Watermarking Cryptographic Functionalities from Standard Lattice Assumptions

Sam Kim
Stanford University
skim13@cs.stanford.edu

David J. Wu
Stanford University
dwu4@cs.stanford.edu

Abstract

A software watermarking scheme allows one to embed a "mark" into a program without significantly altering the behavior of the program. Moreover, it should be difficult to remove the watermark without destroying the functionality of the program. Recently, Cohen et al. (STOC 2016) and Boneh et al. (PKC 2017) showed how to watermark cryptographic functions such as PRFs using the full power of general-purpose indistinguishability obfuscation. Notably, in their constructions, the watermark remains intact even against *arbitrary* removal strategies. A natural question is whether we can build watermarking schemes from standard assumptions that achieve this strong mark-unremovability property.

We give the first construction of a watermarkable family of PRFs that satisfy this strong mark-unremovability property from standard lattice assumptions (namely, the learning with errors (LWE) and the one-dimensional short integer solution (SIS) problems). As part of our construction, we introduce a new cryptographic primitive called a translucent PRF. Next, we give a concrete construction of a translucent PRF family from standard lattice assumptions. Finally, we show that using our new lattice-based translucent PRFs, we obtain the first watermarkable family of PRFs with strong unremovability against arbitrary strategies from standard assumptions.

1 Introduction

A software watermarking scheme enables one to embed a "mark" into a program such that the marked program behaves almost identically to the original program. At the same time, it should be difficult for someone to remove the mark without significantly altering the behavior of the program. Watermarking is a powerful notion that has many applications for digital rights management, such as tracing information leaks or resolving ownership disputes. Although the concept itself is quite natural, and in spite of its numerous potential applications, a rigorous theoretical treatment of the notion was given only recently [BGI+01, HMW07, BGI+12].

Constructing software watermarking with strong security guarantees has proven difficult. Early works on cryptographic watermarking [NSS99, YF11, Nis13] could only achieve mark-unremovability against adversaries who can only make a *restricted* set of modifications to the marked program. The more recent works [CHN+16, BLW17] that achieve the strongest notion of unremovability against *arbitrary* adversarial strategies all rely on the extremely strong cryptographic assumption of general-purpose indistinguishability obfuscation [BGI+01, GGH+13]. In this paper, we focus on constructions that achieve the stronger notion of mark-unremovability against arbitrary removal strategies.

Existing constructions of software watermarking [NSS99, YF11, Nis13, CHN+16, BLW17] with formal security guarantees focus primarily on watermarking cryptographic functions. Following [CHN+16, BLW17], we consider watermarking for PRFs. In this work, we give the first watermarkable family of PRFs from *standard assumptions* that provides mark-unremovability against *arbitrary* adversarial strategies. All previous watermarking constructions [CHN+16, BLW17] that could achieve this notion relied on indistinguishability obfuscation. Indeed, as we discuss in Section 1.2, this notion of software watermarking bears a striking resemblance to the problem of program obfuscation, and so, it is not entirely surprising that the existing constructions all rely on indistinguishability obfuscation.

To construct our watermarkable family of PRFs, we first introduce a new cryptographic primitive we call *translucent constrained PRFs*. We then show how to use translucent constrained PRFs to build a watermarkable family of PRFs. Finally, we leverage a number of lattice techniques (outlined in Section 2) to construct a translucent PRF. Putting these pieces together, we obtain the first watermarkable family of PRFs with strong mark-unremovability guarantees from standard assumptions. Thus, this work broadens our abilities to construct software watermarking, and we believe that by leveraging and extending our techniques, we will see many new constructions of cryptographically-strong watermarking for new functionalities (from standard assumptions) in the future.

1.1 Background

The mathematical foundations of digital watermarking were first introduced by Barak et al. [BGI+01, BGI+12] in their seminal work on cryptographic obfuscation. Unfortunately, their results were largely negative, for they showed that assuming indistinguishability obfuscation, then certain forms of software watermarking cannot exist. Central to their impossibility result is the assumption that the underlying watermarking scheme is *perfect functionality-preserving*. This requirement stipulates that the input/output behavior of the watermarked program is identical to the original unmarked program on all input points. By relaxing this requirement to allow the watermarked program to differ from the original program on a small number (i.e., a negligible fraction) of the points in the domain, Cohen et al. [CHN+16] gave the first construction of an *approximate functionality-preserving* watermarking scheme for a family of pseudorandom functions (PRFs) using indistinguishability obfuscation.

Watermarking circuits. A watermarking scheme for circuits consists of two algorithms: a marking algorithm and a verification algorithm. The marking algorithm is a keyed algorithm takes as input a circuit C and outputs a new circuit C' such that on almost all inputs x, $C'(x) = C(x)$. In other words, the watermarked program preserves the functionality of the original program on almost all inputs. The verification algorithm then takes as input a circuit C' and either outputs "marked" or "unmarked." The correctness requirement is that any circuit output by the marking algorithm should be regarded as "marked" by the verification algorithm. A watermarking scheme is said to be publicly-verifiable if anyone can test whether a circuit is watermarked or not, and secretly-verifiable if only the holder of the watermarking key is able to test whether a program is watermarked.

The primary security property a software watermarking scheme must satisfy is unremovability, which roughly says that given a watermarked circuit C, the adversary cannot produce a new circuit $\tilde{C}$ whose functionality is similar to C, and yet is not considered to be marked (from the perspective of the verification algorithm). The definition can be strengthened by also allowing the adversary to obtain marked circuits of its choosing. A key source of difficulty in achieving unremovability is that we allow the adversary *complete freedom* in crafting its circuit $\tilde{C}$. All existing constructions of watermarking from standard assumptions [NSS99, YF11, Nis13] constrain the output or power of the adversary (e.g., the adversary's output must consist of a tuple of group elements). In contrast, the works of Cohen et al. [CHN+16], Boneh et al. [BLW17], and this work protect against *arbitrary* removal strategies.

A complementary security property to unremovability is unforgeability, which says that an adversary who does not possess the watermarking secret key is unable to construct a new program (i.e., one sufficiently different from any watermarked programs the adversary might have seen) that is deemed to be watermarked (from the perspective of the verification algorithm). As noted by Cohen et al. [CHN+16], unforgeability and unremovability are oftentimes conflicting requirements, and depending on the precise definitions, may not be simultaneously satisfiable. In this work, we consider a natural setting where both conditions are simultaneously satisfiable (and in fact, our construction achieves exactly that).

Watermarking PRFs. Following Cohen et al. [CHN+16] and Boneh et al. [BLW17], we focus on watermarking cryptographic functions, specifically PRFs, in this work. Previously, Cohen et al. [CHN+16] demonstrated that many natural classes of functions, such as any efficiently learnable class of functions, cannot be watermarked. A canonical and fairly natural class of non-learnable functionalities are cryptographic ones. Moreover, watermarking PRFs already suffices for a number of interesting applications; we refer to [CHN+16] for the full details.

Building software watermarking. We begin by describing the high-level blueprint introduced by Cohen et al. [CHN+16] for constructing watermarkable PRFs.[1] To watermark a PRF F with key k, the marking algorithm first evaluates the PRF on several (secret) points $h_1, \ldots, h_d$ to obtain values $t_1, \ldots, t_d$. Then, the marking algorithm uses the values $(t_1, \ldots, t_d)$ to derive a (pseudorandom) pair (x^*, y^*). The watermarked program is a circuit C that on all inputs $x \neq x^*$, outputs $F(k, x)$, while on input x^*, it outputs the special value y^*. To test whether a program C' is marked or not, the verification algorithm first evaluates C' on the secret points $h_1, \ldots, h_d$. It uses the function evaluations to derive the test pair (x^*, y^*). Finally, it evaluates the program at x^* and outputs "marked" if $C'(x^*) = y^*$; otherwise, it outputs "unmarked." For this scheme to be secure against *arbitrary* removing strategies, it must be the case that the watermarked circuit C hides the marked point x^* from the adversary. Moreover, the value y^* at the "reprogrammed" point should not be easily identifiable. Otherwise, an adversary can trivially defeat the watermarking scheme by simply producing a circuit that behaves just like C, but outputs $\perp$ whenever it is queried on the special point x^*. In some sense, security requires that the point x^* is carefully embedded within the description of the watermarked program such that no efficient adversary is able to identify it (or even learn partial information about it). This apparent need to embed a secret within a piece of code is heavily reminiscent of program obfuscation, so not surprisingly, the existing constructions of software watermarking rely on the full power of indistinguishability obfuscation.

Puncturable and programmable PRFs. The starting point of our watermarking construction is the recent construction by Boneh et al. [BLW17] (which follows the Cohen et al. [CHN+16] blueprint sketched above). In their work, they introduce the notion of a *private puncturable* PRF. In a regular puncturable PRF [BW13, KPTZ13, BGI14], the holder of the PRF key can issue a "punctured" key sk_{x^*} such that sk_{x^*} can be used to evaluate the PRF everywhere except at a single point x^*. In a private puncturable PRF, the punctured key sk_{x^*} also hides the punctured point x^*. Intuitively, private puncturing seems to get us halfway to the goal of constructing a watermarkable family of PRFs according to the blueprint above. After all, a private puncturable PRF allows issuing keys that agree with the real PRF almost everywhere, and yet, the holder of the punctured key cannot tell which point was punctured. Unfortunately, in a standard private puncturable PRF, it is difficult to fully reprogram the value at the punctured point, and thus, there is no efficient way to test whether a particular point, and correspondingly, the program, has been marked or not.

To bridge the gap between private puncturable PRFs and watermarkable PRFs, Boneh et al. introduced a stronger notion called a private *programmable* PRF which, as the name suggests, allows one to reprogram the PRF value at the punctured point. This modification allows them to instantiate the Cohen et al. blueprint above. However, private programmable PRFs seem much more difficult to construct than a private puncturable PRF, and the construction in [BLW17] relies on indistinguishability obfuscation. In contrast, Boneh et al. [BLW17] were able to construct private puncturable PRFs from concrete assumptions on multilinear maps, and subsequently, Canneti and Chen [CC17], as well as Boneh et al. [BKM17] showed how to construct private puncturable PRFs from standard lattice assumptions.

1.2 Our Contributions

While the high-level framework of Cohen et al. [CHN+16] provides an elegant approach for building watermarkable PRFs (and by extension, other cryptographic functionalities), realizing it without relying on some form of obfuscation is quite challenging. Our primary contribution in this work is showing that it is possible to construct a watermarkable family of PRFs (in the secret-key setting) while only relying on standard lattice assumptions (namely, on the subexponential hardness[2] of LWE and 1D-SIS). Thus, this work gives the first

[1]There are numerous technicalities in the actual construction, but these are not essential to understanding the main intuition.

[2]The need for the less standard (though still widely used) subexponential hardness is due to the fact that we use "complexity leveraging" [BB04] to show that our watermarkable family of PRFs satisfies adaptive security. If selective security suffices, then our construction is secure assuming polynomial hardness of LWE and 1D-SIS.

construction of a mathematically-sound watermarking construction for a nontrivial family of cryptographic primitives from standard assumptions. In this section, we give a brief overview of our main construction and results. Then, in Section 2, we give a more detailed technical overview of our lattice-based watermarking construction.

Relaxing programmability. The work of Boneh, Lewi, and Wu [BLW17] introduces two closely-related notions: private puncturable PRFs and private programmable PRFs. Despite their similarities, private programmable PRFs give a direct construction of watermarking while private puncturable PRFs do not seem sufficient for watermarking. On the other hand, the recent works of Boneh, Kim, and Montgomery [BKM17] as well as Canetti and Chen [CC17] give constructions of private puncturable PRFs from standard lattice assumptions, while to date, the only construction of a private programmable PRF still relies on indistinguishability obfuscation.

A promising starting point then for realizing watermarking from standard assumptions is to try and extend the lattice-based private puncturing constructions to obtain a private programmable PRF. However, lattice-based constrained PRF constructions [BV15, BFP+15, BKM17, CC17] in general lack the structure for full programmability. For instance, in [BV15, BFP+15, BKM17], the PRF evaluations consist of computing a linear combination of multiple LWE samples and then rounding the result. These vectors only span a subspace of the output space of the PRF, and hence, does not seem to allow arbitrary reprogramming. The recent construction of Canetti and Chen [CC17] based on a secure variant of the Gentry et al. [GGH15] graph-induced multilinear map relies on structured LWE samples where a (secret) matrix branching program is embedded within the LWE secrets themselves. This structural requirement does not seem to enable arbitrary reprogramming of the PRF value at the constrained point.

In this work, we instead take a "meet-in-the-middle" approach to solve this problem. First, we identify an intermediate notion that interpolates between private puncturable PRFs and private programmable PRFs. For reasons described below, we refer to our new primitive as a private translucent PRF. The advantages to defining this new notion are twofold. First, we show how to augment and extend the Boneh et al. [BKM17] private puncturable PRF to obtain a private translucent PRF from standard lattice assumptions. Second, we show that private translucent PRFs still suffice to instantiate the rough blueprint in [CHN+16] for building cryptographic watermarking schemes. Together, these ingredients yield the first (secretly-verifiable) watermarkable family of PRFs from standard assumptions.

Private translucent PRFs. The key cryptographic primitive we introduce in this work is the notion of a translucent puncturable PRF. To keep the description simple, we refer to it as a "translucent PRF" in this section. As described above, private translucent PRFs interpolate between private puncturable PRFs and private programmable PRFs. We begin by describing the notion of a (non-private) translucent PRF. A translucent PRF consists of a set of public parameters pp and a secret testing key tk. Unlike standard puncturable and programmable PRFs, each translucent PRF (specified by $(\mathsf{pp}, \mathsf{tk})$) defines an entire *family* of puncturable PRFs over a domain $\mathcal{X}$ and range $\mathcal{Y}$, and which share a common set of public parameters. More precisely, translucent PRFs implement a $\mathsf{SampleKey}$ algorithm which, on input the public parameters pp, samples a PRF key k from the underlying puncturable PRF family. The underlying PRF family associated with pp is puncturable, so all of the keys k output by $\mathsf{SampleKey}$ can be punctured.

The defining property of a translucent PRF is that when a punctured key sk_{x^*} (derived from some PRF key k output by $\mathsf{SampleKey}$) is used to evaluate the PRF at the punctured point x^*, the resulting value lies in a specific subset $S \subset \mathcal{Y}$. Moreover, when the punctured key sk_{x^*} is used to evaluate at any non-punctured point $x \neq x^*$, the resulting value lies in $\mathcal{Y} \setminus S$ with high probability. The particular subset S is *global* to all PRFs in the punctured PRF family, and moreover, is uniquely determined by the public parameters of the overall translucent PRF. The second requirement we require of a translucent PRF is that the secret testing key tk can be used to test whether a particular value $y \in \mathcal{Y}$ lies in the subset S or not. In other words, given only the evaluation output of a punctured key sk_{x^*} on some input x, the holder of the testing key can efficiently tell whether $x = x^*$ (without any knowledge of sk_{x^*} or its associated PRF key k).

In a *private* translucent PRF, we impose the additional requirement that the underlying puncturable PRF family is privately puncturable (that is, the punctured keys also hide the punctured point). An immediate consequence of the privacy requirement is that whenever a punctured key is used to evaluate the PRF at a punctured point, the output value (contained in S) should look indistinguishable from a random value in the range $\mathcal{Y}$. If elements in S are easily distinguishable from elements in $\mathcal{Y} \setminus S$ (without tk), then an adversary can efficiently test whether a punctured key is punctured at a particular point x, thus breaking privacy. In particular, this means that S must be a sparse hidden subset of $\mathcal{Y}$ such that anyone who does not possess the testing key tk cannot distinguish elements in S from elements in $\mathcal{Y}$. Anyone who possesses the testing key, however, should be able to tell whether a particular element is contained in S or not. Moreover, all of these properties should hold even though it is *easy* to publicly sample elements from S (the adversary can always sample a PRF key k using $\mathsf{SampleKey}$, puncture k at any point x^*, and then evaluate the punctured key at x^*). Sets $S \subset \mathcal{Y}$ that satisfy these properties were referred to as "translucent sets" in the work of Canetti et al. [CDNO97] on constructing deniable encryption. In our setting, the outputs of the punctured PRF keys in a private translucent PRF precisely implement a translucent set system, hence the name "translucent PRF."

From private translucency to watermarking. Once we have a private translucent PRF, it is fairly straightforward to obtain from it a family of watermarkable PRFs. Our construction roughly follows the high-level blueprint described in [CHN+16]. Take any private translucent PRF with public parameters pp and testing key tk. We now describe a (secretly-verifiable) watermarking scheme for the family of private puncturable PRFs associated with pp. The watermarking secret key consists of several randomly chosen domain elements $h_1, \ldots, h_d \in \mathcal{X}$ and the testing key tk for the private translucent PRF. To watermark a PRF key k (output by $\mathsf{SampleKey}$), the marking algorithm evaluates the PRF on $h_1, \ldots, h_d$ and uses the outputs to derive a special point $x^* \in \mathcal{X}$. The watermarked key sk_{x^*} is the key k punctured at the point x^*. By definition, this means that if the watermarked key sk_{x^*} is used to evaluate the PRF at x^*, then the resulting value lies in the hidden sparse subset $S \subseteq \mathcal{Y}$ specific to the private translucent PRF.

To test whether a particular program (i.e., circuit) is marked, the verification algorithm first evaluates the circuit at $h_1, \ldots, h_d$. Then, it uses the evaluations to derive the special point x^*. Finally, the verification algorithm evaluates the program at x^* to obtain a value y^*. Using the testing key tk, the verification algorithm checks to see if y^* lies in the hidden set S associated with the public parameters of the private translucent PRF. Correctness follows from the fact that the punctured key is functionality-preserving (i.e., computes the PRF correctly at all but the punctured point). Security of the watermarking scheme follows from the fact that the watermarked key hides the special point x^*. Furthermore, the adversary cannot distinguish the elements of the hidden set S from random elements in the range $\mathcal{Y}$. Intuitively then, the only effective way for the adversary to remove the watermark is to change the behavior of the marked program on many points (i.e., at least one of $h_1, \ldots, h_d, x^*$). But to do so, we show that such an adversary necessarily corrupts the functionality on a noticeable fraction of the domain. In Section 6, we formalize these notions and show that every private translucent PRF gives rise to a watermarkable family of PRFs.

Message-embedding via t-puncturing. Previous watermarking constructions [CHN+16, BLW17] also supported a stronger notion of watermarking called "message-embedding" watermarking. In a message-embedding scheme, the marking algorithm also takes as input a message $m \in \{0,1\}^t$ and outputs a watermarked program with the message m embedded within it. The verification algorithm is replaced with an extraction algorithm which takes as input a watermarked program (and in the secret-key setting, the watermarking secret key), and either outputs "unmarked" or the embedded message. The unremovability property is strengthened to say that given a program with an embedded message m, the adversary cannot produce a similar program on which the extraction algorithm outputs something other than m. Both the Cohen et al. as well as the Boneh et al. constructions leverage reprogrammability to obtain a message-embedding watermarking scheme—that is, the program's outputs on certain special inputs are modified to contain a (blinded) version of m (which the verification algorithm can then extract).

A natural question is whether our construction based on private translucent PRFs can be used to obtain a message-embedding watermarkable family of PRFs. The key barrier, of course, seems to be the fact that private translucent PRFs do not allow much flexibility in programming the actual value to which a punctured key evaluates on a punctured point. We can only ensure that it lies in some translucent set S. To achieve message-embedding watermarking, we require a different method of embedding the message. Our solution contains two key ingredients:

- First, we introduce a notion of private t-puncturable PRFs, which is a natural extension of puncturing where the punctured keys are punctured on a set of exactly t points in the domain rather than a single point. Fortunately, for small values of t (i.e., polynomial in the security parameter), our private translucent PRF construction (Section 5) can be modified to support keys punctured at t points rather than a single point. The other properties of translucent PRFs remain intact (i.e., whenever a t-punctured key is used to evaluate at any one of the t punctured points, the result of the evaluation lies in the translucent subset $S \subset \mathcal{Y}$).
- To embed a message $m \in \{0,1\}^t$, we follow the same blueprint as before, but instead of deriving a single special point x^*, the marking algorithm instead derives $2 \cdot t$ (pseudorandom) points $x_1^{(0)}, x_1^{(1)}, \ldots, x_t^{(0)}, x_t^{(1)}$. The watermarked key is a t-punctured key, where the t points are chosen based on the bits of the message. Specifically, to embed a message $m \in \{0,1\}^t$ into a PRF key k, the marking algorithm punctures k at the points $x_1^{(m_1)}, \ldots, x_t^{(m_t)}$. The extraction procedure works similarly to the verification procedure in the basic construction. It first evaluates the program on the set of (hidden) inputs, and uses the program outputs to derive the values $x_i^{(b)}$ for all $i = 1, \ldots, t$ and $b \in \{0,1\}$. For each index $i = 1, \ldots, t$, the extraction algorithm tests whether the program's output at $x_i^{(0)}$ or $x_i^{(1)}$ lies within the translucent set S. In this way, the extraction algorithm is able to extract the bits of the message.

Thus, without much additional overhead (i.e., proportional to the bit-length of the embedded messages), we obtain a message-embedding watermarking scheme from standard lattice assumption.

Constructing translucent PRFs. Another technical contribution in this work is a new construction of a private translucent PRF (that supports t-puncturing) from standard lattice assumptions. The starting point of our private translucent PRF construction is the private puncturable PRF construction of Boneh et al. [BKM17]. We provide a detailed technical overview of our algebraic construction in Section 2, and the concrete details of the construction (with accompanying security proofs) in Section 5. Here, we provide some intuition on how we construct a private translucent PRF (for the simpler case of puncturing). Recall first that the construction of Boneh et al. gives rise to a PRF with output space $\mathbb{Z}_p^m$. In our private translucent PRF construction, the translucent set is chosen to be a random *noisy* 1-dimensional subspace within $\mathbb{Z}_p^m$. By carefully exploiting the specific algebraic structure of the Boneh et al. PRF, we ensure that whenever an (honestly-generated) punctured key is used to evaluate on a punctured point, the evaluation outputs a vector in this random subspace (with high probability). The testing key simply consists of a vector that is essentially orthogonal to the hidden subspace. Of course, it is critical here that the hidden subspace is noisy. Otherwise, since the adversary is able to obtain arbitrary samples from this subspace (by generating and puncturing keys of its own), it can trivially learn the subspace, and thus, efficiently decide whether a vector lies in the subspace or not. Using a noisy subspace enables us to appeal to the hardness of LWE and 1D-SIS to argue security of the overall construction. We refer to the technical overview in Section 2 as well as the concrete description in Section 5 for the full details.

1.3 Additional Related Work

Much of the early (and ongoing) work on digital watermarking have focused on watermarking digital media, such as images or video. These constructions tend to be ad hoc, and lack a firm theoretical foundation. We refer to [CMB+07] and the references therein for a comprehensive survey of the field. The work of Hopper, Molnar, and Wagnar [HMW07] gives the first formal and rigorous definitions for a digital watermarking scheme, but they do not provide any concrete constructions. In the same work, Hopper et al. also introduce the formal notion of secretly-verifiable watermarking, which is the focus of this work.

Early works on cryptographic watermarking [NSS99, YF11, Nis13] gave constructions that achieved mark-unremovability against adversaries who could only make a *restricted* set of modifications to the marked program. The work of Nishimaki [Nis13] showed how to obtain message-embedding watermarking using a bit-by-bit embedding of the message within a dual-pairing vector space (specific to his particular construction). Our message-embedding construction in this paper also takes a bit-by-bit approach, but our technique is more general: we show that any translucent t-puncturable PRF suffices for constructing a watermarkable family of PRFs that supports embedding of t-bit messages.

In a recent work, Nishimaki, Wichs, and Zhandry [NWZ16] show how to construct a traitor tracing scheme where arbitrary data can be embedded within a decryption key (which can be recovered by a tracing algorithm). While the notion of message-embedding traitor tracing is conceptually similar to software watermarking, the notions are incomparable. In a traitor tracing scheme, there is a single decryption key and a central authority who issues the marked keys. Conversely, in a watermarking scheme, the keys can be chosen by the *user*, and moreover, different keys (implementing different functions) can be watermarked.

PRFs from LWE. The first PRF constructions from LWE was due to Banerjee, Peikert, and Rosen in [BPR12]. Subsequently, [BLMR13, BP14] gave the first lattice-based key-homomorphic PRFs. These constructions were then generalized to the setting of constrained PRFs in [BV15, BFP+15, BKM17]. Recently, Canetti and Chen [CC17] showed how certain secure modes of operation of the Gentry et al. [GGH15] multilinear map can be used to construct a private constrained PRF for the class of NC^1 constraints (with hardness reducing to the LWE assumption).

ABE and PE from LWE. The techniques used in this work build on a series of works in the area of *attribute-based encryption* [SW05] and *predicate encryption* [BW07, KSW08] from LWE. These include the attribute-based encryption constructions of [ABB10, GVW13, BGG+14, GV15, BV16, BCTW16], and predicate encryption constructions of [AFV11, GMW15, GVW15].[3]

2 Construction Overview

In this section, we give a technical overview of our private translucent t-puncturable PRF from standard lattice assumptions. As described in Section 1, this directly implies a watermarkable family of PRFs from standard lattice assumptions. The formal definitions, constructions and accompanying proofs of security are given in Sections 4 and 5. The watermarking construction is given in Section 6.

The LWE assumption. The learning with errors (LWE) assumption [Reg05], parameterized by n, m, q, χ, states that for a uniformly random vector $\mathbf{s} \in \mathbb{Z}_q^n$ and a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, the distribution $(\mathbf{A}, \mathbf{s}^T\mathbf{A} + \mathbf{e}^T)$ is computationally indistinguishable from the uniform distribution over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$, where $\mathbf{e}$ is sampled from a (low-norm) error distribution χ. To simplify the presentation in this section, we will ignore the precise generation and evolution of the error term $\mathbf{e}$ and just refer to it as "noise."

Matrix embeddings. The starting point of our construction is the recent privately puncturable PRF of Boneh, Kim, and Montgomery [BKM17], which itself builds on the constrained PRF construction of Brakerski and Vaikuntanathan [BV15]. Both of these constructions rely on the matrix embedding mechanism introduced by Boneh et al. [BGG+14] for constructing attribute-based encryption. In [BGG+14], an input $x \in \{0,1\}^\rho$ is embedded as the vector

$$\mathbf{s}^T\big(\mathbf{A}_1 + x_1 \cdot \mathbf{G} \mid \cdots \mid \mathbf{A}_\rho + x_\rho \cdot \mathbf{G}\big) + \mathsf{noise} \in \mathbb{Z}_q^{m\rho}, \tag{2.1}$$

where $\mathbf{A}_1, \ldots, \mathbf{A}_\rho \in \mathbb{Z}_q^{n\times m}$ are uniformly random matrices, $\mathbf{s} \in \mathbb{Z}_q^n$ is a uniformly random vector, and $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ is a special fixed matrix (called the "gadget matrix"). Embedding the inputs in this way enables homomorphic operations on the inputs while keeping the noise small. In particular, given an input $x \in \{0,1\}^\rho$ and any polynomial-size circuit $C : \{0,1\}^\rho \to \{0,1\}$, there is a public operation that allows computing the following vector from Eq. (2.1):

$$\mathbf{s}^T\big(\mathbf{A}_C + C(x) \cdot \mathbf{G}\big) + \mathsf{noise} \in \mathbb{Z}_q^m, \tag{2.2}$$

where the matrix $\mathbf{A}_C \in \mathbb{Z}_q^{n\times m}$ depends only on the circuit C, and *not* on the underlying input x. Thus, we can define a homomorphic operation $\mathsf{Eval}_{\mathsf{pk}}$ on the matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\rho$ where on input a sequence of matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\rho$ and a circuit C, $\mathsf{Eval}_{\mathsf{pk}}(C, \mathbf{A}_1, \ldots, \mathbf{A}_\rho) \to \mathbf{A}_C$.

