inside a PRF. To simplify the description of our construction (and just focus on the main ideas), we also consider the weaker notion of *mark-embedding* watermarking where programs are either considered to be MARKED or UNMARKED. Equivalently, this corresponds to Definition 5.1 where $\mathcal{M} = \{\text{MARKED}\}$. When describing a mark-embedding watermarking scheme, we simplify the $\mathsf{Mark}$ algorithm to only take in two parameters: the watermarking secret key $\mathsf{wsk}$ and the PRF key $k$. In this case, we will also often write UNMARKED in place of $\perp$.

**Correctness.** The correctness requirements on a cryptographic watermarking scheme are twofold. First, a watermarked key should behave like the original key almost everywhere (i.e., the behavior of the watermarked key differs from the original key on only a negligible fraction of the domain). While we might hope that the watermarked key *perfectly* preserves the behavior of the original key, assuming indistinguishability obfuscation, this notion is impossible to achieve [BGI+12]. The second requirement is that if we watermark a key with a message, then the extraction algorithm will be able to extract the same message from the watermarked key. We formalize these two properties below:

**Definition 5.3** (Watermarking Correctness)**.** Fix a security parameter $\lambda$ and let $\Pi_{\mathsf{WM}} = (\mathsf{Setup}, \mathsf{Mark}, \mathsf{Extract})$ be a message-embedding watermarking scheme for a PRF family $\mathsf{F} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ with message space $\mathcal{M}$. Then, we say that $\Pi_{\mathsf{WM}}$ is correct if for all messages $m \in \mathcal{M}$, $(\mathsf{pp}, \mathsf{wsk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $k \leftarrow \mathsf{F.KeyGen}(\mathsf{pp})$, and $C \leftarrow \mathsf{Mark}(\mathsf{wsk}, k, m)$, the following two properties hold:

- **Functionality-preserving:** $C(\cdot) \sim_\varepsilon \mathsf{F}(k, \cdot)$ for some $\varepsilon = \mathsf{negl}(\lambda)$.
- **Extraction correctness:** $\Pr[\mathsf{Extract}(\mathsf{wsk}, C) = m] = 1 - \mathsf{negl}(\lambda)$.

**Remark 5.4** (Stronger Notions of Correctness)**.** Definition 5.3 only requires that the correctness properties hold for *honestly-generated* PRF keys. This was also the case in the lattice-based watermarking scheme from [KW17]. The underlying (algebraic) reason in both cases is that the lattice-based puncturable PRFs used in both constructions are not functionality-preserving for all keys (i.e., the behavior of a punctured key matches that of the real key for most, but not all, of the keys in the domain). In our particular case though, we note that our puncturable extractable PRF is functionality-preserving for adversarially-chosen keys (Definition 4.15, Theorem 4.21), so we can in fact argue correctness even for adversarially-chosen keys. Of course, this does not rule out the *existence* of keys $k \in \mathcal{K}$ where watermarking correctness does not hold. To our knowledge, however, ensuring correctness for honestly-generated (or more generally, adversarially-chosen) keys suffices for all of the candidate applications of watermarking. Nonetheless, we note that there are several constructions of cryptographic watermarking (for PRFs) [CHN+16, YAL+17, YAL+18, QWZ18] that do satisfy the strongest notion of correctness for *all* keys.

**Pseudorandomness.** The second property we require on a watermarkable family of PRFs is the usual notion of pseudorandomness for the PRF family. As discussed in Section 1, we also consider a stronger notion where pseudorandomness should hold even against the watermarking authority (i.e., the holder of the watermarking secret key). While many existing watermarking schemes based on obfuscation or lattices [CHN+16, BLW17, KW17] naturally satisfy this property, both our scheme and that of Quach et al. [QWZ18] do not provide full pseudorandomness. However, in our case, we can achieve the weaker notion of $T$-restricted pseudorandomness (Definition 4.7) against the watermarking authority. Intuitively, this means that pseudorandomness is ensured even against the watermarking authority provided that the authority does not see the PRF evaluations on any of $T$ "special" points. We now define these requirements formally.

**Definition 5.5** (Pseudorandomness)**.** Let $\Pi_{\mathsf{WM}} = (\mathsf{Setup}, \mathsf{Mark}, \mathsf{Extract})$ be a watermarking scheme for a PRF family $\mathsf{F} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. We say that $\Pi_{\mathsf{WM}}$ satisfies pseudorandomness if the PRF family $\mathsf{F}$ is a secure PRF (when the public parameters $\Pi_{\mathsf{WM}}$ are honestly generated via $\mathsf{Setup}$).

**Definition 5.6** ($T$-Restricted Pseudorandomness Against the Authority)**.** Let $\Pi_{\mathsf{WM}} = (\mathsf{Setup}, \mathsf{Mark}, \mathsf{Extract})$ be a watermarking scheme for a PRF family $\mathsf{F}$. We say that $\Pi_{\mathsf{WM}}$ satisfies pseudorandomness against the watermarking authority if the PRF family $\mathsf{F}$ satisfies $T$-restricted pseudorandomness even when the distinguisher is given the watermarking secret key $\mathsf{wsk}$ (equivalently, even if the description of $\mathsf{F}$ includes the trapdoor $\mathsf{td}$ sampled by $\mathsf{Setup}$).

**Remark 5.7** (Extended Pseudorandomness)**.** Quach et al. [QWZ18] introduced the stronger notion of *extended pseudorandomness* which asks that the PRF family $\mathsf{F}$ associated with a watermarking scheme $\Pi_{\mathsf{WM}}$ remain secure even if the adversary is given access to the extraction oracle $\mathsf{Extract}(\mathsf{wsk}, \cdot)$ (but not the watermarking secret key $\mathsf{wsk}$). All of the schemes we present in this work also satisfy this stronger notion. In fact, as we discuss in Remark 5.15, in the secret-key setting, any watermarking scheme that satisfies pseudorandomness and unforgeability automatically satisfies extended pseudorandomness. This argument does not generalize to the setting of public marking. Nonetheless, we are able to show that our watermarking scheme with public marking in the random oracle model (Construction 5.32) also satisfies extended pseudorandomness (using a very similar argument as that needed to argue a relaxed variant of unforgeability for the same construction).

**Remark 5.8** (Security Against the Watermarking Authority)**.** When $T$ is polynomial and the domain is super-polynomial, the notion of $T$-restricted pseudorandomness interpolates between strong pseudorandomness and weak pseudorandomness (or even non-adaptive pseudorandomness) in that we allow the adversary the ability to make adaptive queries on almost but a few points. In fact, in our particular construction, the special set of $T$ points not only has negligible density but is statistically hidden from a normal user. Thus, in normal use of the PRF, it is very unlikely that a user would need to be evaluate the PRF on one of the special points. Thus, in these scenarios, we effectively ensure pseudorandomness against the watermarking authority. In contrast, the Quach et al. [QWZ18] construction has the property that given just two evaluations of any PRF, the watermarking authority can completely break pseudorandomness. Namely, their PRF candidate does not even satisfy weak pseudorandomness against the watermarking authority.

**Remark 5.9** (Stronger Notions of Security Against the Watermarking Authority)**.** Our notion of security against the watermarking authority (Definition 5.6) says that a restricted notion of pseudorandomness holds against an adversary that possesses both the public parameters $\mathsf{pp}$ and the watermarking secret key $\mathsf{wsk}$. This definition still assumes that $\mathsf{pp}$ and $\mathsf{wsk}$ are *honestly* generated. A natural question is whether some meaningful notion of pseudorandomness is possible against an adversary that can choose both $\mathsf{pp}$ and $\mathsf{wsk}$.

In our setting, it is not difficult to see that we still achieve $T$-restricted pseudorandomness against a semi-honest authority that generates the parameters (i.e., even if the adversary is given the randomness used for $\mathsf{Setup}$, the PRF family still satisfies $T$-restricted pseudorandomness). However, if the watermarking authority chooses the public parameters maliciously, then we are no longer able to argue (any) security on the resulting PRF family. In the context of lattice-based PRFs, this issue seems more fundamental in that we would minimally require a PRF that remains pseudorandom even against an adversary that can program the public parameters. Constructing such a PRF family is an interesting challenge. We do note that we can still provide some guarantees in this setting. Specifically, recall from Section 1.2 that our puncturable extractable PRF can essentially be viewed as a *translation* of a puncturable PRF satisfying various structural properties. This means that if we start with a puncturable PRF (e.g., [BV15]) where the public parameters are generated *honestly*, we can bootstrap that construction to a puncturable extractable PRF that provides $T$-restricted pseudorandomness even against an adversary that is allowed to choose the additional parameters needed by the extractable PRF family. We note though that this bootstrapping relies on the algebraic structure of the underlying lattice-based PRF, and is not a black-box construction. Nonetheless, this is conceptually-similar to the stronger property ensured by the obfuscation-based watermarking construction of Cohen et al. [CHN$^+$16], who showed how to obtain a watermarkable PRF family from any puncturable PRF. In their construction, the additional parameters needed to support watermarking are *independent* of the underlying PRF family (and thus, can be adversarially-chosen without compromising the security of the underlying PRF family).

**Unforgeability and unremovability.** The main security notions for a cryptographic watermarking scheme we consider are unremovability and unforgeability. While unforgeability is naturally defined for watermarking schemes supporting secret-marking, the same is not true in the public-key setting as the adversary is able to mark any PRF circuit of its choosing. Nonetheless, as we discussed in Section 1.2, we can still capture the spirit of unforgeability by defining a weaker notion that says that the only watermarked circuits an efficient adversary could produce are those that implement a PRF. In other words, no efficient adversary should be able to produce a new circuit whose behavior is drastically different from that of a valid PRF and which would still be considered watermarked. We refer to this property as "weak unforgeability." We now define the basic watermarking security experiment as well as the different security properties we require:

**Definition 5.10** (Watermarking Experiment [BLW17, adapted])**.** Fix a security parameter $\lambda$. Let $\Pi_{\mathsf{WM}} = (\mathsf{Setup}, \mathsf{Mark}, \mathsf{Extract})$ be a watermarking scheme for the PRF family $\mathsf{F} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. Let $\mathcal{A}$

be an adversary. Then the watermarking experiment $\mathsf{ExptWM}_{\Pi_{\mathsf{WM}},\mathcal{A}}(\lambda)$ proceeds as follows. The challenger begins by sampling $(\mathsf{pp}, \mathsf{wsk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and gives $\mathsf{pp}$ to the adversary $\mathcal{A}$. The adversary $\mathcal{A}$ is then given access to the following oracles:

- **Marking oracle.** On input a message $m \in \mathcal{M}$ and a PRF key $k \in \mathcal{K}$, the challenger returns the circuit $C \leftarrow \mathsf{Mark}(\mathsf{wsk}, k, m)$ to $\mathcal{A}$.
- **Extraction oracle.** On input a message $m \in \mathcal{M}$ and a circuit $C \colon \mathcal{X} \to \mathcal{Y}$, the challenger returns $m \leftarrow \mathsf{Extract}(\mathsf{wsk}, C)$.
- **Challenge oracle.** On input a message $\hat{m} \in \mathcal{M}$, the challenger samples $\hat{k} \leftarrow \mathsf{F.KeyGen}(1^\lambda)$, and returns the circuit $\hat{C} \leftarrow \mathsf{Mark}(\mathsf{wsk}, \hat{k}, \hat{m})$ to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs a circuit $\tilde{C} \colon \mathcal{X} \to \mathcal{Y}$. The output of the experiment, denoted $\mathsf{ExptWM}_{\Pi_{\mathsf{WM}},\mathcal{A}}(\lambda)$, is $\mathsf{Extract}(\mathsf{wsk}, \tilde{C})$.

**Definition 5.11** ($\varepsilon$-Unremovability [CHN$^+$16, BLW17, adapted])**.** Fix a security parameter $\lambda$. Let $\Pi_{\mathsf{WM}} = (\mathsf{Setup}, \mathsf{Mark}, \mathsf{Extract})$ be a watermarking scheme for a PRF family $\mathsf{F} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. We say that an adversary $\mathcal{A}$ for the watermarking experiment is *$\varepsilon$-unremoving-admissible* if the following conditions hold:

- The adversary $\mathcal{A}$ makes exactly one query to the challenge oracle.
- The circuit $\tilde{C}$ that $\mathcal{A}$ outputs satisfies $\tilde{C} \sim_\varepsilon \hat{C}$, where $\hat{C}$ is the circuit output by the challenge oracle.

We say that $\Pi_{\mathsf{WM}}$ is *$\varepsilon$-unremovable* if for all efficient and $\varepsilon$-unremoving-admissible adversaries $\mathcal{A}$,

$$\Pr[\mathsf{ExptWM}_{\Pi_{\mathsf{WM}},\mathcal{A}}(\lambda) \neq \hat{m}] = \mathsf{negl}(\lambda),$$

where $\hat{m}$ is the message $\mathcal{A}$ submitted to the challenge oracle in $\mathsf{ExptWM}_{\Pi_{\mathsf{WM}},\mathcal{A}}(\lambda)$.

**Definition 5.12** ($\delta$-Unforgeability [CHN$^+$16, BLW17, adapted])**.** Fix a security parameter $\lambda$. Let $\Pi_{\mathsf{WM}} = (\mathsf{Setup}, \mathsf{Mark}, \mathsf{Extract})$ be a watermarking scheme for a PRF family $\mathsf{F} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. We say that an adversary $\mathcal{A}$ for the watermarking experiment is *$\delta$-unforging-admissible* if the following conditions hold:

- The adversary $\mathcal{A}$ does not make any challenge oracle queries.
- The circuit $\tilde{C}$ that $\mathcal{A}$ outputs satisfies $\tilde{C} \not\sim_\delta C_\ell$ for all $\ell \in [Q]$, where $Q$ is the number of queries $\mathcal{A}$ made to the marking oracle, $C_\ell$ is the output of the marking oracle on the $\ell^{\text{th}}$ query.

Then, we say that $\Pi_{\mathsf{WM}}$ is *$\delta$-unforgeable* if for all efficient and $\delta$-unforging-admissible adversaries $\mathcal{A}$,

$$\Pr[\mathsf{ExptWM}_{\Pi_{\mathsf{WM}},\mathcal{A}}(\lambda) \neq \bot] = \mathsf{negl}(\lambda).$$

**Definition 5.13** (Weak $\delta$-Unforgeability)**.** Fix a security parameter $\lambda$. Let $\Pi_{\mathsf{WM}} = (\mathsf{Setup}, \mathsf{Mark}, \mathsf{Extract})$ be a watermarking scheme for a PRF family $\mathsf{F} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. We say that an adversary $\mathcal{A}$ for the watermarking experiment is *weak $\delta$-unforging-admissible* if the following conditions hold:

- The adversary $\mathcal{A}$ does not make any challenge oracle queries.
- The circuit $\tilde{C}$ that $\mathcal{A}$ outputs satisfies $\tilde{C}(\cdot) \not\sim_\delta \mathsf{F}(k, \cdot)$ for all $k \in \mathcal{K}$.

Then, we say that $\Pi_{\mathsf{WM}}$ is *weak $\delta$-unforgeable* if for all efficient and weak $\delta$-unforging-admissible adversaries $\mathcal{A}$,

$$\Pr[\mathsf{ExptWM}_{\Pi_{\mathsf{WM}},\mathcal{A}}(\lambda) \neq \perp] = \mathsf{negl}(\lambda).$$

**Remark 5.14** (Unforgeability Notions)**.** We note that any watermarking scheme that is $\delta$-unforgeable is also weak $\delta'$-unforgeable for some $\delta' = \delta + \mathsf{negl}(\lambda)$, provided that the underlying watermarking scheme is correct even for adversarially-chosen keys (Remark 5.4). To see this, observe that any adversary that wins the weak $\delta$-unforgeable game must be able to produce a circuit $\tilde{C}$ that is at least $\delta$-far from $\mathsf{F}(k, \cdot)$ for all $k \in \mathcal{K}$. We argue that $\tilde{C}$ is also an admissible forgery for the standard $\delta'$-unforgeability game for some $\delta' = \delta + \mathsf{negl}(\lambda)$. In the $\delta'$-unforgeability game, the adversary has to output a circuit that is at least $\delta'$-far from $C_\ell$ where $C_\ell$ is the circuit from the $\ell^{\text{th}}$ marking query. If the watermarking scheme is correct even for adversarially-chosen keys, then $C_\ell \sim_\varepsilon \mathsf{F}(k_\ell, \cdot)$, where $k_\ell$ is the key the adversary submitted in its $\ell^{\text{th}}$ marking oracle query and $\varepsilon = \mathsf{negl}(\lambda)$. Thus, if $\tilde{C}' \not\sim_\delta \mathsf{F}(k_\ell, \cdot)$, then $\tilde{C} \not\sim_{\delta'} C_\ell$ for $\delta' = \delta + \varepsilon = \delta + \mathsf{negl}(\lambda)$. This means that the circuit $\tilde{C}$ is a valid forgery for the $\delta'$-unforgeability game.

**Remark 5.15** (From Unforgeability to Extended Pseudorandomness)**.** As discussed in Remark 5.7, the notion of extended pseudorandomness from [QWZ18] requires that pseudorandomness for the PRF associated with a watermarking scheme $\Pi_{\mathsf{WM}}$ hold even if the adversary has access to the extraction oracle. If $\Pi_{\mathsf{WM}}$ satisfies $\delta$-unforgeability for some $\delta > 0$, then pseudorandomness implies extended pseudorandomness. In particular, in the pseudorandomness security game, the adversary is not given any marked circuits; $\delta$-unforgeability then says that the adversary cannot produce any circuit that is considered marked. In other words, the response to *all* of the extraction oracle queries an efficient adversary makes is $\perp$. Correspondingly, extended pseudorandomness reduces to standard pseudorandomness. Thus, our watermarking constructions that satisfy $\delta$-unforgeability (Constructions 5.17 and 5.25) also satisfy extended pseudorandomness. Our watermarking scheme that supports public marking in the random oracle model (Construction 5.32) also satisfies extended pseudorandomness (Theorem 5.38); in that case, the argument does not generically follow from unforgeability, but a similar argument as that used to show weak $\delta$-unforgeability does suffice.

**Remark 5.16** (Relaxing Unforgeabiilty)**.** The definition of unforgeability in Definition 5.12 is the standard one introduced by Cohen et al. [CHN$^+$16]. However, we can consider a relaxation of unforgeability from [KW17] where instead of requiring that the circuit $\tilde{C}$ output by the adversary to be sufficiently far from all circuits $C_1, \ldots, C_Q$ that the adversary received from the marking oracle, we instead require that $\tilde{C}$ be sufficiently far from the PRFs $\mathsf{F}(k_1, \cdot), \ldots, \mathsf{F}(k_\ell, \cdot)$ the adversary submitted to the marking oracle. When the underlying PRF is correct for adversarial keys (Definition 4.15), then these two notions are the same. However, when the underlying PRF does not necessarily provide correctness for adversarially-chosen keys, then achieving this relaxed notion of unforgeability is potentially easier. In our setting, if we just consider the relaxed version of unforgeability, then security of our watermarking scheme only needs to rely on the (polynomial) hardness of solving worst-case lattice problems up to a *quasi-polynomial* approximation factor. All previous constructions of watermarking that support message-embedding from standard assumptions could only be reduced to the hardness of worst-case lattice problems with a *sub-exponential* approximation factor.

## 5.2 Mark-Embedding Watermarking

In this section, we present our construction of a mark-embedding watermarkable family of PRFs (in the secret-key setting) from extractable PRFs. While the message-embedding construction in Sections 5.3 strictly subsumes this construction, we present it for pedagogical reasons. The basic construction captures all of the core ideas that underlie our watermarking scheme from extractable PRFs and minimizes the number of technical details that arise with more complex notions of watermarking. We refer to Section 1.2 for a high-level overview of this construction. In the subsequent sections, we build upon this construction to obtain message-embedding watermarking (and, in the random oracle model, message-embedding watermarking with public marking).

**Construction 5.17** (Mark-Embedding Watermarkable PRFs)**.** Let $\lambda$ be a security parameter. Our *mark-embedding* watermarkable PRF relies on the following primitives:

- Let $\Pi_{\mathsf{EPRF}} = (\mathsf{EX.PrmsGen}, \mathsf{EX.SampleKey}, \mathsf{EX.Eval}, \mathsf{EX.Extract}, \mathsf{EX.Puncture}, \mathsf{EX.PunctureEval})$ be a puncturable extractable PRF with key-space $\mathcal{K}_{\mathsf{EPRF}}$, domain $\mathcal{X}$, and range $\mathcal{Y}$.
- Let $\mathsf{PRF} \colon \mathcal{K}_{\mathsf{PRF}} \times \mathcal{K}_{\mathsf{EPRF}} \to \mathcal{X}^{\lambda}$ be a pseudorandom function.

We construct a watermarkable PRF $\Pi_{\mathsf{WM}} = (\mathsf{Setup}, \mathsf{Mark}, \mathsf{Extract})$ as follows:

- $\mathsf{Setup}(1^{\lambda})$: On input the security parameter $\lambda$, the setup algorithm samples a PRF key $k_{\mathsf{PRF}} \xleftarrow{\text{R}} \mathcal{K}_{\mathsf{PRF}}$, and parameters for the extractable PRF $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{EX.PrmsGen}(1^{\lambda})$. It outputs the public parameters $\mathsf{pp}$ and the watermarking secret key $\mathsf{wsk} = (k_{\mathsf{PRF}}, \mathsf{pp}, \mathsf{td})$.
- $\mathsf{Mark}(\mathsf{wsk}, k)$: On input the watermarking secret key $\mathsf{wsk} = (k_{\mathsf{PRF}}, \mathsf{pp}, \mathsf{td})$, and a key $k \in \mathcal{K}_{\mathsf{EPRF}}$, the marking algorithm derives points $(x_1^*, \ldots, x_{\lambda}^*) \leftarrow \mathsf{PRF}(k_{\mathsf{PRF}}, k)$ and a punctured key $k' \leftarrow \mathsf{EX.Puncture}(\mathsf{pp}, k, (x_1^*, \ldots, x_{\lambda}^*))$. It outputs the circuit $C \colon \mathcal{X} \to \mathcal{Y}$ that implements the punctured evaluation algorithm $\mathsf{EX.PunctureEval}(\mathsf{pp}, k', \cdot)$.
- $\mathsf{Extract}(\mathsf{wsk}, C)$: On input the watermarking secret key $\mathsf{wsk} = (k_{\mathsf{PRF}}, \mathsf{pp}, \mathsf{td})$, and a circuit $C \colon \mathcal{X} \to \mathcal{Y}$, the extraction algorithm first extracts a key $k \leftarrow \mathsf{EX.Extract}(\mathsf{pp}, \mathsf{td}, C)$. If $k = \perp$, output UNMARKED. Otherwise, it computes $(x_1^*, \ldots, x_{\lambda}^*) \leftarrow \mathsf{PRF}(k_{\mathsf{PRF}}, k)$. If $C(x_i^*) \neq \mathsf{EX.Eval}(\mathsf{pp}, k, x_i^*)$ for all $i \in [\lambda]$, then output MARKED. Otherwise, output UNMARKED.

The underlying PRF family $\mathsf{F} \colon \mathcal{K}_{\mathsf{EPRF}} \times \mathcal{X} \to \mathcal{Y}$ (induced by the public parameters $\mathsf{pp}$ for the watermarking scheme) is defined as $\mathsf{F}(k, x) := \mathsf{Eval}(\mathsf{pp}, k, x)$ and $\mathsf{F.KeyGen}$ simply returns $\mathsf{EX.SampleKey}(\mathsf{pp})$. Note that the description of the PRF family $\mathsf{F}$ includes the public parameters $\mathsf{pp}$, but not the other components in the watermarking secret key $\mathsf{wsk}$.

**Correctness.** Correctness of our construction follows from correctness and several statistical properties of the underlying extractable PRF (implied by the different security properties on the extractable PRF). We state the theorem below, but defer the formal proof to Appendix B.1.

**Theorem 5.18** (Correctness)**.** *Suppose* $\Pi_{\mathsf{EPRF}}$ *satisfies perfect correctness for most keys (Definition 4.14), key-injectivity (Definition 4.11),* $(\varepsilon_1, \varepsilon_2)$*-robust extractability for* $\varepsilon_1 = 1/\mathsf{poly}(\lambda)$ *(Definition 4.16), and puncturing security (Definition 4.16), and that* $1/|\mathcal{X}| = \mathsf{negl}(\lambda)$*. Then, the watermarking scheme* $\Pi_{\mathsf{WM}}$ *from Construction 5.17 is correct.*

**Pseudorandomness.** Next, we show that our watermarking scheme satisfies both the usual notion of pseudorandomness (against adversaries that do not have the watermarking secret key) as well as the relaxed version of $T$-pseudorandomness against the watermarking authority.

**Theorem 5.19** (Pseudorandomness)**.** *Suppose* $\Pi_{\mathsf{EPRF}}$ *is pseudorandom (Definition 4.6). Then, the watermarking scheme* $\Pi_{\mathsf{WM}}$ *from Construction 5.17 is pseudorandom (Definition 5.5).*

*Proof.* The PRF family associated with $\Pi_{\mathsf{WM}}$ is the PRF family associated with the underlying extractable PRF $\Pi_{\mathsf{EPRF}}$. Thus, pseudorandomness of $\Pi_{\mathsf{EPRF}}$ implies pseudorandomness of $\Pi_{\mathsf{WM}}$. □

**Theorem 5.20** ($T$-Restricted Pseudorandomness Against the Authority)**.** *Suppose* $\Pi_{\mathsf{EPRF}}$ *satisfies $T$-restricted pseudorandomness given the trapdoor (Definition 4.7). Then, the watermarking scheme* $\Pi_{\mathsf{WM}}$ *from Construction 5.17 satisfies $T$-restricted pseudorandomness against the watermarking authority (Definition 5.6).*

*Proof.* First, we note that the PRF family $\mathsf{F}$ associated with $\Pi_{\mathsf{WM}}$ is precisely the PRF family associated with the underlying extractable PRF $\Pi_{\mathsf{EPRF}}$, which satisfies $T$-restricted pseudorandomness given the trapdoor. Thus, given an adversary $\mathcal{A}$ that breaks $T$-restricted pseudorandomness for $\Pi_{\mathsf{WM}}$, we can easily build an adversary $\mathcal{B}$ that breaks $T$-restricted pseudorandomness of $\Pi_{\mathsf{EPRF}}$. Specifically, algorithm $\mathcal{B}$ gets $(\mathsf{pp}, \mathsf{td})$ and a restricted set $S \subseteq \mathcal{X}$ from the $\Pi_{\mathsf{EPRF}}$ challenger, samples a PRF key $k_{\mathsf{PRF}} \xleftarrow{\text{R}} \mathcal{K}_{\mathsf{PRF}}$ and simulates the watermarking secret key as $\mathsf{wsk} = (k_{\mathsf{PRF}}, \mathsf{pp}, \mathsf{td})$. It gives $(\mathsf{wsk}, S)$ to $\mathcal{A}$. To simulate evaluation queries, algorithm $\mathcal{B}$ simply forwards the queries to its evaluation oracle and echoes the response. At the end of the game, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. It is easy to see that $\mathcal{B}$'s distinguishing advantage is the same as $\mathcal{A}$'s, and the claim follows. □

**Unremovability and unforgeability.** We now state our security theorems for unremovability and unforgeability, but defer their formal analysis to Appendix B.1. We then state a simple corollary that says that the watermarking scheme satisfies the notion of extended pseudorandomness from [QWZ18] (see Remarks 5.7 and 5.15).

**Theorem 5.21** (Unremovability)**.** *Suppose that* $1/|\mathcal{Y}| = \mathsf{negl}(\lambda)$*, that* $\mathsf{PRF}$ *is a secure PRF, and that* $\Pi_{\mathsf{EPRF}}$ *satisfies key-injectivity (Definition 4.11),* $(\varepsilon_1, \varepsilon_2)$*-robust extractability (Definition 4.9), the additional properties in Remark 4.10, and selective puncturing security (Definition 4.16). Then, for all* $\varepsilon \leq \varepsilon_1$*, the mark-embedding watermarkable PRF family* $\Pi_{\mathsf{WM}}$ *from Construction 5.17 satisfies* $\varepsilon$*-unremovability (Definition 5.11).*

**Theorem 5.22** (Unforgeability)**.** *Suppose that* $\mathsf{PRF}$ *is a secure PRF and that* $\Pi_{\mathsf{EPRF}}$ *is almost functionality-preserving for adversarially-chosen keys (Definition 4.15) and satisfies* $(\varepsilon_1, \varepsilon_2)$*-robust extractability (Definition 4.9) as well as the additional properties in Remark 4.10. In addition, suppose that* $1/|\mathcal{X}| = \mathsf{negl}(\lambda)$*. Then, for all* $\delta \geq \varepsilon_2$*, the mark-embedding watermarkable PRF family* $\Pi_{\mathsf{WM}}$ *from Construction 5.17 satisfies* $\delta$*-unforgeability (Definition 5.12).*

**Corollary 5.23** (Extended Pseudorandomness)**.** *Suppose the conditions in Theorems 5.19 and 5.22 hold. Then, the watermarking scheme* $\Pi_{\mathsf{WM}}$ *from Construction 5.17 satisfies extended pseudorandomness (Remark 5.7).*

*Proof.* Follows from the argument outlined in Remark 5.15. □

**Remark 5.24** (Relaxed Unforgeability from Weaker Assumptions)**.** Theorem 5.22 only relies on the extractable PRF $\Pi_{\mathsf{EPRF}}$ being functionality-preserving for adversarially-chosen keys to argue the full notion of unforgeability as defined in Definition 5.12. If we consider the relaxed notion from Remark 5.16 (and considered in [KW17]), then this assumption is no longer necessary for the proof. In this case, all of the other properties specified in Theorem 5.22 can be realizing assuming the hardness of solving worst-case lattice problems with a *quasi-polynomial* (rather than *sub-exponential*) approximation factor (Section 4.3, Remark 4.27). This simplification also applies to the message-embedding watermarking constructions in Sections 5.3 and 5.4. This is the weakest assumption from which we currently have message-embedding watermarking. All previous constructions of message-embedding watermarking either relied on obfuscation [CHN+16, BLW17, YAL+17, YAL+18], or worst-case lattice problems with a sub-exponential approximation factor [KW17, QWZ18].

## 5.3 Message-Embedding Watermarking

In this section, we show how to extend our basic mark-embedding watermarking construction from Construction 5.17 to additionally support message embedding. We refer to Section 1.2 for a high-level overview of our construction.

**Construction 5.25** (Message-Embedding Watermarkable PRFs)**.** Let $\lambda$ be a security parameter. Let $\mathcal{M} = \{0,1\}^t$ be the message space for the watermarking scheme. Our message-embedding watermarkable PRF family relies on the following primitives:

- Let $\Pi_{\mathsf{EPRF}} = (\mathsf{EX.PrmsGen}, \mathsf{EX.SampleKey}, \mathsf{EX.Eval}, \mathsf{EX.Extract}, \mathsf{EX.Puncture}, \mathsf{EX.PunctureEval})$ be a puncturable extractable PRF with key-space $\mathcal{K}_{\mathsf{EPRF}}$, domain $\mathcal{X}$, and range $\mathcal{Y}$.
- Let $\mathsf{PRF}\colon \mathcal{K}_{\mathsf{PRF}} \times (\mathcal{K}_{\mathsf{EPRF}} \times [t] \times \{0,1\}) \to \mathcal{X}^\lambda$ be a pseudorandom function.

We construct a watermarkable PRF $\Pi_{\mathsf{WM}} = (\mathsf{Setup}, \mathsf{Mark}, \mathsf{Extract})$ as follows:

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter $\lambda$, the setup algorithm samples a PRF key $k_{\mathsf{PRF}} \xleftarrow{\text{R}} \mathcal{K}_{\mathsf{PRF}}$, and parameters for the extractable PRF $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{EX.PrmsGen}(1^\lambda)$. It outputs the public parameters $\mathsf{pp}$ and the watermarking secret key $\mathsf{wsk} = (k_{\mathsf{PRF}}, \mathsf{pp}, \mathsf{td})$.
- $\mathsf{Mark}(\mathsf{wsk}, k, m)$: On input the watermarking secret key $\mathsf{wsk} = (k_{\mathsf{PRF}}, \mathsf{pp}, \mathsf{td})$, a key $k \in \mathcal{K}_{\mathsf{EPRF}}$, and a message $m \in \{0,1\}^t$, the marking algorithm derives a collection of points $(x^*_{i,1}, \ldots, x^*_{i,\lambda}) \leftarrow \mathsf{PRF}(k_{\mathsf{PRF}}, (k, i, m_i))$ for each $i \in [t]$. It then constructs a punctured key $k' \leftarrow \mathsf{EX.Puncture}(\mathsf{pp}, k, \{x^*_{i,j}\}_{i \in [t], j \in [\lambda]})$. Finally, it outputs the circuit $C\colon \mathcal{X} \to \mathcal{Y}$ that implements the punctured evaluation algorithm $\mathsf{EX.PunctureEval}(\mathsf{pp}, k', \cdot)$.
- $\mathsf{Extract}(\mathsf{wsk}, C)$: On input the watermarking secret key $\mathsf{wsk} = (k_{\mathsf{PRF}}, \mathsf{pp}, \mathsf{td})$, and a circuit $C\colon \mathcal{X} \to \mathcal{Y}$, the extraction algorithm first extracts a key $k \leftarrow \mathsf{EX.Extract}(\mathsf{pp}, \mathsf{td}, C)$. If $k = \bot$, output $\bot$. Otherwise, for each $i \in [t]$ and $b \in \{0,1\}$, the extraction algorithm computes $(x^*_{i,b,1}, \ldots, x^*_{i,b,\lambda}) \leftarrow \mathsf{PRF}(k_{\mathsf{PRF}}, (k, i, b))$. Let $N_{i,b}$ denote the number of indices $j \in [\lambda]$ where $C(x^*_{i,b,j}) \neq \mathsf{EX.Eval}(\mathsf{pp}, k, x^*_{i,b,j})$. If there exists an index $i \in [t]$ where $N_{i,0}, N_{i,1} < 2\lambda/3$ or $N_{i,0}, N_{i,1} > 2\lambda/3$, then output $\bot$. Otherwise, for each $i \in [t]$, let $b_i \in \{0,1\}$ be the unique bit where $N_{i,b_i} > 2\lambda/3$. Output the message $m = b_1 \cdots b_t$.

The underlying PRF family $\mathsf{F}\colon \mathcal{K}_{\mathsf{EPRF}} \times \mathcal{X} \to \mathcal{Y}$ (induced by the public parameters $\mathsf{pp}$ for the watermarking scheme) is defined as $\mathsf{F}(k,x) := \mathsf{Eval}(\mathsf{pp}, k, x)$ and $\mathsf{F.KeyGen}$ simply returns $\mathsf{EX.SampleKey}(\mathsf{pp})$. Note that the description of the PRF family $\mathsf{F}$ includes the public parameters $\mathsf{pp}$, but not the other components in the watermarking secret key $\mathsf{wsk}$.

**Correctness and security analysis.** We now state our main security theorems, but defer the formal analysis of Theorems 5.29 and 5.30 to Appendix B.2. The theorem statements and analysis are very similar to the corresponding ones from Section 5.2.

**Theorem 5.26** (Correctness)**.** *Suppose $\Pi_{\mathsf{EPRF}}$ satisfies perfect correctness for most keys (Definition 4.14), key-injectivity (Definition 4.11), $(\varepsilon_1, \varepsilon_2)$-robust extractability for $\varepsilon_1 = 1/\mathsf{poly}(\lambda)$ (Definition 4.9), and puncturing security (Definition 4.16), and that $1/|\mathcal{X}| = \mathsf{negl}(\lambda)$. Then, the watermarking scheme $\Pi_{\mathsf{WM}}$ from Construction 5.25 is correct*

*Proof.* Follows by the same argument as that in the proof of Theorem 5.18. □

**Theorem 5.27** (Pseudorandomness)**.** *Suppose $\Pi_{\mathsf{EPRF}}$ is pseudorandom (Definition 4.6). Then, the watermarking scheme $\Pi_{\mathsf{WM}}$ from Construction 5.25 is pseudorandom (Definition 5.5).*

*Proof.* Follows by the same argument as that in the proof of Theorem 5.19. □

**Theorem 5.28** ($T$-Restricted Pseudorandomness Against the Authority)**.** *Suppose* $\Pi_{\mathsf{EPRF}}$ *satisfies* $T$*-restricted pseudorandomness given the trapdoor (Definition 4.7). Then, the watermarking scheme* $\Pi_{\mathsf{WM}}$ *from Construction 5.25 satisfies* $T$*-restricted pseudorandomness against the watermarking authority (Definition 5.6).*

*Proof.* Follows by the same argument as that in the proof of Theorem 5.20. □

**Theorem 5.29** (Unremovability)**.** *Suppose that* $1/|\mathcal{Y}| = \mathsf{negl}(\lambda)$*, that* $\mathsf{PRF}$ *is a secure PRF, and that* $\Pi_{\mathsf{EPRF}}$ *satisfies key-injectivity (Definition 4.11),* $(\varepsilon_1, \varepsilon_2)$*-robust extractability (Definition 4.9), the additional properties in Remark 4.10, and selective puncturing security (Definition 4.16). Then, for all* $\varepsilon \leq \varepsilon_1$*, the message-embedding watermarkable PRF family* $\Pi_{\mathsf{WM}}$ *from Construction 5.25 satisfies* $\varepsilon$*-unremovability (Definition 5.11).*

**Theorem 5.30** (Unforgeability)**.** *Suppose that* $\mathsf{PRF}$ *is a secure PRF and that* $\Pi_{\mathsf{EPRF}}$ *is almost functionality-preserving for adversarially-chosen keys (Definition 4.15) and satisfies* $(\varepsilon_1, \varepsilon_2)$*-robust extractability (Definition 4.9) as well as the additional properties in Remark 4.10. In additional, suppose that* $1/|\mathcal{X}| = \mathsf{negl}(\lambda)$*. Then, for all* $\delta \geq \varepsilon_2$*, the message-embedding watermarkable PRF family* $\Pi_{\mathsf{WM}}$ *from Construction 5.17 satisfies* $\delta$*-unforgeability (Definition 5.12).*

**Corollary 5.31** (Extended Pseudorandomness)**.** *Suppose the conditions in Theorems 5.27 and 5.30 hold. Then, the watermarking scheme* $\Pi_{\mathsf{WM}}$ *from Construction 5.25 satisfies extended pseudorandomness.*

*Proof.* Follows from the argument outlined in Remark 5.15. □

## 5.4 Public Marking in the Random Oracle Model

In this section, we show how a simple variant of Construction 5.25 gives a message-embedding watermarking scheme that supports *public* marking in the random oracle model. At a high level, our construction replaces the pseudorandom function $\mathsf{PRF}(k_{\mathsf{PRF}}, \cdot)$ used to derive the points to puncture with a random oracle. This yields a scheme where anybody can run the marking algorithm. The security analysis follows very similarly to the analysis of the message-embedding watermarking scheme from Section 5.3.