[3] We note that the LWE-based predicate encryption constructions satisfy a weaker security property (compared to [BW07, KSW08]) sometimes referred to as *weak attribute-hiding*.

A puncturable PRF from LWE. Brakerski and Vaikuntanathan [BV15] showed how the homomorphic properties in [BGG+14] can be leveraged to construct a (single-key) constrained PRF for general constraints. Here, we provide a high-level description of their construction specialized to the case of puncturing. First, let eq be the equality circuit where $\mathsf{eq}(x^*, x) = 1$ if $x^* = x$ and 0 otherwise. The public parameters[4] of the scheme in [BV15] consist of randomly generated matrices $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n\times m}$ for encoding the PRF input x and matrices $\mathbf{B}_1, \dots \mathbf{B}_\rho \in \mathbb{Z}_q^{n\times m}$ for encoding the punctured point x^*. The secret key for the PRF is a vector $\mathbf{s} \in \mathbb{Z}_q^n$. Then, on input a point $x \in \{0,1\}^\rho$, the PRF value at x is defined to be

$$\mathsf{PRF}(\mathbf{s}, x) := \lfloor \mathbf{s}^T \cdot \mathbf{A}_{\mathsf{eq},x} \rceil_p \quad \text{where} \quad \mathbf{A}_{\mathsf{eq},x} := \mathsf{Eval}_{\mathsf{pk}}(\mathsf{eq}, \mathbf{B}_1, \dots, \mathbf{B}_\rho, \mathbf{A}_{x_1}, \dots, \mathbf{A}_{x_\rho}),$$

where $\mathbf{A}_0, \mathbf{A}_1, \mathbf{B}_1, \dots, \mathbf{B}_\rho \in \mathbb{Z}_q^{n\times m}$ are the matrices in the public parameters, and $\lfloor\cdot\rceil_p$ is the component-wise rounding operation that maps an element in $\mathbb{Z}_q$ to an element in $\mathbb{Z}_p$ where $p < q$. By construction, $\mathbf{A}_{\mathsf{eq},x}$ is a function of x.

To puncture the key $\mathbf{s}$ at a point $x^* \in \{0,1\}^\rho$, the construction in [BV15] gives out the vector

$$\mathbf{s}^T \cdot \big(\mathbf{A}_0 + 0 \cdot \mathbf{G} \mid \mathbf{A}_1 + 1 \cdot \mathbf{G} \quad \mid \quad \mathbf{B}_1 + x_1^* \cdot \mathbf{G} \mid \cdots \mid \mathbf{B}_\rho + x_\rho^* \cdot \mathbf{G}\big) + \mathsf{noise}. \tag{2.3}$$

To evaluate the PRF at a point $x \in \{0,1\}^\rho$ using a punctured key, the user first homomorphically evaluates the equality circuit eq on input (x^*, x) to obtain the vector $\mathbf{s}^T\big(\mathbf{A}_{\mathsf{eq},x} + \mathsf{eq}(x^*, x)\cdot\mathbf{G}\big) + \mathsf{noise}$. Rounding down this vector yields the correct PRF value whenever $\mathsf{eq}(x^*, x) = 0$, or equivalently, whenever $x \neq x^*$, as required for puncturing. As shown in [BV15], this construction yields a secure (though non-private) puncturable PRF from LWE with some added modifications.

Private puncturing. The reason the Brakerski-Vaikuntanathan puncturable PRF described here does not provide privacy (that is, hide the punctured point) is because in order to operate on the embedded vectors, the evaluator needs to know the underlying inputs. In other words, to homomorphically compute the equality circuit eq on the input (x^*, x), the evaluator needs to know both x and x^*. However, the punctured point x^* is precisely the information we need to hide. Using an idea inspired by the predicate encryption scheme of Gorbunov et al. [GVW15], the construction of Boneh et al. [BKM17] hides the point x^* by first encrypting it using a fully homomorphic encryption (FHE) scheme before applying the matrix embeddings of [BGG+14]. Specifically, in [BKM17], the punctured key has the following form:

$$\begin{aligned}\mathbf{s}^T \cdot \big(\mathbf{A}_0 + 0\cdot\mathbf{G} \mid \mathbf{A}_1 + 1\cdot\mathbf{G} \quad \mid \quad \mathbf{B}_1 + \mathsf{ct}_1\cdot\mathbf{G} \mid \cdots \mid \mathbf{B}_z + \mathsf{ct}_z \cdot \mathbf{G} \\ \mid \quad \mathbf{C}_1 + \mathsf{sk}_1\cdot\mathbf{G} \mid \cdots \mid \mathbf{C}_\tau + \mathsf{sk}_\tau\cdot\mathbf{G}\ \big) + \mathsf{noise},\end{aligned}$$

where $\mathsf{ct}_1, \dots, \mathsf{ct}_z$ are the bits of an FHE encryption ct of the punctured point x^*, and $\mathsf{sk}_1, \dots, \mathsf{sk}_\tau$ are the bits of the FHE secret key sk. Given the ciphertext ct, the evaluator can homomorphically evaluate the equality circuit eq and obtain an FHE encryption of $\mathsf{eq}(x^*, x)$. Next, by leveraging an "asymmetric multiplication property" of the matrix encodings, the evaluator is able to compute the inner product between the encrypted result with the decryption key sk.[5] Recall that for lattice-based FHE schemes (e.g. [GSW13]), decryption consists of evaluating a rounded inner product of the ciphertext with the decryption key. Specifically, the inner product between the ciphertext and the decryption key results in $\frac{q}{2} + e \in \mathbb{Z}_q$ for some "small" error term e.

Thus, it remains to show how to perform the rounding step in the FHE decryption. Simply computing the inner product between the ciphertext and the secret key results in a vector

$$\mathbf{s}^T\left(\mathbf{A}_{\mathsf{FHE},\mathsf{eq},x} + \left(\frac{q}{2}\cdot\mathsf{eq}(x^*, x) + e\right)\cdot\mathbf{G}\right) + \mathsf{noise},$$

[4]Since a constrained PRF is a secret-key primitive, we can always include the public parameters as part of the secret key. However, in the lattice-based constrained PRF constructions [BV15, BFP+15, BKM17], the public parameters can be sampled once and shared across multiple *independent* secret keys. Our construction of translucent PRFs will rely on choosing the public parameter matrices to have a certain structure that is shared across multiple secret keys.

[5]Normally, multiplication of two inputs requires knowledge of both of the underlying inputs. The "asymmetry" in the embedding scheme of [BGG+14] enables multiplications to be done even if only one of the values to be multiplied is known to the evaluator. In the case of computing an inner product between the FHE ciphertext and the FHE secret key, the evaluator knows the bits of the ciphertext, but not the FHE secret key. Thus, the asymmetry enables the evaluator to homomorphically evaluate the inner product without knowledge of the FHE secret key.

where e is the FHE noise (for simplicity, by FHE, we always refer to the specific construction of [GSW13] and its variants hereafter). Even though the error e is small, neither $\mathbf{s}$ nor $\mathbf{G}$ are low-norm and therefore, the noise does not simply round away. The observation made in [BKM17], however, is that the gadget matrix $\mathbf{G}$ contains some low-norm column vectors, namely the identity matrix $\mathbf{I}$ as a submatrix. By restricting the PRF evaluation to just these columns and sampling the secret key $\mathbf{s}$ from the low-norm noise distribution, they show that the FHE error term $\mathbf{s}^T \cdot e \cdot \mathbf{I}$ can be rounded away. Thus, by defining the PRF evaluation to only take these specific column positions of

$$\mathsf{PRF}(\mathbf{s}, x) := \lfloor \mathbf{s}^T \mathbf{A}_{\mathsf{FHE},\mathsf{eq},x} \rceil_p,$$

it is possible to recover the PRF evaluation from the punctured key if and only if $\mathsf{eq}(x^*, x) = 0$.[6]

Trapdoor at punctured key evaluations. We now describe how we extend the private puncturing construction in [BKM17] to obtain a private translucent puncturable PRF where a secret key can be used to test whether a value is the result of using a punctured key to evaluate at a punctured point. We begin by describing an alternative way to perform the rounding step of the FHE decryption in the construction of [BKM17]. First, consider modifying the PRF evaluation at $x \in \{0,1\}^\rho$ to be

$$\mathsf{PRF}(\mathbf{s}, x) := \lfloor \mathbf{s}^T \mathbf{A}_{\mathsf{FHE},\mathsf{eq},x} \cdot \mathbf{G}^{-1}(\mathbf{D}) \rceil_p,$$

where $\mathbf{D} \in \mathbb{Z}_q^{n \times m}$ is a public *binary* matrix and $\mathbf{G}^{-1}$ is the component-wise *bit-decomposition* operator on matrices in $\mathbb{Z}_q^{n \times m}$.[7] The gadget matrix $\mathbf{G}$ is defined so that for any matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$. Then, if we evaluate the PRF using the punctured key and multiply the result by $\mathbf{G}^{-1}(\mathbf{D})$, we obtain the following:

$$\begin{aligned}\left(\mathbf{s}^T \left(\mathbf{A}_{\mathsf{FHE},\mathsf{eq},x} + \left(\frac{q}{2} \cdot \mathsf{eq}(x^*, x) + e\right) \cdot \mathbf{G}\right) + \mathsf{noise}\right) \mathbf{G}^{-1}(\mathbf{D}) \\ = \mathbf{s}^T \underbrace{\left(\mathbf{A}_{\mathsf{FHE},\mathsf{eq},x} \mathbf{G}^{-1}(\mathbf{D}) + \left(\frac{q}{2} \cdot \mathsf{eq}(x^*, x) + e\right) \cdot \mathbf{D}\right)}_{\tilde{\mathbf{A}}_{\mathsf{FHE},\mathsf{eq},x}} + \mathsf{noise}' \\ = \mathbf{s}^T \tilde{\mathbf{A}}_{\mathsf{FHE},\mathsf{eq},x} + \mathsf{noise}'\end{aligned}$$

Since $\mathbf{D}$ is a low-norm (in fact, binary) matrix, the FHE error component $\mathbf{s}^T \cdot e \cdot \mathbf{D}$ is short, and thus, will disappear when we round. Therefore, whenever $\mathsf{eq}(x^*, x) = 0$, we obtain the real PRF evaluation.

The key observation we make is that the algebraic structure of the PRF evaluation allows us to "program" the matrix $\tilde{\mathbf{A}}_{\mathsf{FHE},\mathsf{eq},x}$ whenever $\mathsf{eq}(x^*, x) = 1$ (namely, when the punctured key is used to evaluate at the punctured point). As described here, the FHE ciphertext decrypts to $q/2 + e$ when the message is 1 and e when the message is 0 (where e is a small error term). In the FHE scheme of [GSW13] (and its variants), it is possible to encrypt scalar elements in $\mathbb{Z}_q$, and moreover, to modify the decryption operation so that it outputs the encrypted scalar element (with some error). In other words, decrypting a ciphertext encrypting $w \in \mathbb{Z}_q$ would yield a value $w + e$ for some small error term e. Then, in the PRF construction, instead of encrypting the punctured point x^*, we encrypt a tuple (x^*, w) where $w \in \mathbb{Z}_q$ is used to program the matrix $\tilde{\mathbf{A}}_{\mathsf{FHE},\mathsf{eq},x}$.[8] Next, we replace the basic equality function eq in the construction with a "scaled" equality function that on input $(x, (x^*, w))$, outputs w if $x = x^*$, and 0 otherwise. With these changes, evaluating the punctured PRF at a point x now yields:[9]

$$\mathbf{s}^T \left(\mathbf{A}_{\mathsf{FHE},\mathsf{eq},x} \mathbf{G}^{-1}(\mathbf{D}) + (w \cdot \mathsf{eq}(x^*, x) + e) \cdot \mathbf{D}\right) + \mathsf{noise}.$$

[6]To actually show that the challenge PRF evaluation is pseudorandom at the punctured point, additional modifications must be made such as introducing extra randomizing terms and collapsing the final PRF evaluation to be field elements instead of vectors. We refer to [BKM17] for the full details.

[7]Multiplying by the matrix $\mathbf{G}^{-1}(\mathbf{D})$ can be viewed as an alternative way to restrict the PRF to the column positions corresponding to the identity submatrix in $\mathbf{G}$.

[8]A similar construction is used in [BKM17] to show security. In their construction, they sample and encrypt a random set of w's and use them to blind the real PRF value at the punctured point.

[9]To reduce notational clutter, we redefine the matrix $\mathbf{A}_{\mathsf{FHE},\mathsf{eq},x}$ here to be the matrix associated with homomorphic evaluation of the scaled equality-check circuit.

Since w can be chosen arbitrarily when the punctured key is constructed, a natural question to ask is whether there exists a w such that the matrix $\mathbf{A}_{\mathsf{FHE},\mathsf{eq},x}\mathbf{G}^{-1}(\mathbf{D}) + w \cdot \mathbf{D}$ has a particular structure. This is not possible if w is a scalar, but if there are *multiple* w's, this becomes possible.

To support programming of the matrix $\tilde{\mathbf{A}}_{\mathsf{FHE},\mathsf{eq},x}$, we first take $N = m \cdot n$ (public) binary matrices $\mathbf{D}_\ell \in \{0,1\}^{n \times m}$ where the collection $\{\mathbf{D}_\ell\}_{\ell \in [N]}$ is a basis for the module $\mathbb{Z}_q^{n \times m}$ (over $\mathbb{Z}_q$). This means that any matrix in $\mathbb{Z}_q^{n \times m}$ can be expressed as a unique linear combination $\sum_{\ell \in [N]} w_\ell \mathbf{D}_\ell$ where $\mathbf{w} = (w_1, \ldots, w_N) \in \mathbb{Z}_q^N$ are the coefficients. Then, instead of encrypting a single element w in each FHE ciphertext, we encrypt a vector $\mathbf{w}$ of coefficients. The PRF output is then a sum of N *different* PRF evaluations:

$$\mathsf{PRF}(\mathbf{s}, x) := \left\lfloor \sum_{\ell \in [N]} \mathbf{s}^T \mathbf{A}_{\mathsf{FHE},\mathsf{eq}_\ell,x} \mathbf{G}^{-1}(\mathbf{D}_\ell) \right\rceil_p,$$

where the ℓ^{th} PRF evaluation is with respect to the circuit eq_ℓ that takes as input a pair $(x, (x^*, \mathbf{w}))$ and outputs w_ℓ if $x = x^*$ and 0 otherwise. If we now consider the corresponding computation using the punctured key, evaluation at x yields the vector

$$\sum_{\ell \in [N]} \mathbf{s}^T \left(\mathbf{A}_{\mathsf{FHE},\mathsf{eq}_\ell,x} \mathbf{G}^{-1}(\mathbf{D}_\ell) + (w_\ell \cdot \mathsf{eq}(x^*, x) + e) \cdot \mathbf{D}_\ell \right) + \mathsf{noise} \tag{2.4}$$

The key observation is that for any matrix $\mathbf{W} \in \mathbb{Z}_q^{n \times m}$, the puncturing algorithm can choose the coefficients $\mathbf{w} \in \mathbb{Z}_q^N$ so that

$$\mathbf{W} = \left(\sum_{\ell \in [N]} \mathbf{A}_{\mathsf{FHE},\mathsf{eq}_\ell,x^*} \mathbf{G}^{-1}(\mathbf{D}_\ell) \right) + \sum_{\ell \in [N]} w_\ell \cdot \mathbf{D}_\ell. \tag{2.5}$$

Next, we choose $\mathbf{W}$ to be a lattice trapdoor matrix with associated trapdoor $\mathbf{z}$ (i.e., $\mathbf{W}\mathbf{z} = 0 \bmod q$). From Eq. (2.4) and Eq. (2.5), we have that whenever a punctured key is used to evaluate the PRF at the punctured point, the result is a vector of the form $\lfloor \mathbf{s}^T \mathbf{W} \rceil_p \in \mathbb{Z}_p^m$. Testing whether a vector $\mathbf{y}$ is of this form can be done by computing the inner product of $\mathbf{y}$ with the trapdoor vector $\mathbf{z}$ and checking if the result is small. In particular, when $\mathbf{y} = \lfloor \mathbf{s}^T \mathbf{W} \rceil_p$, we have that

$$\left\langle \lfloor \mathbf{s}^T \mathbf{W} \rceil_p, \mathbf{z} \right\rangle \approx \lfloor \mathbf{s}^T \mathbf{W} \mathbf{z} \rceil_p = 0.$$

In our construction, the trapdoor matrix $\mathbf{W}$ is chosen independently of the PRF key $\mathbf{s}$, and included as part of the public parameters. To puncture a key $\mathbf{s}$, the puncturing algorithm chooses the coefficients $\mathbf{w}$ such that Eq. (2.5) holds. This allows us to program punctured keys associated with different secret keys $\mathbf{s}_i$ to the *same* trapdoor matrix $\mathbf{W}$. The underlying "translucent set" then is the set of vectors of the form $\lfloor \mathbf{s}_i^T \mathbf{W} \rceil_p$. Under the LWE assumption, this set is indistinguishable from random. However, as shown above, using a trapdoor for $\mathbf{W}$, it is easy to determine if a vector lies in this set. Thus, we are able to embed a noisy hidden subspace within the public parameters of the translucent PRF.

We note here that our construction is not expressive enough to give a programmable PRF in the sense of [BLW17], because we do not have full control of the value $\mathbf{y} \in \mathbb{Z}_p^m$ obtained when using the punctured key to evaluate at the punctured point. We only ensure that $\mathbf{y}$ lies in a hidden (but efficiently testable) subspace of $\mathbb{Z}_p^m$. As we show in Section 6, this notion suffices for watermarking.

Puncturing at multiple points. The construction described above yields a translucent puncturable PRF. As noted in Section 1, for message-embedding watermarking, we require a translucent t-puncturable PRF. While we can trivially build a t-puncturable PRF from t instances of a puncturable PRF by xoring the outputs of t independent puncturable PRF instances, this construction does not preserve translucency. Notably, we can no longer detect whether a punctured key was used to evaluate the PRF at one of the punctured points. Instead, to preserve the translucency structure, we construct a translucent t-puncturable PRF by defining it to be the sum of multiple independent PRFs with different (public) parameter matrices, but *sharing the same* *secret key*. Then, to puncture at t different points we first encrypt each of the t punctured points $x_1^*, \ldots, x_t^*$, each with its own set of coefficient vectors $\mathbf{w}_1, \ldots, \mathbf{w}_t$ to obtain t FHE ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_t$. The constrained key then contains the following components:

$$\mathbf{s}^T \cdot \big(\mathbf{A}_0 + 0 \cdot \mathbf{G} \mid \mathbf{A}_1 + 1 \cdot \mathbf{G} \quad \mid \quad \mathbf{B}_{1,1} + \mathsf{ct}_{1,1} \cdot \mathbf{G} \mid \cdots \mid \mathbf{B}_{t,z} + \mathsf{ct}_{t,z} \cdot \mathbf{G} \\ \mid \quad \mathbf{C}_1 + \mathsf{sk}_1 \cdot \mathbf{G} \mid \cdots \mid \mathbf{C}_\tau + \mathsf{sk}_\tau \cdot \mathbf{G} \big) + \mathsf{noise}.$$

To evaluate the PRF at a point $x \in \{0,1\}^\rho$ using the constrained key, one evaluates the PRF on each of the t instances, that is, for all $i \in [t]$,

$$\mathbf{s}^T \left(\sum_{\ell \in [N]} \mathbf{A}_{\mathsf{FHE}, \mathsf{eq}_\ell, i, x} \mathbf{G}^{-1}(\mathbf{D}_\ell) + \mathsf{eq}(x_i^*, x) \cdot \sum_{\ell \in [N]} w_{i,\ell} \cdot \mathbf{D}_\ell \right) + \mathsf{noise}'.$$

The output of the PRF is the (rounded) sum of these evaluations:

$$\mathbf{s}^T \left(\sum_{\substack{i \in [t] \\ \ell \in [N]}} \big(\mathbf{A}_{\mathsf{FHE}, \mathsf{eq}_\ell, i, x} \mathbf{G}^{-1}(\mathbf{D}_\ell) \big) + \sum_{i \in [t]} \left(\mathsf{eq}(x_i^*, x) \cdot \sum_{\ell \in [N]} w_{i,\ell} \cdot \mathbf{D}_\ell \right) \right) + \mathsf{noise}'.$$

Similarly, the real value of the PRF is the (rounded) sum of the t independent PRF evaluations:

$$\mathsf{PRF}(\mathbf{s}, x) := \left\lfloor \mathbf{s}^T \sum_{\substack{i \in [t] \\ \ell \in [N]}} \mathbf{A}_{\mathsf{FHE}, \mathsf{eq}_\ell, i, x} \mathbf{G}^{-1}(\mathbf{D}_\ell) \right\rceil_p.$$

If the point x is not one of the punctured points, then $\mathsf{eq}(x_i^*, x) = 0$ for all $i \in [t]$ and one recovers the real PRF evaluation at x. If x is one of the punctured points (i.e., $x = x_i^*$ for some $i \in [t]$), then the PRF evaluation using the punctured key yields the vector

$$\mathbf{s}^T \left(\sum_{\substack{i \in [t] \\ \ell \in [N]}} \big(\mathbf{A}_{\mathsf{FHE}, \mathsf{eq}_\ell, i, x} \mathbf{G}^{-1}(\mathbf{D}_\ell) \big) + \mathsf{eq}(x_i^*, x) \cdot \sum_{\ell \in [N]} w_{i,\ell} \cdot \mathbf{D}_\ell \right) + \mathsf{noise}'.$$

and as before, we can embed trapdoor matrices $\mathbf{W}_{i^*}$ for all $i^* \in [t]$ by choosing the coefficient vectors $\mathbf{w}_{i^*} = (w_{i^*,1}, \ldots, w_{i^*,N}) \in \mathbb{Z}_q^N$ accordingly:[10]

$$\mathbf{W}_{i^*} = \sum_{\substack{i \in [t] \\ \ell \in [N]}} \Big(\mathbf{A}_{\mathsf{FHE}, \mathsf{eq}_\ell, i, x_{i^*}^*} \mathbf{G}^{-1}(\mathbf{D}_\ell) \Big) + \sum_{\ell \in [N]} w_{i^*,\ell} \cdot \mathbf{D}_\ell.$$

A technical detail. In the actual construction in Section 5.1, we include an additional "auxiliary matrix" $\hat{\mathbf{A}}$ in the public parameters and define the PRF evaluation as the vector

$$\mathsf{PRF}(\mathbf{s}, x) := \left\lfloor \mathbf{s}^T \left(\hat{\mathbf{A}} + \sum_{\substack{i \in [t] \\ \ell \in [N]}} \mathbf{A}_{\mathsf{FHE}, \mathsf{eq}_\ell, i, x} \mathbf{G}^{-1}(\mathbf{D}_\ell) \right) \right\rceil_p.$$

The presence of the additional matrix $\hat{\mathbf{A}}$ does not affect pseudorandomness, but facilitates the argument for some of our other security properties. We give the formal description of our scheme as well as the security analysis in Section 5.

[10]For the punctured keys to hide the set of punctured points, we need a different trapdoor matrix for each punctured point. We provide the full details in Section 5.

3 Preliminaries

We begin by introducing some basic 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 $\text{Funs}[\mathcal{X}, \mathcal{Y}]$ to denote the set of all functions mapping from a domain $\mathcal{X}$ to a range $\mathcal{Y}$. For a finite set S, we write 2^S to denote the *power set* of S, namely the set of all subsets of S. 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. For two vectors $\mathbf{v}, \mathbf{w}$, we write $\mathsf{IP}(\mathbf{v}, \mathbf{w}) = \langle \mathbf{v}, \mathbf{w} \rangle$ to denote the inner product of $\mathbf{v}$ and $\mathbf{w}$.

Unless specified otherwise, we use λ to denote the security parameter. We say a function $f(\lambda)$ is negligible in λ, 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 λ, and $\mathsf{polylog}(\lambda)$ to denote a quantity whose value is bounded by a fixed polynomial in $\log \lambda$ (that is, a function of the form $\log^c \lambda$ for some $c \in \mathbb{N}$). 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 $\mathcal{D}_1 \stackrel{c}{\approx} \mathcal{D}_2$ if the two distributions are computationally indistinguishable (that is, no efficient algorithm can distinguish $\mathcal{D}_1$ from $\mathcal{D}_2$, except with negligible probability). We write $\mathcal{D}_1 \stackrel{s}{\approx} \mathcal{D}_2$ if the two distributions are statistically indistinguishable (that is, the statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is negligible). We also review the (formal) definition of a pseudorandom function:

Definition 3.1 (Pseudorandom Function [GGM84])**.** A pseudorandom function with a key-space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ is a tuple of algorithms $\Pi_{\mathsf{PRF}} = (\mathsf{PRF.KeyGen}, \mathsf{PRF.Eval})$ with the following properties:

- $\mathsf{PRF.KeyGen}(1^\lambda) \to k$: On input the security parameter λ, the key-generation algorithm outputs a key $k \in \mathcal{K}$.
- $\mathsf{PRF.Eval}(k, x) \to y$: On input a PRF key $k \in \mathcal{K}$ and a point $x \in \mathcal{X}$, the evaluation algorithm outputs a value $y \in \mathcal{Y}$.

Definition 3.2 (Pseudorandomness [GGM84])**.** Fix a security parameter λ and let $\Pi_{\mathsf{PRF}} = (\mathsf{PRF.KeyGen}, \mathsf{PRF.Eval})$ be a PRF with domain $\mathcal{X}$ and range $\mathcal{Y}$. Then Π_{PRF} is secure if for all efficient adversaries $\mathcal{A}$, and $k \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$, $f \xleftarrow{\text{R}} \text{Funs}[\mathcal{X}, \mathcal{Y}]$,

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

Vector and matrix norms. In 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 $\left\|\mathbf{x}^T \mathbf{A}\right\| \leq n \cdot \|\mathbf{x}\| \cdot \|\mathbf{A}\|$.

3.1 Lattice Preliminaries

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

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 3.3 (Learning with Errors [Reg05])**.** Fix a security parameter λ 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 $\mathbf{u} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, the following two families of distributions are computationally indistinguishable:

$$(\mathbf{A}, \mathbf{s}^T \mathbf{A} + \mathbf{e}^T) \stackrel{c}{\approx} (\mathbf{A}, \mathbf{u}).$$

When the error distribution χ 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 GapSVP and 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].

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 zero columns. We have the property that for any matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we have that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$.

The 1D-SIS problem. Following [BV15, BKM17], 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].

Definition 3.4 (One-Dimensional Short Integer Solution [Ajt96])**.** Fix a security parameter λ and integers $m = m(\lambda)$, $q = q(\lambda)$, and $\beta = \beta(\lambda)$. The one-dimensional short integer solution (1D-SIS) problem $\mathsf{1D\text{-}SIS}_{m,q,\beta}$ is defined as follows:

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

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

In this work, we require the following "rounded" variant of the 1D-SIS assumption, which was first introduced in [BV15] for constructing single-key circuit-constrained PRFs and used in [BKM17] for constructing privately puncturable PRFs. These works also show that this variant of 1D-SIS is at least as hard as 1D-SIS (for an appropriate choice of parameters).

Definition 3.5 (One-Dimension Rounded Short Integer Solution [BV15, BKM17])**.** Fix a security parameter λ and integers $m = m(\lambda)$, $p = p(\lambda)$, $q = q(\lambda)$, and $\beta = \beta(\lambda)$, where $q = p \cdot \prod_{i \in [n]} p_i$, and $p_1 < p_2 < \cdots < p_n$ are all coprime and also coprime with p. The one-dimensional rounded short integer solution (1D-SIS-R) problem $\mathsf{1D\text{-}SIS\text{-}R}_{m,p,q,\beta}$ problem is defined as follows:

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

and one of the following conditions hold:

$$\langle \mathbf{v}, \mathbf{z} \rangle \in [-\beta, \beta] + (q/p) \cdot \mathbb{Z} \qquad \text{or} \qquad \langle \mathbf{v}, \mathbf{z} \rangle \in [-\beta, \beta] + (q/p)(\mathbb{Z} + 1/2).^{11}$$

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

The works of [BV15, BKM17] show that when $m = O(n \log q)$ and $p_1 \ge \beta \cdot \omega(\sqrt{mn \log n})$, the $\mathsf{1D\text{-}SIS\text{-}R}_{m,p,q,\beta}$ problem is as hard as approximating certain worst-case lattice problems to within a factor of $\beta \cdot \tilde{O}(\sqrt{mn})$.

3.2 Lattice Trapdoors

Although finding a "short vector" in a given lattice is believed to be a hard problem, with some additional auxiliary information such as a *trapdoor* (i.e. a set of short generating vectors of the lattice), 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 only the properties we require in the following theorem.

[11] 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 3.6 (Lattice Trapdoors [Ajt99, GPV08, AP09, MP12, LW15]). *Fix a security parameters λ, and lattice parameters n, m, q. Let $\chi = \chi(\lambda)$ be a B-bounded error distribution. Then, there exists a polynomial time algorithm* $\mathsf{TrapGen}$*:*

- $\mathsf{TrapGen}(1^n, q) \to (\mathbf{W}, \mathbf{z})$*: On input the parameters $n, q \in \mathbb{Z}$, the trapdoor generation algorithm outputs a matrix $\mathbf{W} \in \mathbb{Z}_q^{n \times m}$ and a vector $\mathbf{z} \in \mathbb{Z}^m$ for some $m \in \mathbb{N}$.*

Moreover, the $\mathsf{TrapGen}$ *algorithm satisfies the following properties hold:*

- *The matrix $\mathbf{W}$ is statistically close to uniform.*
- *The vector $\mathbf{z}$ is B-bounded: $\|\mathbf{z}\| \le B$.*
- $\mathbf{W} \cdot \mathbf{z} = \mathbf{0} \bmod q$.

3.3 (Leveled) Homomorphic Encryption

Following the presentation of [GVW15], we give a minimal definition of leveled homomorphic encryption that suffices for our construction. Note that a leveled homomorphic encryption scheme is one that only supports an *a priori* bounded number of homomorphic operations. This is to contrast it with the notion of a fully homomorphic encryption scheme (FHE) scheme supports an *arbitrary* number of homomorphic operations on ciphertexts.[12] A (secret-key) leveled homomorphic encryption scheme over a message space $\{0,1\}^\rho$ is a tuple of polynomial-time algorithms $\Pi_{\mathsf{HE}} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Eval}, \mathsf{HE.Dec})$ defined as follows:

- $\mathsf{HE.KeyGen}(1^\lambda, 1^d, 1^\rho) \to \mathsf{sk}$: On input the security parameter λ, a depth bound d, and a message length ρ, the key generation algorithm outputs a secret key sk.

- $\mathsf{HE.Enc}(\mathsf{sk}, \mu) \to \mathsf{ct}$: On input a secret key sk and a message $\mu \in \{0,1\}^\rho$, the encryption algorithm outputs a ciphertext ct.

- $\mathsf{HE.Eval}(C, \mathsf{ct}) \to \mathsf{ct}'$: On input a circuit $C \colon \{0,1\}^\rho \to \{0,1\}$ of depth at most d and a ciphertext ct, the homomorphic evaluation algorithm outputs another ciphertext ct'.

- $\mathsf{HE.Dec}(\mathsf{sk}, \mathsf{ct}) \to b$: On input a secret key sk and a ciphertext ct, the decryption algorithm outputs a bit b.

Note that we can also define leveled (and fully) homomorphic encryption schemes where the plaintext space is a ring or a finite field. All of the definitions translate analogously.

Definition 3.7 (Correctness). Fix a security parameter λ. A leveled homomorphic encryption scheme $\Pi_{\mathsf{HE}} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Eval}, \mathsf{HE.Dec})$ is (perfectly) correct if for all positive integers $d = d(\lambda)$, $\rho = \rho(\lambda)$, and all messages $\mu \in \{0,1\}^\rho$, all Boolean circuits $C \colon \{0,1\}^\rho \to \{0,1\}$ of depth at most d, setting $\mathsf{sk} \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^d, 1^\rho)$, we have that

$$\Pr\left[\mathsf{HE.Dec}(\mathsf{sk}, \mathsf{HE.Eval}(C, \mathsf{HE.Enc}(\mathsf{sk}, \mu))) = C(\mu)\right] = 1,$$

Definition 3.8 (Semantic Security). Fix a security parameter λ, and let $d = d(\lambda)$, $\rho = \rho(\lambda)$. Then, a leveled homomorphic encryption scheme $\Pi_{\mathsf{HE}} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Eval}, \mathsf{HE.Dec})$ is semantically secure if for all efficient adversaries $\mathcal{A}$ and setting $\mathsf{sk} \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^d, 1^\rho)$,

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

where for $b \in \{0,1\}$, $\mathcal{O}_b(\mathsf{sk}, \cdot, \cdot)$ is the encryption oracle that on input $m_0, m_1 \in \{0,1\}^\rho$, outputs $\mathsf{HE.Enc}(\mathsf{sk}, m_b)$.

[12] Since these two notions are syntactically similar, we often write FHE as shorthand to also refer to leveled homomorphic encryption. All of the constructions in this paper only rely on leveled (rather than fully) homomorphic encryption.

Homomorphic encryption from LWE. There are numerous instantiations of leveled (and fully) homomorphic encryption based on the LWE assumption [BV11, BGV12, Bra12, GSW13, BV14, AP14, CM15, MW16, PS16, MW16]. A key property of existing FHE constructions that we leverage in this work is the asymmetric noise growth when performing homomorphic operations [BV14]. Typically, when two ciphertexts are homomorphically multiplied, the noise in the resulting ciphertext scales proportionally to the size of the underlying plaintext. Thus, plaintexts are usually restricted to be "small" elements (e.g., bits) in $\mathbb{Z}_q$. However, the noise growth of the homomorphic encryption construction of [GSW13] and its variants enjoy an asymmetric property where the noise scales proportionally to only *one* of the underlying plaintext elements being multiplied. Thus, it is possible to encrypt and homomorphically compute on $\mathbb{Z}_q$ elements; correctness is preserved as long as two (large) $\mathbb{Z}_q$ elements are never multiplied together. This is the property we leverage in our construction of translucent PRFs in Section 5.1. There, the plaintext space of the homomorphic encryption scheme is more naturally written as $\{0,1\}^{\rho_0} \times \mathbb{Z}_q^{\rho_1}$ and the circuits that we homomorphically evaluate have the property that they never multiply two (non-binary) elements in $\mathbb{Z}_q$. We can view the plaintext space as $\mathbb{Z}_q^{\rho}$ for $\rho = \rho_0 + \rho_1$, but it will be useful in distinguishing between the inputs that are binary-valued and those that are field elements. Moreover, in the description below (and in our construction), we only require a relaxed version of correctness. On input a ciphertext ct encrypting a field element $w \in \mathbb{Z}_q$, we require that the decryption function $\mathsf{HE.Dec}$ outputs a value that is "close" to w (rather than the exact value of w). We summarize the formal properties of the homomorphic encryption construction based on [GSW13] (and its variants) in the following theorem.

Theorem 3.9 (Homomorphic Encryption from LWE [GSW13, BV14, adapted])**.** *Fix a security parameter λ and lattice parameters n, m, q. Let $\chi = \chi(\lambda)$ be a B-bounded error distribution. There is a leveled homomorphic encryption scheme $\Pi_{\mathsf{HE}} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Eval}, \mathsf{HE.Dec})$ for (arithmetic) circuits of depth $d = d(\lambda)$ over the plaintext space $\{0,1\}^{\rho_0} \times \mathbb{Z}_q^{\rho_1}$ with the following properties:*

- $\mathsf{HE.KeyGen}(1^\lambda, 1^d, 1^\rho)$ *outputs a secret key* $\mathsf{sk} \in \mathbb{Z}_q^\tau$ *where* $\rho = \rho_0 + \rho_1$ *and* $\tau = \mathsf{poly}(\lambda)$*.*
- $\mathsf{HE.Enc}$ *takes a message* $(\mu, \mathbf{w}) \in \{0,1\}^{\rho_0} \times \mathbb{Z}_q^{\rho_1}$ *and outputs a ciphertext* $\mathsf{ct} \in \{0,1\}^z$ *where* $z = \mathsf{poly}(\lambda, d, \rho, \log q)$*.*
- $\mathsf{HE.Eval}$ *takes an arithmetic circuit* $C : \{0,1\}^{\rho_0} \times \mathbb{Z}_q^{\rho_1} \to \mathbb{Z}_q$ *and a ciphertext* $\mathsf{ct} \in \{0,1\}^z$ *and outputs a ciphertext* $\mathsf{ct}' \in \{0,1\}^\tau$*.*
- *On input an arithmetic circuit* $C : \{0,1\}^{\rho_0} \times \mathbb{Z}_q^{\rho_1} \to \mathbb{Z}_q$ *of depth at most d,* $\mathsf{HE.Eval}(C, \cdot)$ *can be computed by a Boolean circuit of depth* $\mathsf{poly}(d, \log z)$*, where z is the length of the ciphertexts output by* $\mathsf{HE.Enc}$*.*
- *Let* $C : \{0,1\}^{\rho_0} \times \mathbb{Z}_q^{\rho_1} \to \mathbb{Z}_q$ *be an arithmetic circuit of depth at most d such that the inputs to every multiplication gate in C contains at most a single non-binary value. Let* $\mathsf{sk} \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^d, 1^\rho)$ *and take a message* $(\mu, \mathbf{w}) \in \{0,1\}^{\rho_0} \times \mathbb{Z}_q^{\rho_1}$*. Let* $\mathsf{ct} \leftarrow \mathsf{HE.Eval}(C, \mathsf{HE.Enc}(\mathsf{sk}, (\mu, \mathbf{w})))$*. If* $C(\mu, \mathbf{w}) = w \in \mathbb{Z}_q$*, then with overwhelming probability,*

$$\mathsf{HE.Dec}(\mathsf{sk}, \mathsf{ct}) = \langle \mathsf{ct}, \mathsf{sk} \rangle = \sum_{k \in [\tau]} \mathsf{sk}_k \cdot \mathsf{ct}_k \in [w - E, w + E]$$

  for some $E = B \cdot m^{O(d)}$*.*
- *The scheme* Π_{HE} *is secure under the* $\mathsf{LWE}_{n,q,\chi}$ *assumption where* $n = \mathsf{poly}(\lambda)$ *and* $q > B \cdot m^{O(d)}$*.*

3.4 Embedding Circuits into Matrices

A core ingredient in our construction is the ability to embed bits $x_1, \dots, x_\rho \in \{0,1\}$ into matrices $\mathbf{A}_1, \dots, \mathbf{A}_\rho \in \mathbb{Z}_q^{n \times m}$ and subsequently evaluate a circuit on these matrices. This technique was first introduced by Boneh et al. [BGG+14], for constructing attribute-based encryption for arithmetic circuits, and has subsequently found applications in other lattice-based constructions such as predicate encryption [GVW15], constrained PRFs [BV15], and private puncturable PRFs [BKM17]. In this work, we rely on the extended matrix embedding for the class of circuits of the form $\mathsf{IP} \circ C$ used in [GVW15, BKM17]. Specifically, if $C : \{0,1\}^\rho \to \{0,1\}^\tau$ is a Boolean circuit, then the circuit $\mathsf{IP} \circ C : \{0,1\}^\rho \times \mathbb{Z}_q^\tau \to \mathbb{Z}_q$ is defined by

$$(\mathsf{IP} \circ C)(\mathbf{x}, \mathbf{y}) = \mathsf{IP}(C(\mathbf{x}), \mathbf{y}) = \langle C(\mathbf{x}), \mathbf{y} \rangle \in \mathbb{Z}_q,$$

Our presentation of the matrix embedding is largely adapted from [GVW15, BKM17], and we refer readers there for a more detailed description. The matrix embedding consists of the following two algorithms $(\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}})$:

- The deterministic algorithm $\mathsf{Eval}_{\mathsf{pk}}$ takes as input a circuit $\mathsf{IP} \circ C : \{0,1\}^\rho \times \mathbb{Z}_q^\tau \to \mathbb{Z}_q$ and $\rho + \tau$ matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\rho, \widetilde{\mathbf{A}}_1, \ldots, \widetilde{\mathbf{A}}_\tau \in \mathbb{Z}_q^{n \times m}$ and outputs a matrix $\mathbf{A}_{\mathsf{IP} \circ C} \in \mathbb{Z}_q^{n \times m}$.
- The deterministic algorithm $\mathsf{Eval}_{\mathsf{ct}}$ takes as input a circuit $\mathsf{IP} \circ C : \{0,1\}^\rho \times \mathbb{Z}_q^\tau \to \mathbb{Z}_q$ and matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\rho, \widetilde{\mathbf{A}}_1, \ldots, \widetilde{\mathbf{A}}_\tau \in \mathbb{Z}_q^{n \times m}$ as in $\mathsf{Eval}_{\mathsf{pk}}$, and in addition, a bit-string $\mathbf{x} \in \{0,1\}^\rho$, and $\rho + \tau$ LWE samples $\mathbf{b}_1, \ldots, \mathbf{b}_\rho, \widetilde{\mathbf{b}}_1, \ldots, \widetilde{\mathbf{b}}_\tau \in \mathbb{Z}_q^n$, associated with the bits of $\mathbf{x} \in \{0,1\}^\rho$ and components of some vector $\mathbf{y} \in \mathbb{Z}_q^\tau$, respectively. Specifically, for $i \in [\rho]$ and $j \in [\tau]$, we can write $\mathbf{b}_i$ and $\widetilde{\mathbf{b}}_j$ as

  $$\mathbf{b}_i = \mathbf{s}^T(\mathbf{A}_i + x_i \mathbf{G}) + \mathbf{e}_i^T \qquad \text{and} \qquad \widetilde{\mathbf{b}}_j = \mathbf{s}^T(\widetilde{\mathbf{A}}_j + y_j \mathbf{G}) + \widetilde{\mathbf{e}}_j^T.$$

  where the noise vectors $\{\mathbf{e}_i\}_{i \in [\rho]}$, $\{\widetilde{\mathbf{e}}_j\}_{j \in [\tau]}$ are sampled from the noise distribution χ^m. The output of $\mathsf{Eval}_{\mathsf{ct}}$ is an LWE sample $\mathbf{s}^T(\mathbf{A}_{\mathsf{IP} \circ C} + (\mathsf{IP} \circ C)(\mathbf{x}, \mathbf{y}) \cdot \mathbf{G}) + \mathbf{e}_{\mathsf{IP} \circ C}$ associated with the output matrix $\mathbf{A}_{\mathsf{IP} \circ C}$ and output value $(\mathsf{IP} \circ C)(\mathbf{x}, \mathbf{y})$. Critically, the input to $\mathsf{Eval}_{\mathsf{ct}}$ just includes $\mathbf{x} \in \{0,1\}^\ell$, and *not* $\mathbf{y} \in \mathbb{Z}_q^\tau$. For notational convenience, when the matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\rho, \widetilde{\mathbf{A}}_1, \ldots, \widetilde{\mathbf{A}}_\tau$ are clear from context, we will not explicitly include them as part of the arguments to $\mathsf{Eval}_{\mathsf{ct}}$.

Next, we state the formal properties satisfied by $(\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}})$.

Theorem 3.10 (Matrix Embeddings [BGG+14, GVW15])**.** *Fix a security parameter λ, and lattice parameters n, m, q. There exists algorithms $(\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}})$ such that for all matrices $\mathbf{A}_1, \ldots \mathbf{A}_\rho, \widetilde{\mathbf{A}}_1, \ldots, \widetilde{\mathbf{A}}_\tau \in \mathbb{Z}_q^{n \times m}$, for all inputs $(\mathbf{x}, \mathbf{y}) \in \{0,1\}^\rho \times \mathbb{Z}_q^\tau$, and for all Boolean circuits $C : \{0,1\}^\rho \to \{0,1\}^\tau$ of depth d, if*

$$\mathbf{b}_i = \mathbf{s}^T(\mathbf{A}_i + x_i \mathbf{G}) + \mathbf{e}_i^T \quad \forall i \in [\rho] \quad \textit{and} \quad \widetilde{\mathbf{b}}_j = \mathbf{s}^T(\widetilde{\mathbf{A}}_j + y_j \mathbf{G}) + \widetilde{\mathbf{e}}_j^T \quad \forall j \in [\tau],$$

for some vector $\mathbf{s} \in \mathbb{Z}_q^n$, and $\|\mathbf{e}_i\|, \|\widetilde{\mathbf{e}}_j\| \le B$ for all $i \in [\rho]$, $j \in [\tau]$, where $B = B(\lambda)$ is a noise bound such that $B \cdot m^{O(d)} < q$, then the following properties hold

- *Letting*
  $$\mathbf{b}_{\mathsf{IP} \circ C} = \mathsf{Eval}_{\mathsf{ct}}(\mathbf{x}, \mathsf{IP} \circ C, \mathbf{A}_1, \ldots, \mathbf{A}_\rho, \widetilde{\mathbf{A}}_1, \ldots, \widetilde{\mathbf{A}}_\tau, \mathbf{b}_1, \ldots, \mathbf{b}_\rho, \widetilde{\mathbf{b}}_1, \ldots, \widetilde{\mathbf{b}}_\tau),$$
  then
  $$\mathbf{b}_{\mathsf{IP} \circ C} = \mathbf{s}^T(\mathbf{A}_{\mathsf{IP} \circ C} + (\mathsf{IP} \circ C)(\mathbf{x}, \mathbf{y}) \cdot \mathbf{G}) + \mathbf{e}_{\mathsf{IP} \circ C},$$
  where $\mathbf{A}_{\mathsf{IP} \circ C} = \mathsf{Eval}_{\mathsf{pk}}(\mathsf{IP} \circ C, \mathbf{A}_1, \ldots, \mathbf{A}_\rho, \widetilde{\mathbf{A}}_1, \ldots, \widetilde{\mathbf{A}}_\tau)$ and $\|\mathbf{e}_{\mathsf{IP} \circ C}\| \le B \cdot m^{O(d)}$.
- *There exists a collection of (efficiently-computable) matrices $\mathbf{R}_1, \ldots, \mathbf{R}_\rho, \widetilde{\mathbf{R}}_1, \ldots, \widetilde{\mathbf{R}}_\tau \in \mathbb{Z}_q^{m \times m}$ such that*

  $$\mathbf{b}_{\mathsf{IP} \circ C}^T = \sum_{i \in [\rho]} \mathbf{b}_i^T \mathbf{R}_i + \sum_{j \in [\tau]} \widetilde{\mathbf{b}}_j^T \widetilde{\mathbf{R}}_j,$$

  where $\mathbf{A}_{\mathsf{IP} \circ C} = \mathsf{Eval}_{\mathsf{pk}}(\mathsf{IP} \circ C, \mathbf{A}_1, \ldots, \mathbf{A}_\rho, \widetilde{\mathbf{A}}_1, \ldots, \widetilde{\mathbf{A}}_\tau)$ and $\|\mathbf{R}_i\|$, $\|\widetilde{\mathbf{R}}_j\| \le m^{O(d)}$ for all $i \in [\rho], j \in [\tau]$.

4 Translucent Constrained PRFs

In this section, we formally define our notion of a translucent constrained PRFs. Recall first that in a constrained PRF [BW13], the holder of the master secret key for the PRF can issue constrained keys which enable PRF evaluation on only the points that satisfy the constraint. Now, each translucent constrained PRF actually defines an entire *family* of constrained PRFs (see the discussion in Section 1.2 and Remark 4.2 for more details). Moreover, this family of constrained PRFs has the special property that the constraining algorithm embeds a hidden subset. Notably, this hidden subset is shared across *all* PRF keys in the constrained PRF family; the hidden subset is specific to the constrained PRF family, and is determined wholly by the parameters of the particular translucent constrained PRF. This means that whenever an (honestly-generated) constrained key is used to evaluate at a point that does not satisfy the constraint, the evaluation lies within this hidden subset. Furthermore, the holder of the constrained key is unable to tell whether a particular output value lies in the hidden subset or not. However, anyone who possesses a secret testing key (specific to the translucent constrained PRF) is able to identify whether a particular value lies in the hidden subset or not. In essence then, the set of outputs of all of the constrained keys in a translucent constrained PRF system defines a translucent set in the sense of [CDNO97]. We now give our formal definitions.

Definition 4.1 (Translucent Constrained PRF)**.** Let λ be a security parameter. A translucent constrained PRF with domain $\mathcal{X}$ and range $\mathcal{Y}$ is a tuple of algorithms $\Pi_{\mathsf{TPRF}} = (\mathsf{TPRF.Setup}, \mathsf{TPRF.SampleKey}, \mathsf{TPRF.Eval}, \mathsf{TPRF.Constrain}, \mathsf{TPRF.ConstrainEval}, \mathsf{TPRF.Test})$ with the following properties:

- $\mathsf{TPRF.Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{tk})$: On input a security parameter λ, the setup algorithm outputs the public parameters pp and a testing key tk.
- $\mathsf{TPRF.SampleKey}(\mathsf{pp}) \to \mathsf{msk}$: On input the public parameter pp, the key sampling algorithm outputs a master PRF key msk.
- $\mathsf{TPRF.Eval}(\mathsf{pp}, \mathsf{msk}, x) \to y$: On input the public parameters pp, a master PRF key msk and a point in the domain $x \in \mathcal{X}$, the PRF evaluation algorithm outputs an element in the range $y \in \mathcal{Y}$.
- $\mathsf{TPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, S) \to \mathsf{sk}_S$: On input the public parameters pp, a master PRF key msk and a set of points $S \subseteq \mathcal{X}$, the constraining algorithm outputs a constrained key sk_S.
- $\mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk}_S, x) \to y$: On input the public parameters pp, a constrained key sk_S, and a point in the domain $x \in \mathcal{X}$, the constrained evaluation algorithm outputs an element in the range $y \in \mathcal{Y}$.
- $\mathsf{TPRF.Test}(\mathsf{pp}, \mathsf{tk}, y') \to \{0,1\}$: On input the public parameters pp, a testing key tk, and a point in the range $y' \in \mathcal{Y}$, the testing algorithm either accepts (with output 1) or rejects (with output 0).

Remark 4.2 (Relation to Constrained PRFs)**.** Every translucent constrained PRF defines an entire *family* of constrained PRFs. In other words, every set of parameters $(\mathsf{pp}, \mathsf{tk})$ output by the setup function $\mathsf{TPRF.Setup}$ of a translucent constrained PRF induces a constrained PRF family (in the sense of [BW13]) for the same class of constraints. Specifically, the key-generation algorithm for the constrained PRF family corresponds to running $\mathsf{TPRF.SampleKey}(\mathsf{pp})$. The constrain, evaluation, and constrained-evaluation algorithms for the constrained PRF family correspond to $\mathsf{TPRF.Constrain}(\mathsf{pp}, \cdot)$, $\mathsf{TPRF.Eval}(\mathsf{pp}, \cdot, \cdot)$, and $\mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \cdot, \cdot)$, respectively.

Correctness. We now define two notions of correctness for a translucent constrained PRF: evaluation correctness and verification correctness. Intuitively, evaluation correctness states that a constrained key behaves the same as the master PRF key (from which it is derived) on the allowed points. Verification correctness states that the testing algorithm can correctly identify whether a constrained key was used to evaluate the PRF at an allowed point (in which case the verification algorithm outputs 0) or at a restricted point (in which case the verification algorithm outputs 1). Like the constrained PRF constructions of [BV15, BKM17], we present definitions for the computational relaxations of both of these properties.

Definition 4.3 (Correctness Experiment). Fix a security parameter λ, and let $\Pi_{\mathsf{TPRF}} = (\mathsf{TPRF.Setup}, \mathsf{TPRF.SampleKey}, \mathsf{TPRF.Eval}, \mathsf{TPRF.Constrain}, \mathsf{TPRF.ConstrainEval}, \mathsf{TPRF.Test})$ be a translucent constrained PRF (Definition 4.1) with domain $\mathcal{X}$ and range $\mathcal{Y}$. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary and let $\mathcal{S} \subseteq 2^{\mathcal{X}}$ be a set system. The (computational) correctness experiment $\mathsf{Expt}_{\Pi_{\mathsf{TPRF}},\mathcal{A},\mathcal{S}}$ is defined as follows:

Experiment $\mathsf{Expt}_{\Pi_{\mathsf{TPRF}},\mathcal{A},\mathcal{S}}(\lambda)$**:**

1. $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{TPRF.Setup}(1^\lambda)$
2. $\mathsf{msk} \leftarrow \mathsf{TPRF.SampleKey}(\mathsf{pp})$
3. $(S, \mathsf{st}_\mathcal{A}) \leftarrow \mathcal{A}_1(1^\lambda, \mathsf{pp})$ where $S \in \mathcal{S}$
4. Output (x, S) where $x \leftarrow \mathcal{A}_2(\mathsf{st}_\mathcal{A}, \mathsf{sk})$ and $\mathsf{sk} \leftarrow \mathsf{TPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, S)$

Definition 4.4 (Correctness). Fix a security parameter λ, and let $\Pi_{\mathsf{TPRF}} = (\mathsf{TPRF.Setup}, \mathsf{TPRF.SampleKey}, \mathsf{TPRF.Eval}, \mathsf{TPRF.Constrain}, \mathsf{TPRF.ConstrainEval}, \mathsf{TPRF.Test})$ be a translucent constrained PRF with domain $\mathcal{X}$ and range $\mathcal{Y}$. We say that Π_{TPRF} is *correct* with respect to a set system $\mathcal{S} \subseteq 2^{\mathcal{X}}$ if it satisfies the following two properties:

- **Evaluation correctness:** For all efficient adversaries $\mathcal{A}$ and setting $(x, S) \leftarrow \mathsf{Expt}_{\Pi_{\mathsf{TPRF}},\mathcal{A},\mathcal{S}}(\lambda)$,

$$\Pr[x \in S \text{ and } \mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk}_S, x) \neq \mathsf{TPRF.Eval}(\mathsf{pp}, \mathsf{msk}, x)] = \mathsf{negl}(\lambda).$$

- **Verification correctness:** For all efficient adversaries $\mathcal{A}$ and taking $(x, S) \leftarrow \mathsf{Expt}_{\Pi_{\mathsf{TPRF}},\mathcal{A},\mathcal{S}}(\lambda)$,

$$\Pr[x \in \mathcal{X} \setminus S \text{ and } \mathsf{TPRF.Test}(\mathsf{pp}, \mathsf{tk}, \mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk}_S, x)) = 1] = 1 - \mathsf{negl}(\lambda)$$

  and

$$\Pr[x \in S \text{ and } \mathsf{TPRF.Test}(\mathsf{pp}, \mathsf{tk}, \mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk}_S, x)) = 1] = \mathsf{negl}(\lambda).$$

Remark 4.5 (Selective Notions of Correctness). In Definition 4.3, the adversary is able to choose the set $S \in \mathcal{S}$ adaptively, that is, after seeing the public parameters pp. We can define a weaker (but still useful) notion of *selective* correctness, where the adversary is forced to commit to its set S before seeing the public parameters. The formal correctness conditions in Definition 4.4 remain unchanged. For certain set systems (e.g., when all sets $S \in \mathcal{S}$ contain a polynomial number of points), *complexity leveraging* [BB04] can be used to boost a scheme that is selectively correct into one that is also adaptively correct, except under a possibly super-polynomial loss in the security reduction. For constructing a watermarkable family of PRFs (Section 6), a selectively-correct translucent PRF already suffices.