**Construction 5.32** (Message-Embedding Watermarkable PRFs with Public Marking)**.** Let $\lambda$ be a security parameter. Let $\mathcal{M} = \{0,1\}^t$ be the message space for the watermarking scheme. Our *mark-embedding* watermarkable PRF with public marking relies on the following primitives:

- Let $\Pi_{\mathsf{EPRF}} = (\mathsf{EX.PrmsGen}, \mathsf{EX.SampleKey}, \mathsf{EX.Eval}, \mathsf{EX.Extract}, \mathsf{EX.Puncture}, \mathsf{EX.PunctureEval})$ be a puncturable extractable PRF with key-space $\mathcal{K}_{\mathsf{EPRF}}$, domain $\mathcal{X}$, and range $\mathcal{Y}$.
- Let $H$ be a hash function $H \colon (\mathcal{K}_{\mathsf{EPRF}} \times [t] \times \{0,1\}) \to \mathcal{X}^\lambda$, which we model as a random oracle in the security analysis.

We construct a publicly-markable watermarkable PRF $\Pi_{\mathsf{WM}} = (\mathsf{Setup}, \mathsf{Mark}, \mathsf{Extract})$ as follows:

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter $\lambda$, the setup algorithm samples parameters for the extractable PRF $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{EX.PrmsGen}(1^\lambda)$. It outputs $\mathsf{pp}$ as the public parameters and sets $\mathsf{wsk} = (\mathsf{pp}, \mathsf{td})$.

- $\mathsf{Mark}(\mathsf{pp}, k, m)$: On input the public parameters $\mathsf{pp}$, a PRF key $k \in \mathcal{K}_{\mathsf{EPRF}}$, and a message $m \in \mathcal{M}$, the marking algorithm derives a collection of points $(x^*_{i,1}, \ldots, x^*_{i,\lambda}) \leftarrow H(k, i, m_i)$ for each $i \in [t]$. It then constructs a punctured key $k' \leftarrow \mathsf{EX.Puncture}(\mathsf{pp}, k, \{x^*_{i,j}\}_{i \in [t], j \in [\lambda]})$. Finally, it outputs the circuit $C \colon \mathcal{X} \to \mathcal{Y}$ that implements the punctured evaluation algorithm $\mathsf{EX.PunctureEval}(\mathsf{pp}, k', \cdot)$.

- $\mathsf{Extract}(\mathsf{wsk}, C)$: On input the watermarking secret key $\mathsf{wsk} = (\mathsf{pp}, \mathsf{td})$, and a circuit $C \colon \mathcal{X} \to \mathcal{Y}$, the extraction algorithm first extracts a key $k \leftarrow \mathsf{EX.Extract}(\mathsf{pp}, \mathsf{td}, C)$. If $k = \perp$, output $\perp$. Otherwise, for each $i \in [t]$ and $b \in \{0,1\}$, the extraction algorithm computes $(x^*_{i,b,1}, \ldots, x^*_{i,b,\lambda}) \leftarrow H(k, i, b)$. Let $N_{i,b}$ denote the number of indices $j \in [\lambda]$ where $C(x^*_{i,b,j}) \neq \mathsf{EX.Eval}(\mathsf{pp}, k, x^*_{i,b,j})$. If there exists an index $i \in [t]$ where $N_{i,0}, N_{i,1} < 2\lambda/3$ or $N_{i,0}, N_{i,1} > 2\lambda/3$, then output $\perp$. Otherwise, for each $i \in [t]$, let $b_i$ be the unique bit where $N_{i,b_i} > 2\lambda/3$. Output the message $m = b_1 \cdots b_t$.

The underlying PRF family $\mathsf{F} \colon \mathcal{K}_{\mathsf{EPRF}} \times \mathcal{X} \to \mathcal{Y}$ (induced by the public parameters $\mathsf{pp}$ for the watermarking scheme) is defined as $\mathsf{F}(k, x) := \mathsf{Eval}(\mathsf{pp}, k, x)$ and $\mathsf{F.KeyGen}$ simply returns $\mathsf{EX.SampleKey}(\mathsf{pp})$. Note that the description of the PRF family $\mathsf{F}$ includes the public parameters $\mathsf{pp}$, but not the other components in the watermarking secret key $\mathsf{wsk}$.

**Correctness and security analysis.** We now state our main security theorems, but defer the formal analysis of Theorems 5.36, 5.22, and 5.38 to Appendix B.3. The theorem statements and analysis are very similar to the corresponding ones from Sections 5.2 and 5.3.

**Theorem 5.33** (Correctness)**.** *Suppose $\Pi_{\mathsf{EPRF}}$ satisfies perfect correctness for most keys (Definition 4.14), key-injectivity (Definition 4.11), $(\varepsilon_1, \varepsilon_2)$-robust extractability for $\varepsilon_1 = 1/\mathsf{poly}(\lambda)$ (Definition 4.9), and puncturing security (Definition 4.16), and that $\lambda / |\mathcal{X}| = \mathsf{negl}(\lambda)$. Then, the watermarking scheme $\Pi_{\mathsf{WM}}$ from Construction 5.32 is correct.*

*Proof.* Follows by the same argument as that in the proof of Theorem 5.18. □

**Theorem 5.34** (Pseudorandomness)**.** *Suppose $\Pi_{\mathsf{EPRF}}$ is pseudorandom (Definition 4.6). Then, the watermarking scheme $\Pi_{\mathsf{WM}}$ from Construction 5.32 is pseudorandom (Definition 5.5).*

*Proof.* Follows by the same argument as that in the proof of Theorem 5.19. □

**Theorem 5.35** ($T$-Restricted Pseudorandomness Against the Authority)**.** *Suppose $\Pi_{\mathsf{EPRF}}$ satisfies $T$-restricted pseudorandomness given the trapdoor (Definition 4.7). Then, the watermarking scheme $\Pi_{\mathsf{WM}}$ from Construction 5.32 satisfies $T$-restricted pseudorandomness against the watermarking authority (Definition 5.6).*

*Proof.* The PRF family associated with $\Pi_{\mathsf{WM}}$ is the PRF family associated with the underlying extractable PRF $\Pi_{\mathsf{EPRF}}$, and the watermarking secret key is precisely the public parameters and trapdoor for $\Pi_{\mathsf{EPRF}}$. Thus $T$-restricted pseudorandomness of $\Pi_{\mathsf{WM}}$ against the watermarking authority reduces directly to $T$-restricted pseudorandomness of $\Pi_{\mathsf{EPRF}}$ given the trapdoor. □

**Theorem 5.36** (Unremovability)**.** *Suppose that $1/|\mathcal{Y}| = \mathsf{negl}(\lambda)$ and that $\Pi_{\mathsf{EPRF}}$ satisfies key-injectivity (Definition 4.11), $(\varepsilon_1, \varepsilon_2)$-robust extractability (Definition 4.9), the additional properties in Remark 4.10, and selective puncturing security (Definition 4.16). Then, for all $\varepsilon \leq \varepsilon_1$, the mark-embedding watermarkable PRF family $\Pi_{\mathsf{WM}}$ from Construction 5.17 satisfies $\varepsilon$-unremovability (Definition 5.11) in the random oracle model.*

**Theorem 5.37** (Weak Unforgeability)**.** *Suppose that* $\Pi_{\mathsf{PRF}}$ *is a secure PRF and that* $\Pi_{\mathsf{EPRF}}$ *satisfies* $(\varepsilon_1, \varepsilon_2)$*-robust extractability (Definition 4.9) and the additional properties in Remark 4.10. Then, for all* $\delta \geq \varepsilon_2$*, the message-embedding watermarkable PRF family* $\Pi_{\mathsf{WM}}$ *from Construction 5.32 satisfies weak* $\delta$*-unforgeability (Definition 5.13) in the random oracle model.*

**Theorem 5.38** (Extended Pseudorandomness)**.** *Suppose the conditions in Theorems 5.34 and 5.37 hold. Then, the watermarking scheme* $\Pi_{\mathsf{WM}}$ *from Construction 5.25 satisfies extended pseudorandomness in the random oracle model.*

## 5.5 Watermarking Instantiations from Lattices

In this section, we instantiate our watermarking constructions from Sections 5.3 and 5.4 using our puncturable extractable PRF from Section 4. This yields the following (new) constructions of PRFs from lattices. To simplify the corollaries, we state them based on the approximation factor for worst-case lattice problems using the parameter settings suggested in Section 4.3 (and Remark 4.27).

**Corollary 5.39** (Message-Embedding Watermarking from Lattices)**.** *Fix a security parameter* $\lambda$*. Take any* $0 < \varepsilon < \delta < 1/2$ *where* $\delta > \varepsilon + 1/\mathsf{poly}(\lambda)$*. Then, assuming it is difficult to approximate to worst-case lattice problems (e.g.,* $\mathsf{GapSVP}$ *or* $\mathsf{SIVP}$*) with a* quasi-polynomial *approximation factor, there exists a secret-key message-embedding watermarking scheme that satisfies* $\varepsilon$*-unremovability,* $\delta$*-unforgeability (the variant in Remark 5.16), and* $T$*-restricted pseudorandomness against the watermarking authority for* $T = \lambda$*. Assuming hardness of approximating worst-case lattice problems with a* sub-exponential *approximation factor, the resulting watermarking scheme satisfies the standard notion of* $\delta$*-unforgeability (Definition 5.12).*

**Corollary 5.40** (Message-Embedding Watermarking with Public Marking from Lattices)**.** *Fix a security parameter* $\lambda$*. Take any* $0 < \varepsilon < \delta < 1/2$ *where* $\delta > \varepsilon + 1/\mathsf{poly}(\lambda)$*. Then, assuming it is difficult to approximate to worst-case lattice problems (e.g.,* $\mathsf{GapSVP}$ *or* $\mathsf{SIVP}$*) with a* quasi-polynomial *approximation factor, there exists a secret-key message-embedding watermarking scheme that satisfies* $\varepsilon$*-unremovability, weak* $\delta$*-unforgeability (the variant in Remark 5.16),* $T$*-restricted pseudorandomness against the watermarking authority for* $T = \lambda$*, and* public marking *in the random oracle model. Assuming hardness of approximating worst-case lattice problems with a* sub-exponential *approximation factor, the resulting watermarking scheme satisfies the standard notion of weak* $\delta$*-unforgeability (Definition 5.13) in the random oracle model.*

**Additional properties.** We conclude by describing several additional features supported by our watermarking scheme.

- **Transferability:** An interesting property of our new watermarking schemes is that the watermarking authority can recover the original *unmarked* key given a watermarked function (via the extraction algorithm of the underlying extractable PRF). Thus, it is possible to *transfer* a watermarked key with one user's identity to a different user's identity (by first extracting the unmarked key and then re-watermarking it with the identity of the new user). It is not clear how to implement such a functionality in previous watermarking schemes [CHN$^+$16, BLW17, KW17, QWZ18].
- **Robust embedding:** Our extractable PRF has the property that even if a circuit $C$ differs from $\mathsf{F}(k, \cdot)$ everywhere, as long as their outputs are "close" (in infinity norm), then key extraction still succeeds. This means that we can consider an even stronger notion of watermarking where we allow the adversary to corrupt the behavior of the PRF *everywhere*, but as long as most of the values are still "close" to their original values, the watermark is still preserved.

## Acknowledgments

We thank Daniel Wichs for helpful discussions and for pointing out the stronger security notion against a corrupt watermarking authority (Remark 5.9). We thank Willy Quach and Giorgos Zirdelis for many insightful discussions. This work was funded by NSF, DARPA, a grant from ONR, and the Simons Foundation. Opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

## References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, 2010.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, 2009.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *ASIACRYPT*, 2011.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC*, 1996.

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In *ICALP*, 1999.

[AP09] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In *STACS*, 2009.

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, 2004.

[BB04b] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, 2004.

[BCI$^+$13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, 2013.

[BCTW16] Zvika Brakerski, David Cash, Rotem Tsabary, and Hoeteck Wee. Targeted homomorphic attribute-based encryption. In *TCC*, 2016.

[BFP$^+$15] Abhishek Banerjee, Georg Fuchsbauer, Chris Peikert, Krzysztof Pietrzak, and Sophie Stevens. Key-homomorphic constrained pseudorandom functions. In *TCC*, 2015.

[BGG$^+$14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT*, 2014.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001.

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

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *PKC*, 2014.

[BISW17] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-based snargs and their application to more efficient obfuscation. In *EUROCRYPT*, 2017.

[BKM17] Dan Boneh, Sam Kim, and Hart William Montgomery. Private puncturable prfs from standard lattice assumptions. In *EUROCRYPT*, 2017.

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

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *CRYPTO*, 2013.

[BLP$^{+}$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, 2013.

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

[BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In *CRYPTO*, 2014.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, 2012.

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

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based fhe as secure as pke. In *ITCS*, 2014.

[BV15] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic prfs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In *TCC*, 2015.

[BV16] Zvika Brakerski and Vinod Vaikuntanathan. Circuit-ABE from LWE: unbounded attributes and semi-adaptive security. In *CRYPTO*, 2016.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, 2007.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT*, 2013.

[CC17] Ran Canetti and Yilei Chen. Constraint-hiding constrained prfs for nc$^{1}$ from LWE. In *EUROCRYPT*, 2017.

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

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

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, 2009.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *TCC*, 2015.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *FOCS*, 1984.

[GINX16] Nicolas Gama, Malika Izabachène, Phong Q Nguyen, and Xiang Xie. Structural lattice reduction: generalized worst-case to average-case reductions and homomorphic cryptosystems. In *EUROCRYPT*, 2016.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *FOCS*, pages 612–621, 2017.

[GMW15] Romain Gay, Pierrick Méaux, and Hoeteck Wee. Predicate encryption for multi-dimensional range queries from lattices. In *PKC*, 2015.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.

[GV15] Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient ABE for branching programs. In *ASIACRYPT*, 2015.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, 2013.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In *CRYPTO*, 2015.

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

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *ACM CCS*, 2013.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, 2008.

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

[LW15] Vadim Lyubashevsky and Daniel Wichs. Simple lattice trapdoor sampling from a broad class of distributions. In *PKC*, 2015.

[Mic04] Daniele Micciancio. Almost perfect lattices, the covering radius problem, and applications to ajtai's connection factor. *SIAM J. Comput.*, 34(1), 2004.

[MM11] Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In *CRYPTO*, 2011.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, 2012.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In *CRYPTO*, 2013.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1), 2007.

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

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

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, 2009.

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

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

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, 2005.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, 2005.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In *FOCS*, pages 600–611, 2017.

[YAL$^+$17] Rupeng Yang, Man Ho Au, Junzuo Lai, Qiuliang Xu, and Zuoxia Yu. Collusion resistant watermarking schemes for cryptographic functionalities. *IACR Cryptology ePrint Archive*, 2017.

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

[YF11] Maki Yoshida and Toru Fujiwara. Toward digital watermarking for cryptographic data. *IEICE Transactions*, 94-A(1), 2011.

# A Analysis of Puncturable Extractable PRF (Construction 4.19)

In this section, we provide the formal analysis of our puncturable extractable PRF (Construction 4.19).

## A.1 Preliminaries

In this section, we define a few concepts that we will use in the security analysis.

**Selective pseudorandomness.** As noted in Remark 4.4, it will often be easier to work with a *selective* notion of pseudorandomness in our proofs. We first give the following alternative definition of pseudorandomness, which is equivalent to the simpler version given in Definition 4.1 (Remark A.2)

**Definition A.1** (Selective and Adaptive Pseudorandomness)**.** Let $\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a function with key-space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$. We additionally allow $\mathsf{F}$ to depend on some public parameters $\mathsf{pp}$ (for instance, $\mathsf{F}$ may be drawn from a family of functions and $\mathsf{pp}$ is used to identify a particular instance). Optionally, $\mathsf{F}$ can include a key-generation algorithm $\mathsf{F.KeyGen}$ that on input the security parameter $\lambda$, outputs a key $k \in \mathcal{K}$. For an adversary $\mathcal{A}$ and a bit $b \in \{0,1\}$, we define the security experiment $\mathsf{ExptPRF}_{\mathsf{F},\mathcal{A}}(\lambda, b)$ as follows:

1. At the beginning of the game, the challenger gives the adversary the description of $\mathsf{F}$ and the public parameters $\mathsf{pp}$ (if there are any). The challenger then samples a key $k \leftarrow \mathsf{F.KeyGen}(1^\lambda)$ and a function $f \xleftarrow{\text{R}} \mathsf{Funs}[\mathcal{X}, \mathcal{Y}]$.
2. The adversary is then allowed to make evaluation queries and a *single* challenge query:
   - **Evaluation queries.** On input an input $x \in \mathcal{X}$, the challenger replies with $\mathsf{F}(k, x)$
   - **Challenge query.** On input an input $x^* \in \mathcal{X}$, the challenger replies with $\mathsf{F}(k, x^*)$ if $b = 0$ and $f(x^*)$ if $b = 1$.
3. At the end of the experiment, algorithm $\mathcal{A}$ outputs a bit, which is the also the output of the experiment.

We say that an adversary $\mathcal{A}$ is *admissible* if $\mathcal{A}$ never makes an evaluation query on $x^* \in \mathcal{X}$, where $x^*$ is its single challenge query. Finally, we say that a function $\mathsf{F}$ satisfies *adaptive pseudorandomness* if for all efficient and admissible adversaries,

$$\left|\Pr\left[\mathsf{ExptPRF}_{\mathsf{F},\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{ExptPRF}_{\mathsf{F},\mathcal{A}}(\lambda, 1) = 1\right]\right| = \mathsf{negl}(\lambda). \tag{A.1}$$

We say that $\mathsf{F}$ satisfies *selective pseudorandomness* if Eq. (A.1) holds only for efficient and admissible adversaries that are required to commit to their challenge point $x^* \in \mathcal{X}$ at the beginning of the security experiment (before seeing the public parameters and before making any evaluation queries).

**Remark A.2** (Relation Between Pseudorandomness Notions)**.** By a simply hybrid argument, any PRF that is adaptively secure according to Definition A.1 is also secure according to Definition 4.1 (with a $1/Q$ loss in the security reduction where $Q$ is the number of queries the adversary makes in the standard PRF security game). As noted also in Remark 4.4, selective pseudorandomness implies adaptive pseudorandomness with a super-polynomial loss in the security reduction (c.f., [BB04a]).

**Remark A.3** (Selective $T$-Restricted Pseudorandomness)**.** We can define a notion of selective $T$-restricted pseudorandomness in the style of Definition A.1. In the selective setting, the challenger chooses the public parameters and the restricted set $S \subseteq \mathcal{X}$ at the beginning of the experiment and gives $T$ to the adversary. Once again, the adversary must commit to its challenge $x^* \in \mathcal{X}$ before seeing the public parameters $\mathsf{pp}$ and making evaluation queries. Note though that the adversary's challenge point is allowed to depend on the restricted set $S$.

**Hybrid LWE.** In this work, we will also use the following simple variant of LWE we call "Hybrid LWE" to simplify our security analysis. The hybrid LWE assumption essentially says that LWE remains hard even if the LWE distinguisher always gets to see valid LWE samples. We show that hardness of the Hybrid LWE assumption is implied by hardness of the standard LWE assumption.

**Definition A.4** (Hybrid LWE)**.** Fix a security parameter $\lambda$, integers $n = n(\lambda)$, $m_1 = m_1(\lambda)$, $m_2 = m_2(\lambda)$, $q = q(\lambda)$, and an error distribution $\chi = \chi(\lambda)$ over the integers. Then the hybrid LWE assumption $\mathsf{HybLWE}_{n,m_1,m_2,q,\chi}$ assumption states that for $\mathbf{A}_1 \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m_1}$, $\mathbf{A}_2 \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m_2}$, $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, $\mathbf{e}_1 \xleftarrow{\text{R}} \chi^{m_1}$, $\mathbf{e}_2 \xleftarrow{\text{R}} \chi^{m_2}$, and $\mathbf{u} \xleftarrow{\text{R}} \mathbb{Z}_q^{m_2}$, the following two distributions are computationally indistinguishable:

$$(\mathbf{A}_1, \mathbf{A}_2, \mathbf{s}\mathbf{A}_1 + \mathbf{e}_1, \mathbf{s}\mathbf{A}_2 + \mathbf{e}_2) \quad \text{and} \quad (\mathbf{A}_1, \mathbf{A}_2, \mathbf{s}\mathbf{A}_1 + \mathbf{e}_1, \mathbf{u}).$$

**Lemma A.5** (LWE Implies Hybrid LWE)**.** *Fix a security parameter $\lambda$ and integers $n = n(\lambda)$, $m_1 = m_1(\lambda)$, $m_2 = m_2(\lambda)$, $q = q(\lambda)$ and an error distribution $\chi = \chi(\lambda)$ over the integers. Let $m = m_1 + m_2$. Under the $\mathsf{LWE}_{n,m,q,\chi}$ assumption, the $\mathsf{HybLWE}_{n,m_1,m_2,q,\chi}$ assumption also holds.*

*Proof.* The proof follows by a simple hybrid argument. In each hybrid experiment below, the experiment first samples $\mathbf{A}_1 \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m_1}$, $\mathbf{A}_2 \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m_2}$, $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, $\mathbf{e}_1 \xleftarrow{\text{R}} \chi^{m_1}$, $\mathbf{e}_2 \xleftarrow{\text{R}} \chi^{m_2}$, $\mathbf{u}_1 \xleftarrow{\text{R}} \mathbb{Z}_q^{m_1}$, and $\mathbf{u}_2 \xleftarrow{\text{R}} \mathbb{Z}_q^{m_2}$. We now specify the output of each hybrid experiment:

- $\textsc{hyb}_0$: Output $(\mathbf{A}_1, \mathbf{A}_2, \mathbf{s}\mathbf{A}_1 + \mathbf{e}_1, \mathbf{s}\mathbf{A}_2 + \mathbf{e}_2)$.
- $\textsc{hyb}_1$: Output $(\mathbf{A}_1, \mathbf{A}_2, \mathbf{u}_1, \mathbf{u}_2)$.
- $\textsc{hyb}_2$: Output $(\mathbf{A}_1, \mathbf{A}_2, \mathbf{s}\mathbf{A}_1 + \mathbf{e}_1, \mathbf{u}_2)$.

Since $m = m_1 + m_2$, the output of hybrids $\textsc{hyb}_0$ and $\textsc{hyb}_1$ is computationally indistinguishable under the $\mathsf{LWE}_{n,m,q,\chi}$ assumption. The output of hybrids $\textsc{hyb}_1$ and $\textsc{hyb}_2$ is computationally indistinguishable under the $\mathsf{LWE}_{n,m_1,q,\chi}$ assumption (implied by the $\mathsf{LWE}_{n,m,q,\chi}$ assumption since $m_1 \le m$). The distributions in $\textsc{hyb}_0$ and $\textsc{hyb}_2$ correspond to the two distributions in the hybrid LWE assumption, and the claim follows. □

**Uniqueness of LWE secrets.** For a suitable choice of parameters, an LWE sample induces a unique secret vector with overwhelming probability. In this work, we rely on a stronger property, which says that with overwhelming probability, there *does not exist* any vectors that can be explained by two LWE secrets. Although the following lemma is folklore, we provide the proof for completeness.

**Lemma A.6** (Uniqueness of LWE Secrets)**.** *Fix a security parameter $\lambda$, lattice parameters $n$, $m$, $q$, and a rounding modulus $p$ where $m \ge 2n \log q$ and $\lceil q/p \rceil \le q/4$. Then, for any two distinct vectors $\mathbf{s}_0, \mathbf{s}_1 \in \mathbb{Z}_q^n$, $\mathbf{s}_0 \neq \mathbf{s}_1$, we have*

$$\Pr_{\mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}} \Big[ \, \exists \; \mathbf{s}_0, \mathbf{s}_1 \in \mathbb{Z}_q^n : \lfloor \mathbf{s}_0 \cdot \mathbf{A} \rceil_p = \lfloor \mathbf{s}_1 \cdot \mathbf{A} \rceil_p \Big] = q^{-n}.$$

*Proof.* Fix any two vectors $\mathbf{s}_0, \mathbf{s}_1 \in \mathbb{Z}_q^n$ with $\mathbf{s}_0 \neq \mathbf{s}_1$ and denote $\tilde{\mathbf{s}} = \mathbf{s}_0 - \mathbf{s}_1 \in \mathbb{Z}_q^n \backslash \{\mathbf{0}\}$. By the (almost-)linearity of the rounding function, we have

$$\begin{aligned}
\Pr_{\mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}} \Big[ \lfloor \mathbf{s}_0 \cdot \mathbf{A} \rceil_p = \lfloor \mathbf{s}_1 \cdot \mathbf{A} \rceil_p \Big] &\le \Pr_{\mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}} \Big[ \lfloor (\mathbf{s}_0 - \mathbf{s}_1) \rceil_p \cdot \mathbf{A} \in \{0, 1\} \Big] \\
&\le \Pr_{\mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}} \Big[ \tilde{\mathbf{s}} \cdot \mathbf{A} \in \big[ 2 \lceil q/p \rceil \big]^m \Big] \\
&= \big( 2 \lceil q/p \rceil \big)^m \cdot q^{-m} \\
&\le 2^{-m}.
\end{aligned}$$

Now, taking a union bound overall all the possible vectors $\tilde{\mathbf{s}} \in \mathbb{Z}_q^n$, the lemma follows. $\square$

**Borderline points.** Before proceeding with the full security analysis, we first prove a statistical fact about the behavior of our PRF. Specifically, we show that with for any fixed key $\mathbf{s} \in [-B, B]^n$, with overwhelming probability over the choice of public parameters, the value $\mathsf{Eval}(\mathsf{pp}, \mathbf{s}, x)$ on all $x \in \{0,1\}^\rho$ will not lie in a "borderline set" (that is, the unrounded PRF evaluation from Eq. (4.4) will be far from all rounding boundaries). We define the $\mathsf{Borderline}$ property more precisely below and then show the lemma.

**Definition A.7** (Borderline Evaluations)**.** Let $m, p, q$ be positive integers with $q > p$. For a bound $E$, we define the set $\mathsf{Borderline}_E$ to be the set of vectors $\mathbf{y} \in \mathbb{Z}_q^m$ where there exists a unit vector $\mathbf{u}_\beta \in \mathbb{Z}_q^m$ for some $\beta \in [m]$ such that $\langle \mathbf{y}, \mathbf{u} \rangle_\beta$ is within $E$ of the "rounding boundary." More precisely,

$$\mathsf{Borderline}_E := \left\{ \mathbf{y} \in \mathbb{Z}_q^m \mid \exists \beta \in [m] : \langle \mathbf{y}, \mathbf{u}_\beta \rangle \in [-E, E] + \frac{q}{p}\left(\mathbb{Z} + \frac{1}{2}\right) \right\}.$$

Next, we define the alternative evaluation algorithm that outputs the *unrounded* PRF evaluation:

- $\mathsf{UREval}(\mathsf{pp}, k, x)$: On input the public parameters $\mathsf{pp}$ (as specified in Eq. (4.2)), a PRF key $k = \mathbf{s}$, and an input $x \in \{0,1\}^\rho \setminus \{\mathbf{0}\}$, the alternative evaluation algorithm outputs the unrounded output $\widetilde{\mathbf{y}}_x$ as defined by Eq. (4.4).

**Lemma A.8** (No Borderline Evaluations for Most Keys)**.** *Suppose that $q$ is a prime with $q = \Omega(np\sqrt{\log q})$. Then, for all $\lambda \in \mathbb{N}$ and any fixed $\mathbf{s} \in [-B, B]^n$, if we sample $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$, then*

$$\Pr\left[\exists x \in \{0,1\}^\rho \setminus \{\mathbf{0}\} : \widetilde{\mathbf{y}}_x \leftarrow \mathsf{UREval}(\mathsf{pp}, \mathbf{s}, x) \wedge \widetilde{\mathbf{y}}_x \in \mathsf{Borderline}_E\right] \leq 2^{\rho+2}\eta m E \cdot \frac{p}{q},$$

*where the probability is taken over the random coins used to sample* $\mathsf{pp}$.

*Proof.* We first bound the probability that $\widetilde{\mathbf{y}}_x \in \mathsf{Borderline}_E$ for some *fixed* input $x \in \{0,1\}^\rho \setminus \{\mathbf{0}\}$. Let

$$\mathsf{pp} = \left(\mathbf{W}^{(\ell)}, \left(\mathbf{A}_j^{(\ell)}\right)_{j\in[\rho]}, \left(\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)}\right)_{\alpha\in[n],\beta\in[m]}, \left(\mathbf{B}_{i,j}^{(\ell)}\right)_{i\in[t],j\in[\rho]}, \left(\mathbf{C}_j^{(\ell)}\right)_{j\in[\rho]}, \mathbf{V}^{(\ell)}\right)_{\ell\in[\eta]}$$

be the public parameters output by $\mathsf{PrmsGen}(1^\lambda)$, and take any $\mathbf{s} \in [-B, B]^n$. From Eq. (4.1),

$$\mathbf{W}^{(\ell)} = \mathbf{A}_{h^{(\ell)}}^{(\ell)} + \mathbf{B}_{h^{(\ell)}}^{(\ell)} \mathbf{G}^{-1}\left(\tilde{\mathbf{B}}_{h^{(\ell)}}^{(\ell)}\right) \mathbf{G}^{-1}\left(\mathbf{V}^{(\ell)}\right) + \mathbf{D}^{(\ell)} \in \mathbb{Z}_q^{n\times m}.$$

Next, by definition of the evaluation algorithm $\mathsf{UREval}(\mathsf{pp}, \mathbf{s}, x)$,

$$\widetilde{\mathbf{y}}_x = \mathbf{s}\left(\mathbf{W}^{(1)} - \mathbf{Z}_x^{(1)} \mid \cdots \mid \mathbf{W}^{(\eta)} - \mathbf{Z}_x^{(\eta)}\right) \in \mathbb{Z}_q^{\eta m}. \tag{A.2}$$

Observe that if $\mathbf{s} = \mathbf{0}$, then $\widetilde{\mathbf{y}}_x = \mathbf{0}$, in which case $\widetilde{\mathbf{y}}_x \notin \mathsf{Borderline}_E$. To complete, the analysis, it suffices to consider only the setting where $\mathbf{s} \neq \mathbf{0}$. To do so, we consider the distribution of $\widetilde{\mathbf{y}}_x$ in the following distributions:

- $\textsc{hyb}_0$: This is the real distribution where $\widetilde{\mathbf{y}}_x \leftarrow \mathsf{UREval}(\mathsf{pp}, \mathbf{s}, x)$ described above.

- $\textsc{hyb}_1$: Instead of computing $\mathbf{W}^{(\ell)}$ according to Eq. (4.1), sample $\mathbf{W}^{(\ell)} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times m}$ for all $\ell \in [\eta]$, and then compute $\widetilde{\mathbf{y}}_x$ using Eq. (A.2) as usual.

- $\textsc{hyb}_2$: Instead of computing $\widetilde{\mathbf{y}}_x$ using Eq. (A.2), sample $\widetilde{\mathbf{y}}_x \xleftarrow{\text{R}} \mathbb{Z}_q^{\eta m}$.

We now bound the statistical distance between each consecutive pair of hybrids as well as the probability of $\widetilde{\mathbf{y}}_x \in \mathsf{Borderline}_E$ in $\textsc{hyb}_2$.

- The only difference between $\textsc{hyb}_0$ and $\textsc{hyb}_1$ is in how the matrices $\mathbf{W}^{(\ell)}$ are sampled. Consider the distribution in $\textsc{hyb}_0$. Since the $\mathsf{PrmsGen}$ algorithm samples $\mathbf{D}^{(\ell)}$ using the trapdoor sampling algorithm $\mathsf{TrapGen}$, the matrices $\mathbf{D}^{(\ell)}$ are distributed statistically close to uniform (and independently of all of the other components in $\mathsf{pp}$). This means that the distribution of $\mathbf{W}^{(\ell)}$ for each $\ell \in [\eta]$ is also statistically close to uniform (and independent of the other components in $\mathsf{pp}$). More precisely, by Theorem 3.5, $\Delta(\mathbf{W}^{(\ell)}, \mathsf{Uniform}(\mathbb{Z}_q^{n \times m})) \le q^{-n}$. Correspondingly,
$$\Delta(\textsc{hyb}_0, \textsc{hyb}_1) = \Delta([\mathbf{W}^{(1)} \mid \cdots \mid \mathbf{W}^{(\eta)}], \mathsf{Uniform}(\mathbb{Z}_q^{n \times \eta m})) \le \eta q^{-n}.$$

- In $\textsc{hyb}_1$, each of the $\mathbf{W}^{(\ell)}$ is uniform and independent of the other components in $\mathsf{pp}$. Thus, if $\mathbf{s} \ne \mathbf{0}$ and $q$ is prime (so $\mathbb{Z}_q$ forms a field), then $\widetilde{\mathbf{y}}_x$ is uniformly distributed over $\mathbb{Z}_q^{\eta m}$. Since we are assuming that $\mathbf{s} \ne \mathbf{0}$, $\textsc{hyb}_1$ and $\textsc{hyb}_2$ are identically distributed.

- In $\textsc{hyb}_2$, $\widetilde{\mathbf{y}}_x$ is uniform over $\mathbb{Z}_q^{\eta m}$, so by a union bound (over the components of $\widetilde{\mathbf{y}}_x$),
$$\Pr[\widetilde{\mathbf{y}}_x \xleftarrow{\text{R}} \mathbb{Z}_q^{\eta m} : \widetilde{\mathbf{y}}_x \in \mathsf{Borderline}_E] \le \eta m \frac{(2E+1)p}{q}.$$

From the above analysis, we have that in $\textsc{hyb}_0$, for any fixed $x \in \{0,1\}^\rho$,
$$\Pr[\mathbf{y}_x \in \mathsf{Borderline}_E] \le \eta m \frac{(2E+1)p}{q} + \eta q^{-n} \le 4\eta m E \cdot \frac{p}{q}$$

We take a union bound over all $x \in \{0,1\}^\rho \setminus \{\mathbf{0}\}$:
$$\Pr\left[\exists x \in \{0,1\}^\rho \setminus \{\mathbf{0}\} : \widetilde{\mathbf{y}}_x \leftarrow \mathsf{UREval}(\mathsf{pp}, k, x) \wedge \widetilde{\mathbf{y}}_x \in \mathsf{Borderline}_E\right] \le 2^\rho \cdot \left(4\eta m E \cdot \frac{p}{q}\right),$$

which proves the claim. □

## A.2 Proof of Theorem 4.20 (Perfect Correctness for Most Keys)

We show that Construction 4.19 is perfectly correct for almost all keys. First, let $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$ and $k = \mathbf{s} \leftarrow \mathsf{SampleKey}(\mathsf{pp})$. From Eq. (4.2), we have
$$\mathsf{pp} = \Big(\mathbf{W}^{(\ell)}, (\mathbf{A}_j^{(\ell)})_{j \in [\rho]}, (\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)})_{\alpha \in [n], \beta \in [m]}, (\mathbf{B}_{i,j}^{(\ell)})_{i \in [t], j \in [\rho]}, (\mathbf{C}_j^{(\ell)})_{j \in [\rho]}, \mathbf{V}^{(\ell)}\Big)_{\ell \in [\eta]}.$$

Take any set $S$ where $|S| = t = \mathsf{poly}(\lambda)$, and let $k_S \leftarrow \mathsf{Puncture}(\mathsf{pp}, k, S)$. From Eq. (4.5), we have
$$k_S = \Big(S, \big(\mathbf{w}^{(\ell)}, (\mathbf{a}_j^{(\ell)})_{j \in [\rho]}, (\tilde{\mathbf{a}}_{\alpha,\beta}^{(\ell)})_{\alpha \in [n], \beta \in [m]}, (\mathbf{b}_{i,j}^{(\ell)})_{i \in [t], j \in [\rho]}\big)_{\ell \in [\eta]}\Big).$$

We show that for all $x \in \{0,1\}^\rho \setminus \{S \cup \{\mathbf{0}\}\}$, with overwhelming probability over the choice of $\mathsf{pp}$ and $\mathbf{s}$, $\mathsf{Eval}(\mathsf{pp}, k, x) = \mathsf{PunctureEval}(\mathsf{pp}, k_S, x)$. Consider the value of the evaluation and punctured evaluation at some $x \in \{0,1\}^\rho \setminus \{S \cup \{\mathbf{0}\}\}$:

- By definition, the $\mathsf{Eval}$ algorithm first computes the "unrounded" value $\widetilde{\mathbf{y}}_x$ where
$$\widetilde{\mathbf{y}}_x = \mathbf{s}\big(\mathbf{W}^{(1)} - \mathbf{Z}_x^{(1)} \mid \cdots \mid \mathbf{W}^{(\eta)} - \mathbf{Z}_x^{(\eta)}\big) \in \mathbb{Z}_q^{\eta m}.$$
  The output is $\mathbf{y}_x = \lfloor \widetilde{\mathbf{y}}_x \rceil_p$.

- By definition, the $\mathsf{PunctureEval}$ algorithm first computes the "unrounded" value $\widetilde{\mathbf{y}}'_x$ where
$$\widetilde{\mathbf{y}}'_x = (\mathbf{w}^{(1)} - \mathbf{z}_x^{(1)} \mid \cdots \mid \mathbf{w}^{(\eta)} - \mathbf{z}_x^{(\eta)}) \in \mathbb{Z}_q^{\eta m},$$
The output is $\mathbf{y}'_x = \lfloor \widetilde{\mathbf{y}}'_x \rceil_p$. Now, for all $x \in \{0,1\}^\rho \setminus (S \cup \{\mathbf{0}\})$, we have that $f_x^{\mathsf{eq}}(\mathbf{0}) = 0$ and $f_x^{\mathsf{con}}(S) = 0$, so by the specification of the $\mathsf{Puncture}$ and $\mathsf{PunctureEval}$ algorithms and Theorem 3.6, we have that for all $\ell \in [\eta]$,
$$\begin{aligned}\mathbf{z}_x^{(\ell)} &= \mathbf{s}\mathbf{A}_x^{(\ell)} + \mathbf{e}_{\mathbf{A}}^{(\ell)} + (\mathbf{s}\mathbf{B}_x^{(\ell)} + \mathbf{e}_{\mathbf{B}}^{(\ell)})\mathbf{G}^{-1}(\mathbf{C}_x^{(\ell)})\mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) \\ &= \mathbf{s}(\mathbf{A}_x^{(\ell)} + \mathbf{B}_x^{(\ell)}\mathbf{G}^{-1}(\mathbf{C}_x^{(\ell)})\mathbf{G}^{-1}(\mathbf{V}^{(\ell)})) + \mathbf{e}^{(\ell)} \\ &= \mathbf{s}\mathbf{Z}_x^{(\ell)} + \mathbf{e}^{(\ell)},\end{aligned}$$
where $\|\mathbf{e}_{\mathbf{A}}^{(\ell)}\|$, $\|\mathbf{e}_{\mathbf{B}}^{(\ell)}\|$ and $\|\mathbf{e}^{(\ell)}\| \le B \cdot m^{O(d)} = B \cdot m^{O(\log\lambda)}$, since $d = O(\log\lambda)$, and $\mathbf{Z}_x^{(\ell)}$ is defined according to Eq. (4.3). Similarly, by definition, $\mathbf{w}^{(\ell)} = \mathbf{s}\mathbf{W}^{(\ell)} + \mathbf{e}_{\mathbf{W}}^{(\ell)}$. Thus, we can write $\widetilde{\mathbf{y}}'_x = \widetilde{\mathbf{y}}_x + \mathbf{e}$ for some $\mathbf{e} \in \mathbb{Z}_q^{\eta m}$ and where $\|\mathbf{e}\| \le B \cdot m^{O(\log\lambda)}$.

Thus, $\mathbf{y}_x = \lfloor \widetilde{\mathbf{y}}_x \rceil_p = \lfloor \widetilde{\mathbf{y}}'_x \rceil_p = \mathbf{y}'_x$ unless $\widetilde{\mathbf{y}}_x \in \mathsf{Borderline}_E$ where $E = B \cdot m^{O(\log\lambda)}$. By Lemma A.8, for the provided parameters, the probability that there exists $x \in \{0,1\}^\rho \setminus \{\mathbf{0}\}$ where $\widetilde{\mathbf{y}}_x \in \mathsf{Borderline}_E$ is negligible, so we conclude that with overwhelming probability (over the randomness in $\mathsf{PrmsGen}$, $\mathsf{SampleKey}$, and $\mathsf{Puncture}$), $\mathsf{Eval}(\mathsf{pp}, k, x) = \mathsf{PunctureEval}(\mathsf{pp}, k_S, x)$ for all $x \in \{0,1\}^\rho \setminus (S \cup \{\mathbf{0}\})$.

## A.3 Proof of Theorem 4.21 (Almost-Functionality-Preserving for All Keys)

We begin by defining two hybrid experiments between an adversary $\mathcal{A}$ and a challenger:

- $\textsc{hyb}_0$: In $\textsc{hyb}_0$, the challenger begins by sampling $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$. It gives $\mathsf{pp}$ to $\mathcal{A}$. Adversary $\mathcal{A}$ replies with a key $k = \mathbf{s} \in [-B, B]^m$ and a set $S \subseteq \{0,1\}^\rho \setminus \{\mathbf{0}\}$ of size $t$. The challenger computes $k_S \leftarrow \mathsf{Puncture}(\mathsf{pp}, k, S)$ and samples $x \xleftarrow{\text{R}} \mathcal{X} \setminus S$. The output of the experiment is 1 if $\mathsf{Eval}(\mathsf{pp}, k, x) \neq \mathsf{PunctureEval}(\mathsf{pp}, k_S, x)$ and 0 otherwise.

- $\textsc{hyb}_1$: Same as $\textsc{hyb}_0$, except when sampling the public parameters $\mathsf{pp}$, instead of defining the matrices $\mathbf{W}^{(\ell)}$ according to (4.1), the challenger instead samples $\mathbf{W}^{(\ell)} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$ for all $\ell \in [\eta]$. The other components of $\mathsf{pp}$ are unchanged.

For an index $i$, we write $\textsc{hyb}_i(\mathcal{A})$ to denote the output distribution of $\textsc{hyb}_i$ with adversary $\mathcal{A}$. Showing that Construction 4.19 is almost-functionality-preserving for adversarial keys is equivalent to showing that for all efficient adversaries $\mathcal{A}$, $\Pr[\textsc{hyb}_0(\mathcal{A}) = 1] = \mathsf{negl}(\lambda)$. To complete the proof, we show that for all efficient adversaries $\mathcal{A}$, $\textsc{hyb}_0(\mathcal{A})$ and $\textsc{hyb}_1(\mathcal{A})$ are statistically indistinguishable and that under the $\mathsf{1D\text{-}SIS\text{-}R}_{m',q,p,E}$ assumption, $\Pr[\textsc{hyb}_1(\mathcal{A}) = 1] = \mathsf{negl}(\lambda)$.

**Lemma A.9.** *Suppose the lattice parameters $n, m, p, q$ satisfy the conditions in Theorem 3.5. Then, for all adversaries $\mathcal{A}$, the distributions $\textsc{hyb}_0(\mathcal{A})$ and $\textsc{hyb}_1(\mathcal{A})$ are statistically indistinguishable.*

*Proof.* The only difference between the two experiments is that in $\textsc{hyb}_1$, the matrices $\mathbf{W}^{(\ell)}$ are uniformly random. To see this, we first appeal to Theorem 3.5 to argue that each $\mathbf{D}^{(\ell)}$ is statistically close to uniform in $\textsc{hyb}_0$. Moreover, they are sampled independently of all the other components in the public parameters. Finally, since $\eta = \mathsf{poly}(\lambda)$, we conclude that the public parameters generated in $\textsc{hyb}_0$ and $\textsc{hyb}_1$ are statistically indistinguishable. □

We now show that for public parameters output by $\textsc{hyb}_1$, no efficient adversary can produce a key $\mathbf{s} \in [-B, B]^n$ that violates correctness.

**Lemma A.10.** *Suppose $\rho = \omega(\log \lambda)$. Let $m' = nm\eta$ and $E = B \cdot m^{O(\log \lambda)}$. Then, under the $\mathsf{1D\text{-}SIS\text{-}R}_{m',p,q,E}$ assumption, for all efficient adversaries $\mathcal{A}$, $\Pr[\textsc{hyb}_1(\mathcal{A}) = 1] = \mathsf{negl}(\lambda)$.*

*Proof.* Suppose there exists an adversary $\mathcal{A}$ where $\textsc{hyb}_1(\mathcal{A}) = 1$ with non-negligible probability. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks the $\mathsf{1D\text{-}SIS\text{-}R}_{m',p,q,E}$ assumption. Algorithm $\mathcal{B}$ proceeds as follows:

1. **Parsing the $\mathsf{1D\text{-}SIS\text{-}R}$ challenge.** At the beginning of the experiment, algorithm $\mathcal{B}$ receives a challenge $\mathbf{d} \in \mathbb{Z}_q^{m'}$ from the $\mathsf{1D\text{-}SIS\text{-}R}_{m',p,q,B}$ challenger where $m' = nm\eta$. It parses the components of $\mathbf{d} = (d_{\alpha,\beta}^{(\ell)})_{\alpha\in[n],\beta\in[m],\ell\in[\eta]}$. For each $\ell \in \eta$, it defines a matrix $\mathbf{D}^{(\ell)} \in \mathbb{Z}_q^{n\times m}$ where the $(\alpha,\beta)^{\text{th}}$ entry of $\mathbf{D}^{(\ell)}$ is $d_{\alpha,\beta}^{(\ell)}$.

2. **Simulating the public parameters.** Before simulating the rest of the public parameters, algorithm $\mathcal{B}$ first samples $x \xleftarrow{\text{R}} \{0,1\}^\rho \setminus \{\mathbf{0}\}$. It then simulates the rest of the public parameters $\mathsf{pp}$ as follows. First, $\mathcal{B}$ samples matrices $(\mathbf{A}_j^{(\ell)})_{j\in[\rho]}$, $(\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)})_{\alpha\in[n],\beta\in[m]}$, $(\mathbf{B}_{i,j}^{(\ell)})_{i\in[t],j\in[\rho]}$, $(\mathbf{C}_j^{(\ell)})_{j\in[\rho]}$, $\mathbf{V}^{(\ell)}$ uniformly at random from $\mathbb{Z}_q^{n\times m}$ for every $\ell \in [\eta]$. Then, for all $\ell \in [\eta]$, it computes

   - $\mathbf{A}_x^{(\ell)} \leftarrow \mathsf{EvalP}_{\mathsf{pk}}\Big(f_x^{\mathsf{eq}}, (\mathbf{A}_j^{(\ell)})_{j\in[\rho]}, (\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)})_{\alpha\in[n],\beta\in[m]}\Big)$,
   - $\mathbf{B}_x^{(\ell)} \leftarrow \mathsf{Eval}_{\mathsf{pk}}\Big(f_x^{\mathsf{con}}, (\mathbf{B}_{i,j}^{(\ell)})_{i\in[t],j\in[\rho]}\Big)$,
   - $\mathbf{C}_x^{(\ell)} \leftarrow \mathsf{Eval}_{\mathsf{pk}}\Big(f_x^{\mathsf{eq}}, (\mathbf{C}_j^{(\ell)})_{j\in[\rho]}\Big)$,

   and defines the matrix

   $$\mathbf{W}^{(\ell)} = \mathbf{A}_x^{(\ell)} + \mathbf{B}_x^{(\ell)}\mathbf{G}^{-1}(\mathbf{C}_x^{(\ell)})\mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) + \mathbf{D}^{(\ell)} \in \mathbb{Z}_q^{n\times m}.$$

   It sets

   $$\mathsf{pp} = \Big(\mathbf{W}^{(\ell)}, \big(\mathbf{A}_j^{(\ell)}\big)_{j\in[\rho]}, \big(\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)}\big)_{\alpha\in[n],\beta\in[m]}, \big(\mathbf{B}_{i,j}^{(\ell)}\big)_{i\in[t],j\in[\rho]}, \big(\mathbf{C}_j^{(\ell)}\big)_{j\in[\rho]}, \mathbf{V}^{(\ell)}\Big)_{\ell\in[\eta]},$$

   and gives $\mathsf{pp}$ to $\mathcal{A}$.

3. **Solving the $\mathsf{1D\text{-}SIS\text{-}R}$ challenge.** After receiving the public parameters $\mathsf{pp}$, $\mathcal{A}$ outputs a key $\mathbf{s} \in [-B,B]^n$ and a set $S \subseteq \{0,1\}^\rho \setminus \{\mathbf{0}\}$ of size $t$. If $x \in S$, then $\mathcal{B}$ aborts the simulation and outputs $\mathsf{Fail}$. Otherwise, algorithm $\mathcal{B}$ computes the vector $\mathbf{y}^{(\ell)} = \mathbf{s} \cdot \mathbf{D}^{(\ell)}$ for all $\ell \in [\eta]$ and checks if there exists an index $\beta^* \in [m]$, $\ell^* \in [\eta]$ for which $\mathbf{y}_{\beta^*}^{(\ell^*)} \in [-E,E] + (q/p)(\mathbb{Z} + 1/2)$. If no such index exists, then $\mathcal{B}$ aborts the experiment and outputs $\mathsf{Fail}$. Otherwise, it defines the vector $\mathbf{z} = (z_{\alpha,\beta,\ell})_{\alpha\in[n],\beta\in[m],\ell\in[\eta]} \in \mathbb{Z}_q^{nm\eta}$ as follows:

   $$z_{\alpha,\beta,\ell} = \begin{cases} s_\alpha & \beta = \beta^*, \ell = \ell^* \\ 0 & \text{otherwise}, \end{cases} \tag{A.3}$$

   where $s_\alpha$ is the $\alpha^{\text{th}}$ component of $\mathbf{s}$. Algorithm $\mathcal{B}$ submits $\mathbf{z} \in \mathbb{Z}^{m'}$ as the solution to the $\mathsf{1D\text{-}SIS\text{-}R}_{m',q,p,E}$ challenge.

To complete the proof, we show that algorithm $\mathcal{B}$ perfectly simulates $\textsc{hyb}_1$ for $\mathcal{A}$ and moreover, that whenever $\textsc{hyb}_1(\mathcal{A})$ outputs 1, then $\mathcal{B}$ successfully solves the $\mathsf{1D\text{-}SIS\text{-}R}$ challenge (up to a negligible loss in advantage).

**Correctness of the simulation.** In the simulation, the matrices $(\mathbf{A}_j^{(\ell)})_{j\in[\rho]}$, $(\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)})_{\alpha\in[n],\beta\in[m]}$, $(\mathbf{B}_{i,j}^{(\ell)})_{i\in[t],j\in[\rho]}$, $(\mathbf{C}_j^{(\ell)})_{j\in[\rho]}$, $\mathbf{V}^{(\ell)}$ are sampled exactly as defined in $\textsc{hyb}_1$. Furthermore, by definition, the $\mathsf{1D\text{-}SIS\text{-}R}_{m',p,q,B}$ challenge $\mathbf{d}\in\mathbb{Z}_q^{m'}$ is a uniform vector over $\mathbb{Z}_q^{m'}$, which means that each $\mathbf{D}^{(\ell)}$ is also uniformly random for all $\ell\in[\eta]$. In addition, since $\mathbf{D}^{(\ell)}$ is sampled independently of all of the other components in $\mathsf{pp}$, it follows that each of the $\mathbf{W}^{(\ell)}$ for $\ell\in[\eta]$ is independently uniform, exactly as required in $\textsc{hyb}_1$. Thus, $\mathcal{B}$ perfectly simulates the $\mathsf{pp}$ according to the distribution in $\textsc{hyb}_1$.

**Correctness of the solution.** By the previous argument, the public parameters perfectly hide the point $x$ from the adversary (the adversary only sees $\mathbf{W}^{(\ell)}$ and not $\mathbf{D}^{(\ell)}$, so the $\mathbf{D}^{(\ell)}$ is a one-time pad encryption of the matrices that depend on $x$). Therefore, the set $S$ output by the adversary is independent of $x$. Since $x$ is sampled uniformly at random from $\{0,1\}^\rho\setminus\{\mathbf{0}\}$, the probability that $x\in S$ is $t/(2^\rho-1)=\mathsf{negl}(\lambda)$ since $t=\mathsf{poly}(\lambda)$ and $\rho=\omega(\log\lambda)$. Thus, without loss of generality, we assume that $x\notin S$ (this only reduces $\mathcal{B}$'s advantage by a negligible factor).

Suppose that the output of $\textsc{hyb}_1(\mathcal{A})$ is 1. This means that $\mathcal{A}$ outputs a key $k=\mathbf{s}\in[-B,B]^n$ and a set $S\subseteq\{0,1\}^\rho\setminus\{\mathbf{0}\}$ such that for $k_S\leftarrow\mathsf{Puncture}(\mathsf{pp},k,S)$, we have $\mathsf{Eval}(\mathsf{pp},k,x)\neq\mathsf{PunctureEval}(\mathsf{pp},k_S,x)$. We consider each of the values $\mathsf{Eval}(\mathsf{pp},k,x)$ and $\mathsf{PunctureEval}(\mathsf{pp},k_S,x)$.

- By definition, the $\mathsf{Eval}$ algorithm first computes the "unrounded" value $\widetilde{\mathbf{y}}_x$
$$\begin{aligned}\widetilde{\mathbf{y}}_x &= \mathbf{s}\big(\mathbf{W}^{(1)}-\mathbf{Z}_x^{(1)}\mid\cdots\mid\mathbf{W}^{(\eta)}-\mathbf{Z}_x^{(\eta)}\big)\\ &= \mathbf{s}\big(\mathbf{D}^{(1)}\mid\cdots\mid\mathbf{D}^{(\eta)}\big)\in\mathbb{Z}_q^{\eta m}.\end{aligned}$$
The output is $\mathbf{y}_x=\lfloor\widetilde{\mathbf{y}}_x\rceil_p$.

- By definition, the $\mathsf{PunctureEval}$ algorithm first computes the "unrounded" value $\widetilde{\mathbf{y}}'_x$
$$\widetilde{\mathbf{y}}'_x=(\mathbf{w}^{(1)}-\mathbf{z}_x^{(1)}\mid\cdots\mid\mathbf{w}^{(\eta)}-\mathbf{z}_x^{(\eta)})\in\mathbb{Z}_q^{\eta m}.$$
The output is $\mathbf{y}'_x=\lfloor\widetilde{\mathbf{y}}'_x\rceil_p$. Now, for all $x\in\{0,1\}^\rho\setminus(S\cup\{\mathbf{0}\})$, we have that $f_x^{\mathsf{eq}}(\mathbf{0})=0$ and $f_x^{\mathsf{con}}(S)=0$, so by the specification of $\mathsf{Puncture}$ and $\mathsf{PunctureEval}$ and Theorem 3.6, we have that for all $\ell\in[\eta]$,
$$\begin{aligned}\mathbf{z}_x^{(\ell)} &= \mathbf{s}\mathbf{A}_x^{(\ell)}+\mathbf{e}_{\mathbf{A}}^{(\ell)}+(\mathbf{s}\mathbf{B}_x^{(\ell)}+\mathbf{e}_{\mathbf{B}}^{(\ell)})\mathbf{G}^{-1}(\mathbf{C}_x^{(\ell)})\mathbf{G}^{-1}(\mathbf{V}^{(\ell)})\\ &= \mathbf{s}(\mathbf{A}_x^{(\ell)}+\mathbf{B}_x^{(\ell)}\mathbf{G}^{-1}(\mathbf{C}_x^{(\ell)})\mathbf{G}^{-1}(\mathbf{V}^{(\ell)}))+\mathbf{e}^{(\ell)}\\ &= \mathbf{s}\mathbf{Z}_x^{(\ell)}+\mathbf{e}^{(\ell)},\end{aligned}$$
where $\|\mathbf{e}_{\mathbf{A}}^{(\ell)}\|$, $\|\mathbf{e}_{\mathbf{B}}^{(\ell)}\|$ and $\|\mathbf{e}^{(\ell)}\|\le B\cdot m^{O(d)}=B\cdot m^{O(\log\lambda)}$, since $d=O(\log\lambda)$, and $\mathbf{Z}_x^{(\ell)}$ is defined according to Eq. (4.3). Similarly, by definition, $\mathbf{w}^{(\ell)}=\mathbf{s}\mathbf{W}^{(\ell)}+\mathbf{e}_{\mathbf{W}}^{(\ell)}$ where $\|\mathbf{e}_{\mathbf{W}}^{(\ell)}\|\le B\cdot m^{O(\log\lambda)}$. Thus, we can write $\widetilde{\mathbf{y}}'_x=\widetilde{\mathbf{y}}_x+\mathbf{e}$ for some $\mathbf{e}\in\mathbb{Z}_q^{\eta m}$ and where $\|\mathbf{e}\|\le B\cdot m^{O(\log\lambda)}$.

Thus, $\mathbf{y}_x=\lfloor\widetilde{\mathbf{y}}_x\rceil_p=\lfloor\widetilde{\mathbf{y}}'_x\rceil_p=\mathbf{y}'_x$ unless $\widetilde{\mathbf{y}}_x\in\mathsf{Borderline}_E$[10] where $E=B\cdot m^{O(\log\lambda)}$. Equivalently, if $\mathbf{y}_x\neq\widetilde{\mathbf{y}}'_x$, then there exists indices $\beta^*\in[m]$ and $\ell^*\in[\eta]$ where
$$\tilde{y}_{x,\beta^*,\ell^*}=\big\langle\mathbf{s},\mathbf{d}_{\beta^*}^{(\ell^*)}\big\rangle\in[-E,E]+(q/p)(\mathbb{Z}+1/2),$$
where $\mathbf{d}_{\beta^*}^{\ell^*}\in\mathbb{Z}_q^n$ is the $(\beta^*)^{\text{th}}$ column of $\mathbf{D}^{(\ell^*)}$ and $\tilde{y}_{x,\beta^*,\ell^*}\in\mathbb{Z}_q$ where we view $\widetilde{\mathbf{y}}_x=(\tilde{y}_{x,\beta,\ell})_{\beta\in[m],\ell\in[\eta]}$. By construction of $\mathbf{z}$ from Eq. (A.3), we thus have
$$\langle\mathbf{d},\mathbf{z}\rangle=\Big\langle\mathbf{s},\mathbf{d}_{\beta^*}^{(\ell^*)}\Big\rangle=\tilde{y}_{x,\beta^*,\ell^*}\in[-E,E]+(q/p)(\mathbb{Z}+1/2).$$

[10] Unlike the proof of Theorem 4.20, we cannot appeal to Lemma A.8 because the secret key $\mathbf{s}$ here is chosen adversarially and depends on the public parameters. This is the reason we rely on the $\mathsf{1D\text{-}SIS\text{-}R}$ assumption.

Furthermore, since each components of $\mathbf{z}$ is either 0 or a component of $\mathbf{s} \in [-B, B]^n$, we have that $\|\mathbf{z}\| \leq B < B \cdot m^{O(\log \lambda)}$. Hence, the vector $\mathbf{z}$ is a correct solution to the 1D-SIS-R challenge $\mathbf{d}$. Thus, if $\textsc{hyb}_1(\mathcal{A})$ outputs 1, then algorithm $\mathcal{B}$ successfully solves the 1D-SIS-R challenge (except with negligible probability). Thus, the advantage of $\mathcal{B}$ is negligibly smaller than $\Pr[\textsc{hyb}_1(\mathcal{A}) = 1]$. The claim follows. □

Combining Lemmas A.9 and A.10, we conclude that under the 1D-SIS-R$_{m',p,q,E}$ assumption, no efficient adversary can find a key $\mathbf{s} \in [-B, B]^n$ and a constraint $S$ where the behavior of the punctured key $k_S$ and the real key $S$ differ on a non-negligible fraction of the non-punctured points. □

## A.4 Proof of Theorem 4.22 (Key-Injectivity)

Key-injectivity is a property only of the public parameters $\mathsf{pp}$. To show this property, we use a simple hybrid argument:

- $\textsc{hyb}_0$: The output of this experiment is the public parameters $\mathsf{pp}$ where $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$. Namely, the public parameters $\mathsf{pp}$ are generated according to Eq. (4.2).
- $\textsc{hyb}_1$: The public parameters $\mathsf{pp}$ are generated as in $\textsc{hyb}_0$, except instead of computing $\mathbf{W}^{(\ell)}$ according to Eq. (4.1), the challenger instead samples $\mathbf{W}^{(\ell)} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$ for all $\ell \in [\eta]$. The other components of $\mathsf{pp}$ are unchanged and the challenger outputs $\mathsf{pp}$.

By the same argument as in the proof of Lemma A.9, the distributions $\textsc{hyb}_0$ and $\textsc{hyb}_1$ are statistically indistinguishable. It thus suffices to show that for the public parameters output by $\textsc{hyb}_1$, key-injectivity holds.

**Lemma A.11.** *Suppose the public parameters* $\mathsf{pp}$ *are generated according to* $\textsc{hyb}_1$*. Then,*

$$\Pr[\exists \mathbf{s}_0, \mathbf{s}_1 \in [-B, B]^n, x \in \{0,1\}^\rho \setminus \{\mathbf{0}\} : \mathsf{Eval}(\mathsf{pp}, \mathbf{s}_0, x) = \mathsf{Eval}(\mathsf{pp}, \mathbf{s}_1, x) \wedge \mathbf{s}_0 \neq \mathbf{s}_1] = \mathsf{negl}(\lambda).$$

*Proof.* Suppose there exist two distinct keys $\mathbf{s}_0 \neq \mathbf{s}_1 \in [-B, B]^n$ and an $x \in \{0,1\}^\rho \setminus \{\mathbf{0}\}$ such that $\mathsf{Eval}(\mathsf{pp}, \mathbf{s}_0, x) = \mathsf{Eval}(\mathsf{pp}, \mathbf{s}_1, x)$. We show that this event happens with negligible probability over the randomness used to sample $\mathbf{W}^{(\ell)}$ in the public parameters. By definition of $\mathsf{Eval}$, we can write

$$\mathsf{Eval}(\mathsf{pp}, \mathbf{s}_0, x) = \lfloor \mathbf{s}_0(\mathbf{W} - \mathbf{Z}_x) \rceil_p \quad \text{and} \quad \mathsf{Eval}(\mathsf{pp}, \mathbf{s}_1, x) = \lfloor \mathbf{s}_1(\mathbf{W} - \mathbf{Z}_x) \rceil_p ,$$

where $\mathbf{W} = (\mathbf{W}^{(1)} \mid \cdots \mid \mathbf{W}^{(\eta)})$, $\mathbf{Z}_x = (\mathbf{Z}_x^{(1)} \mid \cdots \mid \mathbf{Z}^{(\eta)})$, and each $\mathbf{Z}^{(\ell)}$ is as defined in Eq. (4.3). Thus, if $\mathsf{Eval}(\mathsf{pp}, \mathbf{s}_0, x) = \mathsf{Eval}(\mathsf{pp}, \mathbf{s}_1, x)$, it must be the case that $(\mathbf{s}_0 - \mathbf{s}_1)(\mathbf{W} - \mathbf{Z}_x) \in [-B', B']^{\eta m}$, where $B' = q/2p$. Since $\mathbf{W}$ is sampled independently of all components in $\mathbf{Z}$, $(\mathbf{s}_0 - \mathbf{s}_1) \neq \mathbf{0}$, and $q$ is a prime (i.e., $\mathbb{Z}_q$ is a field), this means that that the distribution of $(\mathbf{s}_0 - \mathbf{s}_1)(\mathbf{W} - \mathbf{Z}_x)$ is *uniformly random* over $\mathbb{Z}_q^{\eta m}$. In particular, this means that

$$\Pr_{\mathbf{W} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times \eta m}}[(\mathbf{s}_0 - \mathbf{s}_1)(\mathbf{W} - \mathbf{Z}_x) \in [-B', B']^{\eta m}] \leq (2B'/q)^{\eta m} = 1/p^{\eta m}.$$

We now union bound over all possible values of $(\mathbf{s}_0 - \mathbf{s}_1) \in [-2B, 2B]^n$ and all $x \in \{0,1\}^\rho$ to conclude that

$$\Pr[\exists \mathbf{s}_0, \mathbf{s}_1 \in \mathbb{Z}_q^n, x \in \{0,1\}^\rho \setminus \{\mathbf{0}\} : \mathsf{Eval}(\mathsf{pp}, \mathbf{s}_0, x) = \mathsf{Eval}(\mathsf{pp}, \mathbf{s}_1, x) \wedge \mathbf{s}_0 \neq \mathbf{s}_1] \leq \frac{2^\rho (4B+1)^n}{p^{\eta m}} = \mathsf{negl}(\lambda).$$

Finally, since $\textsc{hyb}_0$ and $\textsc{hyb}_1$ are statistically close, the corresponding probability using the public parameters in $\textsc{hyb}_0$ can only be negligibly different, and the theorem follows. □

### A.5 Proof of Theorem 4.23 (Puncturing Security)

We will prove the theorem under the $\mathsf{HybLWE}_{n,m_1,m_2,q,\chi}$ assumption where $m_1 = \eta m(nm+(t+2)\rho+1)$ and $m_2 = \eta m$ (which is implied by the $\mathsf{LWE}_{n,m,q,\chi}$ assumption by Lemma A.5). We proceed via a sequence of hybrid experiments between an adversary $\mathcal{A}$, and a challenger.

- $\textsc{hyb}_0$: This is the real puncturing security experiment $\mathsf{ExptPunc}_{\Pi_{\mathsf{EPRF}},\mathcal{A}}(\lambda, 0)$ from Definition 4.16. Specifically, the challenger proceeds in each phase of the experiment as follows:
  - **Setup phase**: At the start of the experiment, the adversary $\mathcal{A}$ commits to a challenge set $S \subseteq \{0,1\}^\rho \backslash \{\mathbf{0}\}$ where $S = \{x_i\}_{i\in[t]}$ and a challenge point $x^* \in S$ to the challenger. Then, the challenger generates the public parameters $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$, PRF key $k = \mathbf{s} \leftarrow \mathsf{SampleKey}(\mathsf{pp})$, punctured key $k_S \leftarrow \mathsf{Puncture}(\mathsf{pp}, k, S)$, and the challenge evaluation $\mathbf{y}^* \leftarrow \mathsf{Eval}(\mathsf{pp}, k, x^*) \in \mathbb{Z}_p^{\eta m}$. It gives $\mathsf{pp}$, $k_S$, and $\mathbf{y}^*$ to $\mathcal{A}$.
  - **Query phase**: Adversary $\mathcal{A}$ issues a polynomial number of evaluation queries to the challenger. On an evaluation query $x \in S$, the challenger computes $\mathbf{y}_x \leftarrow \mathsf{Eval}(\mathsf{pp}, k, x)$ and gives $\mathbf{y}_x$ to $\mathcal{A}$.
  - **Output phase**: Adversary $\mathcal{A}$ outputs a bit $b \in \{0,1\}$, which is the output of the experiment.
- $\textsc{hyb}_1$: Same as $\textsc{hyb}_0$ except when generating the public parameters $\mathsf{pp}$ during the setup phase, the challenger samples $\mathbf{W}^{(\ell)} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times m}$ for all $\ell \in [\eta]$.
- $\textsc{hyb}_2$: Same as $\textsc{hyb}_1$ except during the setup phase, instead of sampling the matrices $\big(\mathbf{B}_{i,j}^{(\ell)}\big)_{i\in[t],j\in[\rho]}$ and $\big(\mathbf{C}_j^{(\ell)}\big)_{j\in[\rho]}$ uniformly at random, it first samples another set of matrices $\big(\widehat{\mathbf{B}}_{i,j}^{(\ell)}\big)_{i\in[t],j\in[\rho]}$, $\big(\mathbf{C}_j^{(\ell)}\big)_{j\in[\rho]}$ uniformly at random, and defines for all $\ell \in [\eta]$
  - $\mathbf{B}_{i,j}^{(\ell)} = \widehat{\mathbf{B}}_{i,j}^{(\ell)} - x_{i,j} \cdot \mathbf{G}$ for all $i \in [t]$ and $j \in [\rho]$,
  - $\mathbf{C}_j^{(\ell)} = \widehat{\mathbf{C}}_j^{(\ell)} - x_j^* \cdot \mathbf{G}$ for all $j \in [\rho]$,

  where $x_{i,j}$ is the $j^{\text{th}}$ bit of $x_i \in S$, and $x_j^*$ is the $j^{\text{th}}$ bit of $x^*$.
- $\textsc{hyb}_3$: Same as $\textsc{hyb}_2$ except when computing the challenge value to be $\mathbf{y}^* \leftarrow \mathsf{Eval}(\mathsf{pp}, k, x^*)$, the challenger instead samples $\mathbf{y}^* \xleftarrow{\text{R}} \mathbb{Z}_p^{\eta m}$.
- $\textsc{hyb}_4$: Same as $\textsc{hyb}_3$, except the matrices $\big(\mathbf{B}_{i,j}^{(\ell)}\big)_{i\in[t],j\in[\rho]}$ and $\big(\mathbf{C}_j^{(\ell)}\big)_{j\in[\rho]}$ are sampled uniformly at random.
- $\textsc{hyb}_5$: Same as $\textsc{hyb}_4$, except during the setup phase, the public parameters are generated according to $\mathsf{PrmsGen}$. This corresponds to the ideal puncturing security experiment $\mathsf{ExptPunc}_{\Pi_{\mathsf{EPRF}},\mathcal{A}}(\lambda, 1)$.

For an index $i$ and an adversary $\mathcal{A}$, we write $\textsc{hyb}_i(\mathcal{A})$ to denote the output of experiment $\textsc{hyb}_i$. We now show that the output distributions of each consecutive pair of hybrids are computationally (or statistically) indistinguishable.

**Lemma A.12.** *Suppose that the lattice parameters $n, m, q, p$ satisfy the requirements in Theorem 3.5. Then, for all adversaries $\mathcal{A}$, $|\Pr[\textsc{hyb}_0(\mathcal{A}) = 1] - \Pr[\textsc{hyb}_1(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

*Proof.* Follows by the same argument as that used in the proof of Lemma A.9. □

**Lemma A.13.** *For all adversaries $\mathcal{A}$, $|\Pr[\textsc{hyb}_1(\mathcal{A}) = 1] - \Pr[\textsc{hyb}_2(\mathcal{A}) = 1]| = 0$.*

*Proof.* Since the matrices $\big(\mathbf{B}_{i,j}^{(\ell)}\big)_{i\in[t],j\in[\rho]}$ and $\big(\mathbf{C}_j^{(\ell)}\big)_{j\in[\rho]}$ are sampled uniformly at random, the corresponding set of matrices in the public parameters are also uniformly (and independently) distributed. Thus, the output distributions of $\textsc{hyb}_1$ and $\textsc{hyb}_2$ are identical. □

**Lemma A.14.** *Suppose that $2^\rho B \cdot m^{O(\log\lambda)} \cdot p/q = \mathsf{negl}(\lambda)$, and let $m_1 = \eta m(nm + (t+2)\rho + 1)$ and $m_2 = \eta m$. Under the $\mathsf{HybLWE}_{n,m_1,m_2,q,\chi}$ assumption, for all efficient adversaries $\mathcal{A}$, $|\Pr[\textsc{hyb}_2(\mathcal{A}) = 1] - \Pr[\textsc{hyb}_3(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

*Proof.* Suppose that there is an adversary $\mathcal{A}$ where the output distributions of $\textsc{hyb}_2(\mathcal{A})$ and $\textsc{hyb}_3(\mathcal{A})$ are noticeably different. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks the $\mathsf{HybLWE}_{n,m_1,m_2,q,\chi}$ assumption. Algorithm $\mathcal{B}$ proceeds as follows:

1. **Parsing the $\mathsf{HybLWE}$ challenge.** At the beginning of the experiment, algorithm $\mathcal{B}$ receives a challenge $(\mathbf{A}_1, \mathbf{A}_2, \mathbf{u}_1, \mathbf{u}_2)$ from the $\mathsf{HybLWE}_{n,m_1,m_2,q,\chi}$ challenger. Algorithm $\mathcal{B}$ interprets the matrix $\mathbf{A}_1 \in \mathbb{Z}_q^{n\times m_1}$ as the concatenation of the following collection of matrices (each in $\mathbb{Z}_q^{n\times m}$:
$$\Big(\widehat{\mathbf{W}}^{(\ell)}, \big(\widehat{\mathbf{A}}_j^{(\ell)}\big)_{j\in[\rho]}, \big(\widehat{\mathbf{A}}_{\alpha,\beta}^{(\ell)}\big)_{\alpha\in[n],\beta\in[m]}, \big(\widehat{\mathbf{B}}_{i,j}^{(\ell)}\big)_{i\in[t],j\in[\rho]}, \big(\widehat{\mathbf{C}}_j^{(\ell)}\big)_{j\in[\rho]}, \Big)_{\ell\in[\eta]}.$$
Correspondingly, it interprets the vector $\mathbf{u}_1 \in \mathbb{Z}_q^{m_1}$ as the concatenation of the following collection of vectors (each in $\mathbb{Z}_q^m$):
$$\Big(\hat{\mathbf{w}}^{(\ell)}, \big(\hat{\mathbf{a}}_j^{(\ell)}\big)_{j\in[\rho]}, \big(\hat{\mathbf{a}}_{\alpha,\beta}^{(\ell)}\big)_{\alpha\in[n],\beta\in[m]}, \big(\hat{\mathbf{b}}_{i,j}^{(\ell)}\big)_{i\in[t],j\in[\rho]}, \big(\hat{\mathbf{c}}_j^{(\ell)}\big)_{j\in[\rho]}, \Big)_{\ell\in[\eta]}. \tag{A.4}$$
It interprets the matrix $\mathbf{A}_2 \in \mathbb{Z}_q^{n\times m_2}$ as the concatenation of the collection of matrices $\big(\widehat{\mathbf{V}}^{(\ell)}\big)_{\ell\in[\eta]}$ where each $\widehat{\mathbf{V}}^{(\ell)} \in \mathbb{Z}_q^{n\times m}$. It interprets the vector $\mathbf{u}_2 \in \mathbb{Z}_q^{m_2}$ as the concatenation of the collection of vectors $\big(\hat{\mathbf{v}}^{(\ell)}\big)_{\ell\in[\eta]}$ where $\hat{\mathbf{v}}^{(\ell)} \in \mathbb{Z}_q^m$ for each $\ell \in [\eta]$.