Remark 4.6 (Evaluation Correctness for a Random Point). A useful corollary of evaluation correctness that comes in handy is that whenever the set S of allowed points is a non-negligible fraction of the domain $\mathcal{X}$, (selective) evaluation correctness implies that

$$\Pr_{x \xleftarrow{\text{R}} \mathcal{X}}[\mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk}_S, x) \neq \mathsf{TPRF.Eval}(\mathsf{pp}, \mathsf{msk}, x)] = \mathsf{negl}(\lambda),$$

provided that $\mathsf{pp}, \mathsf{msk}, \mathsf{sk}_S$ are generated using the honest algorithms. In other words, when the set S of allowed points is large, the constrained key agrees with the master PRF key at a random domain element with overwhelming probability.

Translucent puncturable PRFs. A special case of a translucent constrained PRF is a translucent puncturable PRF. Recall that a puncturable PRF [BW13, KPTZ13, BGI14] is a constrained PRF where the constrained keys enable PRF evaluation at all points in the domain $\mathcal{X}$ except at a single, "punctured" point $x^* \in \mathcal{X}$. We can generalize this notion to a *t-puncturable* PRF, which is a PRF that can be punctured at t different points. Formally, we define the analog of a translucent puncturable and t-puncturable PRFs.

Definition 4.7 (Translucent t-Puncturable PRFs). We say that a translucent constrained PRF over a domain $\mathcal{X}$ is a *translucent t-puncturable PRF* if it is constrained with respect to the set system $\mathcal{S}^{(t)} = \{S \subseteq \mathcal{X} : |S| = |\mathcal{X}| - t\}$. The special case of $t = 1$ corresponds to a *translucent puncturable PRF*.

4.1 Security Definitions

We now introduce several security requirements a translucent constrained PRF should satisfy. First, we require that $\mathsf{Eval}(\mathsf{pp}, \mathsf{msk}, \cdot)$ implements a PRF whenever the parameters pp and msk are honestly generated. Next, we require that given a constrained key sk_S for some set S, the real PRF values $\mathsf{Eval}(\mathsf{pp}, \mathsf{msk}, x)$ for points $x \notin S$ remain pseudorandom. This is the notion of constrained pseudorandomness introduced in [BW13]. Using a similar argument as in [BKM17, Appendix A], it follows that a translucent constrained PRF satisfying constrained pseudorandomness is also pseudorandom. Finally, we require that the key sk_S output by $\mathsf{Constrain}(\mathsf{pp}, \mathsf{msk}, S)$ hides the constraint set S. This is essentially the privacy requirement in a private constrained PRF [BLW17].

Definition 4.8 (Pseudorandomness)**.** Let λ be a security parameter, and let $\Pi_{\mathsf{TPRF}} = (\mathsf{TPRF.Setup}, \mathsf{TPRF.SampleKey}, \mathsf{TPRF.Eval}, \mathsf{TPRF.Constrain}, \mathsf{TPRF.ConstrainEval}, \mathsf{TPRF.Test})$ be a translucent constrained PRF with domain $\mathcal{X}$ and range $\mathcal{Y}$. We say that Π_{TPRF} is *pseudorandom* if for $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{TPRF.Setup}(1^\lambda)$, the tuple $(\mathsf{KeyGen}, \mathsf{Eval})$ is a secure PRF (Definition 3.2), where $\mathsf{KeyGen}(1^\lambda)$ outputs a fresh draw $k \leftarrow \mathsf{TPRF.SampleKey}(\mathsf{pp})$ and $\mathsf{Eval}(k, x)$ outputs $\mathsf{TPRF.Eval}(\mathsf{pp}, k, x)$. Note that we implicitly assume that the PRF adversary in this case also is given access to the public parameters pp.

Definition 4.9 (Constrained Pseudorandomness Experiment)**.** Fix a security parameter λ, and let $\Pi_{\mathsf{TPRF}} = (\mathsf{TPRF.Setup}, \mathsf{TPRF.SampleKey}, \mathsf{TPRF.Eval}, \mathsf{TPRF.Constrain}, \mathsf{TPRF.ConstrainEval}, \mathsf{TPRF.Test})$ be a translucent constrained PRF with domain $\mathcal{X}$ and range $\mathcal{Y}$. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary, $\mathcal{S} \subseteq 2^{\mathcal{X}}$ be a set system, and $b \in \{0, 1\}$ be a bit. The constrained pseudorandomness experiment $\mathsf{CExpt}^{(b)}_{\Pi_{\mathsf{TPRF}},\mathcal{A},\mathcal{S}}(\lambda)$ is defined as follows:

Experiment $\mathsf{CExpt}^{(b)}_{\Pi_{\mathsf{TPRF}},\mathcal{A},\mathcal{S}}(\lambda)$**:**

1. $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{TPRF.Setup}(1^\lambda)$
2. $\mathsf{msk} \leftarrow \mathsf{TPRF.SampleKey}(\mathsf{pp})$
3. $(S, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_1^{\mathsf{TPRF.Eval}(\mathsf{pp},\mathsf{msk},\cdot)}(1^\lambda, \mathsf{pp})$ where $S \in \mathcal{S}$
4. Output $b' \leftarrow \mathcal{A}_2^{\mathsf{TPRF.Eval}(\mathsf{pp},\mathsf{msk},\cdot),\mathcal{O}_b(\cdot)}(\mathsf{st}_{\mathcal{A}}, \mathsf{sk})$ where $\mathsf{sk} \leftarrow \mathsf{TPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, S)$ and the challenge oracle $\mathcal{O}_b$ is defined as follows:
   - $\mathcal{O}_0(\cdot) = \mathsf{TPRF.Eval}(\mathsf{pp}, \mathsf{msk}, \cdot)$
   - $\mathcal{O}_1(\cdot) = f(\cdot)$ where $f \xleftarrow{\text{R}} \text{Funs}[\mathcal{X}, \mathcal{Y}]$ is chosen (and fixed) at the beginning of the experiment.

Definition 4.10 (Constrained Pseudorandomness [BW13, adapted])**.** Fix a security parameter λ, and let $\Pi_{\mathsf{TPRF}} = (\mathsf{TPRF.Setup}, \mathsf{TPRF.SampleKey}, \mathsf{TPRF.Eval}, \mathsf{TPRF.Constrain}, \mathsf{TPRF.ConstrainEval}, \mathsf{TPRF.Test})$ be a translucent constrained PRF with domain $\mathcal{X}$ and range $\mathcal{Y}$. We say that an adversary $\mathcal{A}$ is admissible for the constrained pseudorandomness game if all of the queries x that it makes to the evaluation oracle $\mathsf{TPRF.Eval}$ satisfy $x \in S$ and all of the queries it makes to the challenge oracle ($\mathcal{O}_0$ or $\mathcal{O}_1$) satisfy $x \notin S$.[13] Then, we say that Π_{TPRF} satisfies *constrained pseudorandomness* if for all efficient and admissible adversaries $\mathcal{A}$,

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

Theorem 4.11 (Constrained Pseudorandomness Implies Pseudorandomness [BKM17])**.** *Let Π_{TPRF} be a translucent constrained PRF. If Π_{TPRF} satisfies constrained pseudorandomness (Definition 4.10), then it satisfies pseudorandomness (Definition 4.8).*

Proof. Follows by a similar argument as that in [BKM17, Appendix A]. □

[13]In the standard constrained pseudorandomness game introduced in [BW13], the adversary is also allowed to make evaluation queries on values not contained in S. While our construction can be shown to satisfy this stronger property, this is not needed for our watermarking construction. To simplify the presentation and security analysis, we work with this weaker notion here.

Definition 4.12 (Privacy Experiment)**.** Fix a security parameter λ. Let $\Pi_{\mathsf{TPRF}} = (\mathsf{TPRF.Setup}, \mathsf{TPRF.SampleKey}, \mathsf{TPRF.Eval}, \mathsf{TPRF.Constrain}, \mathsf{TPRF.ConstrainEval}, \mathsf{TPRF.Test})$ be a translucent constrained PRF with domain $\mathcal{X}$ and range $\mathcal{Y}$. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary, $\mathcal{S} \subseteq 2^{\mathcal{X}}$ be a set system, and $b \in \{0,1\}$ be a bit. The privacy experiment $\mathsf{PExpt}^{(b)}_{\Pi_{\mathsf{TPRF}},\mathcal{A},\mathcal{S}}(\lambda)$ is defined as follows:

> **Experiment** $\mathsf{PExpt}^{(b)}_{\Pi_{\mathsf{TPRF}},\mathcal{A},\mathcal{S}}(\lambda)$**:**
>
> 1. $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{TPRF.Setup}(1^\lambda)$
> 2. $(S_0, S_1, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_1(1^\lambda, \mathsf{pp})$ where $S_0, S_1 \in \mathcal{S}$
> 3. $\mathsf{sk}_b \leftarrow \mathsf{TPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, S_b)$ where $\mathsf{msk} \leftarrow \mathsf{TPRF.SampleKey}(\mathsf{pp})$
> 4. Output $b' \leftarrow \mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, \mathsf{sk}_b)$

Definition 4.13 (Privacy [BLW17, adapted])**.** Fix a security parameter λ. Let $\Pi_{\mathsf{TPRF}} = (\mathsf{TPRF.Setup}, \mathsf{TPRF.SampleKey}, \mathsf{TPRF.Eval}, \mathsf{TPRF.Constrain}, \mathsf{TPRF.ConstrainEval}, \mathsf{TPRF.Test})$ to be a translucent constrained PRF with domain $\mathcal{X}$ and range $\mathcal{Y}$. We say that Π_{TPRF} is *private* with respect to a set system $\mathcal{S} \subseteq 2^{\mathcal{X}}$ if for all efficient adversaries $\mathcal{A}$,

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

Remark 4.14 (Selective vs. Adaptive Security)**.** We say that a scheme satisfying Definition 4.10 or Definition 4.13 is *adaptively* secure if the adversary chooses the set S (or sets S_0 and S_1) after seeing the public parameters pp for the translucent constrained PRF scheme. As in Definition 4.5, we can define a selective notion of security where the adversary commits to its set S (or S_0 and S_1) at the beginning of the game before seeing the public parameters.

Key injectivity. Another security notion that becomes useful in the context of watermarking is the notion of *key injectivity*. Intuitively, we say a family of PRFs satisfies key injectivity if for all distinct PRF keys k_1 and k_2 (not necessarily uniformly sampled from the key-space), the value of the PRF under k_1 at any point x does not equal the value of the PRF under k_2 at x with overwhelming probability. We note that Cohen et al. [CHN+16] introduce a similar, though incomparable, notion of key injectivity[14] to achieve their strongest notions of watermarking (based on indistinguishability obfuscation). We now give the exact property that suffices for our construction:

Definition 4.15 (Key Injectivity)**.** Fix a security parameter λ and let $\Pi_{\mathsf{TPRF}} = (\mathsf{TPRF.Setup}, \mathsf{TPRF.SampleKey}, \mathsf{TPRF.Eval}, \mathsf{TPRF.Constrain}, \mathsf{TPRF.ConstrainEval}, \mathsf{TPRF.Test})$ be a translucent constrained PRF with domain $\mathcal{X}$. Take $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{TPRF.Setup}(1^\lambda)$, and let $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda \in \mathbb{N}}$ be the set of possible keys output by $\mathsf{TPRF.SampleKey}(\mathsf{pp})$. Then, we say that Π_{TPRF} is key-injective if for all keys $\mathsf{msk}_1, \mathsf{msk}_2 \in \mathcal{K}$, and any $x \in \mathcal{X}$,

$$\Pr[\mathsf{TPRF.Eval}(\mathsf{msk}_1, x) = \mathsf{TPRF.Eval}(\mathsf{msk}_2, x)] = \mathsf{negl}(\lambda),$$

where the probability is taken over the randomness used in $\mathsf{TPRF.Setup}$.

5 Translucent Puncturable PRFs from LWE

In this section, we describe our construction of a translucent t-puncturable PRF. After describing the main construction, we state the concrete correctness and security theorems for our construction. We defer their formal proofs to Appendix A. Our scheme leverages a number of parameters (described in detail at the beginning of Section 5.1). We give concrete instantiations of these parameters based on the requirements of the correctness and security theorems in Section 5.2.

[14] Roughly speaking, Cohen et al. [CHN+16, Definition 7.1] require that for a uniformly random PRF key k, there does not exist a key k' and a point x where $\mathsf{PRF}(k, x) = \mathsf{PRF}(k', x)$. In contrast, our notion requires that any two PRF keys do not agree at any particular point with overwhelming probability.

5.1 Main Construction

In this section, we formally describe our translucent t-puncturable PRF (Definition 4.7). Let λ be a security parameter. Additionally, we define the following scheme parameters:

- (n, m, q, χ) - LWE parameters
- ρ - length of the PRF input
- p - rounding modulus
- t - the number of punctured points (indexed by i)
- N - the dimension of the coefficient vectors $\mathbf{w}_1, \ldots, \mathbf{w}_t$ (indexed by ℓ). Note that $N = m \cdot n$.
- B_{test} - norm bound used by the PRF testing algorithm

Let $\Pi_{\mathsf{HE}} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Enc}, \mathsf{HE.Dec})$ be the (leveled) homomorphic encryption scheme with plaintext space $\{0,1\}^\rho \times \mathbb{Z}_q^N$ from Theorem 3.9. We define the following additional parameters specific to the FHE scheme:

- z - bit-length of a fresh FHE ciphertext (indexed by j)
- τ - bit-length of the FHE secret key (indexed by k)

Next, we define the equality-check circuit $\mathsf{eq}_\ell : \{0,1\}^\rho \times \{0,1\}^\rho \times \mathbb{Z}_q^N \to \mathbb{Z}_q$ where

$$\mathsf{eq}_\ell(x, (x^*, \mathbf{w})) = \begin{cases} w_\ell & \text{if } x = x^* \\ 0 & \text{otherwise,} \end{cases} \tag{5.1}$$

as well as the circuit $C_{\mathsf{Eval}}^{(\ell)} : \{0,1\}^z \times \{0,1\}^\rho \to \{0,1\}^\tau$ for homomorphic evaluation of eq_ℓ:

$$C_{\mathsf{Eval}}^{(\ell)}(\mathsf{ct}, x) = \mathsf{HE.Eval}(\mathsf{eq}_\ell(x, \cdot), \mathsf{ct}). \tag{5.2}$$

Finally, we define the following additional parameters for the depths of these two circuits:

- d_{eq} - depth of the equality-check circuit eq_ℓ
- d - depth of the homomorphic equality-check circuit $C_{\mathsf{Eval}}^{(\ell)}$

For $\ell \in [N]$, we define the matrix $\mathbf{D}_\ell$ to be the ℓ^{th} elementary "basis matrix" for the $\mathbb{Z}_q$-module $\mathbb{Z}_q^{n \times m}$. More concretely,

$$\mathbf{D}_\ell[a, b] = \begin{cases} 1 & \text{if } am + b = \ell \\ 0 & \text{otherwise.} \end{cases}$$

In other words, each matrix $\mathbf{D}_\ell$ has its ℓ^{th} component (when viewing the matrix as a collection of $N = mn$ entries) set to 1 and the remaining components set to 0.

Translucent PRF construction. The translucent t-puncturable PRF $\Pi_{\mathsf{TPRF}} = (\mathsf{TPRF.Setup}, \mathsf{TPRF.Eval}, \mathsf{TPRF.Constrain}, \mathsf{TPRF.ConstrainEval}, \mathsf{TPRF.Test})$ with domain $\{0,1\}^\rho$ and range $\mathbb{Z}_p^m$ is defined as follows:

- $\mathsf{TPRF.Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm samples the following matrices uniformly at random from $\mathbb{Z}_q^{n \times m}$:
  - $\hat{\mathbf{A}}$: an auxiliary matrix used to provide additional randomness
  - $\{\mathbf{A}_b\}_{b \in \{0,1\}}$: matrices to encode the bits of the input to the PRF
  - $\{\mathbf{B}_{i,j}\}_{i \in [t], j \in [z]}$: matrices to encode the bits of the FHE encryptions of the punctured points
  - $\{\mathbf{C}_k\}_{k \in [\tau]}$: matrices to encode the bits of the FHE secret key

  It also samples trapdoor matrices $(\mathbf{W}_i, \mathbf{z}_i) \leftarrow \mathsf{TrapGen}(1^n, q)$ for all $i \in [t]$. Finally, it outputs the public parameters pp and testing key tk:

$$\mathsf{pp} = \left(\hat{\mathbf{A}}, \{\mathbf{A}_b\}_{b \in \{0,1\}}, \{\mathbf{B}_{i,j}\}_{i \in [t], j \in [z]}, \{\mathbf{C}_k\}_{k \in [\tau]}, \{\mathbf{W}_i\}_{i \in [t]}\right) \qquad \mathsf{tk} = \{\mathbf{z}_i\}_{i \in [t]}.$$

- $\mathsf{TPRF.SampleKey}(\mathsf{pp})$: On input the public parameters pp, the key generation algorithm samples a PRF key $\mathbf{s} \leftarrow \chi^n$ and sets $\mathsf{msk} = \mathbf{s}$.
- $\mathsf{TPRF.Eval}(\mathsf{pp}, \mathsf{msk}, x)$: On input the public parameters pp, the PRF key $\mathsf{msk} = \mathbf{s}$, and an input $x = x_1 x_2 \cdots x_\rho \in \{0,1\}^\rho$, the evaluation algorithm first computes

$$\widetilde{\mathbf{B}}_{i,\ell} \leftarrow \mathsf{Eval}_{\mathsf{pk}}\left(C_\ell, \mathbf{B}_{i,1}, \ldots, \mathbf{B}_{i,z}, \mathbf{A}_{x_1}, \ldots, \mathbf{A}_{x_\rho}, \mathbf{C}_1, \ldots, \mathbf{C}_\tau\right)$$

for all $i \in [t]$ and $\ell \in [N]$, and where $C_\ell = \mathsf{IP} \circ C_{\mathsf{Eval}}^{(\ell)}$. Finally, the evaluation algorithm outputs the value

$$\mathbf{y}_x = \left\lfloor \mathbf{s}^T \left(\hat{\mathbf{A}} + \sum_{\substack{i \in [t] \\ \ell \in [N]}} \widetilde{\mathbf{B}}_{i,\ell} \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) \right) \right\rceil_p .$$

- $\mathsf{TPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, \mathsf{T})$:[15] On input the public parameters pp, the PRF key $\mathsf{msk} = \mathbf{s}$ and the set of points $\mathsf{T} = \{x_i^*\}_{i \in [t]}$ to be punctured, the constraining algorithm first computes

$$\widetilde{\mathbf{B}}_{i,i^*,\ell} \leftarrow \mathsf{Eval}_{\mathsf{pk}}(C_\ell, \mathbf{B}_{i,1}, \ldots, \mathbf{B}_{i,z}, \mathbf{A}_{x^*_{i^*,1}}, \ldots, \mathbf{A}_{x^*_{i^*,\rho}}, \mathbf{C}_1, \ldots, \mathbf{C}_\tau)$$

for all $i, i^* \in [t]$ and $\ell \in [N]$ where $C_\ell = \mathsf{IP} \circ C_{\mathsf{Eval}}^{(\ell)}$. Then, for each $i^* \in [t]$, the puncturing algorithm computes the (unique) vector $\mathbf{w}_{i^*} = (w_{i^*,1}, \ldots, w_{i^*,N}) \in \mathbb{Z}_q^N$ where

$$\mathbf{W}_{i^*} = \hat{\mathbf{A}} + \sum_{\substack{i \in [t] \\ \ell \in [N]}} \widetilde{\mathbf{B}}_{i,i^*,\ell} \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) + \sum_{\ell \in [N]} w_{i^*,\ell} \cdot \mathbf{D}_\ell.$$

Next, it samples an FHE key $\mathsf{HE.sk} \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^{d_{\mathsf{eq}}}, 1^{\rho+N})$, and for each $i \in [t]$, it constructs the ciphertext $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(\mathsf{HE.sk}, (x_i^*, \mathbf{w}_i))$ and finally, it defines $\mathsf{ct} = \{\mathsf{ct}_i\}_{i \in [t]}$. It samples error vectors $\mathbf{e}_0 \leftarrow \chi^m$, $\mathbf{e}_{1,b} \leftarrow \chi^m$ for $b \in \{0,1\}$, $\mathbf{e}_{2,i,j} \leftarrow \chi^m$ for $i \in [t]$ and $j \in [z]$, and $\mathbf{e}_{3,k} \leftarrow \chi^m$ for $k \in [\tau]$ and computes the vectors

$$\begin{array}{rcll} \hat{\mathbf{a}}^T &=& \mathbf{s}^T \hat{\mathbf{A}} + \mathbf{e}_0^T & \\ \mathbf{a}_b^T &=& \mathbf{s}^T(\mathbf{A}_b + b \cdot \mathbf{G}) + \mathbf{e}_{1,b}^T & \forall b \in \{0,1\} \\ \mathbf{b}_{i,j}^T &=& \mathbf{s}^T(\mathbf{B}_j + \mathsf{ct}_{i,j} \cdot \mathbf{G}) + \mathbf{e}_{2,i,j}^T & \forall i \in [t], \forall j \in [z] \\ \mathbf{c}_k^T &=& \mathbf{s}^T(\mathbf{C}_k + \mathsf{HE.sk}_k \cdot \mathbf{G}) + \mathbf{e}_{3,k}^T & \forall k \in [\tau]. \end{array}$$

Next, it sets $\mathsf{enc} = \left(\hat{\mathbf{a}}, \{\mathbf{a}_b\}_{b \in \{0,1\}}, \{\mathbf{b}_{i,j}\}_{i \in [t], j \in [z]}, \{\mathbf{c}_k\}_{k \in [\tau]}\right)$. It outputs the constrained key $\mathsf{sk}_\mathsf{T} = (\mathsf{enc}, \mathsf{ct})$.

- $\mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk}_\mathsf{T}, x)$: On input the public parameters pp, a constrained key $\mathsf{sk}_\mathsf{T} = (\mathsf{enc}, \mathsf{ct})$, where $\mathsf{enc} = \left(\hat{\mathbf{a}}, \{\mathbf{a}_b\}_{b \in \{0,1\}}, \{\mathbf{b}_{i,j}\}_{i \in [t], j \in [z]}, \{\mathbf{c}_k\}_{k \in [\tau]}\right)$, $\mathsf{ct} = \{\mathsf{ct}_i\}_{i \in [t]}$, and a point $x \in \{0,1\}^\rho$, the constrained evaluation algorithm computes

$$\widetilde{\mathbf{b}}_{i,\ell} \leftarrow \mathsf{Eval}_{\mathsf{ct}}((\mathsf{ct}_i, x), C_\ell, \mathbf{b}_{i,1}, \ldots, \mathbf{b}_{i,z}, \mathbf{a}_{x_1}, \ldots, \mathbf{a}_{x_\rho}, \mathbf{c}_1, \ldots, \mathbf{c}_\tau)$$

for $i \in [t]$ and $\ell \in [N]$, and where $C_\ell(\mathsf{ct}, x) = \mathsf{IP} \circ C_{\mathsf{Eval}}^{(\ell)}$. Then, it computes and outputs the value

$$\mathbf{y}_x = \left\lfloor \hat{\mathbf{a}} + \sum_{\substack{i \in [t] \\ \ell \in [N]}} \widetilde{\mathbf{b}}_{i,\ell}^T \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) \right\rceil_p .$$

[15] For notational convenience, we modify the syntax of the constrain algorithm to take in a set T of t punctured points rather than a set of allowed points.

- $\mathsf{TPRF.Test}(\mathsf{pp}, \mathsf{tk}, \mathbf{y})$: On input the testing key $\mathsf{tk} = \{\mathbf{z}_i\}_{i \in [t]}$ and a point $\mathbf{y} \in \mathbb{Z}_p^m$, the testing algorithm outputs 1 if $\langle \mathbf{y}, \mathbf{z}_i \rangle \in [-B_{\mathsf{test}}, B_{\mathsf{test}}]$ for some $i \in [t]$ and 0 otherwise.

Correctness theorem. We now state that under the LWE and $\mathsf{1D\text{-}SIS}$ assumptions (with appropriate parameters), our translucent t-puncturable PRF Π_{TPRF} satisfies (selective) evaluation correctness and verification correctness (Definition 4.4, Remark 4.5). We give the formal proof in Appendix A.2.

Theorem 5.1 (Correctness). *Fix a security parameter λ, and define parameters $n, m, p, q, \chi, t, z, \tau, B_{\mathsf{test}}$ as above (such that Theorems 3.9 and 3.10 hold). Let B be a bound on the error distribution χ, and suppose $B_{\mathsf{test}} = B(m+1)$, $p = 2^{\rho^{(1+\varepsilon)}}$ for some constant $\varepsilon > 0$, and $\frac{q}{2pmB} > B \cdot m^{O(d)}$. Then, take $m' = m \cdot (3 + t \cdot z + \tau)$ and $\beta = B \cdot m^{O(d)}$. Under the $\mathsf{LWE}_{n,m',q,\chi}$ and $\mathsf{1D\text{-}SIS\text{-}R}_{m',p,q,\beta}$ assumptions, Π_{TPRF} is (selectively) correct.*

Security theorems. We now state that under the LWE assumption (with appropriate parameters), our translucent t-puncturable PRF Π_{TPRF} satisfies selective constrained pseudorandomness (Definition 4.10), selective privacy (Definition 4.13) and weak key-injectivity (Definition 4.15). We give the formal proofs in Appendix A.3. As a corollary of satisfying constrained pseudorandomness, we have that Π_{TPRF} is also pseudorandom (Definition 4.8, Theorem 4.11).

Theorem 5.2 (Constrained Pseudorandomness). *Fix a security parameter λ, and define parameters $n, m, p, q, \chi, t, z, \tau$ as above (such that Theorems 3.9 and 3.10 hold). Let $m' = m \cdot (3 + t(z+1) + \tau)$, $m'' = m \cdot (3 + t \cdot z + \tau)$ and $\beta = B \cdot m^{O(d)}$ where B is a bound on the error distribution χ. Then, under the $\mathsf{LWE}_{n,m',q,\chi}$ and $\mathsf{1D\text{-}SIS\text{-}R}_{m'',p,q,\beta}$ assumptions, Π_{TPRF} satisfies selective constrained pseudorandomness (Definition 4.10).*

Corollary 5.3 (Pseudorandomness). *Fix a security parameter λ, and define the parameters $n, m, p, q, \chi, t, z, \tau$ as above. Under the same assumptions as in Theorem 5.2, Π_{TPRF} satisfies selective pseudorandomness (Definition 4.8).*

Theorem 5.4 (Privacy). *Fix a security parameter λ, and define parameters $n, m, q, \chi, t, z, \tau$ as above (such that Theorems 3.9 and 3.10 hold). Let $m' = m \cdot (3 + t(z+1) + \tau)$. Then, under the $\mathsf{LWE}_{n,m',q,\chi}$ assumption, and assuming the homomorphic encryption scheme Π_{HE} is semantically secure, Π_{TPRF} is selectively private (Definition 4.13).*

Theorem 5.5 (Key-Injectivity). *If the bound B on the error distribution χ satisfies $B < \hat{p}/2$ where $\hat{p}$ is the smallest prime dividing the modulus q, and $m = \omega(n)$, then the translucent t-puncturable PRF Π_{TPRF} satisfies key-injectivity (Definition 4.15).*

5.2 Concrete Parameter Instantiations

In this section, we give one possible instantiation for the parameters for the translucent t-puncturable PRF construction in Section 5.1. We choose our parameters so that the underlying LWE and 1D-SIS assumptions that we rely on are as hard as approximating worst-case lattice problems to within a subexponential factor $2^{\tilde{O}(n^{1/c})}$ for some constant c (where n is the lattice dimension). Fix a constant c and a security parameter λ.