2. **Simulating the public parameters and trapdoor.** Algorithm $\mathcal{B}$ begins simulating an execution of $\textsc{hyb}_2$ and $\textsc{hyb}_3$ for $\mathcal{A}$. At the start of the setup phase, Adversary $\mathcal{A}$ commits to a challenge set $S = \{x_i\}_{i\in[t]}$ and a challenge point $x^* \in S$. Algorithm $\mathcal{B}$ constructs the components of the public parameters $\mathsf{pp}$ for each $\ell \in [\eta]$ as follows:

   - $\mathbf{W}^{(\ell)} = \widehat{\mathbf{W}}^{(\ell)}$,
   - $\mathbf{A}_j^{(\ell)} = \widehat{\mathbf{A}}_j^{(\ell)}$ for all $j \in [\rho]$,
   - $\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)} = \widehat{\mathbf{A}}_{\alpha,\beta}^{(\ell)}$ for all $\alpha \in [n]$ and $\beta \in [m]$,
   - $\mathbf{B}_{i,j}^{(\ell)} = \widehat{\mathbf{B}}_{i,j}^{(\ell)} - x_{i,j} \cdot \mathbf{G}$ for all $i \in [t], j \in [\rho]$,
   - $\mathbf{C}_j^{(\ell)} = \widehat{\mathbf{C}}_j^{(\ell)} - x_j^* \cdot \mathbf{G}$ for all $j \in [\rho]$,
   - $\mathbf{V}^{(\ell)} = \widehat{\mathbf{V}}^{(\ell)}$,

   where $x_{i,j}$ denotes the $j^{\text{th}}$ component of $x_i \in S$, and $x_j^*$ denote the $j^{\text{th}}$ component of $x^*$. It sets the public parameters $\mathsf{pp}$ to be
$$\mathsf{pp} = \Big(\mathbf{W}^{(\ell)}, \big(\mathbf{A}_j^{(\ell)}\big)_{j\in[\rho]}, \big(\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)}\big)_{\alpha\in[n],\beta\in[m]}, \big(\mathbf{B}_{i,j}^{(\ell)}\big)_{i\in[t],j\in[\rho]}, \big(\mathbf{C}_j^{(\ell)}\big)_{j\in[\rho]}, \mathbf{V}^{(\ell)}\Big)_{\ell\in[\eta]}.$$

3. **Simulating the punctured key.** For the punctured key, $\mathcal{B}$ sets
$$k_S = \Big(S, \big(\hat{\mathbf{w}}^{(\ell)}, (\hat{\mathbf{a}}_j^{(\ell)})_{j\in[\rho]}, (\hat{\mathbf{a}}_{\alpha,\beta}^{(\ell)})_{\alpha\in[n],\beta\in[m]}, (\hat{\mathbf{b}}_{i,j}^{(\ell)})_{i\in[t],j\in[\rho]}\big)_{\ell\in[\eta]}\Big),$$
and gives $k_S$ to $\mathcal{A}$.

4. **Common parameters for simulating PRF evaluations.** Before describing how $\mathcal{B}$ simulates the challenge value and the evaluation queries, we define some common notation. Specifically, for an input $x \in \{0,1\}^\rho$, we define the following vectors and matrices:
$$\begin{aligned}
\mathbf{a}_x^{(\ell)} &\leftarrow \mathsf{EvalP}_{\mathsf{ct}}\big(f_x^{\mathsf{eq}}, \mathbf{0}, (\mathbf{A}_j^{(\ell)})_{j\in[\rho]}, (\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)})_{\alpha\in[n],\beta\in[m]}, (\hat{\mathbf{a}}_j^{(\ell)})_{j\in[\rho]}, (\hat{\mathbf{a}}_{\alpha,\beta}^{(\ell)})_{\alpha\in[n],\beta\in[m]}\big),\\
\mathbf{b}_x^{(\ell)} &\leftarrow \mathsf{Eval}_{\mathsf{ct}}\big(f_x^{\mathsf{con}}, S, (\mathbf{B}_{i,j}^{(\ell)})_{i\in[t],j\in[\rho]}, (\hat{\mathbf{b}}_{i,j}^{(\ell)})_{i\in[t],j\in[\rho]}\big),\\
\mathbf{C}_x^{(\ell)} &\leftarrow \mathsf{Eval}_{\mathsf{pk}}\big(f_x^{\mathsf{eq}}, (\mathbf{C}_j^{(\ell)})_{j\in[\rho]}\big),\\
\mathbf{c}_x^{(\ell)} &\leftarrow \mathsf{Eval}_{\mathsf{ct}}\big(f_x^{\mathsf{eq}}, x, S, (\mathbf{C}_j^{(\ell)})_{j\in[\rho]}, (\hat{\mathbf{c}}_j^{(\ell)})_{j\in[\rho]}\big).
\end{aligned} \tag{A.5}$$

5. **Simulating the challenge evaluation.** To simulate the challenge evaluation $\mathbf{y}^*$, algorithm $\mathcal{B}$ computes $\mathbf{a}_{x^*}^{(\ell)}$, $\mathbf{b}_{x^*}^{(\ell)}$, $\mathbf{C}_{x^*}^{(\ell)}$, and $\mathbf{c}_{x^*}^{(\ell)}$ according to Eq. (A.5) for all $\ell \in [\eta]$. Then, it sets
$$\mathbf{z}_{x^*}^{(\ell)} = \mathbf{a}_{x^*}^{(\ell)} + \mathbf{b}_{x^*}^{(\ell)}\mathbf{G}^{-1}(\mathbf{C}_{x^*}^{(\ell)})\mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) - \mathbf{c}_{x^*}^{(\ell)}\mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) + \hat{\mathbf{v}}^{(\ell)}, \tag{A.6}$$
for all $\ell \in [\eta]$ and finally, it returns the vector
$$\mathbf{y}^* = \Big\lfloor \hat{\mathbf{w}}^{(1)} - \mathbf{z}_{x^*}^{(1)} \mid \cdots \mid \hat{\mathbf{w}}^{(\eta)} - \mathbf{z}_{x^*}^{(\eta)} \Big\rceil_p .$$

6. **Simulating the evaluation queries.** When the adversary $\mathcal{A}$ makes an evaluation query on an input $x \in S$, algorithm $\mathcal{B}$ proceeds by computing $\mathbf{a}_x^{(\ell)}$, $\mathbf{b}_x^{(\ell)}$, $\mathbf{C}_x^{(\ell)}$, and $\mathbf{c}_x^{(\ell)}$ according to Eq. (A.5) for all $\ell \in [\eta]$. Then, it sets
$$\mathbf{z}_x^{(\ell)} = \mathbf{a}_x^{(\ell)} + \mathbf{b}_x^{(\ell)}\mathbf{G}^{-1}(\mathbf{C}_x^{(\ell)})\mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) - \mathbf{c}_x^{(\ell)}, \tag{A.7}$$
for all $\ell \in [\eta]$ and returns the vector
$$\mathbf{y}_x = \Big\lfloor \hat{\mathbf{w}}^{(1)} - \mathbf{z}_x^{(1)} \mid \cdots \mid \hat{\mathbf{w}}^{(\eta)} - \mathbf{z}_x^{(\eta)} \Big\rceil_p .$$

7. **Output of the experiment.** At the end of the experiment, algorithm $\mathcal{A}$ outputs a bit $b \in \{0,1\}$. Algorithm $\mathcal{B}$ simply echoes the bit $b$.

**Correctness of the simulation.** To complete the proof, we show that algorithm $\mathcal{B}$ correctly simulates the views of either $\textsc{hyb}_2$ or $\textsc{hyb}_3$ to $\mathcal{A}$ depending on whether the challenge vectors $\mathbf{u}_2 = (\hat{\mathbf{v}}^{(\ell)})_{\ell\in[\eta]} \in \mathbb{Z}_q^{m_2}$ consist of LWE samples or if they are uniformly random. We consider each components in the simulation separately:

- **Public parameters.** In the $\mathsf{HybLWE}$ security game, the matrix $\mathbf{A}_1$ is sampled uniformly at random from $\mathbb{Z}_q^{n\times m_1}$. Thus, $\mathcal{B}$ constructs the public parameters $\mathsf{pp}$ using the same procedure as that in $\textsc{hyb}_2$ and $\textsc{hyb}_3$.

- **Secret PRF key.** The LWE secret $\mathbf{s}$ (chosen by the $\mathsf{HybLWE}$ challenger) plays the role of the PRF key in the simulation. Note that $\mathsf{HybLWE}$ challenger samples $\mathbf{s}$ from the same distribution as the $\mathsf{SampleKey}$ algorithm used in $\textsc{hyb}_2$ and $\textsc{hyb}_3$.

- **Punctured key.** By definition of $\mathsf{HybLWE}$, we can write the vectors in $\mathbf{u}_1$ (Eq. (A.4)) as follows:

$$\begin{aligned}
\hat{\mathbf{w}}^{(\ell)} &= \mathbf{s}\widehat{\mathbf{W}}^{(\ell)} + \mathbf{e}_{\mathbf{W}}^{(\ell)} &&= \mathbf{s}\mathbf{W}^{(\ell)} + \mathbf{e}_{\mathbf{W}}^{(\ell)} && \\
\hat{\mathbf{a}}_j^{(\ell)} &= \mathbf{s}\widehat{\mathbf{A}}_j^{(\ell)} + \mathbf{e}_{\mathbf{A},j}^{(\ell)} &&= \mathbf{s}\mathbf{A}_j^{(\ell)} + \mathbf{e}_{\mathbf{A},j}^{(\ell)} && \text{for all } j \in [\rho], \\
\hat{\mathbf{a}}_{\alpha,\beta}^{(\ell)} &= \mathbf{s}\widehat{\mathbf{A}}_{\alpha,\beta}^{(\ell)} + \mathbf{e}_{\mathbf{A},\alpha,\beta}^{(\ell)} &&= \mathbf{s}\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)} + \mathbf{e}_{\mathbf{A},\alpha,\beta}^{(\ell)} && \text{for all } \alpha \in [n], \beta \in [m], \\
\hat{\mathbf{b}}_{i,j}^{(\ell)} &= \mathbf{s}\widehat{\mathbf{B}}_{i,j}^{(\ell)} + \mathbf{e}_{\mathbf{B},i,j}^{(\ell)} &&= \mathbf{s}(\mathbf{B}_{i,j}^{(\ell)} + x_{i,j} \cdot \mathbf{G}) + \mathbf{e}_{\mathbf{B},i,j}^{(\ell)} && \text{for all } i \in [t], j \in [\rho], \\
\hat{\mathbf{c}}_j^{(\ell)} &= \mathbf{s}\widehat{\mathbf{C}}_j^{(\ell)} + \mathbf{e}_{\mathbf{C},j}^{(\ell)} &&= \mathbf{s}(\mathbf{C}_j^{(\ell)} + x_j^* \cdot \mathbf{G}) + \mathbf{e}_{\mathbf{C},j}^{(\ell)} && \text{for all } j \in [\rho],
\end{aligned}$$

  for some $\mathbf{s} \in [-B, B]^n$ and error vectors $\mathbf{e}_{\mathbf{W}}^{(\ell)}$, $\mathbf{e}_{\mathbf{A},j}^{(\ell)}$, $\mathbf{e}_{\mathbf{A},\alpha,\beta}^{(\ell)}$, $\mathbf{e}_{\mathbf{B},i,j}^{(\ell)}$, and $\mathbf{e}_{\mathbf{C},j}^{(\ell)}$ that are sampled from $\chi^m$. Hence, the punctured key

$$k_S = \Big(S, \big(\hat{\mathbf{w}}^{(\ell)}, (\hat{\mathbf{a}}_j^{(\ell)})_{j\in[\rho]}, (\hat{\mathbf{a}}_{\alpha,\beta}^{(\ell)})_{\alpha\in[n],\beta\in[m]}, (\hat{\mathbf{b}}_{i,j}^{(\ell)})_{i\in[t],j\in[m]}\big)_{\ell\in[\eta]}\Big),$$

  is correctly simulated as in $\textsc{hyb}_2$ and $\textsc{hyb}_3$.

- **Evaluation queries.** Suppose the adversary makes an evaluation query on an input $x \in S$. In $\textsc{hyb}_2$ and $\textsc{hyb}_3$, the challenger would reply with the real evaluation

$$\overline{\mathbf{y}}_x = \mathsf{Eval}(\mathsf{pp}, \mathbf{s}, x) = \Big\lfloor \mathbf{s}(\mathbf{W}^{(1)} - \mathbf{Z}_x^{(1)} \mid \cdots \mid \mathbf{W}^{(\eta)} - \mathbf{Z}_x^{(\eta)} \Big\rceil_p,$$

  where $\mathbf{W}^{(\ell)} = \widehat{\mathbf{W}}^{(\ell)}$ is the matrix in the public parameters $\mathsf{pp}$ and $\mathbf{Z}^{(\ell)}$ is as defined in Eq. (4.3) for all $\ell \in [\eta]$. We show that with overwhelming probability, algorithm $\mathcal{B}$ replies with the same value. By Theorems 3.6 and 3.7, and the fact that $f_x^{\mathsf{eq}}\big(\mathbf{0}\big) = 0 = f_x^{\mathsf{eq}}\big(x^*\big)$ and $f_x^{\mathsf{con}}(S) = 1$, the vectors $\mathbf{a}_x^{(\ell)}$, $\mathbf{b}_x^{(\ell)}$, and $\mathbf{c}_x^{(\ell)}$ from Eq. (A.5) satisfy the following relations:

$$\begin{aligned}
\mathbf{a}_x^{(\ell)} &= \mathbf{s}\big(\mathbf{A}_x^{(\ell)} + f_x^{\mathsf{eq}}(\mathbf{0}) \cdot \mathbf{D}\big) + \mathbf{e}_{\mathbf{a},x}^{(\ell)} = \mathbf{s}\mathbf{A}_x^{(\ell)} + \mathbf{e}_{\mathbf{a},x}^{(\ell)}, \\
\mathbf{b}_x^{(\ell)} &= \mathbf{s}\big(\mathbf{B}_x^{(\ell)} + f_x^{\mathsf{con}}(S) \cdot \mathbf{G}\big) + \mathbf{e}_{\mathbf{b},x}^{(\ell)} = \mathbf{s}(\mathbf{B}_x^{(\ell)} + \mathbf{G}) + \mathbf{e}_{\mathbf{b},x}^{(\ell)}, \\
\mathbf{c}_x^{(\ell)} &= \mathbf{s}\big(\mathbf{B}_x^{(\ell)} + f_x^{\mathsf{con}}(x^*) \cdot \mathbf{G}\big) + \mathbf{e}_{\mathbf{c},x}^{(\ell)} = \mathbf{s}\mathbf{C}_x^{(\ell)} + \mathbf{e}_{\mathbf{c},x}^{(\ell)},
\end{aligned}$$

  for error vectors $\big\|\mathbf{e}_{\mathbf{a},x}^{(\ell)}\big\|, \big\|\mathbf{e}_{\mathbf{b},x}^{(\ell)}\big\|, \big\|\mathbf{e}_{\mathbf{c},x}^{(\ell)}\big\| \leq B \cdot m^{O(d)} = B \cdot m^{O(\log\lambda)}$ since $d = O(\log\lambda)$. Substituting into Eq. (A.7), this means that in the simulation

$$\begin{aligned}
\mathbf{z}_x^{(\ell)} &= \mathbf{a}_x^{(\ell)} + \mathbf{b}_x^{(\ell)}\mathbf{G}^{-1}(\mathbf{C}_x^{(\ell)})\mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) - \mathbf{c}_x^{(\ell)} \\
&= \mathbf{s}\big(\mathbf{A}^{(\ell)} + \mathbf{B}^{(\ell)}\mathbf{G}^{-1}(\mathbf{C}^{(\ell)})\mathbf{G}^{-1}(\mathbf{V}^{(\ell)})\big) + \mathbf{e}_{\mathbf{z},x}^{(\ell)} \\
&= \mathbf{s}\mathbf{Z}_x^{(\ell)} + \mathbf{e}_{\mathbf{z},x}^{(\ell)},
\end{aligned}$$

  where $\mathbf{Z}_x^{(\ell)}$ is as defined in Eq. (4.3) and where $\|\mathbf{e}_{\mathbf{z},x}^{(\ell)}\| \leq B \cdot m^{O(\log\lambda)}$. Thus, the simulation evaluation $\mathbf{y}_x$ has the form

$$\begin{aligned}
\mathbf{y}_x &= \Big\lfloor \hat{\mathbf{w}}^{(1)} - \mathbf{z}_x^{(1)} \mid \cdots \mid \hat{\mathbf{w}}^{(\eta)} - \hat{\mathbf{e}}_x^{(\eta)} \Big\rceil_p \\
&= \Big\lfloor \mathbf{s}\big(\mathbf{W}^{(1)} - \mathbf{Z}_x^{(1)} + \mathbf{e}^{(1)} \mid \cdots \mid \mathbf{W}^{(\eta)} - \mathbf{Z}_x^{(\eta)} + \mathbf{e}^{(\eta)}\big) \Big\rceil_p
\end{aligned}$$

  using the relation for $\hat{\mathbf{w}}$ from Eq. (A.8) and where $\big\|\mathbf{e}^{(\ell)}\big\| \leq B \cdot m^{O(\log\lambda)}$ for all $\ell \in [\eta]$. This means that the simulated value $\mathbf{y}_x$ and the real evaluation $\overline{\mathbf{y}}_x$ are identical as long as $\overline{\mathbf{y}}_x \notin \mathsf{Borderline}_E$ for $E = B \cdot m^{O(\log\lambda)}$. By Lemma A.8, for the provided parameters, we have $\overline{\mathbf{y}}_x \notin \mathsf{Borderline}_E$ with overwhelming probability. Thus, the evaluation queries are correctly simulated with overwhelming probability.

- **Challenge value.** Using Theorems 3.6 and 3.7, and the fact that $f^{\mathsf{eq}}_{x^*}(\mathbf{0}) = 0$ and $f^{\mathsf{con}}_{x^*}(S) = f^{\mathsf{eq}}_{x^*}(x^*) = 1$, the vectors $\big(\mathbf{a}^{(\ell)}_{x^*}\big)_{\ell \in [\eta]}$, $\big(\mathbf{b}^{(\ell)}_{x^*}\big)_{\ell \in [\eta]}$, and $\big(\mathbf{c}^{(\ell)}_{x^*}\big)_{j \in [t]}$ from Eq. (A.5) satisfy the following relations:

$$\begin{aligned}
\mathbf{a}^{(\ell)}_{x^*} &= \mathbf{s}\big(\mathbf{A}^{(\ell)}_{x^*} + f^{\mathsf{eq}}_{x^*}(\mathbf{0}) \cdot \mathbf{D}\big) + \mathbf{e}^{(\ell)}_{\mathbf{a},x^*} = \mathbf{s}\mathbf{A}^{(\ell)}_{x^*} + \mathbf{e}^{(\ell)}_{\mathbf{a},x^*}, \\
\mathbf{b}^{(\ell)}_{x^*} &= \mathbf{s}\big(\mathbf{B}^{(\ell)}_{x^*} + f^{\mathsf{con}}_{x^*}(S) \cdot \mathbf{G}\big) + \mathbf{e}^{(\ell)}_{\mathbf{b},x^*} = \mathbf{s}(\mathbf{B}^{(\ell)}_{x^*} + \mathbf{G}) + \mathbf{e}^{(\ell)}_{\mathbf{b},x^*}, \\
\mathbf{c}^{(\ell)}_{x^*} &= \mathbf{s}\big(\mathbf{C}^{(\ell)}_{x^*} + f^{\mathsf{eq}}_{x^*}(x^*) \cdot \mathbf{G}\big) + \mathbf{e}^{(\ell)}_{\mathbf{c},x^*} = \mathbf{s}(\mathbf{C}^{(\ell)}_{x^*} + \mathbf{G}) + \mathbf{e}^{(\ell)}_{\mathbf{c},x^*},
\end{aligned} \tag{A.8}$$

for all $\ell \in [\eta]$ where $\big\|\mathbf{e}^{(\ell)}_{\mathbf{a},x^*}\big\|, \big\|\mathbf{e}^{(\ell)}_{\mathbf{b},x^*}\big\|, \big\|\mathbf{e}^{(\ell)}_{\mathbf{c},x^*}\big\| \le B \cdot m^{O(\log \lambda)}$. Substituting into Eq. (A.6), this means that in the simulation

$$\begin{aligned}
\mathbf{z}^{(\ell)}_{x^*} &= \mathbf{a}^{(\ell)}_{x^*} + \mathbf{b}^{(\ell)}_{x^*}\mathbf{G}^{-1}(\mathbf{C}^{(\ell)}_{x^*})\mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) - \mathbf{c}^{(\ell)}_{x^*}\mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) + \hat{\mathbf{v}}^{(\ell)} \\
&= \mathbf{s}(\mathbf{A}^{(\ell)}_{x^*} + \mathbf{B}^{(\ell)}_{x^*}\mathbf{G}^{-1}(\mathbf{C}^{(\ell)}_{x^*})\mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) - \mathbf{V}^{(\ell)}) + \hat{\mathbf{v}}^{(\ell)} + \mathbf{e}^{(\ell)}_{\mathbf{z},x^*} \\
&= \mathbf{s}(\mathbf{Z}^{(\ell)}_{x^*} - \mathbf{V}^{(\ell)}) + \hat{\mathbf{v}}^{(\ell)} + \mathbf{e}^{(\ell)}_{\mathbf{z},x^*},
\end{aligned}$$

using the definition of $\mathbf{Z}^{(\ell)}_{x^*}$ from Eq. (4.3) and where $\big\|\mathbf{e}^{(\ell)}_{\mathbf{z},x^*}\big\| \le B \cdot m^{O(\log \lambda)}$. Thus, for all $\ell \in [\eta]$,

$$\hat{\mathbf{w}}^{(\ell)} - \mathbf{z}^{(\ell)}_{x^*} = \mathbf{s}(\mathbf{W}^{(\ell)} - \mathbf{Z}^{(\ell)}_{x^*} + \mathbf{V}^{(\ell)}) - \hat{\mathbf{v}}^{(\ell)} + \hat{\mathbf{e}}^{(\ell)}, \tag{A.9}$$

using the relation for $\hat{\mathbf{w}}^{(\ell)}$ from Eq. (A.8). We now consider the behavior of the challenger in $\textsc{hyb}_2$ and $\textsc{hyb}_3$ and show that if the vectors $\hat{\mathbf{v}}^{(\ell)}$ are LWE samples, then $\mathcal{B}$ correctly simulates $\textsc{hyb}_2$ for $\mathcal{A}$ (with overwhelming probability); otherwise, if the vectors $\hat{\mathbf{v}}^{(\ell)}$ are uniformly random, then $\mathcal{B}$ correctly simulates $\textsc{hyb}_3$ for $\mathcal{A}$.

  - In $\textsc{hyb}_2$, the challenger answers the challenge query using $\mathsf{Eval}(\mathsf{pp}, \mathbf{s}, x^*)$, which computes and outputs the following:

    $$\bar{\mathbf{y}}^* = \mathsf{Eval}(\mathsf{pp}, \mathbf{s}, x^*) = \Big\lfloor \mathbf{s}(\mathbf{W}^{(1)} - \mathbf{Z}^{(1)}_{x^*} \mid \cdots \mid \mathbf{W}^{(\eta)} - \mathbf{Z}^{(\eta)}_{x^*}) \Big\rceil_p .$$

    Suppose $\hat{\mathbf{v}}^{(\ell)} = \mathbf{s}\mathbf{V}^{(\ell)} + \mathbf{e}^{(\ell)}_{\mathbf{v}}$ where $\mathbf{e}^{(\ell)}_{\mathbf{v}}$ is sampled from $\chi^m$. In this case, Eq. (A.9) becomes

    $$\hat{\mathbf{w}}^{(\ell)} - \mathbf{z}^{(\ell)}_{x^*} = \mathbf{s}(\mathbf{W}^{(\ell)} - \mathbf{Z}^{(\ell)}_{x^*}) + (\hat{\mathbf{e}}^{(\ell)} - \mathbf{e}^{(\ell)}_{\mathbf{v}}),$$

    and the simulated challenge evaluation has the form

    $$\begin{aligned}
    \mathbf{y}^* &= \Big\lfloor \hat{\mathbf{w}}^{(1)} - \mathbf{z}^{(1)}_{x^*} \mid \cdots \mid \hat{\mathbf{w}}^{(\eta)} - \mathbf{z}^{(\eta)}_{x^*} \Big\rceil_p \\
    &= \Big\lfloor \mathbf{s}\big(\mathbf{W}^{(1)} - \mathbf{Z}^{(1)}_{x^*}\big) + (\hat{\mathbf{e}}^{(1)} - \mathbf{e}^{(1)}_{\mathbf{v}}) \mid \cdots \mid \mathbf{s}\big(\mathbf{W}^{(\eta)} - \mathbf{Z}^{(\eta)}_{x^*}\big) + (\hat{\mathbf{e}}^{(\eta)} - \mathbf{e}^{(\eta)}_{\mathbf{v}}) \Big\rceil_p
    \end{aligned}$$

    Thus the simulated value $\mathbf{y}^*$ and the real evaluation $\bar{\mathbf{y}}^*$ are identical as long as $\bar{\mathbf{y}}^* \notin \mathsf{Borderline}_E$ for $E = B \cdot m^{O(\log \lambda)}$. By Lemma A.8, for the provided parameters, $\bar{\mathbf{y}}^* \notin \mathsf{Borderline}_E$ with overwhelming probability, and so algorithm $\mathcal{B}$ correctly simulates the challenge evaluation according to the specification of $\textsc{hyb}_2$.

  - In $\textsc{hyb}_3$, the challenger answers the evaluation query by sampling $\bar{\mathbf{y}}^* \xleftarrow{\text{R}} \mathbb{Z}_p^{\eta m}$. If each $\hat{\mathbf{v}}^{(\ell)}$ is a uniformly random vector over $\mathbb{Z}_q^m$ (and independent of all of the other components), then from Eq. (A.9), $\hat{\mathbf{w}}^{(\ell)} - \mathbf{z}^{(\ell)}_{x^*}$ is also uniformly random over $\mathbb{Z}_p^m$ for all $\ell \in [\eta]$. This means that algorithm $\mathcal{B}$'s response $\mathbf{y}^*$ is uniformly random over $\mathbb{Z}_p^{\eta m}$, in which case $\mathcal{B}$ perfectly simulates the distribution in $\textsc{hyb}_3$.

By the above analysis, we have shown that if the vectors $\hat{\mathbf{v}}^{(\ell)}$ consist of LWE samples, then $\mathcal{B}$ correctly simulates $\textsc{hyb}_3$ for $\mathcal{A}$ (up to negligible error) and if they are uniformly random, then $\mathcal{B}$ correctly simulates $\textsc{hyb}_4$ for $\mathcal{A}$ (up to negligible error). Thus, if the outputs of $\textsc{hyb}_2(\mathcal{A})$ and $\textsc{hyb}_3(\mathcal{A})$ are noticeably different, then $\mathcal{B}$ breaks the $\mathsf{HybLWE}_{n,m_1,m_2,q,\chi}$ assumption with noticeable probability. □

**Lemma A.15.** *For all adversaries $\mathcal{A}$, $|\Pr[\textsc{hyb}_3(\mathcal{A}) = 1] = \Pr[\textsc{hyb}_4(\mathcal{A}) = 1]| = 0$.*

*Proof.* Same as the proof of Lemma A.13. □

**Lemma A.16.** *Suppose that the lattice parameters $n, m, q, p$ satisfy the requirements in Theorem 3.5. Then, for all adversaries $\mathcal{A}$, $|\Pr[\textsc{hyb}_4(\mathcal{A}) = 1] = \Pr[\textsc{hyb}_5(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

*Proof.* Same as the proof of Lemma A.12. □

Combining Lemmas A.12 through A.16, the extractable PRF $\Pi_{\mathsf{EPRF}}$ from Construction 4.19 satisfies puncturing security. □

**Remark A.17** (Adaptive Security via Admissible Hash Functions)**.** In the puncture pseudorandomness proof above, we prove *selective* security (Remark 4.4) to better demonstrate the key ideas of our construction and proofs. Using admissible hash functions and partitioning [BB04b, Wat05], we can tweak our construction to also achieve *adaptive* security. As in [BV15], we can augment the PRF with another set of matrices that (in the security proof) encode an admissible hash function. Since an admissible hash function can be implemented in $\mathsf{NC}^1$, this increases the modulus $q$ (and therefore, the approximation factors for the worst-case lattice problems) by only a polynomial factor.

### A.6 Proof of Theorem 4.25 ($T$-Restricted Pseudorandomness Given Trapdoor)

We begin by specifying the set $S$ of restricted points for the PRF family. First, recall that each choice of public parameters $\mathsf{pp}$ in $\Pi_{\mathsf{EPRF}}$ induces a PRF family $\mathsf{F}_{\mathsf{pp}} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. Take $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$, and write $\mathsf{td} = (h^{(\ell)}, \mathsf{td}_{\mathbf{D}^{(\ell)}})_{\ell \in [\eta]}$. The restricted set for the PRF family $\mathsf{F}_{\mathsf{pp}}$ is the set of special points $\{h^{(\ell)}\}_{\ell \in [\eta]}$. We now show that $\Pi_{\mathsf{EPRF}}$ satisfies $T$-restricted pseudorandomness where $T = \eta$ for this particular choice of restricted points.

**Proof of $T$-resticted pseudorandomness.** We will prove the theorem under the $\mathsf{HybLWE}_{n,m_1,m_2,q,\chi}$ assumption where $m_1 = \eta m(nm + \rho(t+2))$ and $m_2 = \eta m$ (which is implied by the $\mathsf{LWE}_{n,m',q,\chi}$ assumption by Lemma A.5). To show *selective* $T$-restricted pseudorandomness, we use the definition based on Definition A.1 (with the modification from Remark 4.4). We proceed via a sequence of hybrid experiments between an adversary $\mathcal{A}$, and a challenger.

- $\textsc{hyb}_0$: This is the real security experiment $\mathsf{ExptPRF}_{\Pi_{\mathsf{EPRF}},\mathcal{A}}(\lambda, 0)$ from Definition A.1. Specifically, the challenger proceeds in each phase of the experiment as follows:
  - **Setup phase**: At the start of the experiment, the challenger samples a set of $\eta$ special points $h^{(\ell)} \xleftarrow{\text{R}} \{0,1\}^\rho$. It defines the restricted set $\mathcal{T} = \{h^{(\ell)}\}_{\ell \in [\eta]}$ and sends $\mathcal{T}$ to the adversary. The adversary then commits to a challenge point $x^* \in \{0,1\}^\rho \setminus (\mathcal{T} \cup \{\mathbf{0}\})$. The challenger samples the remainder of the public parameters using $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$ using $h^{(\ell)}$ as the special points. It then samples a PRF key $k \leftarrow \mathsf{SampleKey}(\mathsf{pp})$, and computes $\mathbf{y}^* \leftarrow \mathsf{Eval}(\mathsf{pp}, k, x^*)$. It gives $(\mathsf{pp}, \mathsf{td}, \mathbf{y}^*)$ to the adversary. Note that this procedure is identical to the behavior of $\mathsf{PrmsGen}$ as specified by $\Pi_{\mathsf{EPRF}}$.

  - **Query phase**: Adversary $\mathcal{A}$ can then issue any (polynomial) number of evaluation queries $x \in \{0,1\}^\rho \setminus (\mathcal{T} \cup \{\mathbf{0}\})$. For each of these queries, the challenger replies with $y_x \leftarrow \mathsf{Eval}(\mathsf{pp}, k, x)$.
  - **Output phase**: Adversary $\mathcal{A}$ outputs a bit $b \in \{0,1\}$, which is the output of the experiment.

- $\textsc{hyb}_1$: Same as $\textsc{hyb}_0$, during the setup phase, except instead of sampling the matrices $\big(\mathbf{A}_j^{(\ell)}\big)_{j\in[\rho]}$, $(\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)})_{\alpha\in[n],\beta\in[m]}$, $\big(\mathbf{B}_{i,j}^{(\ell)}\big)_{i\in[t],j\in[\rho]}$, and $\big(\mathbf{C}_j^{(\ell)}\big)_{j\in[\rho]}$ uniformly at random, the challenger first samples another set of matrices $\big(\widehat{\mathbf{A}}_j^{(\ell)}\big)_{j\in[\rho]}$, $\big(\widehat{\mathbf{A}}_{\alpha,\beta}^{(\ell)}\big)_{\alpha\in[n],\beta\in[m]}$, $\big(\widehat{\mathbf{B}}_{i,j}^{(\ell)}\big)_{i\in[t],j\in[\rho]}$, and $\big(\widehat{\mathbf{C}}_j^{(\ell)}\big)_{j\in[\rho]}$ uniformly at random from $\mathbb{Z}_q^{n\times m}$, and defines for all $\ell \in [\eta]$

  - $\mathbf{A}_j^{(\ell)} = \widehat{\mathbf{A}}_j^{(\ell)} - h_j^{(\ell)} \cdot \mathbf{G}$ for all $j \in [\rho]$,
  - $\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)} = \widehat{\mathbf{A}}_{\alpha,\beta}^{(\ell)} - d_{\alpha,\beta}^{(\ell)} \cdot \mathbf{G}$ for all $\alpha \in [n]$ and $\beta \in [m]$,
  - $\mathbf{B}_{i,j}^{(\ell)} = \widehat{\mathbf{B}}_{i,j}^{(\ell)} - x_j^* \cdot \mathbf{G}$ for all $i \in [t]$ and $j \in [\rho]$,
  - $\mathbf{C}_j^{(\ell)} = \widehat{\mathbf{C}}_j^{(\ell)} - x_i^* \cdot \mathbf{G}$ for all $j \in [\rho]$,

  where $h_j^{(\ell)}$ denotes the $j^{\text{th}}$ bit of $h^{(\ell)}$, $d_{\alpha,\beta}^{(\ell)}$ denotes the $(\alpha,\beta)^{\text{th}}$ component of $\mathbf{D}^{(\ell)}$, and $x_j^*$ denote the $j^{\text{th}}$ component of $x^*$.

- $\textsc{hyb}_2$: Same as $\textsc{hyb}_1$ except when computing the challenge value $\mathbf{y}^*$ during the setup phase, the challenger samples $\mathbf{y}^* \xleftarrow{\text{R}} \mathbb{Z}_p^{\eta m}$.

- $\textsc{hyb}_3$: Same as $\textsc{hyb}_2$, except the public parameters are generated according to $\mathsf{PrmsGen}$. This corresponds to the ideal $T$-restricted pseudorandomness security experiment $\mathsf{ExptPRF}_{\Pi_{\mathsf{EPRF}},\mathcal{A}}(\lambda, 1)$ from Definition A.1.

For an index $i$ and an adversary $\mathcal{A}$, we write $\textsc{hyb}_i(\mathcal{A})$ to denote the output of experiment $\textsc{hyb}_i$. We now show that the output distributions of each consecutive pair of hybrids are computationally (or statistically) indistinguishable.

**Lemma A.18.** *For all adversaries $\mathcal{A}$, $|\Pr[\textsc{hyb}_0(\mathcal{A}) = 1] = \Pr[\textsc{hyb}_1(\mathcal{A}) = 1]| = 0$.*

*Proof.* Since the matrices $\big(\widehat{\mathbf{A}}_j^{(\ell)}\big)_{j\in[\rho]}$, $\big(\widehat{\mathbf{A}}_{\alpha,\beta}^{(\ell)}\big)_{\alpha\in[n],\beta\in[m]}$, $\big(\widehat{\mathbf{B}}_{i,j}^{(\ell)}\big)_{i\in[t],j\in[\rho]}$, and $\big(\widehat{\mathbf{C}}_j^{(\ell)}\big)_{j\in[\rho]}$ are sampled uniformly at random, the corresponding set of matrices in the public parameters are also uniformly (and independently) distributed. Thus, the output distributions of $\textsc{hyb}_0$ and $\textsc{hyb}_1$ are identical. □

**Lemma A.19.** *Suppose $2^\rho B \cdot m^{O(\log\lambda)} \cdot p/q = \mathsf{negl}(\lambda)$, and let $m_1 = \eta m(nm + \rho(t+2))$ and $m_2 = \eta m$. Then, under the $\mathsf{HybLWE}_{n,m_1,m_2,q,\chi}$ assumption, for all efficient adversaries $\mathcal{A}$, it holds that $|\Pr[\textsc{hyb}_1(\mathcal{A}) = 1] - \Pr[\textsc{hyb}_2(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

*Proof.* This proof is very similar to the proof of Lemma A.14. Suppose there is an adversary $\mathcal{A}$ where the output distributions of $\textsc{hyb}_1(\mathcal{A})$ and $\textsc{hyb}_2(\mathcal{A})$ are noticeably different. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks the $\mathsf{HybLWE}_{n,m_1,m_2,q,\chi}$ assumption. Algorithm $\mathcal{B}$ proceeds as follows:

1. **Parsing the $\mathsf{HybLWE}$ challenge.** At the beginning of the experiment, algorithm $\mathcal{B}$ receives a challenge $(\mathbf{A}_1, \mathbf{A}_2, \mathbf{u}_1, \mathbf{u}_2)$ from the $\mathsf{HybLWE}_{n,m_1,m_2,q,\chi}$ challenger. Algorithm $\mathcal{B}$ interprets

the matrix $\mathbf{A}_1 \in \mathbb{Z}_q^{n \times m_1}$ as the concatenation of the following collection of matrices (each in $\mathbb{Z}_q^{n \times m}$):

$$\Big( \big( \widehat{\mathbf{A}}_j^{(\ell)} \big)_{j \in [\rho]}, \big( \widehat{\mathbf{A}}_{\alpha,\beta}^{(\ell)} \big)_{\alpha \in [n], \beta \in [m]}, \big( \widehat{\mathbf{B}}_{i,j}^{(\ell)} \big)_{i \in [t], j \in [\rho]}, \big( \widehat{\mathbf{C}}_j^{(\ell)} \big)_{j \in [\rho]} \Big)_{\ell \in [\eta]}.$$

Correspondingly, it interprets the vector $\mathbf{u}_1 \in \mathbb{Z}_q^{m_1}$ as the concatenation of the following collection of vectors (each in $\mathbb{Z}_q^m$):

$$\Big( \big( \hat{\mathbf{a}}_j^{(\ell)} \big)_{j \in [\rho]}, \big( \hat{\mathbf{a}}_{\alpha,\beta}^{(\ell)} \big)_{\alpha \in [n], \beta \in [m]}, \big( \hat{\mathbf{b}}_{i,j}^{(\ell)} \big)_{i \in [t], j \in [\rho]}, \big( \hat{\mathbf{c}}_j^{(\ell)} \big)_{j \in [\rho]} \Big)_{\ell \in [\eta]}. \tag{A.10}$$

It interprets the matrix $\mathbf{A}_2 \in \mathbb{Z}_q^{n \times m_2}$ as the concatenation of the collection of matrices $\big( \widehat{\mathbf{V}}^{(\ell)} \big)_{\ell \in [\eta]}$ where each $\widehat{\mathbf{V}}^{(\ell)} \in \mathbb{Z}_q^{n \times m}$. It interprets the vector $\mathbf{u}_2 \in \mathbb{Z}_q^{m_2}$ as the concatenation of the collection of vectors $\big( \hat{\mathbf{v}}^{(\ell)} \big)_{\ell \in [\eta]}$ where each $\hat{\mathbf{v}}^{(\ell)} \in \mathbb{Z}_q^m$.