- We set the PRF input length $\rho = \lambda$. Then, the depth d_{eq} of the equality check circuit eq_ℓ satisfies $d_{\mathsf{eq}} = O(\log \rho) = O(\log \lambda)$.
- We set the lattice dimension $n = \lambda^{2c}$.
- The noise distribution χ is set to be the discrete Gaussian distribution $D_{\mathbb{Z},\sqrt{n}}$. Then the FHE ciphertext length z and the FHE secret key length τ is determined by $\mathsf{poly}(\lambda, d_{\mathsf{eq}}, \rho, \log q) = \mathsf{poly}(\lambda)$. By Theorem 3.9, the depth of the FHE equality check circuit is $d = \mathsf{poly}(d_{\mathsf{eq}}, \log z) = \mathsf{polylog}(\lambda)$. Finally, we set $B_{\mathsf{test}} = B \cdot (m+1)$.

- We choose the modulus q to be large enough to be able to invoke Theorems 3.9 and 3.10. If the initial error distribution χ is B-bounded, then Theorem 3.9 requires that $q > m^{O(d_{\mathsf{eq}})}$ and Theorem 3.10 requires that $q > m^{O(d)}$. Furthermore, for the 1D-SIS-R assumption, we need q to be the product of λ primes $p_1, \ldots, p_\lambda$. For each $i \in [\lambda]$, we set the primes $p_j = 2^{O(n^{1/2c})}$ such that $p_1 < \cdots < p_\lambda$.

- We set $p = 2^{n^{1/2c+\varepsilon}}$ for any $\varepsilon > 0$, so the condition in Theorem 5.1 is satisfied.

- We set $m = \Theta(n \log q)$, and $B_{\mathsf{test}} = B \cdot (m+1)$. For these parameter settings, $m^{O(d)} = m^{\mathsf{polylog}(\lambda)}$ and $q = 2^{\tilde{O}(n^{1/2c})} = 2^{\tilde{O}(\lambda)}$.

Under these parameter setting, the private translucent t-puncturable PRF in Section 5.1 is selectively secure assuming the polynomial hardness of approximating worst-case lattice problems over an n-dimensional lattice to within a subexponential approximation factor $2^{\tilde{O}(n^{1/2c})}$. Using complexity leveraging [BB04], the same construction is adaptively secure assuming subexponential hardness of the same worst-case lattice problems.

6 Watermarkable PRFs from Translucent PRFs

In this section, we formally introduce the notion of a watermarkable family of PRFs. Our definitions are adapted from those of [CHN+16, BLW17]. Then, in Section 6.2, we show how to construct a secretly-extractable, message-embedding watermarkable family of PRFs from translucent t-puncturable PRFs. Combined with our concrete instantiation of translucent t-puncturable PRFs from Section 5, this gives the first watermarkable family of PRFs (with security against arbitrary unremoving strategies) from standard assumptions.

6.1 Watermarking PRFs

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

Definition 6.1 (Watermarkable Family of PRFs [BLW17, adapted]). Fix a security parameter λ and a message space $\{0,1\}^t$. Then, a secretly-extractable, message-embedding watermarking scheme for a PRF $\Pi_{\mathsf{PRF}} = (\mathsf{PRF.KeyGen}, \mathsf{PRF.Eval})$ is a tuple of algorithms $\Pi_{\mathsf{WM}} = (\mathsf{WM.Setup}, \mathsf{WM.Mark}, \mathsf{WM.Extract})$ with the following properties:

- $\mathsf{WM.Setup}(1^\lambda) \to \mathsf{msk}$: On input the security parameter λ, the setup algorithm outputs the watermarking secret key msk.

- $\mathsf{WM.Mark}(\mathsf{msk}, k, m) \to C$: On input the watermarking secret key msk, a PRF key k (to be marked), and a message $m \in \{0,1\}^t$, the mark algorithm outputs a marked circuit C.

- $\mathsf{WM.Extract}(\mathsf{msk}, C') \to m$: On input the master secret key msk and a circuit C', the extraction algorithm outputs a string $m \in \{0,1\}^t \cup \{\perp\}$.

Definition 6.2 (Circuit Similarity). Fix a circuit class $\mathcal{C}$ on n-bit inputs. For two circuits $C, C' \in \mathcal{C}$ and for a non-decreasing function $f : \mathbb{N} \to \mathbb{N}$, we write $C \sim_f C'$ to denote that the two circuits agree on all but an $1/f(n)$ fraction of inputs. More formally, we define

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

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

Remark 6.3 (Public vs. Secret Extraction). Definition 6.1 defines a *secretly-extractable* watermarking scheme, which means that only those who possess the secret key msk are able to extract the message from a marked circuit. A stronger notion of watermarking is *publicly-extractable* watermarking, which means that anyone can test whether a particular program is watermarked or not (and if so, extract the embedded message). Publicly-extractable software watermarking was first introduced by Cohen et al. [CHN+16], who in the same work, gave the first construction of a publicly-extractable watermarking construction for PRFs using indistinguishability obfuscation. While the construction we present operates in the secret-key setting (see Remark 6.4 for a discussion of some of the challenges we encounter when attempting to extend our construction), we stress that even in the secret-key setting, all software watermarking schemes prior to this work (satisfying the strongest notion of unremovability against arbitrary strategies) relied on the full power of indistinguishability obfuscation [CHV15, NW15, BLW17]. Our construction is the first software watermarking construction (of any kind) that is robust against arbitrary removal strategies from standard assumptions.

Remark 6.4 (Difficulty with Public Extraction)**.** It appears difficult to extend our construction to support public extraction. Extending our construction to support public extraction seems to require the contradictory property that the set of "marked points" $x_i^{(m_i)}$ for a circuit is unknown to the adversary (even given the extraction key), and yet, there is an efficient algorithm to sample a sequence of marked points (to run the extraction algorithm). Otherwise, if the public extraction key allows the adversary to efficiently tell whether a particular point is marked, then it can trivially remove the watermark. Cohen et al. [CHN+16] solve this problem by encrypting the marked points (which themselves constitutes a sparse, pseudorandom subset of the domain) and embedding a decryption key inside the (obfuscated) watermarked program. When the program is invoked on an encrypted marked point, the obfuscated program instead outputs a reprogrammed value that can be used to recover the message. We leave as an open problem the construction of a publicly-extractable watermarking scheme from standard assumptions.

Correctness. The correctness property for a watermarking scheme for a PRF family consists of two requirements. The first requirement is that a watermarked key behaves like the original (unmarked) key almost everywhere. In particular, the watermarked key must agree with the unmarked key on all but a negligible fraction of points. While we might desire correctness on all points, Barak et al. [BGI+12] previously showed that assuming indistinguishable obfuscation, perfect functionality-preserving watermarking is generally impossible. Thus, in some sense, approximate correctness is the best we could hope to achieve, and indeed, this is the notion satisfied by existing watermarking candidates [CHN+16, BLW17]. The second correctness requirement is that if we embed a message into a key, then the extraction algorithm should be able to extract the embedded message from the key. We now give the formal definition.

Definition 6.5 (Watermarking Correctness)**.** Fix a security parameter λ. We say that a watermarking scheme $\Pi_{\mathsf{WM}} = (\mathsf{WM.Setup}, \mathsf{WM.Mark}, \mathsf{WM.Extract})$ for a PRF $\Pi_{\mathsf{PRF}} = (\mathsf{PRF.KeyGen}, \mathsf{PRF.Eval})$ with domain $\{0,1\}^n$ is *correct* if for all messages $m \in \{0,1\}^t$, and setting $\mathsf{msk} \leftarrow \mathsf{WM.Setup}(1^\lambda)$, $k \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$, and $C \leftarrow \mathsf{WM.Mark}(\mathsf{msk}, k, m)$, the following two properties hold:

- **Functionality-preserving:** $C(\cdot) \sim_f \mathsf{PRF.Eval}(k, \cdot)$ where $1/f(n) = \mathsf{negl}(\lambda)$ with overwhelming probability.
- **Extraction correctness:** $\Pr[\mathsf{WM.Extract}(\mathsf{msk}, C) = m] = 1 - \mathsf{negl}(\lambda)$.

Remark 6.6 (Stronger Correctness Notions)**.** We note that the correctness properties we introduced in Definition 6.5 are only required to hold when the underlying PRF key is sampled honestly (i.e., using the $\mathsf{PRF.KeyGen}$ algorithm). This is also the notion considered in [BLW17] (in fact, in their construction, the only keys that can be watermarked are those sampled using the honest PRF key-generation algorithm). In contrast, using indistinguishability obfuscation, Cohen et al. [CHN+16] achieve a stronger notion of correctness where functionality-preserving and extraction correctness hold (with high probability) even if the PRF key to be watermarked is chosen maliciously. The reason our construction is unable to achieve the strengthened correctness notion is because our translucent t-puncturable PRF from Section 5.1 only satisfies a computational notion of correctness (rather than a statistical notion). This seems to be a limitation present in several lattice-based constrained PRF constructions [BV15, BKM17] (though not the construction in [CC17]). It is an interesting problem to construct translucent t-puncturable PRFs that achieve statistical correctness; such a construction would give rise to watermarkable PRFs with stronger correctness properties. Finally, it is important to note that our notion of correctness suffices for most, if not all, of the applications of watermarking for PRFs. After all, if the PRF key is not chosen honestly, then the underlying PRF itself is no longer secure.

Security. Following [CHN+16, BLW17], we introduce two different security notions for a watermarking scheme. The first notion is unremovability, which states that no efficient adversary should be able to remove a watermark from a watermarked program without significantly modifying the behavior of the program. The second notion is unforgeability, which states that no efficient adversary should be able to produce a watermarked program that is substantially different from the watermarked program it already possesses. We begin by defining the watermarking experiment.

Definition 6.7 (Watermarking Experiment [BLW17, adapted])**.** Fix a security parameter λ. Let $\Pi_{\mathsf{WM}} = (\mathsf{WM.Setup}, \mathsf{WM.Mark}, \mathsf{WM.Extract})$ be a watermarking scheme for a PRF $\Pi_{\mathsf{PRF}} = (\mathsf{PRF.KeyGen}, \mathsf{PRF.Eval})$ with key-space $\mathcal{K}$, and let $\mathcal{A}$ be an adversary. Then the watermarking experiment $\mathsf{Expt}_{\Pi_{\mathsf{WM}},\mathcal{A}}(\lambda)$ proceeds as follows. The challenger begins by sampling $\mathsf{msk} \leftarrow \mathsf{WM.Setup}(1^\lambda)$. The adversary $\mathcal{A}$ is then given access to the following oracles:

- **Marking oracle.** On input a message $m \in \{0,1\}^t$ and a PRF key $k \in \mathcal{K}$, the challenger returns the circuit $C \leftarrow \mathsf{WM.Mark}(\mathsf{msk}, k, m)$ to $\mathcal{A}$.
- **Challenge oracle.** On input a message $m \in \{0,1\}^t$, the challenger samples a key $k \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$, and returns the circuit $C \leftarrow \mathsf{WM.Mark}(\mathsf{msk}, k, m)$ to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs a circuit C'. The output of the experiment, denoted $\mathsf{Expt}_{\Pi_{\mathsf{WM}},\mathcal{A}}(\lambda)$, is $\mathsf{WM.Extract}(\mathsf{msk}, C')$.

Definition 6.8 (Unremovability [CHN+16, BLW17])**.** Fix a security parameter λ. For a watermarking scheme $\Pi_{\mathsf{WM}} = (\mathsf{WM.Setup}, \mathsf{WM.Mark}, \mathsf{WM.Extract})$ for a PRF $\Pi_{\mathsf{PRF}} = (\mathsf{PRF.KeyGen}, \mathsf{PRF.Eval})$ and an adversary $\mathcal{A}$, we say that $\mathcal{A}$ is *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 $\tilde{C} \sim_f \hat{C}$, where $\hat{C}$ is the circuit output by the challenge oracle and $1/f = \mathsf{negl}(\lambda)$.

Then, we say that Π_{WM} is *unremovable* if for all efficient and unremoving-admissible adversaries $\mathcal{A}$,

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

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

Definition 6.9 (δ-Unforgeability [CHN+16, BLW17])**.** Fix a security parameter λ. For a watermarking scheme $\Pi_{\mathsf{WM}} = (\mathsf{WM.Setup}, \mathsf{WM.Mark}, \mathsf{WM.Extract})$ for a PRF $\Pi_{\mathsf{PRF}} = (\mathsf{PRF.KeyGen}, \mathsf{PRF.Eval})$ and an adversary $\mathcal{A}$, we say that $\mathcal{A}$ is *δ-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 $\tilde{C} \not\sim_f C_\ell$ for all $\ell \in [Q]$, where Q is the number of queries $\mathcal{A}$ made to the marking oracle, C_ℓ is the output of the marking oracle on the ℓ^{th} query, and $1/f > \delta$. Moreover, $\tilde{C} \not\sim_f \mathsf{PRF.Eval}(k_\ell, \cdot)$, where k_ℓ is the key the adversary submitted on its ℓ^{th} query to the marking oracle.

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

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

Remark 6.10 (Giving Access to an Extraction Oracle)**.** As noted in [CHN+16], in the secret-key setting, the watermarking security game (Definition 6.7) can be augmented to allow the adversary oracle access to an extraction oracle (which implements $\mathsf{WM.Extract}(\mathsf{msk}, \cdot)$). However, achieving security in the presence of an extraction oracle introduces many of the same challenges that arise in supporting public extraction (see Remark 6.4). It is an interesting open problem to construct secretly-extractable watermarking from standard assumptions where the adversary is additionally given access to a extraction oracle. The only known constructions today [CHN+16] rely on indistinguishability obfuscation.

Remark 6.11 (Marking Oracle Variations)**.** In our watermarking security game (Definition 6.7), the adversary is allowed to submit arbitrary keys (of its choosing) to the marking oracle. Cohen et al. [CHN+16] also consider a stronger notion where the adversary is allowed to submit arbitrary *circuits* (not corresponding to any particular PRF) to the marking oracle. However, in this model, they can only achieve lunch-time security (i.e., the adversary can only query the marking oracle before issuing its challenge query). In the model where the adversary can only query the marking oracle on valid PRF keys, their construction achieves full security (assuming the PRF family satisfies a key-injectivity property). Similarly, our construction achieves *full* security in this model (in the secret-key setting). Our notion is strictly stronger than the notion considered by Boneh et al. [BLW17]. There, the adversary is *not* allowed to provide a key to the marking oracle, and instead, the marking oracle samples a key (honestly) and gives both the sampled key as well as the watermarked key to the adversary. In contrast, in both our model as well as that in [CHN+16], the adversary is allowed to see watermarked keys on *arbitrary* keys of the adversary's choosing.

Remark 6.12 (Unforgeability and Correctness)**.** The admissibility requirement in the δ-unforgeability game (Definition 6.9) says that the "forged" program the adversary outputs must differ (by at least a δ-fraction) from both the marked programs output by the marking oracle as well as the original programs it submitted to the marking oracle. If all of the PRF keys given to the marking oracle are honestly generated, then by correctness of the watermarking scheme, the marked program and the original program differ only on a negligible fraction of points. In this case, it is redundant to separately require the adversary's program to differ from the unmarked programs. However, for adversarially-chosen keys, it is possible that the unmarked program and the marked program differ on a large fraction of points. While this does not lead to a trivial attack on the scheme, it becomes significantly more difficult to reason about security in these cases. Thus, we relax the unforgeability requirement slightly to require that the adversary produces a circuit that is substantially different from any program it submits or receives from the marking oracle. We note here that if the watermarking scheme satisfies a statistical notion of correctness (similar to [CHN+16]), then this definition becomes equivalent to just requiring that the adversary's program be different only from the marked programs it receives from the marking oracle.

6.2 Watermarking Construction

In this section, we show how any translucent t-puncturable PRF can be used to obtain a watermarkable family of PRFs. Combined with our construction of a translucent t-puncturable PRF from Section 5.1, we obtain the first watermarkable family of PRFs from standard assumptions. We conclude by stating our correctness and security theorems.

Construction 6.13. Fix a security parameter λ and a positive real value $\delta < 1$ such that $d = \lambda/\delta = \mathsf{poly}(\lambda)$. Let $\{0,1\}^t$ be the message space for the watermarking scheme. Our construction relies on the following two ingredients:

- Let $\Pi_{\mathsf{TPRF}} = (\mathsf{TPRF.Setup}, \mathsf{TPRF.SampleKey}, \mathsf{TPRF.Eval}, \mathsf{TPRF.Constrain}, \mathsf{TPRF.ConstrainEval}, \mathsf{TPRF.Test})$ be a translucent t-puncturable PRF (Definition 4.7) with key-space $\mathcal{K}$, domain $\{0,1\}^n$, and range $\{0,1\}^m$.
- Let $\Pi_{\mathsf{PRF}} = (\mathsf{PRF.KeyGen}, \mathsf{PRF.Eval})$ be a secure PRF with domain $(\{0,1\}^m)^d$ and range $(\{0,1\}^n)^{2t}$.

We require $n, m, t = \omega(\log \lambda)$. The secretly-extractable, message-embedding watermarking scheme $\Pi_{\mathsf{WM}} = (\mathsf{WM.Setup}, \mathsf{WM.Mark}, \mathsf{WM.Extract})$ for the PRF associated with Π_{TPRF} is defined as follows:

- $\mathsf{WM.Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm runs $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{TPRF.Setup}(1^\lambda)$. Next, for each $j \in [d]$, it samples $h_j \xleftarrow{\text{R}} \{0,1\}^n$. It also samples a key $\mathsf{k}^* \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$. Finally, it outputs the master secret key $\mathsf{msk} = (\mathsf{pp}, \mathsf{tk}, h_1, \ldots, h_d, \mathsf{k}^*)$.
- $\mathsf{WM.Mark}(\mathsf{msk}, k, m)$: On input the master secret key $\mathsf{msk} = (\mathsf{pp}, \mathsf{tk}, h_1, \ldots, h_d, \mathsf{k}^*)$, a PRF key $k \in \mathcal{K}$ to be marked, and a message $m \in \{0,1\}^t$, the marking algorithm proceeds as follows:
  1. For each $j \in [d]$, set $y_j \leftarrow \mathsf{TPRF.Eval}(\mathsf{pp}, k, h_j)$. Let $\mathbf{y} = (y_1, \ldots, y_d)$.
  2. Compute points $\mathbf{x} = \left(x_1^{(0)}, x_1^{(1)}, \ldots, x_t^{(0)}, x_t^{(1)}\right) \leftarrow \mathsf{PRF.Eval}(\mathsf{k}^*, \mathbf{y})$.
  3. Compute the t-punctured key $\mathsf{sk}_S \leftarrow \mathsf{TPRF.Constrain}(\mathsf{pp}, k, S)$, where the set S is given by $S = \{x \in \{0,1\}^n : x \neq x_i^{(m_i)} \ \forall i \in [t]\}$,
  4. Output the circuit C where $C(\cdot) = \mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk}_S, \cdot)$.
- $\mathsf{WM.Extract}(\mathsf{msk}, C)$: On input the master secret key $\mathsf{msk} = (\mathsf{pp}, \mathsf{tk}, h_1, \ldots, h_d, k)$ and a circuit $C : \{0,1\}^n \to \{0,1\}^m$, the extraction algorithm proceeds as follows:
  1. Compute points $\mathbf{x} = \left(x_1^{(0)}, x_1^{(1)}, \ldots, x_t^{(0)}, x_t^{(1)}\right) \leftarrow \mathsf{PRF.Eval}(\mathsf{k}^*, C(h_1), \ldots, C(h_d))$.
  2. For each $i \in [t]$, and $b \in \{0,1\}$, compute $z_i^{(b)} = \mathsf{TPRF.Test}(\mathsf{pp}, \mathsf{tk}, C(x_i^{(b)}))$.
  3. If there exists some i for which $z_i^{(0)} = z_i^{(1)}$, output $\perp$. Otherwise, output the message $m \in \{0,1\}^t$ where $m_i = 0$ if $z_i^{(0)} = 1$ and $m_i = 1$ if $z_i^{(1)} = 1$.

Security analysis. We now state the correctness and security theorems for our construction, but defer their formal proofs to Appendix B.

Theorem 6.14. *If Π_{TPRF} is a secure translucent t-puncturable PRF, and Π_{PRF} is a secure PRF, then the watermarking schemeΠ_{WM} in Construction 6.13 is correct.*

Theorem 6.15. *If Π_{TPRF} is a selectively-secure translucent t-puncturable PRF, and Π_{PRF} is secure, then the watermarking schemeΠ_{WM} in Construction 6.13 is unremovable.*

Theorem 6.16. *If Π_{TPRF} is a selectively-secure translucent t-puncturable PRF, and Π_{PRF} is secure, then the watermarking schemeΠ_{WM} in Construction 6.13 is δ-unforgeable.*

Acknowledgments

This work was funded by NSF, DARPA, a grant from ONR, the Simons Foundation, and an NSF Graduate Research Fellowship. 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.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *CRYPTO*, 2014.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, 2004.

[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.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.

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

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *CRYPTO*, 2012.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, 2011.

[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.

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In *CRYPTO*, 1997.

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

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

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In *CRYPTO*, 2015.

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

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

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *TCC*, pages 498–527, 2015.

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

[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.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013.

[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 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 *CCS*, 2013.

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

[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.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In *EUROCRYPT*, 2016.

[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.

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

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

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

[PS16] Chris Peikert and Sina Shiehian. Multi-key FHE from lwe, revisited. In *TCC*, 2016.

[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.

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

A Translucent PRF Correctness and Security Analysis

In this section, we give the formal correctness and security analysis of the private translucent t-puncturable PRF construction from Section 5.1. Our analysis leverages a number of similar components. To streamline the presentation in Appendices A.2 and A.3, we first introduce a set of auxiliary algorithms that we will use throughout the analysis in Appendix A.1. We then give the correctness proof in Appendix A.2 and the security proofs in Appendix A.3.

A.1 Correctness and Security Analysis: Auxiliary Algorithms

In this section, we introduce the auxiliary algorithm that will be used in the correctness and security proofs in the subsequent sections.

- $\mathsf{Setup}^*(1^\lambda, \mathsf{T}) \to (\mathsf{pp}^*, \mathsf{msk}^*)$: On input the security parameter λ, and the set $\mathsf{T} = \{x_i^*\}_{i\in[t]}$ of punctured points, the auxiliary setup algorithm first samples matrices $\hat{\mathbf{A}}$, $\{\mathbf{A}'_b\}_{b\in\{0,1\}}$, $\{\mathbf{B}'_{i,j}\}_{i\in[t],j\in[z]}$, and $\{\mathbf{C}'_k\}_{k\in[\tau]}$ uniformly at random from $\mathbb{Z}_q^{n\times m}$ and sample vectors $\{\mathbf{w}_i\}_{i\in[t]}$ uniformly at random from $\mathbb{Z}_q^n$. Then, it generates an FHE secret key $\mathsf{HE.sk} \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^{d_{\mathsf{eq}}}, 1^{\rho+N})$, and for all $i \in [t]$, it constructs ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(\mathsf{HE.sk}, (x_i^*, \mathbf{w}_i))$. It sets $\mathsf{ct} = \{\mathsf{ct}_i\}_{i\in[t]}$. Then, it defines

$$\begin{array}{lcll} \mathbf{A}_b & = & \mathbf{A}'_b - b\cdot\mathbf{G} & \forall b \in \{0,1\} \\ \mathbf{B}_{i,j} & = & \mathbf{B}'_{i,j} - \mathsf{ct}_{i,j}\cdot\mathbf{G} & \forall i \in [t], j\in[z] \\ \mathbf{C}_k & = & \mathbf{C}'_k - \mathsf{HE.sk}_k\cdot\mathbf{G} & \forall k \in [\tau]. \end{array}$$

  Next, for each $i, i^* \in [t]$ and $\ell \in [N]$, the auxiliary setup algorithm computes

$$\widetilde{\mathbf{B}}_{i,i^*,\ell} \leftarrow \mathsf{Eval}_{\mathsf{pk}}(C_\ell, \mathbf{B}'_{i,1}, \ldots, \mathbf{B}'_{i,z}, \mathbf{A}'_{x^*_{i^*,1}}, \ldots, \mathbf{A}'_{x^*_{i^*,\rho}}, \mathbf{C}'_1, \ldots, \mathbf{C}'_\tau)$$

  and sets the trapdoor matrices as

$$\mathbf{W}_{i^*} = \hat{\mathbf{A}} + \sum_{\substack{i\in[t]\\ \ell\in[N]}} \widetilde{\mathbf{B}}_{i,i^*,\ell}\cdot\mathbf{G}^{-1}(\mathbf{D}_\ell) + \sum_{\ell\in[N]} w_{i^*,\ell}\cdot\mathbf{D}_\ell.$$

  Finally, it samples a secret key $\mathbf{s}$ from the error distribution $\mathbf{s} \leftarrow \chi^n$ and returns

$$\mathsf{pp}^* = \Big(\hat{\mathbf{A}}, \{\mathbf{A}_b\}_{b\in\{0,1\}}, \{\mathbf{B}_{i,j}\}_{i\in[t],j\in[z]}, \{\mathbf{C}_k\}_{k\in[\tau]}, \{\mathbf{W}_i\}_{i\in[t]}\Big)$$

$$\mathsf{msk}^* = (\mathbf{s}, \mathsf{HE.sk}, \mathsf{ct}, \mathsf{T}, \{\mathbf{w}_i\}_{i\in[t]}).$$

- $\mathsf{Constrain}_1^*(\mathsf{pp}^*, \mathsf{msk}^*) \to \mathsf{sk}_\mathsf{T}^*$: On input the auxiliary public parameters pp^* and an auxiliary PRF key $\mathsf{msk}^* = (\mathbf{s}, \mathsf{HE.sk}, \mathsf{ct}, \mathsf{T}, \{\mathbf{w}_i\}_{i\in[t]})$, the auxiliary constraining algorithm samples error vectors $\mathbf{e}_0 \leftarrow \chi^m$, $\mathbf{e}_{1,b} \leftarrow \chi^m$ for $b \in \{0,1\}$, $\mathbf{e}_{2,i,j} \leftarrow \chi^m$ for $i \in [t]$ and $j \in [z]$, and $\mathbf{e}_{3,k} \leftarrow \chi^m$ for $k \in [\tau]$ and computes the vectors

$$\begin{array}{lcll} \hat{\mathbf{a}}^T & = & \mathbf{s}^T\hat{\mathbf{A}} + \mathbf{e}_0^T & \\ \mathbf{a}_b^T & = & \mathbf{s}^T(\mathbf{A}_b + b\cdot\mathbf{G}) + \mathbf{e}_{1,b}^T & \forall b\in\{0,1\} \\ \mathbf{b}_{i,j}^T & = & \mathbf{s}^T(\mathbf{B}_j + \mathsf{ct}_{i,j}\cdot\mathbf{G}) + \mathbf{e}_{2,i,j}^T & \forall i\in[t], \forall j\in[z] \\ \mathbf{c}_k^T & = & \mathbf{s}^T(\mathbf{C}_k + \mathsf{HE.sk}_k\cdot\mathbf{G}) + \mathbf{e}_{3,k}^T & \forall k\in[\tau]. \end{array}$$

  It sets $\mathsf{enc} = \big(\hat{\mathbf{a}}, \{\mathbf{a}_b\}_{b\in\{0,1\}}, \{\mathbf{b}_{i,j}\}_{i\in[t],j\in[z]}, \{\mathbf{c}_k\}_{k\in[\tau]}\big)$ and outputs $\mathsf{sk}_\mathsf{T}^* = (\mathsf{enc}, \mathsf{ct})$.

- $\mathsf{Constrain}_2^*(\mathsf{pp}^*, \mathsf{msk}^*) \to \mathsf{sk}_\mathsf{T}^*$: On input the auxiliary public parameters pp^* and an auxiliary PRF key $\mathsf{msk}^* = (\mathbf{s}, \mathsf{HE.sk}, \mathsf{ct}, \mathsf{T}, \{\mathbf{w}_i\}_{i\in[t]})$, the auxiliary constraining algorithm instantiates the encoding $\mathsf{enc} = \big(\hat{\mathbf{a}}, \{\mathbf{a}_b\}_{b\in\{0,1\}}\{\mathbf{b}_{i,j}\}_{i\in[t],j\in[z]}, \{\mathbf{c}_k\}_{k\in[\tau]}\big)$ with uniformly random vectors in $\mathbb{Z}_q^m$ and outputs $\mathsf{sk}_\mathsf{T}^* = (\mathsf{enc}, \mathsf{ct})$.

- $\mathsf{Eval}_1^*(\mathsf{pp}^*, \mathsf{msk}^*, \mathsf{sk}_\mathsf{T}^*, x) \to \tilde{\mathbf{y}}$: On input the auxiliary public parameters pp^*, an auxiliary PRF key $\mathsf{msk}^* = (\mathbf{s}, \mathsf{HE.sk}, \mathsf{ct}, \mathsf{T}, \{\mathbf{w}_i\}_{i\in[t]})$, the auxiliary constrained key $\mathsf{sk}_\mathsf{T}^* = (\mathsf{enc}, \mathsf{ct})$ for some set $\mathsf{T} = \{x_i^*\}_{i\in[t]}$, and an evaluation point $x \in \{0,1\}^\rho$, the auxiliary evaluation algorithm first parses $\mathsf{enc} = \big(\hat{\mathbf{a}}, \{\mathbf{a}_b\}_{b\in\{0,1\}}, \{\mathbf{b}_{i,j}\}_{i\in[t],j\in[z]}, \{\mathbf{c}_k\}_{k\in[\tau]}\big)$ and computes the vector

$$\widetilde{\mathbf{b}}_{i,\ell} \leftarrow \mathsf{Eval}_{\mathsf{ct}}((\mathsf{ct}_i, x), C_\ell, \mathbf{b}_{i,1}, \ldots, \mathbf{b}_{i,z}, \mathbf{a}_{x_1}, \ldots, \mathbf{a}_{x_\rho}, \mathbf{c}_1, \ldots, \mathbf{c}_\tau)$$

for $i \in [t]$ and $\ell \in [N]$. It then checks if $x = x^*_{i^*}$ for some $i^* \in [t]$. If this is not the case, then it returns the value

$$\tilde{\mathbf{y}} = \left\lfloor \hat{\mathbf{a}} + \sum_{\substack{i \in [t] \\ \ell \in [N]}} \widetilde{\mathbf{b}}^T_{i,\ell} \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) \right\rceil_p .$$

Otherwise, if there exists an $i^* \in [t]$ such that $x = x^*_{i^*}$, it samples an error vector $\mathbf{e} \leftarrow \chi^m$ and returns

$$\tilde{\mathbf{y}} = \left\lfloor \hat{\mathbf{a}} + \sum_{\substack{i \in [t] \\ \ell \in [N]}} \widetilde{\mathbf{b}}^T_{i,\ell} \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) - \mathbf{s}^T \sum_{\ell \in [N]} w_{i^*,\ell} \mathbf{D}_\ell - \mathbf{e}^T \right\rceil_p .$$

- $\mathsf{Eval}^*_2(\mathsf{pp}^*, \mathsf{msk}^*, \mathsf{sk}^*_{\mathsf{T}}, x) \to \tilde{\mathbf{y}}$: On input the public parameters pp^*, the auxiliary PRF key $\mathsf{msk}^* = (\mathbf{s}, \mathsf{HE.sk}, \mathsf{ct}, \mathsf{T}, \{\mathbf{w}_i\}_{i \in [t]})$, the auxiliary constrained key $\mathsf{sk}^*_{\mathsf{T}}$ for some set $\mathsf{T} = \{x^*_i\}_{i \in [t]}$, and an evaluation point $x \in \{0,1\}^\rho$, the auxiliary evaluation algorithm first checks if $x = x^*_{i^*}$ for some $i^* \in [t]$. If not, it returns $\perp$. Otherwise, it samples a uniformly random vector $\mathbf{d} \xleftarrow{\text{R}} \mathbb{Z}_q^m$ and returns $\tilde{\mathbf{y}} = \lfloor \mathbf{d} \rceil_p$.

A.2 Correctness Analysis

In this section, we give the formal proof of Theorem 5.1, which states that the translucent t-puncturable PRF in Section 5.1 satisfies both (selective) evaluation correctness and (selective) verification correctness (Definition 4.4). We show the two properties separately in Appendices A.2.1 and A.2.2.

A.2.1 Proof of Selective Evaluation Correctness

In the selective evaluation correctness game, the adversary $\mathcal{A}$ begins by committing to a set $\mathsf{T} = \{x^*_i\}_{i \in [t]}$ of t distinct points in the domain of Π_{TPRF}. Next, let $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{TPRF.Setup}(1^\lambda)$, $\mathsf{msk} \leftarrow \mathsf{TPRF.SampleKey}(\mathsf{pp})$, and $\mathsf{sk}_{\mathsf{T}} \leftarrow \mathsf{TPRF.Constrain}(\mathsf{msk}, \mathsf{T})$. Adversary $\mathcal{A}$ is then given the public parameters pp and the constrained key sk_{T}, and outputs an element in the domain $x \in \{0,1\}^\rho$. Without loss of generality, we can assume that $x \notin \mathsf{T}$, or equivalently $x \neq x^*_i$ for all $i \in [t]$, since otherwise, the adversary's advantage is 0. We now bound the probability that the value $\mathbf{y}_x = \mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk}_{\mathsf{T}}, x)$ obtained using the constrained evaluation algorithm at x differs from the real PRF value $\mathbf{y}'_x = \mathsf{TPRF.Eval}(\mathsf{pp}, \mathsf{msk}, x)$ at x. To argue this, we first recall that the key punctured at $\mathsf{T} = \{x^*_i\}_{i \in [t]}$ contains the following encodings:

$$\begin{array}{rcll} \hat{\mathbf{a}}^T & = & \mathbf{s}^T \hat{\mathbf{A}} + \mathbf{e}_0^T & \\ \mathbf{a}_b^T & = & \mathbf{s}^T(\mathbf{A}_b + b \cdot \mathbf{G}) + \mathbf{e}^T_{1,b} & \forall b \in \{0,1\} \\ \mathbf{b}^T_{i,j} & = & \mathbf{s}^T(\mathbf{B}_j + \mathsf{ct}_{i,j} \cdot \mathbf{G}) + \mathbf{e}^T_{2,i,j} & \forall i \in [t], \forall j \in [z] \\ \mathbf{c}_k^T & = & \mathbf{s}^T(\mathbf{C}_k + \mathsf{HE.sk}_k \cdot \mathbf{G}) + \mathbf{e}^T_{3,k} & \forall k \in [\tau]. \end{array}$$

as well as FHE ciphertexts $\{\mathsf{ct}_i\}_{i \in [t]}$ where ct_i is an FHE encryption of $(x^*_i, \mathbf{w}_i)$. The constrained evaluation algorithm then computes the vectors $\widetilde{\mathbf{b}}_{i,\ell}$

$$\widetilde{\mathbf{b}}_{i,\ell} \leftarrow \mathsf{Eval}_{\mathsf{ct}}((\mathsf{ct}_i, x), C_\ell, \mathbf{b}_{i,1}, \ldots, \mathbf{b}_{i,z}, \mathbf{a}_{x_1}, \ldots, \mathbf{a}_{x_\rho}, \mathbf{c}_1, \ldots, \mathbf{c}_z)$$

for $i \in [t]$ and $\ell \in [N]$ and returns

$$\mathbf{y}_x = \left\lfloor \hat{\mathbf{a}}^T + \sum_{\substack{i \in [t] \\ \ell \in [N]}} \widetilde{\mathbf{b}}^T_{i,\ell} \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) \right\rceil_p . \tag{A.1}$$

Next, by Theorem 3.10, we have that for all $i \in [t]$ and $\ell \in [N]$,

$$
\begin{aligned}
\widetilde{\mathbf{b}}_{i,\ell}^T &= \mathbf{s}^T(\widetilde{\mathbf{B}}_{i,\ell} + \langle \mathsf{HE.Eval}(\mathsf{eq}_\ell(x,\cdot), \mathsf{ct}_i), \mathsf{HE.sk}\rangle \cdot \mathbf{G}) + \mathbf{e}_{i,\ell}^T \\
&= \mathbf{s}^T(\widetilde{\mathbf{B}}_{i,\ell} + (\mathsf{eq}_\ell(x,(x_i^*, \mathbf{w}_i)) + \epsilon_{i,\ell}) \cdot \mathbf{G}) + \mathbf{e}_{i,\ell}^T \\
&= \mathbf{s}^T(\widetilde{\mathbf{B}}_{i,\ell} + \epsilon_{i,\ell} \cdot \mathbf{G}) + \mathbf{e}_{i,\ell}^T,
\end{aligned}
$$

where

$$
\widetilde{\mathbf{B}}_{i,\ell} = \mathsf{Eval}_{\mathsf{pk}}\left(C_\ell, \mathbf{B}_{i,1}, \ldots, \mathbf{B}_{i,z}, \mathbf{A}_{x_1}, \ldots, \mathbf{A}_{x_\rho}, \mathbf{C}_1, \ldots, \mathbf{C}_\tau\right),
$$

and we used Theorem 3.9 in the second equality and the fact that $\mathsf{eq}_\ell(x,(x_i^*, \mathbf{w}_i)) = 0$ when $x_i^* \neq x$ in the third equality. Moreover, by Theorem 3.9, $|\epsilon_{i,\ell}| \le B \cdot m^{O(d_{\mathsf{eq}})}$ and by Theorem 3.10, $\|\mathbf{e}_{i,\ell}\| \le B \cdot m^{O(d)}$. Then,

$$
\begin{aligned}
\hat{\mathbf{a}}^T + \sum_{\substack{i\in[t]\\ \ell\in[N]}} \widetilde{\mathbf{b}}_{i,\ell}^T \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) &= \left(\mathbf{s}^T\hat{\mathbf{A}} + \mathbf{e}_0^T\right) + \sum_{\substack{i\in[t]\\ \ell\in[N]}} \left(\mathbf{s}^T(\widetilde{\mathbf{B}}_{i,\ell} + \epsilon_{i,\ell}\cdot\mathbf{G}) + \mathbf{e}_{i,\ell}^T\right)\mathbf{G}^{-1}(\mathbf{D}_\ell) \\
&= \underbrace{\mathbf{s}^T\left(\hat{\mathbf{A}} + \sum_{\substack{i\in[t]\\ \ell\in[N]}} \widetilde{\mathbf{B}}_{i,\ell}\cdot\mathbf{G}^{-1}(\mathbf{D}_\ell)\right) + \tilde{\mathbf{e}}^T}_{\boldsymbol{\xi}_x^T} = \boldsymbol{\xi}_x^T
\end{aligned}
\tag{A.2}
$$

where

$$
\|\tilde{\mathbf{e}}\| = \left\|\mathbf{e}_0^T + \sum_{\substack{i\in[t]\\ \ell\in[N]}} (\mathbf{e}_{i,\ell}^T \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) + \epsilon_{i,\ell}\cdot\mathbf{s}^T\cdot\mathbf{D}_\ell)\right\| \le B \cdot m^{O(d)},
$$

using the fact that $B, m, N = \mathsf{poly}(\lambda)$. Thus, combining Eq. (A.1) and (A.2), we have that $\mathbf{y}_x = \lfloor \boldsymbol{\xi}_x^T \rceil_p$. Next, by definition, the output $\mathbf{y}_x' = \mathsf{TPRF.Eval}(\mathsf{pp}, \mathsf{msk}, x)$ of the evaluation algorithm is given by

$$
\mathbf{y}_x' = \left\lfloor \mathbf{s}^T\left(\hat{\mathbf{A}} + \sum_{\substack{i\in[t]\\ \ell\in[N]}} \widetilde{\mathbf{B}}_{i,\ell}\cdot\mathbf{G}^{-1}(\mathbf{D}_\ell)\right)\right\rceil_p = \lfloor \boldsymbol{\xi}_x^T - \tilde{\mathbf{e}}^T \rceil_p,
\tag{A.3}
$$

where $\boldsymbol{\xi}_x$ is the quantity defined in Eq. (A.2). Thus $\mathbf{y}_x = \mathbf{y}_x'$ as long as $\boldsymbol{\xi}_x$ does not contain any "borderline" components that can be rounded in the "wrong direction" due to the additional error $\widetilde{\mathbf{e}}$. Let $\mathsf{Borderline}_x$ be the event that there exists an index $\eta \in [m]$ such that $\boldsymbol{\xi}_x^T \mathbf{u}_\eta \in [-E, E] + (q/p)\cdot(\mathbb{Z} + 1/2)$, where $\mathbf{u}_\eta$ is the η^{th} basis vector, and $E = B \cdot m^{O(d)}$ is a bound on $\|\tilde{\mathbf{e}}\|$. To prove the theorem, it suffices to show that it is computationally hard for an adversary to find a point x such that $\mathsf{Borderline}_x$ occurs. To do this, we proceed via a hybrid argument. First, we define our sequence of hybrid experiments.

- **Hybrid H_0:** This is the real experiment. In particular, the adversary begins by committing to a set $\mathsf{T} = \{x_i^*\}_{i\in[t]}$ of punctured points. The challenger then computes $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{TPRF.Setup}(1^\lambda)$, $\mathsf{msk} \leftarrow \mathsf{TPRF.SampleKey}(\mathsf{pp})$, and $\mathsf{sk}_\mathsf{T} = (\mathsf{enc}, \mathsf{ct}) \leftarrow \mathsf{TPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, \mathsf{T})$. Finally, the challenger gives $(\mathsf{pp}, \mathsf{sk}_\mathsf{T})$ to the adversary.

- **Hybrid H_1:** Same as H_0, except the challenger generates the public parameters and PRF key using the auxiliary setup algorithm: $(\mathsf{pp}^*, \mathsf{msk}^*) \leftarrow \mathsf{Setup}^*(1^\lambda, \mathsf{T})$, where $\mathsf{T} = \{x_i^*\}_{i\in[t]}$ is the set of punctured points to which the adversary committed. The challenger generates the constrained key as $\mathsf{sk}_\mathsf{T}^* \leftarrow \mathsf{Constrain}_1^*(\mathsf{pp}^*, \mathsf{msk}^*)$, and gives $(\mathsf{pp}^*, \mathsf{sk}_\mathsf{T}^*)$ to the adversary.

- **Hybrid H_2:** Same as H_1, except the challenger generates the constrained key $\mathsf{sk}_\mathsf{T}^* \leftarrow \mathsf{Constrain}_2^*(\mathsf{pp}^*, \mathsf{msk}^*)$ using the second auxiliary constraining algorithm. It gives $(\mathsf{pp}^*, \mathsf{sk}_\mathsf{T}^*)$ to the adversary.

For a hybrid experiment H and an adversary $\mathcal{A}$, we write $\mathsf{H}(\mathcal{A})$ to denote the indicator random variable for whether the event $\mathsf{Borderline}_x$ occurred in H. We now show that the outputs in each consecutive pair of hybrid experiments are statistically or computationally indistinguishable. This in particular implies that

$$|\Pr[\mathsf{H}_0(\mathcal{A}) = 1] - \Pr[\mathsf{H}_2(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

To finish the proof, we then show that $\Pr[\mathsf{H}_2(\mathcal{A}) = 1] = \mathsf{negl}(\lambda)$.

Lemma A.1. *For all adversaries $\mathcal{A}$, $|\Pr[\mathsf{H}_0(\mathcal{A}) = 1] - \Pr[\mathsf{H}_1(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. We first show that the distribution of the public parameters pp in H_0 is statistically indistinguishable from the distribution of the auxiliary public parameters pp^* in H_1.

- In hybrid H_0, the matrices $\hat{\mathbf{A}}, \{\mathbf{A}_b\}_{b\in\{0,1\}}$, $\{\mathbf{B}_{i,j}\}_{i\in[t],j\in[z]}$, $\{\mathbf{C}_k\}_{k\in[\tau]}$ are uniform and independent over $\mathbb{Z}_q^{n\times m}$, and the matrices $\{\mathbf{W}_i\}_{i\in[t]}$ are independent and statistically close to uniform over $\mathbb{Z}_q^{n\times m}$ by properties of the trapdoor generation algorithm (Theorem 3.6).

- In hybrid H_1, by definition of the auxiliary setup algorithm Setup^*, the matrices $\hat{\mathbf{A}}, \{\mathbf{A}_b\}_{b\in\{0,1\}}$, $\{\mathbf{B}_{i,j}\}_{i\in[t],j\in[z]}$, $\{\mathbf{C}_k\}_{k\in[\tau]}$ are independent and uniform over $\mathbb{Z}_q^{n\times m}$. We conclude by arguing that the matrices $\mathbf{W}_i$ for all $i \in [t]$ are distributed independently and uniformly over $\mathbb{Z}_q^{n\times m}$. By definition, Setup^* first computes the matrices

$$\widetilde{\mathbf{B}}_{i,i^*,\ell} \leftarrow \mathsf{Eval}_{\mathsf{pk}}(C_\ell, \mathbf{B}_{i,1}, \ldots, \mathbf{B}_{i,z}, \mathbf{A}_{x^*_{i^*,1}}, \ldots, \mathbf{A}_{x^*_{i^*,\rho}}, \mathbf{C}_1, \ldots, \mathbf{C}_\tau)$$

  for $i, i^* \in [t], \ell \in [N]$, and defines

$$\mathbf{W}_{i^*} = \hat{\mathbf{A}} + \sum_{\substack{i\in[t]\\ \ell\in[N]}} \widetilde{\mathbf{B}}_{i,i^*,\ell} \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) + \underbrace{\sum_{\ell\in[N]} w_{i^*,\ell} \cdot \mathbf{D}_\ell}_{\widetilde{\mathbf{D}}_i},$$

  where $\mathbf{w}_{i^*} \xleftarrow{\text{R}} \mathbb{Z}_q^N$ for all $i^* \in [t]$. Thus, each matrix $\widetilde{\mathbf{D}}_i$ is a *random* linear combination of basis elements of $\mathbb{Z}_q^{n\times m}$ and distributed independently and uniformly. We conclude that the distribution of pp is statistically indistinguishable from that of pp^*.

To complete the proof, we argue that the distribution of the components in the constrained key $\mathsf{sk}_\mathsf{T} = (\mathsf{enc}, \mathsf{ct})$ in H_0 is statistically indistinguishable from sk^*_T in H_1. This follows from the fact that the matrices $\hat{\mathbf{A}}, \{\mathbf{A}_b\}_{b\in\{0,1\}}$, $\{\mathbf{B}_{i,j}\}_{i\in[t],j\in[z]}$, $\{\mathbf{C}_k\}_{k\in[\tau]}$, and $\{\mathbf{W}_i\}_{i\in[t]}$ are statistically indistinguishable in H_0 and H_1. In particular, this means that the coefficients $\mathbf{w}_i \in \mathbb{Z}_q^N$ in H_0 and H_1 are statistically indistinguishable. Since the ciphertexts $\mathsf{ct} = \{\mathsf{ct}_i\}_{i\in[t]}$ are generated in the exact same manner in H_0 and H_1, we conclude that they are statistically indistinguishable in the two experiments. Finally, since the public matrices, the FHE secret key, and the ciphertexts are either identically distributed or statistically indistinguishable between the two experiments, the encoding enc is statistically indistinguishable between the two experiments and we conclude that the distribution of $(\mathsf{pp}, \mathsf{sk}_\mathsf{T})$ in H_0 is statistically indistinguishable from the distribution of $(\mathsf{pp}^*, \mathsf{sk}^*_\mathsf{T})$ in H_1. □

Lemma A.2. *Under the $\mathsf{LWE}_{n,m',q,\chi}$ assumption (where $m' = m(3 + t \cdot z + \tau)$), for all efficient adversaries $\mathcal{A}$, $|\Pr[\mathsf{H}_1(\mathcal{A}) = 1] - \Pr[\mathsf{H}_2(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that can distinguish H_1 from H_2 with some non-negligible probability ε. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks the $\mathsf{LWE}_{n,m,q,\chi}$ assumption. Algorithm $\mathcal{B}$ works as follows:

1. First, it receives a challenge $(\hat{\mathbf{A}}, \hat{\mathbf{a}})$, $\{(\mathbf{A}'_b, \mathbf{a}'_b)\}_{b\in\{0,1\}}$, $\{(\mathbf{B}'_{i,j}, \mathbf{b}'_{i,j})\}_{i\in[t],j\in[z]}$, and $\{(\mathbf{C}'_k, \mathbf{c}'_k)\}_{k\in[\tau]}$ from the LWE challenger.

2. Algorithm $\mathcal{B}$ starts running $\mathcal{A}$. When $\mathcal{A}$ commits to its set $\mathsf{T} = \{x_i^*\}_{i \in [t]}$, algorithm $\mathcal{B}$ runs the auxiliary setup algorithm Setup^*, except it uses the matrices $\hat{\mathbf{A}}$, $\{\mathbf{A}'_b\}_{b \in \{0,1\}}$, $\{\mathbf{B}'_{i,j}\}_{i \in [t], j \in [z]}$, and $\{\mathbf{C}'_k\}_{k \in [\tau]}$ from the LWE challenge in place of the corresponding matrices in Setup^*. It generates the rest of the public parameters pp^* as described in Setup^*.

3. To simulate the constrained key sk_T^*, algorithm $\mathcal{B}$ sets $\mathsf{enc} = \big(\hat{\mathbf{a}}, \{\mathbf{a}'_b\}_{b \in \{0,1\}}, \{\mathbf{b}'_{i,j}\}_{i \in [t], j \in [z]}, \{\mathbf{c}'_k\}_{k \in [\tau]}\big)$ to be the vectors from the LWE challenge. The ciphertexts ct are constructed exactly as in H_1 and H_2 (as described in Setup^*). Finally, $\mathcal{B}$ gives the public parameters pp^* and the constrained key $\mathsf{sk}_\mathsf{T}^* = (\mathsf{enc}, \mathsf{ct})$ to $\mathcal{A}$.

4. At the end of the game, $\mathcal{A}$ outputs a vector x. Algorithm $\mathcal{B}$ computes $\boldsymbol{\xi}_x$ as defined in Eq. (A.2), and outputs 1 if $\boldsymbol{\xi}_x^T \mathbf{u}_\eta \in [-E, E] + (q/p) \cdot (\mathbb{Z} + 1/2)$ where $\mathbf{u}_\eta$ is the η^{th} basis vector, and 0 otherwise.

It is easy to see that if the challenge consists of valid LWE challenge vectors, then $\mathcal{B}$ has perfectly simulated H_1, whereas if the challenge consists of uniformly random vectors, then $\mathcal{B}$ has perfectly simulated H_2. Moreover, algorithm $\mathcal{B}$ outputs 1 if and only if the adversary's output x triggers the $\mathsf{Borderline}_x$ event. By assumption then, $\mathcal{B}$ is able to break the $\mathsf{LWE}_{n,m',q,\chi}$ assumption with the same probability ε. ☐

Lemma A.3. *Under the* $\mathsf{1D\text{-}SIS\text{-}R}_{m',p,q,\beta}$ *assumption (where* $m' = m(3 + t \cdot z + \tau)$ *and* $\beta = B \cdot m^{O(d)}$*), for all efficient adversaries,* $\mathcal{A}$*,* $\Pr[\mathsf{H}_2(\mathcal{A}) = 1] = \mathsf{negl}(\lambda)$.

Proof. We begin with a high-level overview of the proof. In H_3, the encoding enc in the punctured key is uniformly random, and thus, can be viewed as the challenge vector $\mathbf{v}$ in a 1D-SIS-R instance (Definition 3.5). Next, according to Theorem 3.10, the constrained evaluation algorithm $\mathsf{TPRF.ConstrainEval}$ is effectively computing a "short" linear combination of the vectors in enc. Thus, if an adversary is able to find a point x such that the constrained evaluation algorithm yields a boundary value, then the same point x is a solution to the 1D-SIS-R instance.