2. **Simulating the public parameters and trapdoor.** Algorithm $\mathcal{B}$ begins simulating an execution of $\textsc{hyb}_1$ and $\textsc{hyb}_2$ for $\mathcal{A}$. At the start of the setup phase, algorithm $\mathcal{B}$ samples a set of test points $h^{(\ell)} \xleftarrow{\text{R}} \{0,1\}^\rho$ for $\ell \in [\eta]$. It gives the set $\mathcal{T} = \{h^{(\ell)}\}_{\ell \in [\eta]}$ to $\mathcal{A}$. Adversary $\mathcal{A}$ then commits to a challenge point $x^* \in \{0,1\}^\rho \setminus (\mathcal{T} \cup \{\mathbf{0}\})$. Next, for $\ell \in [\eta]$, algorithm $\mathcal{B}$ samples trapdoor matrices $(\mathbf{D}^{(\ell)}, \mathsf{td}_{\mathbf{D}^{(\ell)}}) \leftarrow \mathsf{TrapGen}(1^\lambda)$. Algorithm $\mathcal{B}$ now constructs the components of the public parameters $\mathsf{pp}$ as follows for each $\ell \in [\eta]$:

   - $\mathbf{A}_j^{(\ell)} = \widehat{\mathbf{A}}_j^{(\ell)} - h_j^{(\ell)} \cdot \mathbf{G}$ for all $j \in [\rho]$,
   - $\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)} = \widehat{\mathbf{A}}_{\alpha,\beta}^{(\ell)} - d_{\alpha,\beta}^{(\ell)} \cdot \mathbf{G}$ for all $\alpha \in [n]$ and $\beta \in [m]$,
   - $\mathbf{B}_{i,j}^{(\ell)} = \widehat{\mathbf{B}}_{i,j}^{(\ell)} - x_j^* \cdot \mathbf{G}$ for all $i \in [t], j \in [\rho]$,
   - $\mathbf{C}_j^{(\ell)} = \widehat{\mathbf{C}}_j^{(\ell)} - x_j^* \cdot \mathbf{G}$ for all $j \in [\rho]$,
   - $\mathbf{V}^{(\ell)} = \widehat{\mathbf{V}}^{(\ell)}$,

   where $h_j^{(\ell)}$ denotes the $j^{\text{th}}$ bit of $h^{(\ell)}$, $d_{\alpha,\beta}^{(\ell)}$ denotes the $(\alpha,\beta)^{\text{th}}$ component of $\mathbf{D}^{(\ell)}$, and $x_j^*$ denote the $j^{\text{th}}$ component of $x^*$. Then, for $\ell \in [\eta]$, it computes

   - $\mathbf{A}_{h^{(\ell)}}^{(\ell)} \leftarrow \mathsf{EvalP}_{\mathsf{pk}}\Big( f_{h^{(\ell)}}^{\mathsf{eq}}, (\mathbf{A}_j^{(\ell)})_{j \in [\rho]}, (\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)})_{\alpha \in [n], \beta \in [m]} \Big)$,
   - $\mathbf{B}_{h^{(\ell)}}^{(\ell)} \leftarrow \mathsf{Eval}_{\mathsf{pk}}\Big( f_{h^{(\ell)}}^{\mathsf{con}}, (\mathbf{B}_{i,j}^{(\ell)})_{i \in [t], j \in [\rho]} \Big)$,
   - $\mathbf{C}_{h^{(\ell)}}^{(\ell)} \leftarrow \mathsf{Eval}_{\mathsf{pk}}\Big( f_{h^{(\ell)}}^{\mathsf{eq}}, (\mathbf{C}_j^{(\ell)})_{j \in [\rho]} \Big)$,

   and

   $$\mathbf{W}^{(\ell)} = \mathbf{A}_{h^{(\ell)}}^{(\ell)} + \mathbf{B}_{h^{(\ell)}}^{(\ell)} \mathbf{G}^{-1}(\mathbf{C}_{h^{(\ell)}}^{(\ell)}) \mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) + \mathbf{D}^{(\ell)} \in \mathbb{Z}_q^{n \times m}.$$

   Finally, it sets the public parameters $\mathsf{pp}$ to be

   $$\mathsf{pp} = \Big( \mathbf{W}^{(\ell)}, \big( \mathbf{A}_j^{(\ell)} \big)_{j \in [\rho]}, \big( \widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)} \big)_{\alpha \in [n], \beta \in [m]}, \big( \mathbf{B}_{i,j}^{(\ell)} \big)_{i \in [t], j \in [\rho]}, \big( \mathbf{C}_j^{(\ell)} \big)_{j \in [\rho]}, \mathbf{V}^{(\ell)} \Big)_{\ell \in [\eta]}$$

   and the trapdoor $\mathsf{td} = \big( h^{(\ell)}, \mathsf{td}_{\mathbf{D}^{(\ell)}} \big)_{\ell \in [\eta]}$.

3. **Common parameters for simulating PRF evaluations.** Before describing how $\mathcal{B}$ simulates the challenge value and the evaluation queries, we define some common notation. First, define the multi-set $S_{x^*}$ that consists of $t$ copies of $x^*$ (that is, the bitwise representation of $S_{x^*}$ is $x^* \| x^* \| \cdots \| x^* \in \{0,1\}^{t\rho}$). Next, for an input $x \in \{0,1\}^\rho$, define the following vectors and matrices:

$$\begin{aligned}
\mathbf{a}_x^{(\ell)} &\leftarrow \mathsf{EvalP}_{\mathsf{ct}}\big(f_x^{\mathsf{eq}}, h^{(\ell)}, \big(\mathbf{A}_j^{(\ell)}\big)_{j\in[\rho]}, \big(\widetilde{\mathbf{A}}_{\alpha,\beta}^{(\ell)}\big)_{\alpha\in[n],\beta\in[m]}, \big(\hat{\mathbf{a}}_j^{(\ell)}\big)_{j\in[\rho]}, \big(\hat{\mathbf{a}}_{\alpha,\beta}^{(\ell)}\big)_{\alpha\in[n],\beta\in[m]}\big),\\
\mathbf{b}_x^{(\ell)} &\leftarrow \mathsf{Eval}_{\mathsf{ct}}\big(f_x^{\mathsf{con}}, S_{x^*}, \big(\mathbf{B}_{i,j}^{(\ell)}\big)_{i\in[t],j\in[\rho]}, \big(\hat{\mathbf{b}}_{i,j}^{(\ell)}\big)_{i\in[t],j\in[\rho]}\big),\\
\mathbf{C}_x^{(\ell)} &\leftarrow \mathsf{Eval}_{\mathsf{pk}}\big(f_x^{\mathsf{eq}}, \big(\mathbf{C}_j^{(\ell)}\big)_{j\in[\rho]}\big),\\
\mathbf{c}_x^{(\ell)} &\leftarrow \mathsf{Eval}_{\mathsf{ct}}\big(f_x^{\mathsf{eq}}, x^*, \big(\mathbf{C}_j^{(\ell)}\big)_{j\in[\rho]}, \big(\hat{\mathbf{c}}_j^{(\ell)}\big)_{j\in[\rho]}\big).
\end{aligned} \tag{A.11}$$

In addition algorithm $\mathcal{B}$ first computes $\mathbf{a}_{h^{(\ell)}}^{(\ell)}$, $\mathbf{b}_{h^{(\ell)}}^{(\ell)}$, $\mathbf{C}_{h^{(\ell)}}^{(\ell)}$ for all $\ell \in [\eta]$ according to Eq. (A.11), and computes for all $\ell \in [\eta]$,

$$\mathbf{w}^{(\ell)} = \mathbf{a}_{h^{(\ell)}}^{(\ell)} + \mathbf{b}_{h^{(\ell)}}^{(\ell)} \mathbf{G}^{-1}(\mathbf{C}_{h^{(\ell)}}^{(\ell)}) \mathbf{G}^{-1}(\mathbf{V}^{(\ell)}). \tag{A.12}$$

4. **Simulating the challenge value.** To simulate the challenge evaluation $\mathbf{y}^*$, algorithm $\mathcal{B}$ computes $\mathbf{a}_{x^*}^{(\ell)}$, $\mathbf{b}_{x^*}^{(\ell)}$, $\mathbf{c}_{x^*}^{(\ell)}$ according to Eq. (A.11), and the vector

$$\mathbf{z}_{x^*}^{(\ell)} = \mathbf{a}_{x^*}^{(\ell)} + \mathbf{b}_{x^*}^{(\ell)} \mathbf{G}^{-1}(\mathbf{C}_{x^*}^{(\ell)}) \mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) - \mathbf{c}_{x^*}^{(\ell)} \mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) + \hat{\mathbf{v}}^{(\ell)}, \tag{A.13}$$

for all $\ell \in [\eta]$. It defines the vector

$$\mathbf{y}^* = \Big\lfloor (\mathbf{w}^{(1)} - \mathbf{z}_{x^*}^{(1)}) \mid \cdots \mid (\mathbf{w}^{(\eta)} - \mathbf{z}_{x^*}^{(\eta)}) \Big\rceil_p,$$

where the vectors $\mathbf{w}^{(\ell)}$ are defined in Eq. (A.12). Algorithm $\mathcal{B}$ gives $(\mathsf{pp}, \mathsf{td}, \mathbf{y}^*)$ to $\mathcal{A}$.

5. **Simulating the evaluation queries.** When the adversary $\mathcal{A}$ makes an evaluation query $x \in \{0,1\}^\rho \setminus (\mathcal{T} \cup \{\mathbf{0}\} \cup \{x^*\})$, algorithm $\mathcal{B}$ computes $\mathbf{a}_x^{(\ell)}, \mathbf{b}_x^{(\ell)}, \mathbf{C}_x^{(\ell)}$ according to Eq. (A.11) and the vector

$$\mathbf{z}_x^{(\ell)} = \mathbf{a}_x^{(\ell)} + \mathbf{b}_x^{(\ell)} \mathbf{G}^{-1}(\mathbf{C}_x^{(\ell)}) \mathbf{G}^{-1}(\mathbf{V}^{(\ell)}), \tag{A.14}$$

for all $\ell \in [\eta]$. It replies with

$$\mathbf{y}_x = \Big\lfloor (\mathbf{w}^{(1)} - \mathbf{z}_x^{(1)}) \mid \cdots \mid (\mathbf{w}^{(\eta)} - \mathbf{z}_x^{(\eta)}) \Big\rceil_p,$$

where the $\mathbf{w}^{(\ell)}$ are as defined in Eq. (A.12).

6. **Output of the experiment.** At the end of the experiment, algorithm $\mathcal{A}$ outputs a bit $b \in \{0,1\}$. Algorithm $\mathcal{B}$ simply echoes the bit $b$.

**Correctness of the simulation.** To complete the proof, we show that algorithm $\mathcal{B}$ correctly simulates the views of either $\textsc{hyb}_1$ or $\textsc{hyb}_2$ to $\mathcal{A}$ depending on whether the challenge vectors $\mathbf{u}_2 = (\hat{\mathbf{v}}^{(\ell)})_{\ell\in[\eta]} \in \mathbb{Z}_q^{m_2}$ consist of LWE samples or if they are uniformly random. First, by definition of $\mathsf{HybLWE}$, we can write the vectors in $\mathbf{u}_1$ (Eq. (A.10)) as follows:

$$\begin{array}{lllll}
\hat{\mathbf{a}}_j^{(\ell)} & = & \mathbf{s}\widehat{\mathbf{A}}_j^{(\ell)} + \mathbf{e}_{\mathbf{A},j}^{(\ell)} & = & \mathbf{s}(\mathbf{A}_j^{(\ell)} + h_j^{(\ell)} \cdot \mathbf{G}) + \mathbf{e}_{\mathbf{A},j}^{(\ell)} & \text{for all } j \in [\rho],\\
\hat{\mathbf{a}}_{\alpha,\beta}^{(\ell)} & = & \mathbf{s}\widehat{\mathbf{A}}_{\alpha,\beta}^{(\ell)} + \mathbf{e}_{\mathbf{A},\alpha,\beta}^{(\ell)} & = & \mathbf{s}(\mathbf{A}_{\alpha,\beta}^{(\ell)} + d_{\alpha,\beta}^{(\ell)} \cdot \mathbf{G}) + \mathbf{e}_{\mathbf{A},\alpha,\beta}^{(\ell)} & \text{for all } \alpha \in [n], \beta \in [m],\\
\hat{\mathbf{b}}_{i,j}^{(\ell)} & = & \mathbf{s}\widehat{\mathbf{B}}_{i,j}^{(\ell)} + \mathbf{e}_{\mathbf{B},i,j}^{(\ell)} & = & \mathbf{s}(\widehat{\mathbf{B}}_{i,j}^{(\ell)} + x_j^* \cdot \mathbf{G}) + \mathbf{e}_{\mathbf{B},i,j}^{(\ell)} & \text{for all } i \in [t], j \in [\rho],\\
\hat{\mathbf{c}}_j^{(\ell)} & = & \mathbf{s}\widehat{\mathbf{C}}_j^{(\ell)} + \mathbf{e}_{\mathbf{C},j}^{(\ell)} & = & \mathbf{s}(\mathbf{C}_j^{(\ell)} + x_j^* \cdot \mathbf{G}) + \mathbf{e}_{\mathbf{C},j}^{(\ell)} & \text{for all } j \in [\rho],
\end{array}$$

for some $\mathbf{s} \in \mathbb{Z}_q^n$ and error vectors $\mathbf{e}_{\mathbf{A},j}^{(\ell)}$, $\mathbf{e}_{\mathbf{A},\alpha,\beta}^{(\ell)}$, $\mathbf{e}_{\mathbf{B},i,j}^{(\ell)}$, and $\mathbf{e}_{\mathbf{C},j}^{(\ell)}$ sampled from $\chi^m$. By Theorems 3.6 and 3.7, and the fact that $f_{h^{(\ell)}}^{\mathsf{eq}}(h^{(\ell)}) = 1$, $f_{h^{(\ell)}}^{\mathsf{con}}(S_{x^*}) = 0$, the vectors $\mathbf{a}_{h^{(\ell)}}^{(\ell)}$ and $\mathbf{b}_{h^{(\ell)}}^{(\ell)}$ from Eq. (A.11) satisfy the following:

$$\begin{aligned}
\mathbf{a}_{h^{(\ell)}}^{(\ell)} &= \mathbf{s}\big(\mathbf{A}_{h^{(\ell)}}^{(\ell)} + f_{h^{(\ell)}}^{\mathsf{eq}}(h^{(\ell)}) \cdot \mathbf{D}\big) + \mathbf{e}_{\mathbf{a},h^{(\ell)}}^{(\ell)} = \mathbf{s}\big(\mathbf{A}_{h^{(\ell)}}^{(\ell)} + \mathbf{D}\big) + \mathbf{e}_{\mathbf{a},h^{(\ell)}}^{(\ell)}, \\
\mathbf{b}_{h^{(\ell)}}^{(\ell)} &= \mathbf{s}\big(\mathbf{B}_{h^{(\ell)}}^{(\ell)} + f_{h^{(\ell)}}^{\mathsf{con}}(S_{x^*}) \cdot \mathbf{G}\big) + \mathbf{e}_{\mathbf{b},h^{(\ell)}}^{(\ell)} = \mathbf{s}\mathbf{B}_{h^{(\ell)}}^{(\ell)} + \mathbf{e}_{\mathbf{b},h^{(\ell)}}^{(\ell)},
\end{aligned}$$

for all $\ell \in [\eta]$ where $\big\|\mathbf{e}_{\mathbf{a},h^{(\ell)}}^{(\ell)}\big\|, \big\|\mathbf{e}_{\mathbf{b},h^{(\ell)}}^{(\ell)}\big\| \leq B \cdot m^{O(d)} = m^{O(\log \lambda)}$ since $d = O(\log \lambda)$. Substituting into Eq. (A.12), this means that in the simulation,

$$\begin{aligned}
\mathbf{w}^{(\ell)} &= \mathbf{a}_{h^{(\ell)}}^{(\ell)} + \mathbf{b}_{h^{(\ell)}}^{(\ell)} \mathbf{G}^{-1}(\mathbf{C}_{h^{(\ell)}}^{(\ell)}) \mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) \\
&= \mathbf{s}\big(\mathbf{A}_{h^{(\ell)}}^{(\ell)} + \mathbf{D} + \mathbf{B}_{h^{(\ell)}}^{(\ell)} \mathbf{G}^{-1}(\mathbf{C}_{h^{(\ell)}}^{(\ell)}) \mathbf{G}^{-1}(\mathbf{V}^{(\ell)})\big) + \mathbf{e}_{\mathbf{w}}^{(\ell)} \\
&= \mathbf{s}\mathbf{W}^{(\ell)} + \mathbf{e}_{\mathbf{w}}^{(\ell)} \qquad \text{(A.15)}
\end{aligned}$$

for all $\ell \in [\eta]$ and where $\|\mathbf{e}_{\mathbf{w}}^{(\ell)}\| \leq B \cdot m^{O(\log \lambda)}$. Now, we consider each component in the simulation separately:

- **Public parameters and trapdoor.** In the $\mathsf{HybLWE}$ security game, the matrix $\mathbf{A}_1$ is sampled uniformly at random from $\mathbb{Z}_q^{n \times m_1}$. Thus, $\mathcal{B}$ constructs the public parameters $\mathsf{pp}$ and trapdoor $\mathsf{td}$ using the same procedure as that in $\text{HYB}_1$ and $\text{HYB}_2$.
- **Secret PRF key.** The LWE secret $\mathbf{s}$ (chosen by the $\mathsf{HybLWE}$ challenger) plays the role of the PRF key in the simulation. Note that the $\mathsf{HybLWE}$ challenger samples $\mathbf{s}$ from the same distribution as the $\mathsf{SampleKey}$ algorithm used in $\text{HYB}_1$ and $\text{HYB}_2$.
- **Evaluation queries.** Suppose the adversary makes an evaluation query on an input $x \in \{0,1\}^\rho \setminus (\mathcal{T} \cup \{\mathbf{0}\} \cup \{x^*\})$. In $\text{HYB}_1$ and $\text{HYB}_2$, the challenger would reply with the real evaluation
$$\bar{\mathbf{y}}_x = \mathsf{Eval}(\mathsf{pp}, \mathbf{s}, x) = \Big\lfloor \mathbf{s}\big(\mathbf{W}^{(1)} - \mathbf{Z}_x^{(1)} \mid \cdots \mid \mathbf{W}^{(\eta)} - \mathbf{Z}_x^{(\eta)}\big) \Big\rceil_p,$$
where $\mathbf{W}^{(\ell)}$ is the matrix in the public parameters $\mathsf{pp}$ and $\mathbf{Z}_x^{(\ell)}$ is as defined in Eq. (4.3) for all $\ell \in [\eta]$. We show that with overwhelming probability, algorithm $\mathcal{B}$ replies with the same value. By Theorems 3.6 and 3.7 and the fact that $f_x^{\mathsf{eq}}\big(h^{(\ell)}\big) = 0 = f_x^{\mathsf{con}}(S_{x^*})$, the vectors $\mathbf{a}_x^{(\ell)}$ and $\mathbf{b}_x^{(\ell)}$ from Eq. (A.11) satisfy the following relations:
$$\begin{aligned}
\mathbf{a}_x^{(\ell)} &= \mathbf{s}\big(\mathbf{A}_{h^{(\ell)}}^{(\ell)} + f_x^{\mathsf{eq}}(h^{(\ell)}) \cdot \mathbf{D}\big) + \mathbf{e}_{\mathbf{a},x}^{(\ell)} = \mathbf{s}\mathbf{A}_{h^{(\ell)}}^{(\ell)} + \mathbf{e}_{\mathbf{a},x}^{(\ell)}, \\
\mathbf{b}_x^{(\ell)} &= \mathbf{s}(\mathbf{B}_{h^{(\ell)}}^{(\ell)} + f_x^{\mathsf{con}}(S_{x^*}) \cdot \mathbf{G}) + \mathbf{e}_{\mathbf{b},x}^{(\ell)} = \mathbf{s}\mathbf{B}_{h^{(\ell)}}^{(\ell)} + \mathbf{e}_{\mathbf{b},x}^{(\ell)},
\end{aligned}$$
for error vectors $\|\mathbf{e}_{\mathbf{a},x}^{(\ell)}\|, \|\mathbf{e}_{\mathbf{b},x}^{(\ell)}\| \leq B \cdot m^{O(\log \lambda)}$. Substituting into Eq. (A.14), this means that in the simulation,
$$\begin{aligned}
\mathbf{z}_x^{(\ell)} &= \mathbf{a}_x^{(\ell)} + \mathbf{b}_x^{(\ell)} \mathbf{G}^{-1}(\mathbf{C}_x^{(\ell)}) \mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) \\
&= \mathbf{s}\big(\mathbf{A}_x^{(\ell)} + \mathbf{B}_x^{(\ell)} \mathbf{G}^{-1}(\mathbf{C}_x^{(\ell)}) \mathbf{G}^{-1}(\mathbf{V}^{(\ell)})\big) + \mathbf{e}_{\mathbf{z},x}^{(\ell)} \\
&= \mathbf{s}\mathbf{Z}_x^{(\ell)} + \mathbf{e}_{\mathbf{z},x}^{(\ell)},
\end{aligned}$$

using the definition of $\mathbf{Z}_x^{(\ell)}$ from Eq. (4.3) and where $\|\mathbf{e}_{\mathbf{z},x}^{(\ell)}\| \leq B \cdot m^{O(\log \lambda)}$. Thus, for all $\ell \in [\eta]$

$$\mathbf{w}^{(\ell)} - \mathbf{z}_x^{(\ell)} = \mathbf{s}\big(\mathbf{W}^{(\ell)} - \mathbf{Z}_x^{(\ell)}\big) + \hat{\mathbf{e}}^{(\ell)},$$

using the relation for $\mathbf{w}^{(\ell)}$ from Eq. (A.15) and where $\|\hat{\mathbf{e}}^{(\ell)}\| \leq B \cdot m^{O(\log \lambda)}$. Thus, the simulated evaluation $\mathbf{y}_x$ satisfies the following:

$$\begin{aligned} \mathbf{y}_x &= \left\lfloor (\mathbf{w}^{(1)} - \mathbf{z}_x^{(1)}) \mid \cdots \mid (\mathbf{w}^{(\eta)} - \mathbf{z}_x^{(\eta)}) \right\rceil_p \\ &= \left\lfloor \mathbf{s}\big(\mathbf{W}^{(1)} - \mathbf{Z}_x^{(1)}\big) + \hat{\mathbf{e}}^{(1)} \mid \cdots \mid \mathbf{s}\big(\mathbf{W}^{(\eta)} - \mathbf{Z}_x^{(\eta)}\big) + \hat{\mathbf{e}}^{(\eta)} \right\rceil_p \end{aligned}$$

This means that the simulated value $\mathbf{y}_x$ and the real evaluation $\overline{\mathbf{y}}_x$ are identical as long as $\overline{\mathbf{y}}_x \notin \mathsf{Borderline}_E$ for $E = B \cdot m^{O(\log \lambda)}$. By Lemma A.8, $\overline{\mathbf{y}}_x \notin \mathsf{Borderline}_E$ with overwhelming probability. Thus, the evaluation queries are correctly simulated with overwhelming probability.

- **Challenge value.** By Theorems 3.6 and 3.7 and the fact that $f_{x^*}^{\mathsf{eq}}\big(h^{(\ell)}\big) = 0$, $f_{x^*}^{\mathsf{con}}(S_{x^*}) = 1$, and $f_{x^*}^{\mathsf{eq}}\big(x^*\big) = 1$, the vectors $\mathbf{a}_{x^*}^{(\ell)}$, $\mathbf{b}_{x^*}^{(\ell)}$, $\mathbf{c}_{x^*}^{(\ell)}$ satisfy the following relations:

$$\begin{aligned} \mathbf{a}_{x^*}^{(\ell)} &= \mathbf{s}(\mathbf{A}_{x^*}^{(\ell)} + f_{x^*}^{\mathsf{eq}}(h^{(\ell)}) \cdot \mathbf{D}) + \mathbf{e}_{\mathbf{a},x^*}^{(\ell)} = \mathbf{s}\mathbf{A}_{x^*}^{(\ell)} + \mathbf{e}_{\mathbf{a},x^*}^{(\ell)} \\ \mathbf{b}_{x^*}^{(\ell)} &= \mathbf{s}(\mathbf{B}_{x^*}^{(\ell)} + f_{x^*}^{\mathsf{con}}(S_{x^*}) \cdot \mathbf{G}) + \mathbf{e}_{\mathbf{b},x^*}^{(\ell)} = \mathbf{s}(\mathbf{B}_{x^*}^{(\ell)} + \mathbf{G}) + \mathbf{e}_{\mathbf{b},x^*}^{(\ell)} \\ \mathbf{c}_{x^*}^{(\ell)} &= \mathbf{s}(\mathbf{C}_{x^*}^{(\ell)} + f_{x^*}^{\mathsf{eq}}(x^*) \cdot \mathbf{G}) + \mathbf{e}_{\mathbf{c},x^*}^{(\ell)} = \mathbf{s}(\mathbf{C}_{x^*}^{(\ell)} + \mathbf{G}) + \mathbf{e}_{\mathbf{c},x^*}^{(\ell)} \end{aligned}$$

for error vectors $\|\mathbf{e}_{\mathbf{a},x^*}^{(\ell)}\|, \|\mathbf{e}_{\mathbf{b},x^*}^{(\ell)}\|, \|\mathbf{e}_{\mathbf{c},x^*}^{(\ell)}\| \leq B \cdot m^{O(\log \lambda)}$. Substituting into Eq. (A.13), this means that in the simulation,

$$\begin{aligned} \mathbf{z}_{x^*}^{(\ell)} &= \mathbf{a}_{x^*}^{(\ell)} + \mathbf{b}_{x^*}^{(\ell)}\mathbf{G}^{-1}(\mathbf{C}_{x^*}^{(\ell)})\mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) - \mathbf{c}_{x^*}^{(\ell)}\mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) + \hat{\mathbf{v}}^{(\ell)} \\ &= \mathbf{s}\big(\mathbf{A}_{x^*}^{(\ell)} + \mathbf{B}_{x^*}^{(\ell)}\mathbf{G}^{-1}(\mathbf{C}_{x^*}^{(\ell)})\mathbf{G}^{-1}(\mathbf{V}^{(\ell)}) - \mathbf{V}^{(\ell)}\big) + \hat{\mathbf{v}}^{(\ell)} + \mathbf{e}_{\mathbf{z},x^*}^{(\ell)} \\ &= \mathbf{s}\big(\mathbf{Z}_{x^*}^{(\ell)} - \mathbf{V}^{(\ell)}\big) + \hat{\mathbf{v}}^{(\ell)} + \mathbf{e}_{\mathbf{z},x^*}^{(\ell)} \end{aligned}$$

using the definition of $\mathbf{Z}_{x^*}^{(\ell)}$ from Eq. (4.3) and where $\|\mathbf{e}_{\mathbf{z},x^*}^{(\ell)}\| \leq B \cdot m^{O(\log \lambda)}$. Thus, for all $\ell \in [\eta]$,

$$\mathbf{w}^{(\ell)} - \mathbf{z}_{x^*}^{(\ell)} = \mathbf{s}\big(\mathbf{W}^{(\ell)} - \mathbf{Z}_{x^*}^{(\ell)} + \mathbf{V}^{(\ell)}\big) - \hat{\mathbf{v}}^{(\ell)} + \hat{\mathbf{e}}^{(\ell)}, \tag{A.16}$$

using the relation for $\mathbf{w}^{(\ell)}$ from Eq. (A.15) and where $\big\|\hat{\mathbf{e}}^{(\ell)}\big\| \leq B \cdot m^{O(\log \lambda)}$. We now consider the behavior of the challenger in HYB$_1$ and HYB$_2$ and show that if the vectors $\hat{\mathbf{v}}^{(\ell)}$ are LWE samples, then $\mathcal{B}$ correctly simulates HYB$_1$ for $\mathcal{A}$ (with overwhelming probability); otherwise, if the vectors $\mathbf{v}^{(\ell)}$ are uniformly random, then $\mathcal{B}$ correctly simulates HYB$_2$ for $\mathcal{A}$.

  - In HYB$_1$, the challenger answers the challenge query using $\mathsf{Eval}(\mathsf{pp}, \mathbf{s}, x^*)$ which computes and outputs the following:

$$\overline{\mathbf{y}}^* = \mathsf{Eval}(\mathsf{pp}, \mathbf{s}, x^*) = \left\lfloor \mathbf{s}\big(\mathbf{W}^{(1)} - \mathbf{Z}_{x^*}^{(1)} \mid \cdots \mid \mathbf{W}^{(\eta)} - \mathbf{Z}_{x^*}^{(\eta)}\big) \right\rceil_p .$$

    Suppose $\hat{\mathbf{v}}^{(\ell)} = \mathbf{s}\mathbf{V}^{(\ell)} + \mathbf{e}_{\mathbf{v}}^{(\ell)}$ where $\mathbf{e}_{\mathbf{v}}^{(\ell)}$ is sampled from $\chi^m$. In this case, Eq. (A.16) becomes

$$\mathbf{w}^{(\ell)} - \mathbf{z}_{x^*}^{(\ell)} = \mathbf{s}\big(\mathbf{W}^{(\ell)} - \mathbf{Z}_{x^*}^{(\ell)}\big) + (\hat{\mathbf{e}}^{(\ell)} - \mathbf{e}_{\mathbf{v}}^{(\ell)}),$$

and the simulated challenge evaluation has the form

$$\begin{aligned}\mathbf{y}^* &= \left\lfloor (\mathbf{w}^{(1)} - \mathbf{z}_{x^*}^{(1)}) \mid \cdots \mid (\mathbf{w}^{(\eta)} - \mathbf{z}_{x^*}^{(\eta)}) \right\rceil_p \\ &= \left\lfloor \mathbf{s}\big(\mathbf{W}^{(1)} - \mathbf{Z}_{x^*}^{(1)}\big) + (\hat{\mathbf{e}}^{(1)} - \mathbf{e}_{\mathbf{v}}^{(1)}) \mid \cdots \mid \mathbf{s}\big(\mathbf{W}^{(\eta)} - \mathbf{Z}_{x^*}^{(\eta)}\big) + (\hat{\mathbf{e}}^{(\eta)} - \mathbf{e}_{\mathbf{v}}^{(\eta)}) \right\rceil_p\end{aligned}$$

Thus, the simulated value $\mathbf{y}^*$ and the real evaluation $\overline{\mathbf{y}}^*$ are identical as long as $\overline{\mathbf{y}}^* \notin \mathsf{Borderline}_E$ for $E = B \cdot m^{O(\log \lambda)}$. By Lemma A.8, $\overline{\mathbf{y}}^* \notin \mathsf{Borderline}_E$ with overwhelming probability, and so algorithm $\mathcal{B}$ correctly simulates the challenge evaluation according to the specification of $\textsc{hyb}_1$ if the vectors $\mathbf{v}^{(\ell)}$ are LWE samples.

– In $\textsc{hyb}_2$, the challenger answers the evaluation query by sampling $\overline{\mathbf{y}}^* \xleftarrow{\text{R}} \mathbb{Z}_p^{\eta m}$. If each $\hat{\mathbf{v}}^{(\ell)}$ is a uniformly random vector over $\mathbb{Z}_q^m$ (and independent of all of the other components), then according to Eq. (A.16), $\mathbf{w}^{(\ell)} - \mathbf{z}_{x^*}^{(\ell)}$ is also uniformly random over $\mathbb{Z}_q^m$ for all $\ell \in [\eta]$. This means that algorithm $\mathcal{B}$'s response $\mathbf{y}^*$ is uniformly random over $\mathbb{Z}_p^{\eta m}$, in which case $\mathcal{B}$ perfectly simulates the distribution in $\textsc{hyb}_2$.

By the above analysis, we have shown that if the vectors $\mathbf{v}^{(\ell)}$ consist of valid LWE samples, then $\mathcal{B}$ correctly simulates $\textsc{hyb}_1$ for $\mathcal{A}$ (up to negligible error) and if they are uniformly random, then $\mathcal{B}$ correctly simulates $\textsc{hyb}_2$ for $\mathcal{A}$ (up to negligible error). Thus, if the outputs of $\textsc{hyb}_1(\mathcal{A})$ and $\textsc{hyb}_2(\mathcal{A})$ are noticeably different, then $\mathcal{B}$ breaks the $\mathsf{HybLWE}_{n,m_1,m_2,q,\chi}$ with noticeable probability. □

**Lemma A.20.** *For all adversaries $\mathcal{A}$, $|\Pr[\textsc{hyb}_2(\mathcal{A}) = 1] - \Pr[\textsc{hyb}_3(\mathcal{A}) = 1]| = 0$.*

*Proof.* Same as the proof of Lemma A.18. □

Combining Lemmas A.18 through A.20, construction $\Pi_{\mathsf{PRF}}$ satisfies $T$-restricted pseudorandomness for $T = \eta$. □

## A.7 Proof of Theorem 4.26 (Robust Extractability)

By inspection, the $\mathsf{TestCandidate}$ algorithm in $\Pi_{\mathsf{EPRF}}$ satisfies the generalized candidate testing properties from Remark 4.10. It thus suffices to show that $\Pi_{\mathsf{EPRF}}$ satisfies robust extractability. First, we show that the $\mathsf{TestCandidate}$ algorithm for $\Pi_{\mathsf{EPRF}}$ satisfies the requirements for robust extractability. Then, we show that the output of the real extraction experiment $\mathsf{ExtReal}_{\mathcal{A}}(\lambda, \varepsilon_1, \varepsilon_2)$ is statistically indistinguishable from the output of the ideal extraction algorithm $\mathsf{ExtIdeal}_{\mathcal{A}}(\lambda, \varepsilon_1, \varepsilon_2)$.

**Statistical properties of $\mathsf{TestCandidate}$.** Both properties follow from Hoeffding's inequality. Take $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$, any key $\mathbf{s} \in [-B, B]^n$ and any circuit $C \colon \{0,1\}^\rho \to \mathbb{Z}_p^{\eta m}$.

- Suppose $C(\cdot) \sim_{\varepsilon_1} \mathsf{Eval}(\mathsf{pp}, \mathbf{s}, \cdot)$. Then

  $$\Pr[x_i \xleftarrow{\text{R}} \{0,1\}^\rho : C(x_i) \neq \mathsf{Eval}(\mathsf{pp}, \mathbf{s}, x_i)] \leq \varepsilon_1 = \varepsilon - \delta.$$

  Let $N_{\mathbf{s}}$ be the number of indices $i \in [\xi]$ where $C(x_i^*) \neq \mathsf{Eval}(\mathsf{pp}, \mathbf{s}, x_i^*)$, $\xi = \lambda/\delta^2$, and $x_1^*, \ldots, x_\xi^* \xleftarrow{\text{R}} \{0,1\}^\rho$. By Hoeffding's inequality,

  $$\Pr[N_{\mathbf{s}} > \varepsilon\xi] \leq \Pr[|N_{\mathbf{s}} - (\varepsilon - \delta)\xi| > \delta\xi] \leq 2^{-\Omega(\delta^2\xi)} = 2^{-\Omega(\lambda)} = \mathsf{negl}(\lambda).$$

  Thus, in this case, $\mathsf{TestCandidate}$ outputs 1 with probability $1 - \mathsf{negl}(\lambda)$.

- Suppose $C(\cdot) \not\sim_{\varepsilon_2} \mathsf{Eval}(\mathsf{pp}, \mathbf{s}, \cdot)$. Then

  $$\Pr[x_i \xleftarrow{\mathrm{R}} \{0,1\}^\rho : C(x_i) \neq \mathsf{Eval}(\mathsf{pp}, \mathbf{s}, x_i)] > \varepsilon_2 = \varepsilon + \delta.$$

  Again by Hoeffding's inequality,

  $$\Pr[N_{\mathbf{s}} \leq \varepsilon\xi] = \Pr[|N_{\mathbf{s}} - (\varepsilon + \delta)\xi| \geq \delta\xi] \leq 2^{-\Omega(\delta^2\xi)} = 2^{-\Omega(\lambda)} = \mathsf{negl}(\lambda).$$

  Thus, in this case, $\mathsf{TestCandidate}$ outputs 1 with negligible probability.

**Indistinguishability of $\mathsf{ExtReal}$ and $\mathsf{ExtIdeal}$.** We proceed via a sequence of hybrid experiments between an (unbounded) adversary $\mathcal{A}$, and an (unbounded) challenger.