Formally, suppose there exists an adversary $\mathcal{A}$ that outputs a point $x \in \{0,1\}^\rho$ such that $\mathsf{Borderline}_x$ occurs with non-negligible probability ε. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks $\mathsf{1D\text{-}SIS\text{-}R}_{m',p,q,\beta}$. At the beginning of the game, algorithm $\mathcal{B}$ is given its 1D-SIS-R challenge vector $\mathbf{v} \in \mathbb{Z}_q^{m'}$. Then, $\mathcal{B}$ begins simulating H_3 algorithm $\mathcal{A}$. At the beginning of the game, $\mathcal{A}$ commits to a set $\mathsf{T} = \{x_i^*\}_{i \in [t]}$ of punctured points. Algorithm $\mathcal{B}$ then runs $\mathsf{Setup}^*(1^\lambda, \mathsf{T})$ to obtain the public parameters pp^*. When simulating the $\mathsf{Constrain}_2^*$ algorithm, algorithm $\mathcal{B}$ substitutes the challenge vector $\mathbf{v}$ for enc (in particular, $\mathcal{B}$ treats $\mathbf{v}$ as the concatenation of the vectors $\hat{\mathbf{a}}, \{\mathbf{a}_b\}$, $\{\mathbf{b}_{i,j}\}$, $\{\mathbf{c}_k\}$). The other components of the secret key are constructed exactly as in H_3. It then gives pp^* and $\mathsf{sk}_\mathsf{T} = (\mathsf{enc}, \mathsf{ct})$ to the adversary and receives $\mathcal{A}$'s guess x. Since $\mathbf{v}$ is uniformly distributed, algorithm $\mathcal{B}$ perfectly simulates H_3 for $\mathcal{A}$. By assumption then, with probability ε, $\mathcal{A}$ outputs a point x such that $\mathsf{Borderline}_x$ occurs. This means that there exists some $\eta \in [m]$ such that

$$\boldsymbol{\xi}_x^T \mathbf{u}_\eta = \left(\hat{\mathbf{a}}^T + \sum_{\substack{i \in [t] \\ \ell \in [N]}} \widetilde{\mathbf{b}}_{i,\ell}^T \cdot \mathbf{G}^{-1}(\mathbf{D}_i) \right) \mathbf{u}_\eta \in [-E, E] + \frac{q}{p} \cdot (\mathbb{Z} + 1/2). \tag{A.4}$$

where $E = B \cdot m^{O(d)}$ and $\mathbf{u}_\eta \in \mathbb{Z}_q^m$ is the η^{th} canonical basis vector. By Theorem 3.10, we have that for all $i \in [t]$ and $\ell \in [N]$,j

$$\widetilde{\mathbf{b}}_{i,\ell}^T = \sum_{b \in \{0,1\}} \mathbf{a}_b^T \mathbf{R}_{b,i,\ell}^{(1)} + \sum_{j \in [z]} \mathbf{b}_{i,j}^T \mathbf{R}_{i,j,\ell}^{(2)} + \sum_{k \in [\tau]} \mathbf{c}_k^T \mathbf{R}_{k,i,\ell}^{(3)}$$

for some matrices $\{\mathbf{R}_{b,\ell,i}^{(1)}\}_{b \in \{0,1\}}$, $\{\mathbf{R}_{j,\ell,i}^{(2)}\}_{j \in [z]}$, $\{\mathbf{R}_{k,\ell,i}^{(3)}\}_{k \in [\tau]}$ where $\left\|\mathbf{R}_{b,\ell,i}^{(1)}\right\|$, $\left\|\mathbf{R}_{j,\ell,i}^{(2)}\right\|$, $\left\|\mathbf{R}_{k,\ell,i}^{(3)}\right\| \leq m^{O(d)}$. This means that we can write $\boldsymbol{\xi}_x$ as

$$\boldsymbol{\xi}_x^T = \hat{\mathbf{a}}^T + \sum_{\substack{i \in [t] \\ \ell \in [N]}} \widetilde{\mathbf{b}}_{i,\ell}^T \cdot \mathbf{G}^{-1}(\mathbf{D}_i) = \mathbf{v}^T \mathbf{R}.$$

for some $\mathbf{R} \in \mathbb{Z}_q^{m' \times m}$ where $\|\mathbf{R}\| \leq m^{O(d)}$. Substituting into Eq. (A.4), we have that

$$\boldsymbol{\xi}_x^T \mathbf{u}_\eta = \mathbf{v}^T \mathbf{R} \cdot \mathbf{u}_\eta = \langle \mathbf{v}, \mathbf{R} \cdot \mathbf{u}_\eta \rangle \in [-E, E] + \frac{q}{p} \cdot (\mathbb{Z} + 1/2).$$

Moreover, $\|\mathbf{R} \cdot \mathbf{u}_\eta\| \leq \|\mathbf{R}\| \leq m^{O(d)} = \beta$, and so the vector $\mathbf{R} \cdot \mathbf{u}_\eta$ is a valid solution to the 1D-SIS-R challenge. We conclude that $\mathcal{B}$ succeeds in breaking the $\mathsf{1D\text{-}SIS\text{-}R}_{m',p,q,\beta}$ with the same (non-negligible) advantage ε. The claim follows. ☐

Combining Lemmas A.1 through A.3, we conclude that under the LWE and 1D-SIS-R assumptions (for the parameters given in Theorem 5.1), no efficient adversary is able to find an input $x \notin \mathsf{T}$ such that the $\mathsf{Borderline}_x$ event occurs. Equivalently, no efficient adversary can find an $x \notin \mathsf{T}$ where $\mathsf{TPRF.Eval}(\mathsf{pp}, \mathsf{msk}, x) \neq \mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk_T}, x)$. Thus Π_{TPRF} satisfies (selective) evaluation correctness. ☐

A.2.2 Proof of Selective Verification Correctness

In the selective verification correctness game, the adversary $\mathcal{A}$ first commits to a set $\mathsf{T} = \{x_i^*\}_{i \in [t]}$ of punctured points. It is then provided with the public parameters pp and the constrained key $\mathsf{sk_T}$. Finally, $\mathcal{A}$ wins the game if at least one of the following conditions is satisfied:

- **Case 1**: it outputs a point $x \in \mathsf{T}$ such that the testing algorithm rejects:

$$\mathsf{TPRF.Test}(\mathsf{pp}, \mathsf{tk}, \mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk_T}, x_i^*)) = 0.$$

- **Case 2**: it outputs a point $x \notin \mathsf{T}$ such that the testing algorithm accepts:

$$\mathsf{TPRF.Test}(\mathsf{pp}, \mathsf{tk}, \mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk_T}, x)) = 1.$$

We define Bad_1 to be the event that an adversary outputs a point x that satisfies the first case. We define the event Bad_2 analogously. We now show that for all efficient adversaries $\mathcal{A}$, the probability of either Bad_1 or Bad_2 occurring is negligible.

Lemma A.4. *Under the parameter settings given in Theorem 5.1, for all adversaries $\mathcal{A}$,* $\Pr[\mathsf{Bad}_1] = 0$.

Proof. Let $x \in \mathcal{X}$ be the output of $\mathcal{A}$, and suppose $x \in \mathsf{T}$. Then, there exists an index $i^* \in [t]$ such that $x = x_{i^*}^*$. On input the public parameters pp, the constrained key $\mathsf{sk_T}$, and the point $x_{i^*}^*$, the constrained evaluation algorithm first computes

$$\widetilde{\mathbf{b}}_{i,\ell} \leftarrow \mathsf{Eval_{ct}}((\mathsf{ct}, x_{i^*}^*), C_\ell, \mathbf{b}_{i,1}, \ldots, \mathbf{b}_{i,z}, \mathbf{a}_{x_{i^*,1}^*}, \ldots, \mathbf{a}_{x_{i^*,\rho}^*}, \mathbf{c}_1, \ldots, \mathbf{c}_t)$$

for $i \in [t]$ and $\ell \in [N]$ and the returns the value

$$\mathbf{y}_{x_{i^*}^*} = \left\lfloor \hat{\mathbf{a}}^T + \sum_{\substack{i \in [t] \\ \ell \in [N]}} \widetilde{\mathbf{b}}_{i,\ell}^T \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) \right\rceil_p . \tag{A.5}$$

By Theorems 3.9 and 3.10, the vectors $\widetilde{\mathbf{b}}_{i,\ell}$ can be written as

$$\widetilde{\mathbf{b}}_{i,\ell} = \mathbf{s}^T \left(\widetilde{\mathbf{B}}_{i,\ell} + (\mathsf{eq}(x_{i^*}^*, x_i^*) \cdot w_{i,\ell} + \epsilon_{i,\ell}) \cdot \mathbf{G} \right) + \mathbf{e}_{i,\ell}^T,$$

where

$$\widetilde{\mathbf{B}}_{i,\ell} = \mathsf{Eval_{pk}}(C_\ell, \mathbf{B}_{i,1}, \ldots, \mathbf{B}_{i,z}, \mathbf{A}_{x_{i^*,1}^*}, \ldots, \mathbf{A}_{x_{i^*,\rho}^*}, \mathbf{C}_1, \ldots, \mathbf{C}_\tau),$$

and $|\epsilon_{i,\ell}| \le B \cdot m^{O(d_{\mathsf{eq}})}$ and $\|\mathbf{e}_{i,\ell}\| \le B \cdot m^{O(d)}$. Substituting into Eq. (A.5), we have

$$\begin{aligned}
\mathbf{y}_{x^*_{i^*}} &= \left\lfloor \left(\mathbf{s}^T\hat{\mathbf{A}} + \mathbf{e}_0^T\right) + \sum_{\substack{i\in[t]\\ \ell\in[N]}} \left(\mathbf{s}^T\left(\widetilde{\mathbf{B}}_{i,\ell}\cdot\mathbf{G}^{-1}(\mathbf{D}_\ell) + \mathsf{eq}(x^*_{i^*}, x^*_i)\cdot w_{i,\ell}\cdot\mathbf{D}_\ell + \epsilon_{i,\ell}\cdot\mathbf{D}_\ell\right) + \mathbf{e}_{i,\ell}^T\cdot\mathbf{G}^{-1}(\mathbf{D}_\ell)\right)\right\rceil_p \\
&= \left\lfloor \mathbf{s}^T\left(\hat{\mathbf{A}} + \sum_{\substack{i\in[t]\\ \ell\in[N]}}\left(\widetilde{\mathbf{B}}_{i,\ell}\cdot\mathbf{G}^{-1}(\mathbf{D}_\ell) + \mathsf{eq}(x^*_{i^*}, x^*_i)\cdot w_{i,\ell}\cdot\mathbf{D}_\ell\right)\right) + \mathbf{e}^T\right\rceil_p \\
&= \left\lfloor \mathbf{s}^T\left(\hat{\mathbf{A}} + \sum_{\substack{i\in[t]\\ \ell\in[N]}}\left(\widetilde{\mathbf{B}}_{i,\ell}\cdot\mathbf{G}^{-1}(\mathbf{D}_\ell)\right) + \sum_{\ell\in[N]} w_{i^*,\ell}\cdot\mathbf{D}_\ell\right) + \mathbf{e}^T\right\rceil_p
\end{aligned}$$

where

$$\mathbf{e}^T = \mathbf{e}_0^T + \sum_{\substack{i\in[t]\\ \ell\in[N]}}\left(\epsilon_{i,\ell}\cdot\mathbf{s}^T\cdot\mathbf{D}_\ell + \mathbf{e}_{i,\ell}^T\cdot\mathbf{G}^{-1}(\mathbf{D}_\ell)\right),$$

and $\|\mathbf{e}\| \le B\cdot m^{O(d)}$. Now, by construction of the $\mathsf{TPRF.Constrain}$ algorithm, the vector $\mathbf{w}_{i^*} \in \mathbb{Z}_q^N$ is chosen such that

$$\mathbf{W}_{i^*} = \hat{\mathbf{A}} + \sum_{\substack{i\in[t]\\ \ell\in[N]}} \widetilde{\mathbf{B}}_{i,\ell}\cdot\mathbf{G}^{-1}(\mathbf{D}_\ell) + \sum_{\ell\in[N]} w_{i^*,\ell}\cdot\mathbf{D}_\ell$$

and so we have

$$\mathbf{y}_{x^*_{i^*}} = \left\lfloor \mathbf{s}^T\mathbf{W}_{i^*} + \mathbf{e}^T\right\rceil_p.$$

Next, the testing algorithm $\mathsf{TPRF.Test}$ computes the inner product

$$\begin{aligned}
\left\langle \mathbf{y}_{x^*_{i^*}}, \mathbf{z}_{i^*}\right\rangle &= \left\lfloor \mathbf{s}^T\mathbf{W}_{i^*} + \mathbf{e}^T\right\rceil_p\cdot\mathbf{z}_{i^*} \\
&= \left\lfloor \mathbf{s}^T\mathbf{W}_{i^*}\mathbf{z}_{i^*} + \mathbf{e}^T\mathbf{z}_{i^*}\right\rceil_p + \tilde{e} \\
&= \left\lfloor \mathbf{e}^T\mathbf{z}_{i^*}\right\rceil_p + \tilde{e}
\end{aligned}$$

where $|\tilde{e}| \le B\cdot(m+1) = B_{\mathsf{test}}$ is the rounding error. Here, we used the fact that $\mathbf{z}_{i^*}$ is a (short) trapdoor vector for $\mathbf{W}_{i^*}$ (Theorem 3.6), as well as the fact that the rounding operation $\lfloor\cdot\rceil_p$ is *almost* additively homomorphic in that for any $x, y \in \mathbb{Z}_q$, we have that $\lfloor x+y\rceil_p = \lfloor x\rceil_p + \lfloor y\rceil_p + b$ for $b \in \{0,1\}$. Since $\|\mathbf{e}\| \le B\cdot m^{O(d)}$ and $\mathbf{z}_{i^*}$ is B-bounded, we have that $\left|\mathbf{e}^T\mathbf{z}_{i^*}\right| < \frac{q}{2p}$, in which case $\left\lfloor\mathbf{e}^T\mathbf{z}_{i^*}\right\rceil_p = 0$. Thus, $\langle\mathbf{y}_x, \mathbf{z}_{i^*}\rangle = \tilde{e} \in [-B_{\mathsf{test}}, B_{\mathsf{test}}]$. In this case, $\mathsf{TPRF.Test}$ outputs 1 with probability 1, and the claim follows. ☐

Lemma A.5. *Under the parameter settings given in Theorem 5.1, and the* $\mathsf{LWE}_{n,m',q,\chi}$ *and* $\mathsf{1D\text{-}SIS\text{-}R}_{m',p,q,\beta}$ *assumptions (where $m' = m(3+t\cdot z+\tau)$ and $\beta = B\cdot m^{O(d)}$), for all efficient adversaries $\mathcal{A}$,* $\Pr[\mathsf{Bad}_2] = \mathsf{negl}(\lambda)$.

Proof. In the correctness experiment, the challenger samples $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{TPRF.Setup}(1^\lambda)$ and $\mathsf{msk} \leftarrow \mathsf{TPRF.SampleKey}(\mathsf{pp})$. We first show that over the random choices of these algorithms

$$\Pr[\exists x \in \{0,1\}^\rho : \mathsf{TPRF.Test}(\mathsf{pp}, \mathsf{tk}, \mathsf{TPRF.Eval}(\mathsf{pp}, \mathsf{msk}, x)) = 1] = \mathsf{negl}(\lambda). \tag{A.6}$$

As we subsequently show, the claim then follows by invoking evaluation correctness. To show Eq. (A.6), we union bound over all $x \in \{0,1\}^\rho$. First, take any $x \in \{0,1\}^\rho$ and let $\mathbf{y}_x = \mathsf{TPRF.Eval}(\mathsf{pp}, \mathsf{msk}, x)$. Consider the probability that $\mathsf{TPRF.Test}(\mathsf{pp}, \mathsf{tk}, \mathbf{y}_x) = 1$. By definition,

$$\mathbf{y}_x = \lfloor\mathbf{s}^T(\hat{\mathbf{A}} + \mathbf{B}')\rceil_p \text{ where } \mathbf{B}' = \sum_{\substack{i\in[t]\\ \ell\in[N]}} \widetilde{\mathbf{B}}_{i,\ell}\cdot\mathbf{G}^{-1}(\mathbf{D}_\ell).$$

Now, the matrix $\hat{\mathbf{A}}$ is sampled uniformly at random over $\mathbb{Z}_q^{n\times m}$. Since $\mathbf{s}$ is non-zero with overwhelming probability, there is at least a single entry $i \in [n]$ such that $s_i \neq 0$. Moreover, since $\mathbf{s}$ is sampled from a B-bounded distribution, $|s_i| \leq B$. In particular, this means that s_i is invertible over $\mathbb{Z}_q$.[16] Since $\hat{\mathbf{A}}$ is sampled uniformly and independently of $\mathbf{B}'$, $\hat{\mathbf{A}} + \mathbf{B}'$ is also distributed uniformly at random. Since s_i is invertible over $\mathbb{Z}_q$ (and independent of $\hat{\mathbf{A}} + \mathbf{B}'$), this implies that the product $\mathbf{s}^T(\hat{\mathbf{A}} + \mathbf{B}')$ is a uniformly random vector in $\mathbb{Z}_q^m$. Finally, since q is a multiple of p, we conclude that $\mathbf{y}_x = \lfloor \mathbf{s}^T\hat{\mathbf{A}} + \mathbf{B}' \rceil_p$ is uniform over $\mathbb{Z}_p^m$.

Consider now the output $\mathsf{TPRF.Test}(\mathsf{pp}, \mathsf{tk}, \mathbf{y}_x)$. Let $\mathsf{tk} = \{\mathbf{z}_i\}_{i\in[t]}$. Since $\mathbf{y}_x$ is distributed uniformly over $\mathbb{Z}_p^m$ and independent of $\mathbf{z}_i$ for all $i \in [t]$, then for any $i \in [t]$, we have that

$$\Pr[\langle \mathbf{y}_x, \mathbf{z}_i\rangle \in [-B_{\mathsf{test}}, B_{\mathsf{test}}]] = 2B_{\mathsf{test}}/p.$$

Union bounding over all $i \in [t]$, we have that

$$\Pr[\mathsf{TPRF.Test}(\mathsf{pp}, \mathsf{tk}, \mathbf{y}_x) = 1] = \Pr[\exists i \in [t] \colon \langle \mathbf{y}_x, \mathbf{z}_i\rangle \in [-B_{\mathsf{test}}, B_{\mathsf{test}}]] \leq \frac{2 \cdot B_{\mathsf{test}} \cdot t}{p}.$$

Finally, to show Eq. (A.6), we union bound over all $x \in \{0,1\}^\rho$ to argue that over the randomness used to sample the public parameters (in particular, the matrix $\hat{\mathbf{A}}$),

$$\Pr[\exists x \in \{0,1\}^\rho \colon \mathsf{TPRF.Test}(\mathsf{pp}, \mathsf{tk}, \mathsf{TPRF.Eval}(\mathsf{pp}, \mathsf{msk}, x)) = 1] \leq \frac{2^{\rho+1} \cdot B_{\mathsf{test}} \cdot t}{p} = \mathsf{negl}(\lambda),$$

since $p = 2^{(\rho^{1+\varepsilon})}$, and $B_{\mathsf{test}}, t = \mathsf{poly}(\lambda)$. Thus, we conclude that if the adversary outputs a point $x \notin \mathsf{T}$ where $\mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk}_{\mathsf{T}}, x) = \mathsf{TPRF.Eval}(\mathsf{pp}, \mathsf{msk}, x)$, then with overwhelming probability (over the randomness used to sample the public parameters),

$$\Pr[\mathsf{TPRF.Test}(\mathsf{pp}, \mathsf{tk}, \mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk}_{\mathsf{T}}, x)) = 1] = \mathsf{negl}(\lambda).$$

However, by evaluation correctness (shown in Appendix A.2.1), with overwhelming probability, no efficient adversary in the correctness game can find a point $x \notin \mathsf{T}$ where $\mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk}_{\mathsf{T}}, x) \neq \mathsf{TPRF.Eval}(\mathsf{pp}, \mathsf{msk}, x)$. The claim follows. ☐

Combining Lemmas A.4 and A.5, we have that $\mathcal{A}$ wins the game with negligible probability. We conclude that Π_{TPRF} satisfies selective verification correctness. ☐

A.3 Security Analysis

In this section, we give the formal proofs of the security theorems from Section 5.1 (Theorems 5.2, 5.4, and 5.5). Note that Corollary 5.3 follows immediately from Theorems 4.11 and 5.2.

A.3.1 Proof of Theorem 5.2

Let $\mathcal{A}$ be an adversary and $\mathcal{S}^{(t)}$ be the set system corresponding to the family of t-puncturable constraints (Definition 4.7). We begin by defining a sequence of hybrid experiments:

- **Hybrid** H_0: This is the real experiment $\mathsf{CExpt}^{(0)}_{\Pi_{\mathsf{TPRF}},\mathcal{A},\mathcal{S}^{(t)}}$ (Definition 4.9). Specifically, the adversary $\mathcal{A}$ begins by committing to a set $\mathsf{T} = \{x_i^*\}_{i\in[t]}$ of t distinct points in the domain of Π_{TPRF}. The challenger then samples $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{TPRF.Setup}(1^\lambda)$, $\mathsf{msk} \leftarrow \mathsf{TPRF.SampleKey}(\mathsf{pp})$, and $\mathsf{sk}_{\mathsf{T}} \leftarrow \mathsf{TPRF.Constrain}(\mathsf{msk}, \mathsf{T})$. Then, the adversary is given pp, sk_{T}, access to an honest evaluation oracle $\mathsf{TPRF.Eval}(\mathsf{msk}, \cdot)$ for points $x \notin \mathsf{T}$, and access to a challenge evaluation oracle for points $x \in \mathsf{T}$. In $\mathsf{CExpt}^{(0)}_{\Pi_{\mathsf{TPRF}},\mathcal{A},\mathcal{S}^{(t)}}$, the challenge evaluation oracle outputs the PRF value $\mathsf{TPRF.Eval}(\mathsf{msk}, \cdot)$.

[16] Recall that q is a product of primes p_j such that $p_j > B$.

- **Hybrid** H_1: Same as H_0, except that the challenger generates the public parameters pp^* and the PRF key msk^* using the auxiliary setup algorithm: $(\mathsf{pp}^*, \mathsf{msk}^*) \leftarrow \mathsf{Setup}^*(1^\lambda, \mathsf{T})$, where $\mathsf{T} = \{x_i^*\}_{i \in [t]}$ is the set of points the adversary commits to at the beginning of the experiment. In addition, the challenger generates the constrained key as $\mathsf{sk}_\mathsf{T}^* \leftarrow \mathsf{Constrain}_1^*(\mathsf{pp}^*, \mathsf{msk}^*)$, and gives $(\mathsf{pp}^*, \mathsf{sk}_\mathsf{T}^*)$ to the adversary. Both the evaluation and challenge queries are handled as in H_0: on a query $x \in \mathcal{X}$, the challenger replies with $\mathsf{TPRF.Eval}(\mathsf{pp}^*, \mathsf{msk}^*, x)$.

- **Hybrid** H_2: Same as H_1, except that the challenger answers the evaluation and challenge queries using the auxiliary evaluation algorithm Eval_1^*. Specifically, on an evaluation or a challenge query $x \in \mathcal{X}$, the challenger replies with $\mathsf{Eval}_1^*(\mathsf{msk}^*, \mathsf{sk}_\mathsf{T}^*, x)$.

- **Hybrid** H_3: Same as H_2, except that the challenger generates the constrained key using the auxiliary constraining algorithm $\mathsf{Constrain}_2^*$: $\mathsf{sk}_\mathsf{T} \leftarrow \mathsf{Constrain}_2^*(\mathsf{msk}^*)$. Moreover, the challenger answers the challenge queries using the auxiliary evaluation algorithm Eval_2^*. In particular, on a challenge query $x \in \mathsf{T}$, the challenger replies with $\mathsf{Eval}_2^*(\mathsf{msk}^*, \mathsf{sk}_\mathsf{T}^*, x)$. The evaluation queries are handled as in H_2 (using Eval_1^*).

- **Hybrid** H_4: Same as H_3, except the challenger generates the constrained key using the auxiliary constraining algorithm $\mathsf{Constrain}_1^*$: $\mathsf{sk}_\mathsf{T} \leftarrow \mathsf{Constrain}_1^*(\mathsf{msk}^*)$. Both the evaluation and the challenge oracle queries are handled as in H_3.

- **Hybrid** H_5: Same as H_4, except the challenger answers the evaluation queries using the real evaluation algorithm $\mathsf{TPRF.Eval}(\mathsf{pp}, \mathsf{msk}, \cdot)$. The challenge queries are handled as in H_4 (using Eval_2^*).

- **Hybrid** H_6: Same as H_5, except the challenger generates the public parameters pp and the constrained key sk_T honestly using $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{TPRF.Setup}(1^\lambda)$, $\mathsf{msk} \leftarrow \mathsf{TPRF.SampleKey}(\mathsf{pp})$, and $\mathsf{sk}_\mathsf{T} = (\mathsf{enc}, \mathsf{ct}) \leftarrow \mathsf{TPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, \mathsf{T})$. This is the experiment $\mathsf{CExpt}^{(1)}_{\Pi_{\mathsf{TPRF}}, \mathcal{A}, \mathcal{S}^{(t)}}$ (Definition 4.9).

For a hybrid experiment H and an adversary $\mathcal{A}$, we write $\mathsf{H}(\mathcal{A})$ to denote the random variable for the output of $\mathcal{A}$ in hybrid H. We now show that the distribution of the adversary's outputs in each consecutive pair of hybrid experiments is either statistically or computationally indistinguishable.

Lemma A.6. *For all adversaries $\mathcal{A}$, $|\Pr[\mathsf{H}_0(\mathcal{A}) = 1] - \Pr[\mathsf{H}_1(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. The only difference between H_0 and H_1 is that the public parameters and the constrained key are generated according to the auxiliary algorithms Setup^* and $\mathsf{Constrain}_1^*$ in H_1, respectively, rather than the real algorithms. By the same argument as in the proof of Lemma A.1 (for evaluation correctness), we have that the distribution of $(\mathsf{pp}, \mathsf{sk}_\mathsf{T})$ in H_0 is statistically indistinguishable from the distribution of $(\mathsf{pp}^*, \mathsf{sk}_\mathsf{T}^*)$ in H_1. Finally, since the evaluation oracle queries are handled identically in the two experiments, we conclude that the adversary's view in H_0 and H_1 is statistically indistinguishable. The lemma follows. ☐

Before showing that hybrid H_1 and H_2 are computationally indistinguishable (Lemma A.8), we first show that hybrids H_2 and H_3 are computationally indistinguishable (Lemma A.7). This will greatly simplify the argument needed to show indistinguishability of hybrids H_1 and H_2 in Lemma A.8.

Lemma A.7. *Under the $\mathsf{LWE}_{n,m',q,\chi}$ assumption (where $m' = m(3+t(z+1)+\tau)$), for all efficient adversaries $\mathcal{A}$, $|\Pr[\mathsf{H}_2(\mathcal{A}) = 1] - \Pr[\mathsf{H}_3(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. Our argument is very similar to the proof of Lemma A.2, with the exception that we additionally have to reason about the challenge oracle queries in this case. In particular, we show that if there exists an adversary $\mathcal{A}$ that can distinguish H_2 from H_3 with some non-negligible probability, then we can use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks the $\mathsf{LWE}_{n,m',q,\chi}$ assumption with the same probability. Algorithm $\mathcal{B}$ behaves as follows:

1. First, $\mathcal{B}$ receives a challenge $(\hat{\mathbf{A}}, \hat{\mathbf{a}})$, $\{(\mathbf{A}_b', \mathbf{a}_b')\}_{b \in \{0,1\}}$, $\{(\mathbf{B}_{i,j}', \mathbf{b}_{i,j}')\}_{i \in [t], j \in [z]}$, $\{(\mathbf{C}_k', \mathbf{c}_k')\}_{k \in [\tau]}$, and $\{(\mathbf{H}_i', \mathbf{h}_i')\}_{i \in [t]}$ from the LWE challenger.

2. Algorithm $\mathcal{B}$ starts running $\mathcal{A}$. When $\mathcal{A}$ commits to its set $\mathsf{T} = \{x_{i^*}\}_{i\in[t]}$, algorithm $\mathcal{B}$ runs the auxiliary setup algorithm Setup^*, except it instantiates pp^* as follows:
   - It uses the matrices $\hat{\mathbf{A}}$, $\{\mathbf{A}'_b\}_{b\in\{0,1\}}$, $\{\mathbf{B}'_{i,j}\}_{i\in[t],j\in[z]}$, and $\{\mathbf{C}'_k\}_{k\in[\tau]}$ from the LWE challenge in place of the corresponding matrices in Setup^*.
   - It uses the matrices $\mathbf{H}'_i$ from the LWE challenge to instantiate the vectors $\{\mathbf{w}_i\}_{i\in[t]}$. Namely, it sets $\mathbf{w}_\ell$ such that $\mathbf{H}'_i = \sum_{\ell\in[N]} w_{i,\ell}\mathbf{D}_\ell$.

   Finally, $\mathcal{B}$ constructs the remaining components of pp^* and msk^* exactly as described in Setup^* algorithm, with the exception that it does not sample a secret key $\mathbf{s}$ in msk^*.

3. To simulate the constrained key sk^*_T, algorithm $\mathcal{B}$ sets $\mathsf{enc} = \big(\hat{\mathbf{a}}, \{\mathbf{a}'_b\}_{b\in\{0,1\}}, \{\mathbf{b}'_{i,j}\}_{i\in[t],j\in[z]}, \{\mathbf{c}'_k\}_{k\in[\tau]}\big)$ to be the vectors from the LWE challenge. The ciphertexts ct are constructed exactly as described in Setup^*. Finally, $\mathcal{B}$ gives the public parameters pp^* and the constrained key $\mathsf{sk}^*_\mathsf{T} = (\mathsf{enc}, \mathsf{ct})$ to $\mathcal{A}$.

4. To simulate the honest evaluation queries for $x \notin \mathsf{T}$, $\mathcal{B}$ computes the vector
$$\widetilde{\mathbf{b}}_{i,\ell} \leftarrow \mathsf{Eval}_{\mathsf{ct}}((\mathsf{ct}_i, x), C_\ell, \mathbf{b}_{i,1}, \ldots, \mathbf{b}_{i,z}, \mathbf{a}_{x_1}, \ldots, \mathbf{a}_{x_\rho}, \mathbf{c}_1, \ldots, \mathbf{c}_\tau)$$
   for $i \in [t]$ and $\ell \in [N]$ and returns the value
$$\tilde{\mathbf{y}} = \left\lfloor \hat{\mathbf{a}} + \sum_{\substack{i\in[t]\\ \ell\in[N]}} \widetilde{\mathbf{b}}^T_{i,\ell} \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) \right\rceil_p .$$

5. Whenever $\mathcal{A}$ makes a challenge oracle query on a point $x \in \mathsf{T}$ (in particular, this means that $x = x_{i^*}$ for some $i^* \in [t]$), algorithm $\mathcal{B}$ responds as follows. It first computes the vector
$$\widetilde{\mathbf{b}}_{i,\ell} \leftarrow \mathsf{Eval}_{\mathsf{ct}}((\mathsf{ct}_i, x), C_\ell, \mathbf{b}_{i,1}, \ldots, \mathbf{b}_{i,z}, \mathbf{a}_{x_1}, \ldots, \mathbf{a}_{x_\rho}, \mathbf{c}_1, \ldots, \mathbf{c}_\tau)$$
   for $i \in [t]$ and $\ell \in [N]$ and returns the value
$$\tilde{\mathbf{y}} = \left\lfloor \hat{\mathbf{a}} + \sum_{\substack{i\in[t]\\ \ell\in[N]}} \widetilde{\mathbf{b}}^T_{i,\ell} \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) - \mathbf{h}'_{i^*} \right\rceil_p .$$

6. Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

We now argue that the public parameters pp^*, the constrained key sk^*_T, the honest evaluation queries, and the challenge oracle queries are correctly simulated.