- $\textsc{hyb}_0$: This is the real extractability experiment $\mathsf{ExtReal}_{\mathcal{A}}(\lambda, \varepsilon_1, \varepsilon_2)$ from Definition 4.9. Specifically, the challenger proceeds in each phase of the experiment as follows:
  - **Setup phase:** The challenger samples $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PrmsGen}(1^\lambda)$ and gives $\mathsf{pp}$ to $\mathcal{A}$.
  - **Query phase:** Whenever $\mathcal{A}$ makes an extraction query on a circuit $C \colon \{0,1\}^\rho \to \mathbb{Z}_p^{\eta m}$, the challenger replies with $\mathsf{Extract}(\mathsf{pp}, \mathsf{td}, C)$.
  - **Output phase:** At the end of the experiment, the adversary $\mathcal{A}$ outputs a bit $b \in \{0,1\}$, which is also the output of the experiment.

- $\textsc{hyb}_1$: Same as $\textsc{hyb}_0$, except the challenger implements the $\mathsf{ExtractCandidates}$ sub-algorithm (used by $\mathsf{Extract}$) using the following (inefficient) algorithm that does *not* rely on the lattice trapdoors $\mathsf{td}_{\mathbf{D}^{(\ell)}}$ in $\mathsf{td}$:
  - $\mathsf{ExtractCandidates}(\mathsf{pp}, \mathsf{td}, C)$: On input the public parameters $\mathsf{pp}$ (as specified in Eq. (4.2)), a trapdoor $\mathsf{td} = \big(h^{(\ell)}, \mathsf{td}_{\mathbf{D}^{(\ell)}}\big)_{\ell \in [\eta]}$, and a circuit $C \colon \{0,1\}^\rho \to \mathbb{Z}_p^{\eta m}$, the candidate extraction algorithm first evaluates the circuit $C$ on the test points $h^{(\ell)}$ to obtain $(\mathbf{y}_1^{(\ell)} \mid \cdots \mid \mathbf{y}_\eta^{(\ell)}) \leftarrow C(h^{(\ell)})$ for all $\ell \in [\eta]$. For each $\ell \in [\eta]$, if there exists a unique $\mathbf{s} \in [-B, B]^n$ such that

    $$\mathbf{y}_\ell^{(\ell)} = \Big\lfloor \mathbf{s}\big(\mathbf{W}^{(\ell)} - \mathbf{Z}_{h^{(\ell)}}^{(\ell)}\big) \Big\rceil_p, \tag{A.17}$$

    where $\mathbf{Z}_{h^{(\ell)}}^{(\ell)}$ is defined according to Eq. (4.3), then set $\mathbf{s}^{(\ell)} = \mathbf{s}$, and otherwise, set $\mathbf{s}^{(\ell)} = \perp$. Output the set of all $\mathbf{s}^{(\ell)}$ where $\mathbf{s}^{(\ell)} \neq \perp$.

- $\textsc{hyb}_2$: Same as $\textsc{hyb}_1$, except during the setup phase, the challenger samples the trapdoor matrices $\mathbf{D}^{(\ell)} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$ in the public parameters $\mathsf{pp}$ uniformly at random for all $\ell \in [\eta]$. It leaves the trapdoors $\mathsf{td}_{\mathbf{D}^{(\ell)}}$ unspecified (since nothing in the experiment depends on them).

- $\textsc{hyb}_3$: Same as $\textsc{hyb}_2$ except the challenger implements the $\mathsf{ExtractCandidates}$ sub-algorithm (used by $\mathsf{Extract}$) using the following (inefficient) algorithm that does *not* depend on any trapdoor $\mathsf{td}$:
  - $\mathsf{ExtractCandidates}(\mathsf{pp}, C)$: On input the public parameters $\mathsf{pp}$ (as specified in Eq. (4.2)), and a circuit $C \colon \{0,1\}^\rho \to \mathbb{Z}_p^{\eta m}$, check if there exists a unique key $\mathbf{s} \in [-B, B]^n$ such that $C(\cdot) \sim_{\varepsilon_2} \mathsf{Eval}(\mathsf{pp}, \mathbf{s}, \cdot)$. If so, then output the singleton set $\{\mathbf{s}\}$; otherwise, output the empty set $\varnothing$.

  By design, this is the ideal extractability experiment $\mathsf{ExtIdeal}_{\mathcal{A}}(\lambda, \varepsilon_1, \varepsilon_2)$.

**Lemma A.21.** *Suppose the conditions in Theorem 3.5 hold, $m \geq 2n \log q$, and $\lceil q/p \rceil \leq q/4$. Then, for all adversaries $\mathcal{A}$ that makes $Q = \mathsf{poly}(\lambda)$ extraction queries, we have $\left| \Pr[\textsc{hyb}_0(\mathcal{A}) = 1] - \Pr[\textsc{hyb}_1(\mathcal{A}) = 1] \right| = \mathsf{negl}(\lambda)$.*

*Proof.* The only difference between the experiments $\textsc{hyb}_0$ and $\textsc{hyb}_1$ is in the way the challenger implements the $\mathsf{ExtractCandidates}$ algorithm.

- In $\textsc{hyb}_0$, the $\mathsf{ExtractCandidates}$ algorithm computes $\mathbf{s}^{(\ell)} \leftarrow \mathsf{Invert}(\mathsf{td}_{\mathbf{D}^{(\ell)}}, \mathbf{y}_\ell^{(\ell)})$ for each $\ell \in [\eta]$, where the values $(\mathbf{y}_1^{(\ell)} \mid \cdots \mid \mathbf{y}_\eta^{(\ell)})$ are derived from $C(h^{(\ell)})$.
- In $\textsc{hyb}_1$, the $\mathsf{ExtractCandidates}$ algorithm computes $(\mathbf{y}_1^{(\ell)} \mid \cdots \mid \mathbf{y}_\eta^{(\ell)}) \leftarrow C(h^{(\ell)})$ as above, and then tries to finds a key $\mathbf{s} \in [-B, B]^n$ where

$$\mathbf{y}_\ell^{(\ell)} = \left\lfloor \mathbf{s}\big(\mathbf{W}^{(\ell)} - \mathbf{Z}_{h^{(\ell)}}^{(\ell)}\big) \right\rceil_p = \left\lfloor \mathbf{s} \cdot \mathbf{D}^{(\ell)} \right\rceil_p .$$

  If such a vector exists, then it sets $\mathbf{s}^{(\ell)} = \mathbf{s}$, and otherwise, it sets $\mathbf{s}^{(\ell)} = \perp$.

The rest of the logic is identical in $\textsc{hyb}_0$ and $\textsc{hyb}_1$. Hence, it suffices to show that the distribution of the vectors $\mathbf{s}^{(\ell)}$ for $\ell \in [\eta]$ in the two experiments are statistically indistinguishable. By construction, $\mathbf{D}^{(\ell)} = \mathbf{W}^{(\ell)} - \mathbf{Z}_{h^{(\ell)}}^{(\ell)}$ for all $\ell \in [\eta]$ in both experiments. By Theorem 3.5, if there exists a vector $\mathbf{s} \in [-B, B]^n$ for which $\left\lfloor \mathbf{s} \cdot \mathbf{D}^{(\ell)} \right\rceil_p = \mathbf{y}_\ell^{(\ell)}$, the challenger in $\textsc{hyb}_0$ will set $\mathbf{s}^{(\ell)} = \mathbf{s}$ with overwhelming probability. If no such vector $\mathbf{s}$ exists, then it will set $\mathbf{s}^{(\ell)} = \perp$ with overwhelming probability. Again by Theorem 3.5, the matrices $\mathbf{D}^{(\ell)}$ are statistically close to uniform over $\mathbb{Z}_q^{n \times m}$, so by Lemma A.6, if there exists $\mathbf{s} \in [-B, B]^n$ such that $\left\lfloor \mathbf{s} \cdot \mathbf{D}^{(\ell)} \right\rceil_p = \mathbf{y}_\ell^{(\ell)}$, then $\mathbf{s}$ is *unique* with overwhelming probability. Thus, with overwhelming probability, the output of $\mathsf{ExtractCandidates}$ in $\textsc{hyb}_0$ and $\textsc{hyb}_1$ will be identical. The lemma then follows by a union bound over the number of times $Q = \mathsf{poly}(\lambda)$ the $\mathsf{Extract}$ algorithm is invoked in the two experiments. □

**Lemma A.22.** *Suppose the conditions in Theorem 3.5 hold. Then, for all adversaries $\mathcal{A}$, we have that $\left| \Pr[\textsc{hyb}_1(\mathcal{A}) = 1] - \Pr[\textsc{hyb}_2(\mathcal{A}) = 1] \right| = \mathsf{negl}(\lambda)$.*

*Proof.* Follows immediately from Theorem 3.5. □

**Lemma A.23.** *Suppose $\Pi_{\mathsf{EPRF}}$ satisfies key-injectivity (Definition 4.11), $\eta = \omega(\log \lambda)$, $m \geq 2n \log q$, $\lceil q/p \rceil \leq q/4$, and $\varepsilon_2 < 1/2$. Then, for all adversaries $\mathcal{A}$ that makes $Q = \mathsf{poly}(\lambda)$ extraction queries, we have that $\left| \Pr[\textsc{hyb}_2(\mathcal{A}) = 1] - \Pr[\textsc{hyb}_3(\mathcal{A}) = 1] \right| = \mathsf{negl}(\lambda)$.*

*Proof.* We use a hybrid argument. First, let $\textsc{hyb}_{2,0} \equiv \textsc{hyb}_2$. Then, for $i \in [Q]$, we define $\textsc{hyb}_{2,i}$ to be the experiment where the first $i$ extraction queries the adversary makes are answered according according to the specification in $\textsc{hyb}_2$ and the remaining $Q - i$ queries are answered according to the specification in $\textsc{hyb}_3$. By construction, $\textsc{hyb}_{2,Q} \equiv \textsc{hyb}_3$. We now show that for all $i \in [Q]$, $\textsc{hyb}_{2,i-1} \overset{s}{\approx} \textsc{hyb}_{2,i}$. It suffices to consider the $i^{\text{th}}$ extraction query the adversary $\mathcal{A}$ makes. Let $C$ be the circuit $\mathcal{A}$ submits to the challenger on its $i^{\text{th}}$ query. We consider two cases:

**Case 1.** Suppose there exists a key $\mathbf{s}^* \in [-B, B]^n$ where $C(\cdot) \sim_{\varepsilon_2} \mathsf{Eval}(\mathsf{pp}, \mathbf{s}^*, \cdot)$. Since $\Pi_{\mathsf{EPRF}}$ satisfies key-injectivity and $\varepsilon_2 < 1/2$, we can appeal to Lemma 4.12 to argue that $\mathbf{s}^*$ is in fact unique (with overwhelming probability). Consider the challenger's behavior in $\textsc{hyb}_{2,i-1}$ and $\textsc{hyb}_{2,i}$:

- In $\textsc{hyb}_{2,i}$, the $\mathsf{ExtractCandidates}$ algorithm will output $\{\mathbf{s}^*\}$. In this case, the challenger then computes $b_{\mathbf{s}^*} \leftarrow \mathsf{TestCandidate}(\mathsf{pp}, C, \mathbf{s}^*)$. The challenger replies to the extraction query with $\perp$ if $b_{\mathbf{s}^*} = 0$ and $\mathbf{s}^*$ if $b_{\mathbf{s}^*} = 1$.

- In $\textsc{hyb}_{2,i-1}$, the $\mathsf{ExtractCandidates}$ algorithm first computes $(\mathbf{y}_1^{(\ell)} \mid \cdots \mid \mathbf{y}_\eta^{(\ell)}) \leftarrow C(h^{(\ell)})$ for all $\ell \in [\eta]$, and defines $\mathbf{s}^{(\ell)} \in [-B, B]^n$ to be the unique vector (if there is one) where

  $$\mathbf{y}_\ell^{(\ell)} = \left\lfloor \mathbf{s}^{(\ell)} (\mathbf{W}^{(\ell)} - \mathbf{Z}_{h^{(\ell)}}^{(\ell)}) \right\rceil_p = \left\lfloor \mathbf{s}^{(\ell)} \cdot \mathbf{D}^{(\ell)} \right\rceil_p .$$

  Note that if $\mathbf{s}^{(\ell)}$ exists, then it is unique (with overwhelming probability) by Lemma A.6. We now proceed as follows:

  - First, we show that with overwhelming probability, there exists some $\ell \in [\eta]$ such that $C(h^{(\ell)}) = \mathsf{Eval}(\mathsf{pp}, \mathsf{s}^*, h^{(\ell)})$. By construction of $\textsc{hyb}_{2,i-1}$, the view of the adversary prior to its $i^{\text{th}}$ extraction query is entirely independent of $h^{(\ell)}$ for all $\ell \in [\eta]$. This means that the challenger can sample $h^{(\ell)} \xleftarrow{\text{R}} \{0,1\}^\rho$ *after* the adversary has submitted its $i^{\text{th}}$ extraction query $C$. Since $C(\cdot) \sim_{\varepsilon_2} \mathsf{Eval}(\mathsf{pp}, \mathsf{s}^*, \cdot)$, and all of the points $h^{(\ell)}$ are sampled uniformly and independently from $\{0,1\}^\rho$,

    $$\Pr[\forall \ell \in [\eta] : C(h^{(\ell)}) \neq \mathsf{Eval}(\mathsf{pp}, \mathsf{s}^*, h^{(\ell)})] \leq \varepsilon_2^\eta \leq 1/2^\eta = \mathsf{negl}(\lambda),$$

    since $\varepsilon_2 < 1/2$ and $\eta = \omega(\log \lambda)$.

  - Thus, with overwhelming probability, there exists $\ell \in [\eta]$ where

    $$C(h^{(\ell)}) = \mathsf{Eval}(\mathsf{pp}, \mathsf{s}^*, h^{(\ell)}) = \left\lfloor \mathbf{s}^* (\mathbf{D}^{(1)} \mid \cdots \mid \mathbf{D}^{(\eta)}) \right\rceil_p ,$$

    where the last equality follows by definition of $\mathsf{Eval}$ and the public parameters $\mathsf{pp}$. This means that in $\textsc{hyb}_{2,i-1}$, the challenger will set $\mathbf{s}^{(\ell)} = \mathbf{s}^*$. Now, let $S \subseteq \mathbb{Z}_q^n$ be the set of candidate keys output by $\mathsf{ExtractCandidates}$ on the $i^{\text{th}}$ query in $\textsc{hyb}_{2,i-1}$. To compute the response to the extraction query, the challenger computes $b_{\mathbf{s}} \leftarrow \mathsf{TestCandidate}(\mathsf{pp}, C, \mathbf{s})$ for each $\mathbf{s} \in S$. It outputs $\mathbf{s}$ if $b_{\mathbf{s}} = 1$. By the above analysis, with overwhelming probability, $\mathbf{s}^* \in S$. Suppose there exists another key $\mathbf{s} \in S$ where $\mathbf{s} \neq \mathbf{s}^*$. Since $\mathbf{s}^*$ is the *unique* key where $C(\cdot) \sim_{\varepsilon_2} \mathsf{Eval}(\mathsf{pp}, \mathsf{s}^*, \cdot)$, it follows that $C(\cdot) \not\sim_{\varepsilon_2} \mathsf{Eval}(\mathsf{pp}, \mathsf{s}, \cdot)$. Then, by the statistical properties of $\mathsf{TestCandidate}$, the bit $b_{\mathbf{s}} = 0$ for all $\mathbf{s} \neq \mathbf{s}^*$ with overwhelming probability. Since $|S| \leq \eta = \mathsf{poly}(\lambda)$, it follows that with overwhelming probability $b_{\mathbf{s}} = 0$ for all $\mathbf{s} \in S$ where $\mathbf{s} \neq \mathbf{s}^*$. In this case, the challenger in $\textsc{hyb}_{2,i-1}$ replies with $\mathbf{s}^*$ if $b_{\mathbf{s}^*} = 1$ and $\perp$ if $b_{\mathbf{s}^*} = 0$. This is exactly the challenger's behavior in $\textsc{hyb}_{2,i}$. Hence, the challenger's behavior in $\textsc{hyb}_{2,i-1}$ and $\textsc{hyb}_{2,i}$ is statistically indistinguishable.

**Case 2.** Suppose that for all keys $\mathbf{s} \in [-B, B]^n$, it is the case that $C(\cdot) \not\sim_{\varepsilon_2} \mathsf{Eval}(\mathsf{pp}, \mathsf{s}, \cdot)$. We again consider the challenger's behavior on the $i^{\text{th}}$ extraction query in $\textsc{hyb}_{2,i-1}$ and $\textsc{hyb}_{2,i}$.

- In $\textsc{hyb}_{2,i}$, the $\mathsf{ExtractCandidates}$ algorithm will always output $\varnothing$ in this case. Thus, the challenger always responds with $\perp$.

- In $\textsc{hyb}_{2,i-1}$, let $S \subseteq \mathbb{Z}_q^n$ be the set of candidate keys output by $\mathsf{ExtractCandidates}$ on the $i^{\text{th}}$ query in $\textsc{hyb}_{2,i-1}$. The challenger in $\textsc{hyb}_{2,i-1}$ computes $b_{\mathbf{s}} \leftarrow \mathsf{TestCandidate}(\mathsf{pp}, C, \mathbf{s})$ for each $\mathbf{s} \in S$. For all $\mathbf{s} \in S$, since $C(\cdot) \not\sim_{\varepsilon_2} \mathsf{Eval}(\mathsf{pp}, \mathsf{s}, \cdot)$, it follows by the statistical properties of $\mathsf{TestCandidate}$ that $b_{\mathbf{s}} = 0$ with overwhelming probability. Since $|S| \leq \eta = \mathsf{poly}(\lambda)$, it follows by a union bound that $b_{\mathbf{s}} = 0$ for all $\mathbf{s} \in S$ with overwhelming probability. Equivalently, the challenger replies with $\perp$ with overwhelming probability.

We see that in both cases, the challenger's response to the $i^{\text{th}}$ extraction query in $\textsc{hyb}_{2,i-1}$ and $\textsc{hyb}_{2,i}$ are statistically indistinguishable. Finally, since the adversary makes at most $Q = \mathsf{poly}(\lambda)$ queries (and so, there are polynomially many hybrids), we conclude that $\textsc{hyb}_2$ and $\textsc{hyb}_3$ are statistically indistinguishable. □

Combining Lemmas A.21 through A.23, we conclude that the extractable PRF construction $\Pi_{\mathsf{EPRF}}$ from Construction 4.19 satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability. □

# B Analysis of Watermarking Schemes

In this section, we provide the formal analysis of our watermarking schemes from Section 5.

## B.1 Analysis of Mark-Embedding Watermarking Scheme

In this section, we provide the security analysis of our basic mark-embedding watermarking scheme (Construction 5.17).

**Proof of Theorem 5.18 (Correctness).** We show that both of the correctness requirements are satisfied. Take $(\mathsf{pp}, \mathsf{wsk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $k \leftarrow \mathsf{F.KeyGen}(\mathsf{pp})$ and $C \leftarrow \mathsf{Mark}(\mathsf{wsk}, k)$. By definition of $\Pi_{\mathsf{WM}}$, this means that $k \leftarrow \mathsf{EX.SampleKey}(\mathsf{pp})$, and moreover $C := \mathsf{EX.PunctureEval}(\mathsf{pp}, k', \cdot)$, where $k' \leftarrow \mathsf{EX.Puncture}(\mathsf{pp}, k, \{x_i^*\}_{i \in [\lambda]})$ where $(x_1^*, \ldots, x_\lambda^*) \leftarrow \mathsf{PRF}(k_{\mathsf{PRF}}, k) \in \mathcal{X}^\lambda$.

- **Functionality-preserving:** Since the public parameters $\mathsf{pp}$ and the key $k$ are sampled honestly, by correctness of $\Pi_{\mathsf{EPRF}}$, with overwhelming probability

$$C(x) = \mathsf{EX.PunctureEval}(\mathsf{pp}, k', x) = \mathsf{EX.Eval}(\mathsf{pp}, k, x) = \mathsf{F}(k, x)$$

  for all $x \notin \{x_i^*\}_{i \in [\lambda]}$. This means that $C(\cdot) \sim_\varepsilon \mathsf{F}(k, \cdot)$ for $\varepsilon = \lambda / |\mathcal{X}| = \mathsf{negl}(\lambda)$.

- **Extraction correctness:** From above, we have that with overwhelming probability, $C(\cdot) \sim_\varepsilon \mathsf{EX.Eval}(\mathsf{pp}, k, \cdot)$ for $\varepsilon = \mathsf{negl}(\lambda)$. Since $\varepsilon_1 = 1/\mathsf{poly}(\lambda)$, by $(\varepsilon_1, \varepsilon_2)$-robust extractability (and key-injectivity) of $\Pi_{\mathsf{EPRF}}$, we have that with overwhelming probability, $\mathsf{EX.Extract}(\mathsf{pp}, \mathsf{td}, C)$ outputs $k$. In this case, the extraction algorithm checks whether $C(x_i^*) \neq \mathsf{EX.Eval}(\mathsf{pp}, k, x_i^*)$ for all $i \in [\lambda]$, where $(x_1^*, \ldots, x_\lambda^*) \leftarrow \mathsf{PRF}(k_{\mathsf{PRF}}, k)$. By definition of $C$, this is equivalent to checking whether $\mathsf{EX.PunctureEval}(\mathsf{pp}, k', x_i^*) \neq \mathsf{EX.Eval}(\mathsf{pp}, k, x_i^*)$ for all $i \in [\lambda]$. By definition $k'$ is obtained by puncturing $k$ at $(x_1^*, \ldots, x_\lambda^*)$, so by puncturing security[11] of $\Pi_{\mathsf{EPRF}}$, with overwhelming probability, $\mathsf{EX.PunctureEval}(\mathsf{pp}, k', x_i^*) \neq \mathsf{EX.Eval}(\mathsf{pp}, k, x_i^*)$ for all $i \in [\lambda]$. Hence, with overwhelming probability, $\mathsf{Extract}(\mathsf{wsk}, C) = \textsc{marked}$.

**Proof of Theorem 5.21 (Unremovability).** We proceed via a sequence of hybrid arguments. Without loss of generality, we assume that the adversary makes at most one challenge query (otherwise, the adversary is not admissible).

- $\textsc{hyb}_0$: This is the real watermarking security game $\mathsf{ExptWM}_{\Pi_{\mathsf{WM}}, \mathcal{A}}(\lambda)$ from Definition 5.10. Specifically, the experiment proceeds as follows.
  1. First, the challenger samples $k_{\mathsf{PRF}} \xleftarrow{\text{R}} \mathcal{K}_{\mathsf{PRF}}$ and $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{EX.PrmsGen}(1^\lambda)$. It gives $\mathsf{pp}$ to the adversary.

[11] Specifically, if the punctured key agreed with the real key at a punctured point with noticeable probability, then an adversary can break security of the punctured PRF with the same advantage.

2. The challenger responds to the adversary's oracle queries as follows:
   - **Marking oracle.** On input a key $k \in \mathcal{K}_{\mathsf{EPRF}}$, the challenger first computes the test points $(x_1^*, \ldots, x_\lambda^*) \leftarrow \mathsf{PRF}(k_{\mathsf{PRF}}, k)$ as well as the punctured key $k' \leftarrow \mathsf{EX.Puncture}(\mathsf{pp}, k, (x_1^*, \ldots, x_\lambda^*))$. Finally, it outputs the circuit $C \colon \mathcal{X} \to \mathcal{Y}$ where $C(\cdot) = \mathsf{EX.PunctureEval}(\mathsf{pp}, k', \cdot)$.
   - **Extraction oracle.** On input a circuit $C \colon \mathcal{X} \to \mathcal{Y}$, the challenger first computes $k \leftarrow \mathsf{EX.Extract}(\mathsf{pp}, \mathsf{td}, C)$. If $k = \bot$, it outputs UNMARKED. Otherwise, it computes $(x_1^*, \ldots, x_\lambda^*) \leftarrow \mathsf{PRF}(k_{\mathsf{PRF}}, k)$ and outputs MARKED if $C(x_i^*) \neq \mathsf{EX.Eval}(\mathsf{pp}, k, x_i^*)$ for all $i \in [\lambda]$, and UNMARKED otherwise.
   - **Challenge oracle.** The challenger first samples a key $\hat{k} \leftarrow \mathsf{EX.SampleKey}(\mathsf{pp})$. Then, it computes the test points $(\hat{x}_1^*, \ldots, \hat{x}_\lambda^*) \leftarrow \mathsf{PRF}(k_{\mathsf{PRF}}, \hat{k})$ as well as the punctured key $\hat{k}' \leftarrow \mathsf{EX.Puncture}(\mathsf{pp}, \hat{k}, (\hat{x}_1^*, \ldots, \hat{x}_\lambda^*))$. It outputs the challenge circuit $\hat{C} \colon \mathcal{X} \to \mathcal{Y}$ where $\hat{C}(\cdot) = \mathsf{EX.PunctureEval}(\mathsf{pp}, \hat{k}', \cdot)$.
3. At the end of the experiment, after the adversary outputs its circuit $\tilde{C} \colon \mathcal{X} \to \mathcal{Y}$, the challenger treats $\tilde{C}$ as an extraction query and proceeds accordingly. The output of the extraction query on $\tilde{C}$ is the output of the experiment.

- HYB$_1$: Same as HYB$_0$, except that at the beginning of the experiment, the challenger samples a uniformly random function $f \xleftarrow{\text{R}} \mathsf{Funs}[\mathcal{K}_{\mathsf{EPRF}}, \mathcal{X}^\lambda]$. For the rest of the experiment, the challenger always uses $f(\cdot)$ in place of $\mathsf{PRF}(k_{\mathsf{PRF}}, \cdot)$.

- HYB$_2$: Same as HYB$_1$, except the challenger initializes an empty set $T$ at the beginning of the experiment. During the experiment, whenever the adversary makes a marking oracle query for a key $k \in \mathcal{K}_{\mathsf{EPRF}}$, the challenger adds $k$ to $T$ (if it is not already contained in $T$). When the adversary makes a challenge query, the challenger adds the challenge key $\hat{k} \in \mathcal{K}_{\mathsf{EPRF}}$ to $T$. During the experiment (and when computing the output), the challenger uses the following procedure to implement the extraction oracle queries:

  - **Extraction oracle.** On input a circuit $C \colon \mathcal{X} \to \mathcal{Y}$, the challenger computes $b_k \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, C, k)$ for each $k \in T$. If $b_k = 0$ for all $k \in T$, it outputs UNMARKED. If $b_k = 1$ for some $k \in T$, the challenger computes $(x_1^*, \ldots, x_\lambda^*) \leftarrow f(k)$. If $C(x_i^*) \neq \mathsf{EX.Eval}(\mathsf{pp}, k, x_i^*)$ for all $i \in [\lambda]$, the challenger outputs MARKED; otherwise, it outputs UNMARKED.

  In particular, the challenger's behavior in HYB$_2$ no longer depends on the trapdoor $\mathsf{td}$.

- HYB$_3$: Same as HYB$_2$, except whenever the challenger computes $\mathsf{EX.TestCandidate}(\mathsf{pp}, \cdot, \hat{k})$, where $\hat{k}$ is the challenge key, it instead computes $\mathsf{EX.TestCandidate}(\mathsf{pp}, \cdot, \hat{C})$, where $\hat{C}$ is the challenge circuit.

- HYB$_4$: Same as HYB$_3$, except at the beginning of the security game, the challenger samples random values $\hat{y}_1^*, \ldots, \hat{y}_\lambda^* \xleftarrow{\text{R}} \mathcal{Y}$. Then, whenever the challenger needs to compute $\mathsf{EX.Eval}(\mathsf{pp}, \hat{k}, \hat{x}_i^*)$ for $i \in [\lambda]$ after the adversary has made a challenge query, the challenger uses the value $\hat{y}_i^*$ instead. Here, $\hat{k}$ is the challenge key and $(\hat{x}_1^*, \ldots, \hat{x}_\lambda^*) \leftarrow f(\hat{k})$.

- HYB$_5$: Same as HYB$_4$, except the challenger no longer adds the challenge key $\hat{k}$ to the set $T$ when responding to the challenge oracle. Moreover, it uses the following procedure to implement the extraction oracle:

- **Extraction oracle.** The pre-challenge extraction queries are handled as in $\textsc{hyb}_4$. On a post-challenge extraction query $C \colon \mathcal{X} \to \mathcal{Y}$, the challenger computes $b_{\hat{k}} \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, C, \hat{C})$, and outputs $\textsc{marked}$ if $b_{\hat{k}} = 1$. Otherwise, the challenger proceeds as in $\textsc{hyb}_4$.

For a hybrid experiment $\textsc{hyb}_i$, we write $\textsc{hyb}_i(\mathcal{A})$ to denote the output distribution of hybrid experiment $\textsc{hyb}_i$ with adversary $\mathcal{A}$. For an integer $i$, we define the advantage $\mathsf{Adv}_{i,i+1}(\mathcal{A})$ as

$$\mathsf{Adv}_{i,i+1}(\mathcal{A}) := \left|\Pr[\textsc{hyb}_0(\mathcal{A}) \neq \textsc{marked}] - \Pr[\textsc{hyb}_1(\mathcal{A}) \neq \textsc{marked}]\right|.$$

We now show that for all efficient adversaries $\mathcal{A}$, the advantage between each consecutive pair of hybrid experiments is negligible, and moreover, that $\Pr[\textsc{hyb}_5(\mathcal{A}) \neq \textsc{marked}] = \mathsf{negl}(\lambda)$.

**Lemma B.1.** *Suppose $\Pi_{\mathsf{PRF}}$ is a secure PRF. Then, for all efficient adversaries $\mathcal{A}$, $\mathsf{Adv}_{0,1}(\mathcal{A}) = \mathsf{negl}(\lambda)$.*

*Proof.* Suppose there is an efficient adversary $\mathcal{A}$ where $\mathsf{Adv}_{0,1}(\mathcal{A})$ is non-negligible. We use $\mathcal{A}$ to construct an efficient adversary $\mathcal{B}$ for $\Pi_{\mathsf{PRF}}$. Algorithm $\mathcal{B}$ simulates experiment $\textsc{hyb}_0$ for $\mathcal{A}$ exactly as prescribed, except whenever it needs to compute $\mathsf{PRF}(k_{\mathsf{PRF}}, \cdot)$ on an input $k \in \mathcal{K}_{\mathsf{EPRF}}$, algorithm $\mathcal{B}$ forwards $k$ to the $\Pi_{\mathsf{PRF}}$ challenger and uses the response $(x_1^*, \ldots, x_\lambda^*) \in \mathcal{X}^\lambda$ as the value for $\mathsf{PRF}(k_{\mathsf{PRF}}, k)$. At the end, $\mathcal{B}$ computes the output of the experiment and outputs 1 if the output is not $\textsc{marked}$, and 0 otherwise. If the $\Pi_{\mathsf{PRF}}$ challenger responds with pseudorandom values, then $\mathcal{B}$ perfectly simulates $\textsc{hyb}_0$ for $\mathcal{A}$, and if the challenger responds with evaluations of a truly random function, then $\mathcal{B}$ perfectly simulates $\textsc{hyb}_1$ for $\mathcal{A}$. Thus, the distinguishing advantage of $\mathcal{B}$ is precisely $\mathsf{Adv}_{0,1}$, and the claim follows. □

**Lemma B.2.** *Suppose $\Pi_{\mathsf{EPRF}}$ satisfies key-injectivity and $(\varepsilon_1, \varepsilon_2)$-robust extractability for $\varepsilon_2 < 1/2$. Then, for all (unbounded) adversaries $\mathcal{A}$, $\mathsf{Adv}_{1,2}(\mathcal{A}) = \mathsf{negl}(\lambda)$.*

*Proof.* We begin by defining an intermediate hybrid experiment $\textsc{hyb}_1'$ as follows:

- $\textsc{hyb}_1'$: Same as $\textsc{hyb}_1$, except the challenger uses the following (inefficient) algorithm to implement the extraction oracle:
  - **Extraction oracle.** On input a circuit $C \colon \mathcal{X} \to \mathcal{Y}$, the challenger checks if there is a unique key $k \in \mathcal{K}_{\mathsf{EPRF}}$ where $C(\cdot) \sim_{\varepsilon_2} \mathsf{EX.Eval}(\mathsf{pp}, k, \cdot)$. If so, the challenger computes $b_k \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, C, k)$. If no such $k$ exists or if $b_k = 0$, the challenger replies with $\textsc{unmarked}$. Otherwise, it computes $(x_1^*, \ldots, x_\lambda^*) \leftarrow f(k)$ and outputs $\textsc{marked}$ if for all $i \in [\lambda]$, $C(x_i) = \mathsf{EX.Eval}(\mathsf{pp}, k, x_i)$, and $\textsc{unmarked}$ otherwise.

Essentially, $\textsc{hyb}_1'$ is identical to $\textsc{hyb}_1$ except we replace $\mathsf{EX.Extract}(\mathsf{pp}, \mathsf{td}, \cdot)$ with the ideal extraction procedure from Definition 4.9.

**Claim B.3.** *If $\Pi_{\mathsf{EPRF}}$ satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability, then for all (unbounded) adversaries $\mathcal{A}$, the output distributions of $\textsc{hyb}_1(\mathcal{A})$ and $\textsc{hyb}_1'(\mathcal{A})$ are statistically indistinguishable.*

*Proof.* Follows immediately from the fact that $\Pi_{\mathsf{EPRF}}$ satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability. □

Next, we show that the output distributions of $\textsc{hyb}_1'$ and $\textsc{hyb}_2$ are also statistically indistinguishable. Let $Q = \mathsf{poly}(\lambda)$ be a bound on the number of extraction queries $\mathcal{A}$ makes. Define $\textsc{hyb}_{1,0}' \equiv \textsc{hyb}_1'$, and for $\ell \in [Q]$, define a hybrid experiment $\textsc{hyb}_{1,\ell}'$ as follows:

- $\text{HYB}'_{1,\ell}$: Same as $\text{HYB}'_1$ except the first $\ell$ extraction queries are handled using the procedure in $\text{HYB}_2$, while the remaining extraction queries are implemented using the procedure in $\text{HYB}'_1$.

Define $\text{HYB}'_{1,Q+1} \equiv \text{HYB}_2$; namely, $\text{HYB}'_{1,Q+1}$ is the hybrid where all of the extraction queries $\mathcal{A}$ makes is handled according to $\text{HYB}_2$, as is the final "simulated" extraction oracle query used to compute the output of the experiment.

**Claim B.4.** *If $\varepsilon_2 < 1/2$ and $\Pi_{\mathsf{EPRF}}$ satisfies key-injectivity, then for all (unbounded) adversaries $\mathcal{A}$ and all $\ell \in [Q+1]$, the output distributions of $\text{HYB}'_{1,\ell-1}(\mathcal{A})$ and $\text{HYB}'_{1,\ell}(\mathcal{A})$ are statistically indistinguishable.*

*Proof.* By construction, hybrid $\text{HYB}'_{1,\ell-1}$ and $\text{HYB}'_{1,\ell}$ are identical experiments except on how the challenger implements the $\ell^{\text{th}}$ extraction query. Let $C_\ell \colon \mathcal{X} \to \mathcal{Y}$ be the $\ell^{\text{th}}$ query the adversary submits to the extraction oracle (or the "simulated" extraction query used to determine the output of the experiment). We argue that the challenger's response in the two hybrids is statistically indistinguishable. We consider two possibilities:

- Suppose there exists some key $k^* \in \mathcal{K}_{\mathsf{EPRF}}$ where $C_i(\cdot) \sim_{\varepsilon_2} \mathsf{EX.Eval}(\mathsf{pp}, k^*, \cdot)$. Since $\varepsilon_2 < 1/2$ and $\Pi_{\mathsf{EPRF}}$ satisfies key-injectivity, by Lemma 4.12, the key $k^*$ is unique. Consider now the challenger's behavior in the two experiments:
  - In $\text{HYB}'_{1,\ell-1}$, the challenger computes $b_{k^*} \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, C_\ell, k^*)$. If $b_{k^*} = 0$, it outputs UNMARKED. Otherwise, it computes $(x_1^*, \ldots, x_\lambda^*) \leftarrow f(k^*)$ and outputs MARKED if for all $i \in [\lambda]$, $C_\ell(x_i^*) = \mathsf{EX.Eval}(\mathsf{pp}, k^*, x_i^*)$, and UNMARKED otherwise.
  - In $\text{HYB}'_{1,\ell}$, the challenger first computes $b_k \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, C_\ell, k)$ for all $k \in T$. From above, for all $k \neq k^*$, $C_\ell(\cdot) \not\sim_{\varepsilon_2} \mathsf{EX.Eval}(\mathsf{pp}, k, \cdot)$. Since $\Pi_{\mathsf{EPRF}}$ satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability and $|T| = \mathsf{poly}(\lambda)$, with overwhelming probability, $b_k = 0$ for all $k \neq k^*$. Thus, if $k^* \notin T$, or if $k^* \in T$ and $b_{k^*} = 0$, then the challenger outputs UNMARKED. If $k^* \in T$ and $b_{k^*} = 1$, then the challenger computes $(x_1^*, \ldots, x_\lambda^*) \leftarrow f(k^*)$ and outputs MARKED if for all $i \in [\lambda]$, $C_\ell(x_i^*) \neq \mathsf{EX.Eval}(\mathsf{pp}, k^*, x_i^*)$, and UNMARKED otherwise.

  We now consider two cases:

  - Suppose the adversary previously made a marking oracle query on $k^*$ (or if $k^* = \hat{k}$ is the challenge key). Then, in $\text{HYB}'_{1,i}$, $k^* \in T$. By the above analysis, the challenger's response in $\text{HYB}'_{1,i-1}$ and $\text{HYB}'_{1,i}$ is identical with overwhelming probability (since with overwhelming probability, the challenger in both experiments implement the same extraction procedure).
  - Suppose the adversary has not made a marking oracle query on $k^*$ and $k^*$ is not the challenge key. By construction, this means that $k^* \notin T$ in $\text{HYB}'_{1,i}$. In this case, with overwhelming probability, the challenger's response in $\text{HYB}'_{1,i}$ is UNMARKED. We show that this is also the case in $\text{HYB}'_{1,i-1}$.

    By construction, in $\text{HYB}'_{1,i-1}$, the challenger outputs MARKED only if for all $i \in [\lambda]$, $C_\ell(x_i^*) = \mathsf{EX.Eval}(\mathsf{pp}, k^*, x_i^*)$. We show that this happens with negligible probability. By construction, if the adversary never makes a marking query on $k^*$ and $k^*$ is not the challenge query, then the adversary's view in $\text{HYB}'_{1,i-1}$ up to the $i^{\text{th}}$ extraction query is *independent* of $f(k^*)$. This is because for all queries prior to the $i^{\text{th}}$ extraction query, the challenger in $\text{HYB}'_{1,i-1}$ only needs to compute $f(k)$ for keys that appear or have appeared in either a marking query or a challenge query (namely, keys in the set $T$). Since the view of $\mathcal{A}$ prior to the $i^{\text{th}}$ extraction query is independent of $f(k^*)$, the challenger can *lazily* sample $f(k^*) \in \mathcal{X}^\lambda$ *after* $\mathcal{A}$ has submitted its query to the extraction oracle. Since $C_\ell(\cdot) \sim_{\varepsilon_2} \mathsf{EX.Eval}(\mathsf{pp}, k^*, \cdot)$,

$$\Pr[(x_1^*, \ldots, x_\lambda^*) \xleftarrow{\text{R}} \mathcal{X}^\lambda : C_\ell(x_i^*) \neq \mathsf{EX.Eval}(\mathsf{pp}, k^*, x_i^*) \text{ for all } i \in [\lambda]] \leq (\varepsilon_2)^\lambda = \mathsf{negl}(\lambda).$$

Thus, with overwhelming probability, the challenger in $\textsc{hyb}'_{1,i-1}$ outputs $\textsc{unmarked}$.

In both cases, the distribution of the challenger's response on the $i^{\text{th}}$ extraction query in $\textsc{hyb}'_{1,i-1}$ and $\textsc{hyb}'_{1,i}$ is statistically indistinguishable.

- Suppose that for all keys $k \in \mathcal{K}_{\mathsf{EPRF}}$, $C_\ell(\cdot) \not\sim_{\varepsilon_2} \mathsf{EX.Eval}(\mathsf{pp}, k, \cdot)$. Then, in $\textsc{hyb}'_{1,i-1}$, the challenger always responds with $\textsc{unmarked}$. Next, since $\Pi_{\mathsf{EPRF}}$ satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability, $\Pr[\mathsf{EX.TestCandidate}(\mathsf{pp}, C, k) = 1] = \mathsf{negl}(\lambda)$ for all $k \in \mathcal{K}_{\mathsf{EPRF}}$. In $\textsc{hyb}_{1,i}$, the challenger first evaluates $b_k \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, C, k)$ for all $k \in T$. Since $|T| = \mathsf{poly}(\lambda)$, with overwhelming probability, $b_k = 0$ for all $k \in T$. This means that the challenger in $\textsc{hyb}'_{1,i}$ outputs $\textsc{unmarked}$ with overwhelming probability. Thus, the adversary's views in $\textsc{hyb}'_{1,i-1}$ and $\textsc{hyb}'_{1,i}$ are statistically indistinguishable.

In both cases, the challenger's responses in $\textsc{hyb}'_{1,i-1}$ and $\textsc{hyb}'_{1,i}$ are statistically indistinguishable, and the claim follows. □

Combining Claims B.3 and B.4 and using the fact that $Q = \mathsf{poly}(\lambda)$, we claim that the output distributions of $\textsc{hyb}_1$ and $\textsc{hyb}_2$ are statistically indistinguishable. □

**Lemma B.5.** *Suppose algorithm* $\mathsf{TestCandidate}$ *in* $\Pi_{\mathsf{EPRF}}$ *satisfies the additional properties in Remark 4.10. Then, for all (unbounded) adversaries* $\mathcal{A}$, $\mathsf{Adv}_{2,3}(\mathcal{A}) = \mathsf{negl}(\lambda)$.

*Proof.* The only difference between experiments $\textsc{hyb}_2$ and $\textsc{hyb}_3$ is that the challenger in $\textsc{hyb}_3$ replaces $\mathsf{EX.TestCandidate}(\mathsf{pp}, \cdot, \hat{k})$ with $\mathsf{EX.TestCandidate}(\mathsf{pp}, \cdot, \hat{C})$. First, by definition, $\hat{C}(\cdot) = \mathsf{EX.PunctureEval}(\mathsf{pp}, \hat{k}', \cdot)$ where $\hat{k}' \leftarrow \mathsf{EX.Puncture}(\mathsf{pp}, \hat{k}, (\hat{x}_1^*, \ldots, \hat{x}_\lambda^*))$. Thus, by correctness of $\Pi_{\mathsf{EPRF}}$ (Definition 4.14),

$$\Pr[x \xleftarrow{\text{R}} \mathcal{X} \setminus \{\hat{x}_1^*, \ldots, \hat{x}_\lambda^*\} : \hat{C}(x) \neq \mathsf{EX.Eval}(\mathsf{pp}, \hat{k}, x)] = \mathsf{negl}(\lambda).$$

This means that

$$\Pr_{x \xleftarrow{\text{R}} \mathcal{X}}[\hat{C}(x) \neq \mathsf{EX.Eval}(\mathsf{pp}, \hat{k}, x)] \leq \frac{\lambda}{|\mathcal{X}|} + \mathsf{negl}(\lambda) = \mathsf{negl}(\lambda).$$

Thus $\hat{C} \sim_\varepsilon \mathsf{EX.Eval}(\mathsf{pp}, \hat{k}, \cdot)$ where $\varepsilon = \mathsf{negl}(\lambda)$. Finally, since $\hat{k}$ is sampled using $\mathsf{EX.SampleKey}$, we can conclude via Remark 4.10 that the distribution of $\{\mathsf{EX.TestCandidate}(\mathsf{pp}, \cdot, \hat{k})\}$ in $\textsc{hyb}_2$ and the distribution of $\{\mathsf{EX.TestCandidate}(\mathsf{pp}, \cdot, \hat{C})\}$ are statistically indistinguishable. Since the adversary in the unremovability game makes $\mathsf{poly}(\lambda)$ queries, the number of times the challenger needs to run $\mathsf{EX.TestCandidate}$ is also $\mathsf{poly}(\lambda)$. Thus, the output distributions for $\textsc{hyb}_2$ and $\textsc{hyb}_3$ are statistically indistinguishable. □

**Lemma B.6.** *Suppose* $\Pi_{\mathsf{EPRF}}$ *satisfies key-injectivity and selective puncturing security and* $1/|\mathcal{Y}| = \mathsf{negl}(\lambda)$. *Then, for all efficient adversaries* $\mathcal{A}$, $\mathsf{Adv}_{3,4}(\mathcal{A}) = \mathsf{negl}(\lambda)$.

*Proof.* The difference between $\textsc{hyb}_3$ and $\textsc{hyb}_4$ is that in $\textsc{hyb}_4$, the evaluations $\mathsf{EX.Eval}(\mathsf{pp}, \hat{k}, \hat{x}_i^*)$ using the challenge key are replaced with random values $\hat{y}_i^*$ for all $i \in [\lambda]$. Thus, we appeal to puncturing security of $\Pi_{\mathsf{EPRF}}$ to argue that the outputs of the two experiments are computationally indistinguishable. Towards that end, suppose there exists an adversary $\mathcal{A}$ where $\mathsf{Adv}_{3,4}(\mathcal{A})$ is non-negligible. We use $\mathcal{A}$ to build a distinguisher $\mathcal{B}$:

1. Algorithm $\mathcal{B}$ samples $(\hat{x}_1^*, \ldots, \hat{x}_\lambda^*) \xleftarrow{\text{R}} \mathcal{X}^\lambda$ and sends $(\hat{x}_1^*, \ldots, \hat{x}_\lambda^*)$ to the puncturing security challenger. The challenger replies with the public parameters $\mathsf{pp}$ and a punctured key $\hat{k}'$. Algorithm $\mathcal{B}$ makes challenge queries at $\hat{x}_1^*, \ldots, \hat{x}_\lambda^*$ to obtain values $\hat{y}_1^*, \ldots, \hat{y}_\lambda^*$.

2. Algorithm $\mathcal{B}$ begins simulating an execution of $\textsc{hyb}_3$ and $\textsc{hyb}_4$ for $\mathcal{A}$ by providing $\mathcal{A}$ the public parameters $\mathsf{pp}$. To simplify the description, we assume that $\mathcal{B}$ implements the random function $f \colon \mathcal{K}_{\mathsf{EPRF}} \to \mathcal{X}^\lambda$ by lazily sampling the entries of $f$. That is, whenever $\mathcal{B}$ needs to evaluate $f$ on a new input $k \in \mathcal{K}_{\mathsf{EPRF}}$, it samples and stores $(x_1, \ldots, x_\lambda) \xleftarrow{\text{R}} \mathcal{X}^\lambda$ and sets $f(k) := (x_1, \ldots, x_\lambda)$. If $\mathcal{B}$ already computed $f(k)$ on a previous query, then $\mathcal{B}$ uses the same value. Now, it answers $\mathcal{A}$'s queries as follows:

   - **Marking oracle.** On input a key $k \in \mathcal{K}_{\mathsf{EPRF}}$, algorithm $\mathcal{B}$ implements the marking oracle using the procedure described in $\textsc{hyb}_3$ and $\textsc{hyb}_4$. In addition, it samples a random point $x \in \mathcal{X}$ and checks whether $\mathsf{EX.Eval}(\mathsf{pp}, k, x) = \mathsf{EX.PunctureEval}(\mathsf{pp}, \hat{k}', x)$. If so, then $\mathcal{B}$ halts the simulation and outputs 1 if $\hat{y}_1^* = \mathsf{EX.Eval}(\mathsf{pp}, k, \hat{x}_1^*)$ and 0 otherwise.
   - **Extraction oracle.** Algorithm $\mathcal{B}$ simulates the pre-challenge queries as described in $\textsc{hyb}_3$ and $\textsc{hyb}_4$. On a post-challenge query $C \colon \mathcal{X} \to \mathcal{Y}$, algorithm $\mathcal{B}$ computes $b_k \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, C, k)$ for $k \in T$ and $b_{\hat{k}} \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, C, \hat{C})$, where $\hat{C}$ is the challenge circuit.
     - If $b_{\hat{k}} = 0$ and $b_k = 0$ for all $k \in T$, algorithm $\mathcal{B}$ outputs $\textsc{unmarked}$.
     - If $b_k = 1$ for some $k \in T$, algorithm $\mathcal{B}$ computes $(x_1^*, \ldots, x_\lambda^*) \leftarrow f(k)$ and outputs $\textsc{marked}$ if $C(x_i^*) \neq \mathsf{EX.Eval}(\mathsf{pp}, k, x_i^*)$ for $i \in [\lambda]$ and $\textsc{unmarked}$ otherwise.
     - If $b_{\hat{k}} = 1$, algorithm $\mathcal{B}$ outputs $\textsc{marked}$ if $C(\hat{x}_i^*) \neq \hat{y}_i^*$ for $i \in [\lambda]$ and $\textsc{unmarked}$ otherwise.
   - **Challenge oracle.** Algorithm $\mathcal{B}$ replies with the circuit $\hat{C} := \mathsf{EX.PunctureEval}(\mathsf{pp}, \hat{k}, \cdot)$, where $\hat{k}$ is the punctured key $\mathcal{B}$ received from the puncturing security challenger.

3. At the end of the experiment, let $\tilde{C} \colon \mathcal{X} \to \mathcal{Y}$ be the circuit the adversary outputs. If $\mathcal{A}$ aborts before outputting a circuit, $\mathcal{B}$ sets $\mathsf{output} = \perp$. Otherwise, $\mathcal{B}$ simulates an extraction oracle query on $\tilde{C}$ and sets $\mathsf{output}$ to be the output of the extraction oracle. Finally, $\mathcal{B}$ outputs 1 if $\mathsf{output} \neq \textsc{marked}$ and 0 otherwise.

We now show that algorithm $\mathcal{B}$ simulates an execution of $\textsc{hyb}_3$ if the challenge values $\hat{y}_1^*, \ldots, \hat{y}_\lambda^*$ correspond to the real evaluations of the PRF, and that it corresponds to an execution of $\textsc{hyb}_4$ if they are uniformly random values. In particular, let $\hat{k}$ be the PRF key sampled by the puncturing security challenger (unknown to $\mathcal{B}$). The test points $(\hat{x}_1^*, \ldots, \hat{x}_\lambda^*)$ play the role of $f(\hat{k})$ in the simulation. As long as $\mathcal{B}$ never has to compute $f(\hat{k})$ in the simulation, then it correctly simulates the behavior of $f$. By construction, the only times where $\mathcal{B}$ needs to sample new outputs of $f$ are in response to marking queries. We thus consider the two cases:

- Suppose $\mathcal{A}$ never makes a marking query on $\hat{k}$. In this case, if the challenge values $\hat{y}_1^*, \ldots, \hat{y}_\lambda^*$ are pseudorandom (i.e., the outputs of $\mathsf{EX.Eval}$), the view $\mathcal{B}$ simulates for $\mathcal{A}$ is statistically close to $\textsc{hyb}_3$. If the challenge values are uniformly random, the view $\mathcal{B}$ simulates for $\mathcal{A}$ is statistically close to $\textsc{hyb}_4$. In particular, the only difference between the simulated experiments and the real experiments is the extra abort condition $\mathcal{B}$ introduces when answering marking queries. Suppose $\mathcal{A}$ makes a marking query on a key $k \neq \hat{k}$. We argue that $\mathcal{B}$ halts with negligible probability. First, by correctness of $\Pi_{\mathsf{EPRF}}$,

$$\Pr_{x \xleftarrow{\text{R}} \mathcal{X}} [\mathsf{EX.PunctureEval}(\mathsf{pp}, \hat{k}', x) \neq \mathsf{EX.Eval}(\mathsf{pp}, \hat{k}, x)] \leq \frac{\lambda}{|\mathcal{X}|} + \mathsf{negl}(\lambda) = \mathsf{negl}(\lambda). \tag{B.1}$$

Next, by key-injectivity of $\Pi_{\mathsf{EPRF}}$, we have that for $k \neq \hat{k}$

$$\Pr_{x \xleftarrow{\text{R}} \mathcal{X}} [\mathsf{EX.Eval}(\mathsf{pp}, k, x) = \mathsf{EX.Eval}(\mathsf{pp}, \hat{k}, x)] = \mathsf{negl}(\lambda).$$

Thus, the probability that $\mathcal{B}$ aborts when responding to a marking oracle query is $\mathsf{negl}(\lambda)$. Since $\mathcal{A}$ makes a polynomial number of queries, we conclude that the view $\mathcal{B}$ simulates for $\mathcal{A}$ is statistically close to $\textsc{hyb}_3$ or $\textsc{hyb}_4$ depending on whether the challenge values are pseudorandom or truly random. In this case, the distinguishing advantage of $\mathcal{B}$ is precisely $\mathsf{Adv}_{3,4}(\mathcal{A})$.

- Suppose $\mathcal{A}$ makes a marking oracle query on $\hat{k}$. By correctness of $\Pi_{\mathsf{EPRF}}$ (Eq. (B.1)), algorithm $\mathcal{B}$ in this case will halt with overwhelming probability. When the challenge values are pseudorandom (namely, if $\hat{y}_1^* = \mathsf{EX.Eval}(\mathsf{pp}, \hat{k}, \hat{x}_1^*)$), algorithm $\mathcal{B}$ outputs 1 with probability 1. When the challenge values are truly random, algorithm $\mathcal{B}$ outputs 1 with probability $1/|\mathcal{Y}|$. Thus, algorithm $\mathcal{B}$ is able to break puncturing security of $\Pi_{\mathsf{EPRF}}$ with probability that is negligibly close to $1 - 1/|\mathcal{Y}|$, which is non-negligible.

In both cases, the distinguishing advantage of $\mathcal{B}$ is non-negligible, and the claim follows. □

**Lemma B.7.** *For all (unbounded) adversaries $\mathcal{A}$, $\mathsf{Adv}_{4,5}(\mathcal{A}) = \mathsf{negl}(\lambda)$.*

*Proof.* By construction, the only difference between $\textsc{hyb}_4$ and $\textsc{hyb}_5$ is in how the post-challenge extraction queries are handled and how the experiment's output is computed. Let $Q = \mathsf{poly}(\lambda)$ be a bound on the number of post-challenge extraction queries the adversary makes. We define a sequence of $Q$ intermediate hybrids as follows:

- $\textsc{hyb}_{4,\ell}$: Same as $\textsc{hyb}_4$, except the first $\ell$ post-challenge extraction queries are handled as in $\textsc{hyb}_5$ while the remaining post-challenge extraction queries are handled as in $\textsc{hyb}_4$.

We additionally define $\textsc{hyb}_{4,Q+1} \equiv \textsc{hyb}_5$. Now, we show that the outputs of each pair of hybrid experiments is statistically indistinguishable.

**Claim B.8.** *For all (unbounded) adversaries $\mathcal{A}$ and all $\ell \in [Q+1]$, the distributions $\textsc{hyb}_{4,\ell-1}(\mathcal{A})$ and $\textsc{hyb}_{4,\ell}(\mathcal{A})$ are statistically indistinguishable.*

*Proof.* By construction, $\textsc{hyb}_{4,\ell-1}$ and $\textsc{hyb}_{4,\ell}$ are identical except on how the challenger responds to the $\ell^{\text{th}}$ post-challenge extraction query. Let $C_\ell \colon \mathcal{X} \to \mathcal{Y}$ be the circuit the challenger submits in the $\ell^{\text{th}}$ extraction query. By design, the challenger's behavior in the two experiments are only different if $b_{\hat{k}} \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, C_\ell, \hat{C}) = 1$. Suppose this is the case, and consider the challenger's behavior in the two experiments:

- In $\textsc{hyb}_{4,\ell-1}$, the challenger computes $(x_1^*, \ldots, x_\lambda^*) \leftarrow f(\hat{k})$ and outputs $\textsc{marked}$ if $C_\ell(x_i^*) \neq \hat{y}_i^*$ for all $\ell \in [\lambda]$. Otherwise, it outputs $\textsc{unmarked}$.

  By construction of $\textsc{hyb}_{4,\ell-1}$, the adversary's view up to its $\ell^{\text{th}}$ post-challenge extraction query is entirely independent of $\hat{y}_1^*, \ldots, \hat{y}_\lambda^*$, which are sampled uniformly at random at the beginning of the experiment. Thus, we can defer the sampling of $\hat{y}_1^*, \ldots, \hat{y}_\lambda^*$ until *after* the adversary has submitted its extraction query $C_\ell$. Then,

  $$\Pr[\hat{y}_1^*, \ldots, \hat{y}_\lambda^* \xleftarrow{\text{R}} \mathcal{Y} : C_\ell(x_i^*) \neq \hat{y}_i^* \text{ for all } i \in [\lambda]] = (1 - 1/|\mathcal{Y}|)^\lambda = 1 - \mathsf{negl}(\lambda).$$

  Thus, with overwhelming probability, the challenge outputs $\textsc{marked}$ in this case.

- In $\textsc{hyb}_{4,\ell}$, the challenger always outputs $\textsc{marked}$ in this case. □

Since $Q = \mathsf{poly}(\lambda)$, we conclude via Claim B.8 that the distributions $\textsc{hyb}_4(\mathcal{A})$ and $\textsc{hyb}_5(\mathcal{A})$ are statistically indistinguishable. □

**Lemma B.9.** *If $\Pi_{\mathsf{EPRF}}$ satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability, then for all (unbounded) and $\varepsilon$-unremoving-admissible adversaries $\mathcal{A}$ where $\varepsilon \leq \varepsilon_1$, $\Pr[\textsc{hyb}_5(\mathcal{A}) \neq \textsc{marked}] = \mathsf{negl}(\lambda)$.*

*Proof.* If $\mathcal{A}$ is $\varepsilon$-unremoving-admissible, then at the end of the experiment, the circuit $\tilde{C} \colon \mathcal{X} \to \mathcal{Y}$ output by $\mathcal{A}$ satisfies $\tilde{C} \sim_\varepsilon \hat{C}$, where $\hat{C}$ is the circuit the challenger uses to respond to the challenge query. Next, since $\Pi_{\mathsf{EPRF}}$ satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability and $\varepsilon \leq \varepsilon_1$, this means that $b_{\hat{k}} = 1$ with overwhelming probability where $b_{\hat{k}} \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, \tilde{C}, \hat{C})$. By construction, in $\textsc{hyb}_5$, the challenger outputs $\textsc{unmarked}$ in this case, as required. □

Combining Lemmas B.1 through B.9, the message-embedding watermarking scheme $\Pi_{\mathsf{WM}}$ from Construction 5.25 satisfies $\varepsilon$-unremovability. □

**Proof of Theorem 5.22 (Unforgeability).** The proof relies on a similar sequence of hybrids as that in the proof of Theorem 5.21. We briefly describe them below:

- $\textsc{hyb}_0$: This is the real watermarking security game $\mathsf{ExptWM}_{\Pi_{\mathsf{WM}},\mathcal{A}}(\lambda)$ from Definition 5.10.
- $\textsc{hyb}_1$: Same as $\textsc{hyb}_1$ from the proof of Theorem 5.21.
- $\textsc{hyb}_2$: Same as $\textsc{hyb}_2$ from the proof of Theorem 5.21.

Indistinguishability of hybrids $\textsc{hyb}_0$, $\textsc{hyb}_1$, and $\textsc{hyb}_2$ follow exactly as in the proof of Theorem 5.21. It suffices to show that $\Pr[\textsc{hyb}_2(\mathcal{A}) \neq \textsc{unmarked}] = \mathsf{negl}(\lambda)$.

**Lemma B.10.** *Suppose that $\Pi_{\mathsf{EPRF}}$ is almost functionality-preserving for adversarial keys, satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability as well as the additional properties in Remark 4.10, and that $1/|\mathcal{X}| = \mathsf{negl}(\lambda)$. Then for all efficient $\delta$-unforging-admissible adversaries $\mathcal{A}$ where $\delta \geq \varepsilon_2$, $\Pr[\textsc{hyb}_2(\mathcal{A}) \neq \textsc{unmarked}] = \mathsf{negl}(\lambda)$.*

*Proof.* Let $k_1, \ldots, k_Q$ be the keys $\mathcal{A}$ submits to the marking oracle, and let $C_1, \ldots, C_Q \colon \mathcal{X} \to \mathcal{Y}$ be the circuits $\mathcal{A}$ receives from the marking oracle. Since $\mathcal{A}$ is $\delta$-unforging-admissible, it must be the case that $\tilde{C} \not\sim_{\varepsilon_2} C_\ell$ where $C_\ell := \mathsf{EX.PunctureEval}(\mathsf{pp}, k_\ell, \cdot)$ and $k_\ell \leftarrow \mathsf{EX.Puncture}(\mathsf{pp}, k_\ell, S_\ell)$ where $S_\ell \subseteq \mathcal{X}$ and $|S_\ell| = \mathsf{poly}(\lambda)$. Since $\Pi_{\mathsf{EPRF}}$ is almost functionality-preserving for adversarially-chosen keys, $\mathcal{A}$ is efficient, and $|S_\ell| / |\mathcal{X}| = \mathsf{negl}(\lambda)$, it follows (via a union bound) that there exists $\varepsilon = \mathsf{negl}(\lambda)$ such that with overwhelming probability, $C_\ell \sim_\varepsilon \mathsf{EX.Eval}(\mathsf{pp}, k_\ell, \cdot)$ for all $\ell \in [Q]$. In particular, this means that for each $\ell \in [Q]$, there exists a circuit $\tilde{C}'_\ell \colon \mathcal{X} \to \mathcal{Y}$ such that $\tilde{C}'_\ell \sim_\varepsilon \tilde{C}$

and $C'_\ell \not\sim_{\varepsilon_2} \mathsf{EX.Eval}(\mathsf{pp}, k_\ell, \cdot)$. Since $\Pi_{\mathsf{EPRF}}$ satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability, this means that $\Pr[\mathsf{EX.TestCandidate}(\mathsf{pp}, \tilde{C}'_\ell, k_\ell) = 1] = \mathsf{negl}(\lambda)$. Since $\Pi_{\mathsf{EPRF}}$ satisfies the additional properties in Remark 4.10, and $\tilde{C} \sim_\varepsilon \tilde{C}'_\ell$ for $\varepsilon = \mathsf{negl}(\lambda)$, this means that $\Pr[\mathsf{EX.TestCandidate}(\mathsf{pp}, \tilde{C}, k_\ell) = 1] = \mathsf{negl}(\lambda)$ for each $\ell \in [Q]$. By a union bound,

$$\Pr[\exists \ell \in [Q] : \mathsf{EX.TestCandidate}(\mathsf{pp}, \tilde{C}, k_\ell) = 1] = \mathsf{negl}(\lambda).$$

By construction of $\textsc{hyb}_2$, the set $T$ of keys submitted to the marking oracle is $T = \{k_1, \ldots, k_Q\}$. Since $\mathsf{EX.TestCandidate}(\mathsf{pp}, \tilde{C}, k) = 0$ for all $k \in T$ with overwhelming probability, the output in $\textsc{hyb}_2$ is $\textsc{unmarked}$ with overwhelming probability. □

We conclude that the mark-embedding watermarking scheme $\Pi_{\mathsf{WM}}$ from Construction 5.17 satisfies $\delta$-unforgeability.

## B.2 Analysis of Message-Embedding Watermarking Scheme

In this section, we provide the security analysis of the message-embedding watermarking scheme (Construction 5.25).

**Proof of Theorem 5.29 (Unremovability).** We use the same hybrid structure as in the proof of Theorem 5.21. For completeness, we provide the full description of the hybrid experiments:

- $\textsc{hyb}_0$: This is the real watermarking game $\mathsf{ExptWM}_{\Pi_{\mathsf{WM}}, \mathcal{A}}(\lambda)$ from Definition 5.10. Specifically, the experiment proceeds as follows.
  1. First, the challenger samples $k_{\mathsf{PRF}} \xleftarrow{\text{R}} \mathcal{K}_{\mathsf{PRF}}$ and $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{EX.PrmsGen}(1^\lambda)$. It gives $\mathsf{pp}$ to the adversary.
  2. The challenger responds to the adversary's oracle queries as follows:
     - **Marking oracle.** On input a key $k \in \mathcal{K}_{\mathsf{EPRF}}$ and a message $m \in \{0,1\}^t$, the challenger derives a collection of points $(x^*_{i,1}, \ldots, x^*_{i,\lambda}) \leftarrow \mathsf{PRF}(k_{\mathsf{PRF}}, (k, i, m_i))$ for each $i \in [t]$. It then constructs a punctured key $k' \leftarrow \mathsf{EX.Puncture}(\mathsf{pp}, k, \{x^*_{i,j}\}_{i \in [t], j \in [\lambda]})$. Finally, it outputs the circuit $C \colon \mathcal{X} \to \mathcal{Y}$ that implements the punctured evaluation algorithm $\mathsf{EX.PunctureEval}(\mathsf{pp}, k', \cdot)$.
     - **Extraction oracle.** On input a circuit $C \colon \mathcal{X} \to \mathcal{Y}$, the challenger computes $k \leftarrow \mathsf{EX.Extract}(\mathsf{pp}, \mathsf{td}, C)$. If $k = \bot$, it outputs $\bot$. Otherwise, for each $i \in [t]$ and $b \in \{0,1\}$, the challenger computes $(x^*_{i,b,1}, \ldots, x^*_{i,b,\lambda}) \leftarrow \mathsf{PRF}(k_{\mathsf{PRF}}, (k, i, b))$. Let $N_{i,b}$ be the number of indices $j \in [\lambda]$ where $C(x^*_{i,b,j}) \neq \mathsf{EX.Eval}(\mathsf{pp}, k, x^*_{i,b,j})$. If there exists an index $i \in [t]$ where $N_{i,0}, N_{i,1} < 2\lambda/3$ or $N_{i,0}, N_{i,1} > 2\lambda/3$, then output $\bot$. Otherwise, for each $i \in [t]$, let $b_i \in \{0,1\}$ be the unique bit where $N_{i,b_i} > 2\lambda/3$. The challenger responds with the message $m = b_1 \cdots b_t$.
     - **Challenge oracle.** On input a message $\hat{m} \in \{0,1\}^t$, the challenger samples a key $\hat{k} \leftarrow \mathsf{EX.SampleKey}(\mathsf{pp})$. Then, it derives a collection of points $(\hat{x}^*_{i,1}, \ldots, \hat{x}^*_{i,\lambda}) \leftarrow \mathsf{PRF}(k_{\mathsf{PRF}}, (k, i, \hat{m}_i))$ for each $i \in [t]$. The challenger computes the punctured key $\hat{k}' \leftarrow \mathsf{EX.Puncture}(\mathsf{pp}, \hat{k}, \{\hat{x}^*_{i,j}\}_{i \in [t], j \in [\lambda]})$. Finally, it outputs the circuit $\hat{C} \colon \mathcal{X} \to \mathcal{Y}$ that implements the punctured evaluation algorithm $\mathsf{EX.PunctureEval}(\mathsf{pp}, \hat{k}', \cdot)$.
  3. At the end of the experiment, after the adversary outputs its circuit $\tilde{C} \colon \mathcal{X} \to \mathcal{Y}$, the challenger treats $\tilde{C}$ as an extraction query and proceeds accordingly. The output of the extraction query on $\tilde{C}$ is the output of the experiment.

- $\text{HYB}_1$: Same as $\text{HYB}_0$, except at the beginning of the experiment, the challenger samples a uniformly random function $f \xleftarrow{\text{R}} \mathsf{Funs}[(\mathcal{K}_{\mathsf{EPRF}} \times [t] \times \{0,1\}), \mathcal{X}^\lambda]$. For the rest of the experiment, the challenger always uses $f(\cdot)$ in place of $\mathsf{PRF}(k_{\mathsf{PRF}}, \cdot)$.

- $\text{HYB}_2$: Same as $\text{HYB}_1$, except the challenge initializes an empty set $T$ at the beginning of the experiment. During the experiment, whenever the adversary makes a marking oracle query for a key $k \in \mathcal{K}_{\mathsf{EPRF}}$, the challenger adds $k$ to $T$ (if it is not already contained in $T$). When the adversary makes a challenge query, the challenger adds the challenge key $\hat{k} \in \mathcal{K}_{\mathsf{EPRF}}$ to $T$. During the experiment (and when computing the output), the challenger uses the following procedure to implement the extraction oracle queries:
  - **Extraction oracle.** On input a circuit $C \colon \mathcal{X} \to \mathcal{Y}$, the challenger computes $b_k \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, C, k)$ for each $k \in T$. If $b_k = 0$ for all $k \in T$, it outputs $\perp$. If $b_k = 1$ for some $k \in T$, the challenger computes $(x^*_{i,b,1}, \ldots, x^*_{i,b,\lambda}) \leftarrow f(k, i, b)$ for all $i \in [t]$ and $b \in \{0,1\}$. Let $N_{i,b}$ denote the number of indices $j \in [\lambda]$ where $C(x^*_{i,b,j}) \neq \mathsf{EX.Eval}(\mathsf{pp}, k, x^*_{i,b,j})$. If there exists an index $i \in [t]$ where $N_{i,0}, N_{i,1} < 2\lambda/3$ or $N_{i,0}, N_{i,1} > 2\lambda/3$, then output $\perp$. Otherwise, for each $i \in [t]$, let $b_i \in \{0,1\}$ be the unique bit where $N_{i,b_i} > 2\lambda/3$. Output the message $m = b_1 \cdots b_t$.

- $\text{HYB}_3$: Same as $\text{HYB}_2$, except whenever the challenger computes $\mathsf{EX.TestCandidate}(\mathsf{pp}, \cdot, \hat{k})$, where $\hat{k}$ is the challenge key, it instead computes $\mathsf{EX.TestCandidate}(\mathsf{pp}, \cdot, \hat{C})$ where $\hat{C}$ is the challenge circuit.

- $\text{HYB}_4$: Same as $\text{HYB}_3$, except at the beginning of the security game, the challenger samples random values $\hat{y}^*_{i,j} \xleftarrow{\text{R}} \mathcal{Y}$ for $i \in [t]$ and $j \in [\lambda]$. Then, whenever the challenger needs to compute $\mathsf{EX.Eval}(\mathsf{pp}, \hat{k}, \hat{x}^*_{i,j})$ for $i \in [t]$ and $j \in [\lambda]$ after the adversary has made a challenge query, the challenger uses the value $\hat{y}^*_{i,j}$ instead. Here, $\hat{k}$ is the challenge key, $(\hat{x}^*_{i,1}, \ldots, \hat{x}^*_{i,\lambda})$ is the value of $f(\hat{k}, i, \hat{m}_i)$, and $\hat{m}_i$ is the challenge message. In addition, if $\mathcal{A}$ ever queries the marking oracle on the challenge key $\hat{k}$, the experiment always outputs $\hat{m}$.

- $\text{HYB}_5$: Same as $\text{HYB}_4$ except the challenger no longer adds the challenge key $\hat{k}$ to the set $T$ when responding to the challenge oracle. Moreover, it uses the following procedure to implement the extraction oracle:
  - **Extraction oracle.** The pre-challenge extraction queries are handled as in $\text{HYB}_4$. On a post-challenge extraction query $C \colon \mathcal{X} \to \mathcal{Y}$, the challenger computes $b_{\hat{k}} \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, C, \hat{C})$ and outputs $\hat{m}$ if $b_{\hat{k}} = 1$. Otherwise, the challenge proceeds as in $\text{HYB}_4$.

For a hybrid experiment $\text{HYB}_i$, we write $\text{HYB}_i(\mathcal{A})$ to denote the output distribution of $\text{HYB}_i$ with adversary $\mathcal{A}$. For an integer $i$, we define the advantage $\mathsf{Adv}_{i,i+1}(\mathcal{A})$ as

$$\mathsf{Adv}_{i,i+1}(\mathcal{A}) := \left| \Pr[\text{HYB}_i(\mathcal{A}) \neq \hat{m}] - \Pr[\text{HYB}_{i+1}(\mathcal{A}) \neq \hat{m}] \right| .$$

We now show that for all efficient adversaries $\mathcal{A}$, the advantage between each consecutive pair of hybrid experiments is negligible, and moreover, that $\Pr[\text{HYB}_5(\mathcal{A}) \neq \hat{m}] = \mathsf{negl}(\lambda)$.

**Lemma B.11.** *Suppose $\Pi_{\mathsf{PRF}}$ is a secure PRF. Then for all efficient adversaries $\mathcal{A}$, $\mathsf{Adv}_{0,1}(\mathcal{A}) = \mathsf{negl}(\lambda)$.*

*Proof.* Follows by by an analogous argument as that in the proof of Lemma B.1. □

**Lemma B.12.** *Suppose $\Pi_{\mathsf{EPRF}}$ satisfies key-injectivity and $(\varepsilon_1, \varepsilon_2)$-robust extractability for $\varepsilon_2 < 1/2$. Then, for all (unbounded) adversaries $\mathcal{A}$, $\mathsf{Adv}_{1,2}(\mathcal{A}) = \mathsf{negl}(\lambda)$.*

*Proof.* Similar to the proof of Lemma B.2, we begin by defining an intermediate hybrid $\textsc{hyb}'_1$:

- $\textsc{hyb}'_1$: Same as $\textsc{hyb}_1$, except the challenger uses the following (inefficient) algorithm to implement the extraction oracle:
  - **Extraction oracle.** On input a circuit $C \colon \mathcal{X} \to \mathcal{Y}$, the challenger first checks if there is a unique key $k \in \mathcal{K}_{\mathsf{EPRF}}$ where $C(\cdot) \sim_{\varepsilon_2} \mathsf{EX.Eval}(\mathsf{pp}, k, \cdot)$. If so, the challenger computes $b_k \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, C, k)$. If no such $k$ exists or if $b_k = 0$, the challenger replies with $\perp$. Otherwise, the challenger computes $(x^*_{i,b,1}, \ldots, x^*_{i,b,\lambda}) \leftarrow f(k, i, b)$ for all $i \in [t]$ and $b \in \{0, 1\}$. Let $N_{i,b}$ denote the number of indices $j \in [\lambda]$ where $C(x^*_{i,b,j}) \neq \mathsf{EX.Eval}(\mathsf{pp}, k, x^*_{i,b,j})$. If there exists an index $i \in [t]$ where $N_{i,0}, N_{i,1} < 2\lambda/3$ or $N_{i,0}, N_{i,1} > 2\lambda/3$, the challenger responds with $\perp$. Otherwise, for each $i \in [t]$, let $b_i \in \{0, 1\}$ be the unique bit where $N_{i,b_i} > 2\lambda/3$. The challenger replies with the message $m = b_1 \cdots b_t$.

As in the proof of Lemma B.2, the output distributions $\textsc{hyb}_1$ and $\textsc{hyb}'_1$ are statistically indistinguishable if $\Pi_{\mathsf{EPRF}}$ satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability. To conclude, we show that the output distributions of $\textsc{hyb}'_1$ and $\textsc{hyb}_2$ are also statistically indistinguishable. Once again, we rely on a query-by-query hybrid. Let $Q = \mathsf{poly}(\lambda)$ be a bound on the number of extraction queries $\mathcal{A}$ makes. Define $\textsc{hyb}'_{1,0} \equiv \textsc{hyb}'_1$ and for $\ell \in [Q]$, define a hybrid experiment $\textsc{hyb}'_{1,\ell}$ for $\ell \in [Q]$ as follows:

- $\textsc{hyb}'_{1,\ell}$: Same as $\textsc{hyb}'_1$ except the first $\ell$ extraction queries are handled using the procedure in $\textsc{hyb}_2$ while the remaining extraction queries are handled using the procedure in $\textsc{hyb}'_1$.

Define $\textsc{hyb}'_{1,Q+1} \equiv \textsc{hyb}_2$; namely, $\textsc{hyb}'_{1,Q+1}$ is the hybrid where all of the extraction queries $\mathcal{A}$ makes is handled according to $\textsc{hyb}_2$, as is the final "simulated" extraction query used to compute the output of the experiment.

**Claim B.13.** *If $\varepsilon_2 < 1/2$ and $\Pi_{\mathsf{EPRF}}$ satisfies key-injectivity, then for all (unbounded) adversaries $\mathcal{A}$ and all $\ell \in [Q + 1]$, the output distributions of $\textsc{hyb}'_{1,\ell-1}(\mathcal{A})$ and $\textsc{hyb}'_{1,\ell}(\mathcal{A})$ are statistically indistinguishable.*

*Proof.* By construction, hybrids $\textsc{hyb}'_{1,\ell-1}$ and $\textsc{hyb}'_{1,\ell}$ are identical experiments except in the way the challenger response to the $\ell^{\text{th}}$ extraction query. Let $C_\ell \colon \mathcal{X} \to \mathcal{Y}$ be the $\ell^{\text{th}}$ query the adversary submits to the extraction oracle (or the "simulated" extraction query used to determine the output of the experiment). We argue that the challenger's response to this query in the two experiments is statistically indistinguishable. We consider two cases:

- Suppose there exists $k^* \in \mathcal{K}_{\mathsf{EPRF}}$ where $C(\cdot) \sim_{\varepsilon_2} \mathsf{EX.Eval}(\mathsf{pp}, k^*, \cdot)$. Since $\varepsilon_2 < 1/2$ and $\Pi_{\mathsf{EPRF}}$ satisfies key-injectivity, Lemma 4.12 says that $k^*$ is unique. In particular, for all $k \neq k^*$, $C_\ell(\cdot) \not\sim_{\varepsilon_2} \mathsf{EX.Eval}(\mathsf{pp}, k, \cdot)$. Consider now the challenger's behavior in the two experiments:
  - In $\textsc{hyb}'_{1,\ell-1}$, the challenger computes $b_{k^*} \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, C_\ell, k^*)$. If $b_{k^*} = 0$, the challenger outputs $\perp$. Otherwise, the challenger computes $(x^*_{i,b,1}, \ldots, x^*_{i,b,\lambda}) \leftarrow f(k^*, i, b)$ for all $i \in [t]$ and $b \in \{0, 1\}$ and continues according to the procedure in $\textsc{hyb}'_1$ and $\textsc{hyb}_2$.
  - In $\textsc{hyb}'_{1,\ell}$, the challenger computes $b_k \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, C_\ell, k)$ for all $k \in T$. Since $C_\ell(\cdot) \not\sim_{\varepsilon_2} \mathsf{EX.Eval}(\mathsf{pp}, k, \cdot)$ for all $k \neq k^*$, we can appeal to $(\varepsilon_1, \varepsilon_2)$-robust extractability of $\Pi_{\mathsf{EPRF}}$ to argue that $b_k = 0$ with overwhelming probability for all $k \neq k^*$. Thus, if $k^* \notin T$ or $b_{k^*} = 0$, then with overwhelming probability, the challenger replies with $\perp$. If $b_{k^*} = 1$, then the challenger computes $(x^*_{i,b,1}, \ldots, x^*_{i,b,\lambda}) \leftarrow f(k^*, i, b)$ for all $i \in [t]$ and $b \in \{0, 1\}$ and continues according to the procedure in $\textsc{hyb}'_1$ and $\textsc{hyb}_2$.

We consider two cases:

  - Suppose the adversary previously made a marking oracle query on $k^*$ or $\hat{k} = k^*$ is the challenge key. In this case, $k^* \in T$. From the above analysis, the challenger's response in the two experiments is statistically indistinguishable.
  - Suppose the adversary has not made a marking oracle query on $k^*$ and $k^* \neq \hat{k}$ is not the challenge key. In this case, $k^* \notin T$, and by the above analysis, the challenger in $\textsc{hyb}'_{1,\ell}$ outputs $\perp$ with overwhelming probability. We show that this is also the case in $\textsc{hyb}'_{1,\ell-1}$.

    If $b_{k^*} = 0$, then the challenger in $\textsc{hyb}'_{1,\ell-1}$ always outputs $\perp$. Suppose this is not the case. Then, in $\textsc{hyb}'_{1,\ell-1}$, the challenger computes $(x^*_{i,b,1}, \ldots, x^*_{i,b,\lambda}) \leftarrow f(k^*, i, b)$ for all $i \in [t]$ and $b \in \{0,1\}$, and then counts the number $N_{i,b}$ of indices $j \in [\lambda]$ where $C_\ell(x^*_{i,b,j}) \neq \mathsf{EX.Eval}(\mathsf{pp}, k^*, x^*_{i,b,j})$. We argue now that with overwhelming probability (over the choice of the random function $f$) that $N_{i,b} < 2\lambda/3$ for all $i \in [t]$ and $b \in \{0,1\}$. In this case, the challenger outputs $\perp$. The argument proceeds very similarly to the corresponding one used in the proof of Claim B.4. In particular, since the adversary has not made a marking oracle query on $k^*$ and $k^*$ is not the challenge query, the adversary's view in $\textsc{hyb}'_{1,\ell-1}$ up to the point of its $\ell^{\text{th}}$ query is independent of $f(k^*, i, b)$. This is because by construction, the challenger in $\textsc{hyb}_{1,\ell-1}$ only needs to evaluates $f$ on keys $k$ that are contained in $T$ (up until processing the $\ell^{\text{th}}$ extraction query). Thus, the challenger can lazily sample the value of $f(k^*, i, b) := (x^*_{i,b,1}, \ldots, x^*_{i,b,\lambda})$ *after* the adversary makes its $\ell^{\text{th}}$ extraction query. By assumption, $C_\ell(\cdot) \sim_{\varepsilon_2} \mathsf{EX.Eval}(\mathsf{pp}, k^*, \cdot)$, which means that

    $$\Pr[x^* \xleftarrow{\text{R}} \mathcal{X} : C_\ell(x^*) \neq \mathsf{EX.Eval}(\mathsf{pp}, k^*, x^*)] \leq \varepsilon_2 < 1/2.$$

    Then, by a Chernoff bound,

    $$\Pr[(x^*_{i,b,1}, \ldots, x^*_{i,b,\lambda}) \xleftarrow{\text{R}} \mathcal{X}^\lambda : N_{i,b} \geq 2\lambda/3] \leq 2^{-\Omega(\lambda)} = \mathsf{negl}(\lambda).$$

    By a union bound, with overwhelming probability, $N_{i,b} < 2\lambda/3$ for all $i \in [t]$ and $b \in \{0,1\}$. Correspondingly, the challenger in $\textsc{hyb}_{1,\ell-1'}$ outputs $\perp$ with overwhelming probability in this case.

- Suppose that for all keys $k \in \mathcal{K}_{\mathsf{EPRF}}$, $C_\ell(\cdot) \not\sim_{\varepsilon_2} \mathsf{EX.Eval}(\mathsf{pp}, k, \cdot)$. In $\textsc{hyb}'_{1,\ell-1}$, the challenger will always respond with $\perp$. By $(\varepsilon_1, \varepsilon_2)$-robust extractability of $\Pi_{\mathsf{EPRF}}$, the challenger in $\textsc{hyb}'_{1,\ell}$ will also respond with $\perp$ with overwhelming probability.

In both cases, the distribution of the challenger's response to the $\ell^{\text{th}}$ extraction query in the two hybrids is statistically indistinguishable, and the claim follows. □

Since $Q = \mathsf{poly}(\lambda)$, we conclude from Claim B.13 that $\textsc{hyb}'_1(\mathcal{A}) \overset{s}{\approx} \textsc{hyb}_2(\mathcal{A})$. Finally, since $\textsc{hyb}_1 \overset{s}{\approx} \textsc{hyb}'_1$, the lemma follows. □

**Lemma B.14.** *Suppose algorithm* $\mathsf{TestCandidate}$ *in* $\Pi_{\mathsf{EPRF}}$ *satisfies the additional properties in Remark 4.10. Then, for all (unbounded) adversaries* $\mathcal{A}$, $\mathsf{Adv}_{2,3}(\mathcal{A}) = \mathsf{negl}(\lambda)$.

*Proof.* Follows by an analogous argument as that in the proof of Lemma B.5. □

**Lemma B.15.** *Suppose* $\Pi_{\mathsf{EPRF}}$ *satisfies key-injectivity, selective puncturing security and* $1/|\mathcal{Y}| = \mathsf{negl}(\lambda)$. *Then, for all efficient adversaries* $\mathcal{A}$, $\mathsf{Adv}_{3,4}(\mathcal{A}) = \mathsf{negl}(\lambda)$.

*Proof.* Follows by an analogous argument as that in the proof of Lemma B.6. Note that in contrast to $\textsc{hyb}_4$ in the proof of Theorem 5.21, experiment $\textsc{hyb}_4$ here always outputs $\hat{m}$ if $\mathcal{A}$ queries the marking oracle on $\hat{k}$. By a similar argument as that used in the proof of Lemma B.6, we can show that any efficient adversary that makes a marking oracle query on the challenge key $\hat{k}$ implies an efficient adversary that breaks puncturing security of $\Pi_{\mathsf{EPRF}}$. Thus, an efficient adversary can only trigger this condition and cause the experiment to output $\hat{m}$ with negligible probability. □

**Lemma B.16.** *If $\Pi_{\mathsf{EPRF}}$ satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability with $\varepsilon_2 < 1/2$, then for all (unbounded) adversaries $\mathcal{A}$, $\mathsf{Adv}_{4,5}(\mathcal{A}) = \mathsf{negl}(\lambda)$.*

*Proof.* The only difference between $\textsc{hyb}_4$ and $\textsc{hyb}_5$ is in the way the challenger responds to the post-challenge extraction queries. Similar to the proof of Lemma B.7, we proceed with a query-by-query hybrid. Let $Q = \mathsf{poly}(\lambda)$ be a bound on the number of post-challenge extraction queries the adversary makes. We define a sequence of $Q$ intermediate hybrids as follows:

- $\textsc{hyb}_{4,\ell}$: Same as $\textsc{hyb}_4$ except the first $\ell$ post-challenge extraction queries are handled as in $\textsc{hyb}_5$ while the remaining post-challenge extraction queries are handled as in $\textsc{hyb}_4$.

We additionally define $\textsc{hyb}_{4,Q+1} \equiv \textsc{hyb}_5$. We now show that the output distributions of each consecutive pair of hybrids are statistically indistinguishable.

**Claim B.17.** *If $\Pi_{\mathsf{EPRF}}$ satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability with $\varepsilon_2 < 1/2$, then for all (unbounded) adversaries $\mathcal{A}$ and all $\ell \in [Q+1]$, the distributions $\textsc{hyb}_{4,\ell-1}(\mathcal{A})$ and $\textsc{hyb}_{4,\ell}(\mathcal{A})$ are statistically indistinguishable.*

*Proof.* By construction, $\textsc{hyb}_{4,\ell-1}$ and $\textsc{hyb}_{4,\ell}$ are identical except in the way the challenge responds to the $\ell^{\text{th}}$ post-challenge extraction query $C_\ell \colon \mathcal{X} \to \mathcal{Y}$. We begin by making two observations that will simplify the analysis:

- Let $b_{\hat{k}} = \mathsf{EX.TestCandidate}(\mathsf{pp}, C_\ell, \hat{C}) = 1$. The challenger's behavior in $\textsc{hyb}_{4,\ell-1}$ and $\textsc{hyb}_{4,\ell}$ is identical unless $b_{\hat{k}} = 1$. Thus, it suffices to only consider the setting where $b_{\hat{k}} = 1$.

- If $\mathcal{A}$ makes a marking oracle query on the challenge key $\hat{k}$, the outputs of $\textsc{hyb}_{4,\ell-1}$ and $\textsc{hyb}_{4,\ell}$ is identical (both experiments output $\hat{m}$). It suffices to consider the case where $\mathcal{A}$ never makes a marking oracle query on $\hat{k}$.

We now consider the challenger's behavior in the two experiments.

- In $\textsc{hyb}_{4,\ell-1}$, the challenger first computes $(\hat{x}^*_{i,b,1}, \ldots, \hat{x}^*_{i,b,\lambda}) \leftarrow f(\hat{k}, i, b)$ for all $i \in [t]$ and $b \in \{0,1\}$. Next, for each $i \in [t]$, the challenger computes the following:

  - The number $N_{i,\hat{m}_i}$ of indices $j \in [\lambda]$ where $C_\ell(\hat{x}^*_{i,\hat{m}_i,j}) \neq \hat{y}^*_{i,j}$.
  - The number $N_{i,1-\hat{m}_i}$ of indices $j \in [\lambda]$ where $C_\ell(\hat{x}^*_{i,1-\hat{m}_i,j}) \neq \mathsf{EX.Eval}(\mathsf{pp}, \hat{k}, \hat{x}^*_{i,1-\hat{m}_i,j})$.

  We now argue that with overwhelming probability, $N_{i,\hat{m}_i} > 2\lambda/3$ and $N_{i,1-\hat{m}_i} < 2\lambda/3$. Since we assume $\mathcal{A}$ does not make a marking oracle query on $\hat{k}$, the adversary's view up to the point it makes its $\ell^{\text{th}}$ post-challenge extraction query is independent of $f(\hat{k}, i, b)$ for all $i \in [t]$ and $b \in \{0,1\}$ and $\hat{y}^*_{i,j}$ for all $i \in [t]$ and $j \in [\lambda]$. This means that we can defer the sampling of $f(\hat{k}, i, b)$ and $\hat{y}^*_{i,j}$ until *after* the adversary has submitted its extraction query $C_\ell$. In particular, this means that for each $i \in [t]$, we have that

$$\Pr[\hat{y}^*_{i,1}, \ldots, \hat{y}^*_{i,\lambda} \xleftarrow{\text{R}} \mathcal{Y} : C_\ell(\hat{x}^*_{i,\hat{m}_i,j}) \neq \hat{y}^*_{i,j} \text{ for all } j \in [\lambda]] = (1 - 1/|\mathcal{Y}|)^\lambda = 1 - \mathsf{negl}(\lambda).$$

Equivalently, this means that with overwhelming probability, $N_{i,\hat{m}_i} = \lambda > 2\lambda/3$. Next, since $\hat{b}_k = 1$, it follows by $(\varepsilon_1, \varepsilon_2)$-robust extractability of $\Pi_{\mathsf{EPRF}}$ that with overwhelming probability, $C_\ell \sim_{\varepsilon_2} \mathsf{EX.Eval}(\mathsf{pp}, \hat{k}, \cdot)$, which implies that

$$\Pr[x^* \xleftarrow{\text{R}} \mathcal{X} : C_\ell(x^*) \neq \mathsf{EX.Eval}(\mathsf{pp}, \hat{k}, x^*)] \leq \varepsilon_2 < 1/2.$$

Moreover, since the adversary's view up to its $\ell^{\text{th}}$ post-challenge extraction query is independent of $f(\hat{k}, i, 1 - \hat{m}_i) := (\hat{x}^*_{i,1-\hat{m}_i,1}, \ldots, \hat{x}^*_{i,1-\hat{m}_i,\lambda})$ for all $i \in [t]$, the challenger can sample the value of $(\hat{x}^*_{i,1-\hat{m}_i,1}, \ldots, \hat{x}^*_{i,1-\hat{m}_i,\lambda}) \xleftarrow{\text{R}} \mathcal{X}^\lambda$ after the adversary has submitted its query. By a Chernoff bound,

$$\Pr[(\hat{x}^*_{i,1-\hat{m}_i,1}, \ldots, \hat{x}^*_{i,1-\hat{m}_i,\lambda}) \xleftarrow{\text{R}} \mathcal{X}^\lambda : N_{i,1-\hat{m}_i} \geq 2\lambda/3] \leq 2^{-\Omega(\lambda)} = \mathsf{negl}(\lambda).$$

Using a union bound, we conclude that with overwhelming probability (over the choice of $f$ and $\hat{y}^*_{i,j}$), $N_{i,\hat{m}_i} > 2\lambda/3$ and $N_{i,1-\hat{m}_i} < 2\lambda/3$ for all $i \in [t]$. In this case, the challenger in $\textsc{hyb}_{4,\ell}$ outputs $\hat{m}$.

- In $\textsc{hyb}_{4,\ell}$, the challenger always outputs the challenge message $\hat{m}$.

With overwhelming probability, we see that on the $\ell^{\text{th}}$ query, the challenger in both $\textsc{hyb}_{4,\ell-1}$ and $\textsc{hyb}_{4,\ell}$ will output the challenge message $\hat{m}$ whenever $b_{\hat{k}} = 1$. The claim follows. □

Since $Q = \mathsf{poly}(\lambda)$, we conclude via Claim B.17 that the distributions $\textsc{hyb}_4(\mathcal{A})$ and $\textsc{hyb}_5(\mathcal{A})$ are statistically indistinguishable. □

**Lemma B.18.** *If $\Pi_{\mathsf{EPRF}}$ satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability, then for all (unbounded) and $\varepsilon$-unmoving-admissible adversaries $\mathcal{A}$ where $\varepsilon < \varepsilon_1$,* $\Pr[\textsc{hyb}_5(\mathcal{A}) \neq \hat{m}] = \mathsf{negl}(\lambda)$.

*Proof.* Follows by an analogous argument as that in the proof of Lemma B.9. □

Combining Lemmas B.11 through B.18, the message-embedding watermarking scheme $\Pi_{\mathsf{WM}}$ from Construction 5.25 satisfies $\varepsilon$-unremovability. □

**Proof of Theorem 5.30 (Unforgeability).** The proof relies on a similar sequence of hybrids as that in the proof of Theorem 5.29 and follows an analogous structure as the proof of Theorem 5.22. We briefly describe the hybrid experiments below:

- $\textsc{hyb}_0$: This is the real watermarking security game $\mathsf{ExptWM}_{\Pi_{\mathsf{WM}},\mathcal{A}}(\lambda)$ from Definition 5.10.
- $\textsc{hyb}_1$: Same as $\textsc{hyb}_1$ from the proof of Theorem 5.29.
- $\textsc{hyb}_2$: Same as $\textsc{hyb}_2$ from the proof of Theorem 5.29.

Indistinguishability of hybrids $\textsc{hyb}_0$, $\textsc{hyb}_1$, and $\textsc{hyb}_2$ follow exactly as in the proof of Theorem 5.29. It suffices to show that $\Pr[\textsc{hyb}_2(\mathcal{A}) \neq \perp] = \mathsf{negl}(\lambda)$.

**Lemma B.19.** *If $\Pi_{\mathsf{EPRF}}$ satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability, then for all (unbounded) and $\delta$-unforging-admissible adversaries $\mathcal{A}$ where $\delta \geq \varepsilon_2 + 1/\mathsf{poly}(\lambda)$,* $\Pr[\textsc{hyb}_2(\mathcal{A}) \neq \perp] = \mathsf{negl}(\lambda)$.

*Proof.* Follows by an analogous argument as in the proof of Lemma B.10. □

We conclude that the message-embedding watermarking scheme $\Pi_{\mathsf{WM}}$ from Construction 5.25 satisfies $\delta$-unforgeability. □

### B.3 Analysis of Publically-Markable Watermarking Scheme

In this section, we provide the security analysis of the publically-markable watermarking scheme (Construction 5.32) in the random oracle model.

**Proof of Theorem 5.36 (Unremovability).** The proof is almost identical to that of the proof of Theorem 5.29. For completeness, we describe the full sequence of hybrid experiments below:

- $\textsc{hyb}_0$: This is the real watermarking security game $\mathsf{ExptWM}_{\Pi_{\mathsf{WM}},\mathcal{A}}(\lambda)$ from Definition 5.10. Specifically, the experiment proceeds as follows.
  1. First, the challenger samples $(\mathsf{pp},\mathsf{td}) \leftarrow \mathsf{EX.PrmsGen}(1^\lambda)$. It gives $\mathsf{pp}$ to the adversary.
  2. The challenger responds to the adversary's oracle queries as follows:
     - **Random oracle.** On input a triple $(k,i,b) \in \mathcal{K}_{\mathsf{EPRF}} \times [t] \times \{0,1\}$, the challenger replies with $H(k,i,b)$.
     - **Extraction oracle.** On input a circuit $C\colon \mathcal{X} \to \mathcal{Y}$, the challenger first computes $k \leftarrow \mathsf{EX.Extract}(\mathsf{pp},\mathsf{td},C)$. If $k = \perp$, it outputs $\perp$. Otherwise, it computes $(x^*_{i,b,1},\ldots,x^*_{i,b,\lambda}) \leftarrow H(k,i,b)$ for all $i \in [t]$ and $b \in \{0,1\}$. Let $N_{i,b}$ denote the number of indices $j \in [\lambda]$ where $C(x^*_{i,b,j}) \neq \mathsf{EX.Eval}(\mathsf{pp},k,x^*_{i,b,j})$. If there exists an index $i \in [t]$ where $N_{i,0}, N_{i,1} < 2\lambda/3$ or $N_{i,0}, N_{i,1} > 2\lambda/3$, output $\perp$. Otherwise, for each $i \in [t]$, let $b_i$ be the unique bit where $N_{i,b_i} > 2\lambda/3$, and output the message $m = b_1 \cdots b_t$.
     - **Challenge oracle.** On input a message $\hat{m} \in \{0,1\}^t$, the challenger samples a key $\hat{k} \leftarrow \mathsf{EX.SampleKey}(\mathsf{pp})$ and computes $(\hat{x}^*_{i,1},\ldots,\hat{x}^*_{i,\lambda}) \leftarrow H(\hat{k},i,\hat{m}_i)$ for each $i \in [t]$. Then, the challenger computes the key $\hat{k}' \leftarrow \mathsf{EX.Puncture}(\mathsf{pp},\hat{k},\{\hat{x}^*_{i,j}\}_{i\in[t],j\in[\lambda]})$. It outputs the challenge circuit $\hat{C}\colon \mathcal{X} \to \mathcal{Y}$ where $\hat{C}(\cdot) = \mathsf{EX.PunctureEval}(\mathsf{pp},\hat{k}',\cdot)$.
  3. At the end of the experiment, after the adversary outputs its circuit $\tilde{C}\colon \mathcal{X} \to \mathcal{Y}$, the challenger treats $\tilde{C}$ as an extraction query and proceeds accordingly. The output of the extraction query on $\tilde{C}$ is the output of the experiment.

- $\textsc{hyb}_1$: Same as $\textsc{hyb}_0$, except the challenger initializes an empty set $T$ at the beginning of the experiment. During the experiment, whenever the adversary makes a random oracle query on a value $(k,i,b) \in \mathcal{K}_{\mathsf{EPRF}} \times [t] \times \{0,1\}$, the challenger adds $k$ to $T$ (if it is not already contained in $T$). When the adversary makes a challenge query, the challenger adds the challenge key $\hat{k} \in \mathcal{K}_{\mathsf{EPRF}}$ to $T$. During the experiment (and when computing the output), the challenger uses the following procedure to implement the extraction oracle queries:
  - **Extraction oracle.** On input a circuit $C\colon \mathcal{X} \to \mathcal{Y}$, the challenger computes $b_k \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp},C,k)$ for $k \in T$. If $b_k = 0$ for all $k \in T$, it outputs $\perp$. If $b_k = 1$ for some $k \in T$, the challenger computes $(x^*_{i,b,1},\ldots,x^*_{i,b,\lambda}) \leftarrow H(k,i,b)$ for all $i \in [t]$ and $b \in \{0,1\}$. Let $N_{i,b}$ denote the number of indices $j \in [\lambda]$ where $C(x^*_{i,b,j}) \neq \mathsf{EX.Eval}(\mathsf{pp},k,x^*_{i,b,j})$. If there exists an index $i \in [t]$ where $N_{i,0}, N_{i,1} < 2\lambda/3$ or $N_{i,0}, N_{i,1} > 2\lambda/3$, then output $\perp$. Otherwise, for each $i \in [t]$, let $b_i \in \{0,1\}$ be the unique bit where $N_{i,b_i} > 2\lambda/3$. Output the message $m = b_1 \cdots b_t$.

  In particular, the challenger's behavior in $\textsc{hyb}_1$ no longer depends on the trapdoor $\mathsf{td}$.

- $\textsc{hyb}_2$: Same as $\textsc{hyb}_1$, except whenever the challenger computes $\mathsf{EX.TestCandidate}(\mathsf{pp},\cdot,\hat{k})$, where $\hat{k}$ is the challenge key, it instead computes $\mathsf{EX.TestCandidate}(\mathsf{pp},\cdot,\hat{C})$, where $\hat{C}$ is the challenge circuit.

- $\textsc{hyb}_3$: Same as $\textsc{hyb}_2$, except at the beginning of the security game, the challenger samples random values $\hat{y}^*_{i,j} \xleftarrow{\text{R}} \mathcal{Y}$ for $i \in [t]$ and $j \in [\lambda]$. Then, whenever the challenger needs to compute $\mathsf{EX.Eval}(\mathsf{pp}, \hat{k}, \hat{x}^*_{i,j})$ for $i \in [t]$ and $j \in [\lambda]$ after the adversary has made a challenge query, the challenger uses the value $\hat{y}^*_{i,j}$ instead. Here, $\hat{k}$ is the challenge key, $(\hat{x}^*_{i,1}, \ldots, \hat{x}^*_{i,\lambda})$ is the value of $H(\hat{k}, i, \hat{m}_i)$, and $\hat{m}_i$ is the challenge message. In addition, if $\mathcal{A}$ ever queries the marking oracle on the challenge key $\hat{k}$, the experiment always outputs $\hat{m}$.

- $\textsc{hyb}_4$: Same as $\textsc{hyb}_3$, except the challenger no longer adds the challenge key $\hat{k}$ to the set $T$ when responding to the challenge oracle. Moreover, it uses the following procedure to implement the extraction oracle:
  - **Extraction oracle.** The pre-challenge extraction queries are handled as in $\textsc{hyb}_4$. On a post-challenge extraction query $C \colon \mathcal{X} \to \mathcal{Y}$, the challenger computes $b_{\hat{k}} \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, C, \hat{C})$, and outputs $\hat{m}$ if $b_{\hat{k}} = 1$. Otherwise, the challenger proceeds as in $\textsc{hyb}_4$.

For a hybrid experiment $\textsc{hyb}_i$, we write $\textsc{hyb}_i(\mathcal{A})$ to denote the output distribution of hybrid experiment $\textsc{hyb}_i$ with adversary $\mathcal{A}$. For an integer $i$, we define the advantage $\mathsf{Adv}_{i,i+1}(\mathcal{A})$ as

$$\mathsf{Adv}_{i,i+1}(\mathcal{A}) := \left| \Pr[\textsc{hyb}_0(\mathcal{A}) \neq \hat{m}] - \Pr[\textsc{hyb}_1(\mathcal{A}) \neq \hat{m}] \right|.$$

We now show that for all efficient adversaries $\mathcal{A}$, the advantage between each consecutive pair of hybrid experiments is negligible, and moreover, that $\Pr[\textsc{hyb}_4(\mathcal{A}) \neq \hat{m}] = \mathsf{negl}(\lambda)$. Indistinguishability of each pair of hybrid experiments follows analogously to indistinguishability of the corresponding pair of hybrids in the proof of Theorem 5.29. We state the lemmas below.

**Lemma B.20.** *Suppose* $\Pi_{\mathsf{EPRF}}$ *satisfies key-injectivity and* $(\varepsilon_1, \varepsilon_2)$*-robust extractability for* $\varepsilon_2 < 1/2$. *Then, for all (unbounded) adversaries* $\mathcal{A}$, $\mathsf{Adv}_{0,1}(\mathcal{A}) = \mathsf{negl}(\lambda)$.

*Proof.* Follows by an analogous argument as in the proof of Lemma B.12, except the random oracle $H(\cdot)$ takes the place of the random function $f(\cdot)$ and random oracle queries play the role of marking oracle queries. □

**Lemma B.21.** *Suppose algorithm* $\mathsf{TestCandidate}$ *in* $\Pi_{\mathsf{EPRF}}$ *satisfies the additional properties in Remark 4.10. Then, for all (unbounded) adversaries* $\mathcal{A}$, $\mathsf{Adv}_{1,2}(\mathcal{A}) = \mathsf{negl}(\lambda)$.

*Proof.* Follows by an analogous argument as in the proof of Lemmas B.5 and B.14. □

**Lemma B.22.** *Suppose* $\Pi_{\mathsf{EPRF}}$ *satisfies key-injectivity, selective puncturing security, and* $1/|\mathcal{Y}| = \mathsf{negl}(\lambda)$. *Then, for all efficient adversaries* $\mathcal{A}$, $\mathsf{Adv}_{2,3}(\mathcal{A}) = \mathsf{negl}(\lambda)$.

*Proof.* Follows by an analogous argument as in the proofs of Lemmas B.6 and B.15, except the random oracle $H(\cdot)$ takes the place of the random function $f(\cdot)$ and random oracle queries play the role of marking oracle queries. □

**Lemma B.23.** *If* $\Pi_{\mathsf{EPRF}}$ *satisfies* $(\varepsilon_1, \varepsilon_2)$*-robust extractability with* $\varepsilon_2 < 1/2$, *then for all (unbounded) adversaries* $\mathcal{A}$, $\mathsf{Adv}_{3,4}(\mathcal{A}) = \mathsf{negl}(\lambda)$.

*Proof.* Follows by an analogous argument as in the proofs of Lemma B.16, except the random oracle $H(\cdot)$ takes the place of the random function $f(\cdot)$ and random oracle queries play the role of marking oracle queries. □

**Lemma B.24.** *If $\Pi_{\mathsf{EPRF}}$ satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability, then for all (unbounded) and $\varepsilon$-unremoving-admissible adversaries $\mathcal{A}$ where $\varepsilon < \varepsilon_1$, $\Pr[\textsc{hyb}_4(\mathcal{A}) \neq \hat{m}] = \mathsf{negl}(\lambda)$.*

*Proof.* Follows by an analogous argument as in the proof of Lemmas B.9 and B.18. □

Combining Lemmas B.20 through B.24, the message-embedding watermarking scheme $\Pi_{\mathsf{WM}}$ from Construction 5.32 satisfies $\varepsilon$-unremovability in the random oracle model. □

**Proof of Theorem 5.37 (Weak Unforgeability).** The proof relies on a similar sequence of hybrids as that in the proof of Theorem 5.36 and follows an analogous structure as the proofs of Theorems 5.22 and 5.30. We briefly describe the hybrid experiments below:

- $\textsc{hyb}_0$: This is the real watermarking security game $\mathsf{ExptWM}_{\Pi_{\mathsf{WM}},\mathcal{A}}(\lambda)$ from Definition 5.10.
- $\textsc{hyb}_1$: Same as $\textsc{hyb}_1$ from the proof of Theorem 5.36.

Indistinguishability of hybrids $\textsc{hyb}_0$ and $\textsc{hyb}_1$ follow exactly as in the proof of Theorem 5.36. It suffices to show that $\Pr[\textsc{hyb}_1(\mathcal{A}) \neq \perp] = \mathsf{negl}(\lambda)$.

**Lemma B.25.** *If $\Pi_{\mathsf{EPRF}}$ satisfies $(\varepsilon_1, \varepsilon_2)$-robust extractability, then for all (unbounded) and weak $\delta$-unforging-admissible adversaries $\mathcal{A}$ where $\delta \geq \varepsilon_2$, $\Pr[\textsc{hyb}_1(\mathcal{A}) \neq \perp] = \mathsf{negl}(\lambda)$.*

*Proof.* Since $\mathcal{A}$ is weak $\delta$-unforging admissible, the circuit $\tilde{C} \colon \mathcal{X} \to \mathcal{Y}$ output by $\mathcal{A}$ satisfies $\tilde{C}(\cdot) \not\sim_\delta \mathsf{EX.Eval}(\mathsf{pp}, k, \cdot)$ for all $k \in \mathcal{K}$. Since $\delta \geq \varepsilon_2$, this means that $\tilde{C}(\cdot) \not\sim_{\varepsilon_2} \mathsf{EX.Eval}(\mathsf{pp}, k, \cdot)$ for all $k \in \mathcal{K}$. By robust extractability and union bounding over all $|T| = \mathsf{poly}(\lambda)$ marking oracle queries the adversary makes, we have that

$$\Pr[\exists k \in T : \mathsf{EX.TestCandidate}(\mathsf{pp}, \tilde{C}, k) = 1] = \mathsf{negl}(\lambda).$$

By construction then, the output in $\textsc{hyb}_1$ is $\perp$ with overwhelming probability, and the claim follows. □

We conclude that the message-embedding watermarking scheme $\Pi_{\mathsf{WM}}$ from Construction 5.32 satisfies weak $\delta$-unforgeability in the random oracle model. □

**Proof of Theorem 5.38 (Extended Pseudorandomness).** Recall that in the extended pseudorandomness security game is the standard pseudorandomness game for the watermarking game, except the adversary is additionally given access to an extraction oracle. We take a similar proof strategy as that used in the proof of Theorem 5.37 for arguing unforgeability.

- $\textsc{hyb}_0$: This is the extended pseudorandomness game where the challenger answer the adversary's queries using the real PRF evaluation. Specifically, the game proceeds as follows:
  1. At the beginning of the game, the challenger samples the parameters for the watermarking scheme $(\mathsf{pp}, \mathsf{wsk}) \leftarrow \mathsf{Setup}(1^\lambda)$. It also samples a PRF key $k \leftarrow \mathsf{EX.SampleKey}(\mathsf{pp})$. It gives $\mathsf{pp}$ to the adversary.
  2. The adversary is now allowed to make random oracle queries, evaluation queries, and extraction queries, which the challenger responds to as follows:
     - **Random oracle.** On input $(k, i, b) \in k_{\mathsf{PRF}} \times [t] \times \{0,1\}$, the challenger replies with $H(k, i, b)$.
     - **Evaluation oracle.** On input a point $x \in \mathcal{X}$, the challenger replies with $\mathsf{F}(k, x)$
     - **Extraction oracle.** On input a circuit $C \colon \mathcal{X} \to \mathcal{Y}$, the challenger replies with $\mathsf{Extract}(\mathsf{wsk}, C)$.

3. At the end of the game, the adversary outputs a bit $b \in \{0,1\}$, which is also the output of the experiment.

- $\text{HYB}_1$: Same as $\text{HYB}_0$ except the challenger initializes an empty set $T$ at the beginning of the experiment. During the experiment, whenever the adversary makes a random oracle query on a value $(k, i, b) \in \mathcal{K}_{\mathsf{EPRF}} \times [t] \times \{0,1\}$, the challenger adds $k$ to $T$ (if it is not already contained in $T$). During the experiment, the challenger uses the following procedure to implement the extraction oracle queries:

  - **Extraction oracle.** On input a circuit $C \colon \mathcal{X} \to \mathcal{Y}$, the challenger computes $b_k \leftarrow \mathsf{EX.TestCandidate}(\mathsf{pp}, C, k)$ for each $k \in T$. If $b_k = 0$ for all $k \in T$, it outputs $\perp$. If $b_k = 1$ for some $k \in T$, the challenger computes $(x^*_{i,b,1}, \ldots, x^*_{i,b,\lambda}) \leftarrow H(k, i, b)$ for all $i \in [t]$ and $b \in \{0,1\}$. Let $N_{i,b}$ denote the number of indices $j \in [\lambda]$ where $C(x^*_{i,b,j}) \neq \mathsf{EX.Eval}(\mathsf{pp}, k, x^*_{i,b,j})$. If there exists an index $i \in [t]$ where $N_{i,0}, N_{i,1} < 2\lambda/3$ or $N_{i,0}, N_{i,1} > 2\lambda/3$, then output $\perp$. Otherwise, for each $i \in [t]$, let $b_i \in \{0,1\}$ be the unique bit where $N_{i,b_i} > 2\lambda/3$. Output the message $m = b_1 \cdots b_t$.

  In particular, the challenger's behavior in $\text{HYB}_1$ no longer depends on the watermarking secret key $\mathsf{wsk}$.

- $\text{HYB}_2$: Same as $\text{HYB}_1$ except at the beginning of the experiment, the challenger samples a random function $f \xleftarrow{\text{R}} \mathsf{Funs}[\mathcal{X}, \mathcal{Y}]$, and responds to the evaluation oracle queries on $x \in \mathcal{X}$ with $f(x)$.

- $\text{HYB}_3$: Same as $\text{HYB}_2$ except the challenger responds to the extraction oracle queries on a circuit $C \colon \mathcal{X} \to \mathcal{Y}$ using the honest extraction procedure $\mathsf{Extract}(\mathsf{wsk}, C)$. This is the extended pseudorandomness game where the challenger answers the adversary's queries using random values.

We note that the outputs of $\text{HYB}_0$ and $\text{HYB}_1$ are computationally indistinguishable by robust extractability of $\Pi_{\mathsf{EPRF}}$ via the same argument as that given in the proof of Lemma B.20. Observe now that the behavior of the extraction oracle in $\text{HYB}_1$ and $\text{HYB}_2$ no longer depends on any secrets of the watermarking scheme (and so the adversary itself can simulate the behavior of the extraction oracle). Thus, the outputs of $\text{HYB}_1$ and $\text{HYB}_2$ are computationally indistinguishable by standard pseudorandomness of $\Pi_{\mathsf{WM}}$ (Theorem 5.34). Finally, indistinguishability of $\text{HYB}_2$ and $\text{HYB}_3$ follow again by robust extractability of $\Pi_{\mathsf{EPRF}}$ (via the same argument as that used in the proof of Lemma B.20). Since the outputs of $\text{HYB}_0$ and $\text{HYB}_3$ are computationally indistinguishable, we conclude that $\Pi_{\mathsf{WM}}$ satisfies extended pseudorandomness in the random oracle model. □