- By definition, the matrices $\hat{\mathbf{A}}$, $\{\mathbf{A}'_b\}_{b\in\{0,1\}}$, $\{\mathbf{B}'_{i,j}\}_{i\in[t],j\in[z]}$, and $\{\mathbf{C}'_k\}_{k\in[\tau]}$ are distributed uniformly and independently over $\mathbb{Z}_q^{n\times m}$, exactly as those sampled by Setup^*. In addition, since each $\mathbf{H}'_i$ is also uniformly random over $\mathbb{Z}_q^{n\times m}$, it follows that each $\mathbf{w}_i$ is uniform over $\mathbb{Z}_q^N$ (since the collection $\{\mathbf{D}_\ell\}_{\ell\in[N]}$ constitutes a basis for $\mathbb{Z}_q^{n\times m}$). Thus, algorithm $\mathcal{B}$ perfectly simulates the behavior of Setup^* in H_2 and H_3 (except it does not explicitly sample a secret vector $\mathbf{s}$).
- Next, if the challenge vectors $\big(\hat{\mathbf{a}}, \{\mathbf{a}'_b\}_{b\in\{0,1\}}, \{\mathbf{b}'_{i,j}\}_{i\in[t],j\in[z]}, \{\mathbf{c}'_k\}_{k\in[\tau]}\big)$ are LWE samples, then $\mathcal{B}$ has correctly simulated the distribution of sk^*_T in H_2. If instead they are uniformly random, then $\mathcal{B}$ has correctly simulated the distribution of sk^*_T in H_3.
- For the honest evaluation queries for $x \notin \mathsf{T}$, it is easy to see that the simulation is correct since $\mathcal{B}$ is simply computing the auxiliary evaluation function Eval^*_1, which is used in both hybrid experiments.

- For the challenge queries, if $\{\mathbf{h}'_i\}_{i\in[t]}$ are LWE samples, then we have for all $i^* \in [t]$,

$$\mathbf{h}'_{i^*} = \mathbf{s}^T\mathbf{H}'_{i^*} + \mathbf{e}^T_{i^*} = \mathbf{s}^T \sum_{\ell\in[N]} w_{i^*,\ell}\mathbf{D}_\ell + \mathbf{e}^T_{i^*},$$

  where $\mathbf{s}$ is the LWE secret and $\mathbf{e}_i$ is an error term. Therefore, the value

$$\tilde{\mathbf{y}} = \left\lfloor \hat{\mathbf{a}} + \sum_{\substack{i\in[t]\\ \ell\in[N]}} \widetilde{\mathbf{b}}^T_{i,\ell} \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) - \mathbf{h}'_{i^*} \right\rceil_p .$$

  is a perfect simulation of Eval^*_1 in H_2. Alternatively, if the vectors $\mathbf{h}'_{i^*}$ are uniformly random, then $\mathcal{B}$ correctly simulates the challenge oracle responses with Eval^*_2 according to H_3.

We conclude that if algorithm $\mathcal{B}$ obtains samples from the LWE distribution, then the view it simulates for $\mathcal{A}$ is identical to the view of $\mathcal{A}$ in H_2. Otherwise, if $\mathcal{B}$ obtains samples from a uniformly random distribution, then the view it simulates for $\mathcal{A}$ is identical to the view of $\mathcal{A}$ in H_3. Thus, we conclude that if $\mathcal{A}$ is able to distinguish H_2 from H_3 with non-negligible probability, $\mathcal{B}$ can break the $\mathsf{LWE}_{n,m',q,\chi}$ assumption with the same probability. □

Lemma A.8. *Under the* $\mathsf{LWE}_{n,m',q,\chi}$ *assumption (where* $m' = m(3 + t(z + 1) + \tau)$*), and* $\mathsf{1D\text{-}SIS\text{-}R}_{m'',p,q,\beta}$ *assumptions (where* $m'' = m(3 + t \cdot z + \tau)$ *and* $\beta = B \cdot m^{O(d)}$*) for all efficient adversaries* $\mathcal{A}$*, we have that* $|\Pr[\mathsf{H}_1(\mathcal{A}) = 1] - \Pr[\mathsf{H}_2(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. The only difference between hybrids H_1 and H_2 is that in H_2, the honest evaluation and challenge queries are answered using the auxiliary evaluation algorithm $\mathsf{Eval}^*_1(\mathsf{pp}^*, \mathsf{msk}^*, \mathsf{sk}^*_\mathsf{T}, \cdot)$ rather than the real evaluation algorithm $\mathsf{TPRF.Eval}(\mathsf{pp}^*, \mathsf{sk}^*_\mathsf{T}, \cdot)$. For clarity of presentation, we consider the case for the evaluation queries and challenge queries separately.

Evaluation oracle queries. By the admissibility condition, the adversary is only allowed to query the evaluation oracle on inputs $x \notin \mathsf{T}$. In this case, the auxiliary evaluation algorithm $\mathsf{Eval}^*_1(\mathsf{pp}^*, \mathsf{msk}^*, \mathsf{sk}^*_\mathsf{T}, x)$ simply implements the constrained evaluation algorithm $\mathsf{TPRF.ConstrainEval}(\mathsf{pp}^*, \mathsf{sk}^*_\mathsf{T}, x)$ using the auxiliary public parameters and constrained key. As long as $\mathsf{TPRF.Eval}(\mathsf{pp}^*, \mathsf{sk}^*_\mathsf{T}, x) = \mathsf{TPRF.ConstrainEval}(\mathsf{pp}^*, \mathsf{sk}^*_\mathsf{T}, x)$, the distribution of the evaluation queries in both H_1 and H_2 are identical. However, this is precisely the guarantee provided by evaluation correctness (Definition 4.4). More precisely, we can apply the same argument as in the proof of Theorem 5.1 in Appendix A.2.1 to show that the constrained evaluation agrees with the true evaluation. Thus, we conclude that the responses to all (admissible) evaluation oracle queries in H_1 and H_2 are identical with overwhelming probability.

Challenge oracle queries. We now consider the challenge oracle queries. In particular, we argue that the outputs of $\mathsf{Eval}^*_1(\mathsf{pp}^*, \mathsf{msk}^*, \mathsf{sk}^*_\mathsf{T}, \cdot)$ and $\mathsf{TPRF.Eval}(\mathsf{pp}^*, \mathsf{sk}^*_\mathsf{T}, \cdot)$ on the challenge queries $x \in \mathsf{T}$ are computationally indistinguishable. We start by recalling how the challenge queries are handled in the two hybrid experiments:

- In H_1, on input a point x, the challenger computes

$$\widetilde{\mathbf{B}}_{i,\ell} \leftarrow \mathsf{Eval}_{\mathsf{pk}}(C_\ell, \mathbf{B}_{i,1}, \ldots, \mathbf{B}_{i,z}, \mathbf{A}_{x_{i,1}}, \ldots, \mathbf{A}_{x_{i,\rho}}, \mathbf{C}_1, \ldots, \mathbf{C}_\tau).$$

  for $i \in [t]$, $\ell \in [N]$ and returns the value

$$\mathbf{y}_x = \left\lfloor \mathbf{s}^T \left(\hat{\mathbf{A}} + \sum_{\substack{i\in[t]\\ \ell\in[N]}} \widetilde{\mathbf{B}}_{i,\ell} \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) \right) \right\rceil_p$$

- In H_2, on input a point $x = x^*_{i^*}$, for some $i^* \in [t]$, the challenger computes $\tilde{\mathbf{y}} = \mathsf{Eval}^*_1(\mathsf{pp}^*, \mathsf{msk}^*, \mathsf{sk}^*_{\mathsf{T}}, x)$ by first computing

$$\widetilde{\mathbf{b}}_{i,\ell} \leftarrow \mathsf{Eval}_{\mathsf{ct}}((\mathsf{ct}_i, x), C_\ell, \mathbf{b}_{i,1}, \ldots, \mathbf{b}_{i,z}, \mathbf{a}_{x_1}, \ldots, \mathbf{a}_{x_\rho}, \mathbf{c}_1, \ldots, \mathbf{c}_\tau)$$

for $i \in [t]$ and $\ell \in [N]$. It then samples an error vector $\mathbf{e} \leftarrow \chi^m$ and returns

$$\tilde{\mathbf{y}} = \left\lfloor \hat{\mathbf{a}} + \sum_{\substack{i \in [t] \\ \ell \in [N]}} \widetilde{\mathbf{b}}_{i,\ell}^T \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) - \mathbf{s}^T \sum_{\ell \in [N]} w_{i^*,\ell} \mathbf{D}_\ell - \mathbf{e}^T \right\rceil_p .$$

By Theorems 3.9 and 3.10 and the definition of $\{\mathbf{A}_b\}_{b \in \{0,1\}}$, $\{\mathbf{B}_{i,j}\}_{i \in [t], j \in [z]}$, and $\{\mathbf{C}_k\}_{k \in [\tau]}$ in Setup^*, we can write

$$\widetilde{\mathbf{b}}_{i,\ell}^T = \mathbf{s}^T \left(\widetilde{\mathbf{B}}_{i,\ell} + (\mathsf{eq}(x, x^*_i) \cdot w_{i,\ell} + \epsilon_{i,\ell}) \cdot \mathbf{G} \right) + \mathbf{e}_{i,\ell}^T$$

for $\|\mathbf{e}_{i,\ell}\| \le B \cdot m^{O(d)}$. Since $\mathsf{eq}(x, x^*_i) = 0$ for $i \neq i^*$ and $\mathsf{eq}(x, x^*_{i^*}) = 1$, we can rewrite $\tilde{\mathbf{y}}$ as follows

$$\begin{aligned} \tilde{\mathbf{y}} &= \left\lfloor \hat{\mathbf{a}} + \sum_{\substack{i \in [t] \\ \ell \in [N]}} \left[\mathbf{s}^T \left(\widetilde{\mathbf{B}}_{i,\ell}^T \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) + \epsilon_{i,\ell} \mathbf{D}_\ell \right) + \mathbf{e}_{i,\ell}^T \mathbf{G}^{-1}(\mathbf{D}_\ell) \right] + \mathbf{s}^T \sum_{\ell \in [N]} (w_{i^*,\ell} \mathbf{D}_\ell - w_{i^*,\ell} \mathbf{D}_\ell) - \mathbf{e}^T \right\rceil_p \\ &= \left\lfloor \mathbf{s}^T \left(\hat{\mathbf{A}} + \sum_{\substack{i \in [t] \\ \ell \in [N]}} \widetilde{\mathbf{B}}_{i,\ell}^T \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) \right) + \widetilde{\mathbf{e}}^T \right\rceil_p , \end{aligned}$$

where $\widetilde{\mathbf{e}}^T = \sum_{i \in [t], \ell \in [N]} \left(\epsilon_{i,\ell} \cdot \mathbf{s}^T \mathbf{D}_\ell + \mathbf{e}_{i,\ell}^T \mathbf{G}^{-1}(\mathbf{D}_\ell) \right) - \mathbf{e}^T$. Note that $\|\widetilde{\mathbf{e}}\| \le B \cdot m^{O(d)}$.

For notational convenience, define $\boldsymbol{\xi}_x \in \mathbb{Z}_q^m$ to be the "unrounded" PRF value in H_2:

$$\boldsymbol{\xi}_x^T = \mathbf{s}^T \left(\hat{\mathbf{A}} + \sum_{\substack{i \in [t] \\ \ell \in [N]}} \widetilde{\mathbf{B}}_{i,\ell}^T \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) \right) + \widetilde{\mathbf{e}}^T.$$

Then, we can write $\tilde{\mathbf{y}}_x = \lfloor \boldsymbol{\xi}_x^T \rceil_p$ and $\mathbf{y}_x = \lfloor \boldsymbol{\xi}_x^T - \widetilde{\mathbf{e}}^T \rceil_p$. Thus, we see that $\mathbf{y}_x = \tilde{\mathbf{y}}_x$ as long as the vector $\boldsymbol{\xi}_x$ does not contain any "borderline" components that can be rounded in the wrong direction due to $\widetilde{\mathbf{e}}$. Similar to the proof of evaluation correctness in Appendix A.2.1, we define $\mathsf{Borderline}_x$ to be the event that there exists an index $\eta \in [m]$ such that

$$\boldsymbol{\xi}_x^T \mathbf{u}_\eta \in [-E, E] + \frac{q}{p} \cdot (\mathbb{Z} + 1/2)$$

where $E = m^{O(d)}$ is a bound on $\|\widetilde{\mathbf{e}}\|$ and $\mathbf{u}_\eta$ is the η^{th} basis vector. To conclude the proof, we show that $\Pr[\mathsf{Borderline}_x] = \mathsf{negl}(\lambda)$ in H_2. Our argument consists of two steps. First, we argue that in H_3, the "unrounded" PRF evaluation does *not* contain any borderline components. This in turn implies that in H_2, the unrounded PRF value $\boldsymbol{\xi}_x$ does not contain any borderline components—otherwise, algorithm $\mathcal{B}$ from the proof of Lemma A.7 can be used to distinguish H_2 from H_3, in violation of Lemma A.7. We now show this more formally.

- In hybrid H_3, on a challenge query $x = x^*_i$, the response is computed by first sampling $\mathbf{d} \xleftarrow{\text{R}} \mathbb{Z}_q^m$ and then rounding $\tilde{\mathbf{y}}_x = \lfloor \mathbf{d} \rceil_p$. Since $E \cdot p/q = \mathsf{negl}(\lambda)$, we conclude that for each $\eta \in [m]$,

$$\Pr[\mathbf{d}^T \mathbf{u}_\eta \in [-E, E] + (q/p) \cdot (\mathbb{Z} + 1/2)] = \mathsf{negl}(\lambda).$$

Thus, with overwhelming probability, $\mathbf{d}$ does not contain any borderline components.

- Suppose in H_2 that the vector $\boldsymbol{\xi}_x$ contains a borderline component with non-negligible probability. But then the algorithm $\mathcal{B}$ from the proof of Lemma A.7 can be used to distinguish H_2 from H_3: the algorithm $\mathcal{B}$ simply outputs 1 if the unrounded vector contains a borderline component. From our analysis in Lemma A.7, the unrounded vector $\boldsymbol{\xi}_x$ is distributed exactly as in H_2 if $\mathcal{B}$ received samples from the LWE distribution whereas the unrounded vector $\boldsymbol{\xi}_x$ is distributed as in H_3 if $\mathcal{B}$ received samples from the uniform distribution. Thus, under $\mathsf{LWE}_{n,m',q,\chi}$, it must be the case that $\boldsymbol{\xi}_x$ does not contain any borderline components with overwhelming probability.

Under the $\mathsf{LWE}_{n,m',q,\chi}$ assumption, we have that $\Pr[\mathsf{Borderline}_x] = \mathsf{negl}(\lambda)$. In this case, $\mathbf{y}_x = \tilde{\mathbf{y}}_x$. We conclude that the distributions of responses to the challenge queries in H_1 and H_2 are computationally indistinguishable. □

Lemma A.9. *Under the* $\mathsf{LWE}_{n,m',q,\chi}$ *assumption (where* $m' = m(3+t(z+1)+\tau)$*), for all efficient adversaries* $\mathcal{A}$*,* $|\Pr[\mathsf{H}_3(\mathcal{A}) = 1] - \Pr[\mathsf{H}_4(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$*.*

Proof. Follows from a similar argument as Lemma A.7 (except the behavior of the challenge oracle is identical in the two experiments). □

Lemma A.10. *Under the* $\mathsf{LWE}_{n,m',q,\chi}$ *assumption (where* $m' = m(3+t(z+1)+\tau)$*) and the* $\mathsf{1D\text{-}SIS\text{-}R}_{m'',p,q,\beta}$ *(where* $m'' = m(3+tz+\tau)$*) for all efficient adversaries* $\mathcal{A}$*,* $|\Pr[\mathsf{H}_4(\mathcal{A}) = 1] - \Pr[\mathsf{H}_5(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$*.*

Proof. Follows from a similar argument as Lemma A.8, except we only have to reason about how the evaluation oracle queries are handled. The challenge queries are handled identically in the two experiments. □

Lemma A.11. *For all adversaries* $\mathcal{A}$*,* $|\Pr[\mathsf{H}_5(\mathcal{A}) = 1] - \Pr[\mathsf{H}_6(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$*.*

Proof. Follows from the same argument as Lemma A.6. □

Combining Lemmas A.6 through A.11, we conclude that Π_{TPRF} satisfies (selective) constrained pseudorandomness. □

A.3.2 Proof of Theorem 5.4

Recall first that a constrained key $\mathsf{sk}_\mathsf{T} = (\mathsf{enc}, \mathsf{ct})$ for a set $\mathsf{T} = \{x_i^*\}_{i\in[t]}$ consists of two components: a set of encodings enc and a collection of ciphertexts $\mathsf{ct} = \{\mathsf{ct}_i\}_{i\in[t]}$. In our proof of constrained pseudorandomness (Theorem 5.2, Appendix A.3.1), we demonstrated that the set of encodings enc in the constrained key is indistinguishable from a collection of random vectors. Together with semantic security of the FHE ciphertexts ct_i, we have that the constrained key sk_T hides the set T.

Formally, we proceed with a hybrid argument. Let $\mathcal{A}$ be an adversary and $\mathcal{S}^{(t)}$ be the set system corresponding to the family of t-puncturable constraints (Definition 4.7). In the proof, we show the selective notion of privacy (Remark 4.14) where we assume that the adversary commits to its two challenge sets S_0 and S_1 at the beginning of the experiment. We now introduce our hybrid experiments.

- **Hybrid H_0:** This is the real experiment $\mathsf{PExpt}^{(0)}_{\Pi_{\mathsf{TPRF}},\mathcal{A},\mathcal{S}^{(t)}}$ where the challenger, on input two sets $S_0, S_1 \in \mathcal{S}^{(t)}$, gives the adversary the constrained key $\mathsf{sk}_0 = (\mathsf{enc}, \mathsf{ct}) \leftarrow \mathsf{TPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, S_0)$, where pp and msk are sampled exactly as in the real experiment.

- **Hybrid H_1:** Same as H_0, except the encodings in sk_0 are replaced by a uniformly random string. More precisely, the challenger first computes $(\mathsf{enc}, \mathsf{ct}) \leftarrow \mathsf{TPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, S_0)$. Then, it samples $\mathbf{r} \xleftarrow{\text{R}} \{0,1\}^{|\mathsf{enc}|}$ and returns $\mathsf{sk}_0 = (\mathbf{r}, \mathsf{ct})$ to the adversary.

- **Hybrid H_2:** Same as H_1, except that the challenger computes $(\mathsf{enc}, \mathsf{ct}) \leftarrow \mathsf{TPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, S_1)$. Then, it samples $\mathbf{r} \xleftarrow{\text{R}} \{0,1\}^{|\mathsf{enc}|}$ and returns $\mathsf{sk}_1 = (\mathbf{r}, \mathsf{ct})$ to the adversary.

- **Hybrid H_3:** This is the real experiment $\mathsf{PExpt}^{(1)}_{\Pi_{\mathsf{TPRF}},\mathcal{A},\mathcal{S}^{(t)}}$, where the challenger, on input two sets $S_0, S_1 \in \mathcal{S}^{(t)}$, replies to the adversary with the constrained key $\mathsf{sk}_1 \leftarrow \mathsf{TPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, S_1)$.

Lemma A.12. *Under the* $\mathsf{LWE}_{n,m',q,\chi}$ *assumption (where* $m' = m(3 + t(z + 1) + \tau)$*), for all efficient adversaries* $\mathcal{A}$*,* $|\Pr[\mathsf{H}_0(\mathcal{A}) = 1] - \Pr[\mathsf{H}_1(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. The lemma follows directly by the indistinguishability of hybrid experiments H_0 and H_3 in the proof of Theorem 5.2 in Appendix A.3.1 (Lemmas A.6, A.7, and A.8). In particular, note that the adversary in the (selective) privacy game is *strictly* weaker than the adversary in the (selective) constrained pseudorandomness game since it is not give access to either a challenge oracle or an evaluation oracle. Thus, we can invoke the corresponding lemmas from Appendix A.3.1. Moreover, we note that the 1D-SIS-R assumption needed in Lemma A.8 is not necessary in the case of privacy because the challenger does not need to simulate the evaluation oracle queries. In Lemma A.8, the 1D-SIS-R assumption is needed to argue that the evaluation questions are properly simulated. □

Lemma A.13. *If* Π_{HE} *is semantically secure (Definition 3.8), then for all efficient adversaries* $\mathcal{A}$*, it follows that* $|\Pr[\mathsf{H}_1(\mathcal{A}) = 1] - \Pr[\mathsf{H}_2(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. We argue that by semantic security of Π_{HE}, the adversary's views in H_1 and H_2 are computationally indistinguishable. First, the public parameters pp are identically distributed in the two distributions. Let $\mathsf{sk}_0 = (\mathbf{r}_0, \mathsf{ct}_0)$ and $\mathsf{sk}_1 = (\mathbf{r}_1, \mathsf{ct}_1)$ be the constrained keys the adversary receives in H_1 and H_2, respectively. By construction, $\mathbf{r}_0$ and $\mathbf{r}_1$ are uniform over $\{0,1\}^{|\mathsf{enc}|}$ and independent of all other parameters. Thus, it suffices to argue that the ciphertexts ct_0 and ct_1 are computationally indistinguishable. But since ct_0 and ct_1 consists of a (polynomial-sized) collection of ciphertexts encrypted under Π_{HE} (with a secret key that is unknown to the adversary $\mathcal{A}$), semantic security of Π_{HE} implies that the distribution of ct_0 is computationally indistinguishable from the distribution of ct_1. The claim follows. □

Lemma A.14. *Under the* $\mathsf{LWE}_{n,m',q,\chi}$ *assumption (where* $m' = m(3 + t(z + 1) + \tau)$*), for all efficient adversaries* $\mathcal{A}$*,* $|\Pr[\mathsf{H}_2(\mathcal{A}) = 1] - \Pr[\mathsf{H}_3(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. Follows from the same argument as Lemma A.12. □

Combining Lemmas A.12 through A.14, we have that experiments $\mathsf{PExpt}^{(0)}_{\Pi_{\mathsf{TPRF}},\mathcal{A},\mathcal{S}^{(t)}}$ and $\mathsf{PExpt}^{(1)}_{\Pi_{\mathsf{TPRF}},\mathcal{A},\mathcal{S}^{(t)}}$ are computationally indistinguishable. Thus, Π_{TPRF} is (selectively) private. □

A.3.3 Proof of Theorem 5.5

Let $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{TPRF.Setup}(1^\lambda)$ and take any $\mathsf{msk} = \mathbf{s} \in [-B, B]^n$. Take $x \in \{0,1\}^\rho$. Then, to compute the PRF value at x, the evaluation algorithm $\mathsf{TPRF.Eval}(\mathsf{msk}, x)$ first computes the matrices

$$\widetilde{\mathbf{B}}_{i,\ell} \leftarrow \mathsf{Eval}_{\mathsf{pk}}(C_\ell, \mathbf{B}_{i,1}, \ldots, \mathbf{B}_{i,\ell}, \mathbf{A}_{x_1}, \ldots, \mathbf{A}_{x_\rho}, \mathbf{C}_1, \ldots, \mathbf{C}_\tau)$$

for all $i \in [t]$ and $\ell \in [N]$. It then outputs the vector

$$\mathbf{y}_x = \left\lfloor \mathbf{s}^T \left(\hat{\mathbf{A}} + \sum_{\substack{i \in [t] \\ \ell \in [N]}} \widetilde{\mathbf{B}}_{i,\ell} \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell) \right) \right\rceil_p .$$

To simplify notation, let $\mathbf{B}' = \sum_{i \in [t], \ell \in [N]} \widetilde{\mathbf{B}}_{i,\ell} \cdot \mathbf{G}^{-1}(\mathbf{D}_\ell)$. Now, suppose that there are two keys $\mathsf{msk}_1 = \mathbf{s}_1, \mathsf{msk}_2 = \mathbf{s}_2 \in [-B, B]^n$ where $\mathsf{TPRF.Eval}(\mathsf{msk}_1, x) = \mathsf{TPRF.Eval}(\mathsf{msk}_2, x)$ for some $x \in \{0,1\}^\rho$. Then,

$$\left\lfloor \mathbf{s}_1^T \left(\hat{\mathbf{A}} + \mathbf{B}' \right) \right\rceil_p = \left\lfloor \mathbf{s}_2^T \left(\hat{\mathbf{A}} + \mathbf{B}' \right) \right\rceil_p .$$

This means that the vectors $\mathbf{s}_1^T(\hat{\mathbf{A}} + \mathbf{B}')$ and $\mathbf{s}_2^T(\hat{\mathbf{A}} + \mathbf{B}')$ are "close" or more precisely,

$$\mathbf{s}_1^T(\hat{\mathbf{A}} + \mathbf{B}') - \mathbf{s}_2^T(\hat{\mathbf{A}} + \mathbf{B}') = (\mathbf{s}_1^T - \mathbf{s}_2^T)(\hat{\mathbf{A}} + \mathbf{B}') \in [-B', B']^m,$$

where $B' = \frac{q}{2p}$. To complete the proof, we show that such a vector $\hat{\mathbf{s}} = (\mathbf{s}_1 - \mathbf{s}_2)$ exists in $\mathbb{Z}_q^n$ with only negligible probability over the randomness used to sample the public parameter matrices (specifically, over the choice of the random coins used to sample $\hat{\mathbf{A}}$).

Lemma A.15. *Fix any matrix $\mathbf{B}' \in \mathbb{Z}_q^{n\times m}$ where $m = \omega(n)$. Then, if the bound B on the error distribution χ satisfies $B < \hat{p}/2$, where $\hat{p}$ is the smallest prime dividing the modulus q, and $B' = q/2p$, we have that*

$$\Pr_{\hat{\mathbf{A}} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times m}} \left[\exists\, \hat{\mathbf{s}} \in [-2B, 2B]^n \setminus \{\mathbf{0}\} \colon \hat{\mathbf{s}}^T(\hat{\mathbf{A}} + \mathbf{B}') \in [-B', B']^m\right] = \mathsf{negl}(\lambda).$$

Proof. We bound the probability that there exists a non-zero $\hat{\mathbf{s}} \in [-2B, 2B]^n$ such that $\hat{\mathbf{s}}^T\hat{\mathbf{A}} = -\hat{\mathbf{s}}^T\mathbf{B}' + \mathbf{e}^T$ where $\mathbf{e} \in [-B', B']^m$. Take any non-zero $\hat{\mathbf{s}} \in [-2B, 2B]^n$. Since $\hat{\mathbf{s}} \neq \mathbf{0}$, there exists an index $i \in [n]$ such that $\hat{s}_i \neq 0$. Moreover, since $|\hat{s}_i| \le 2B < \hat{p}$, $\hat{s}_i$ is invertible over $\mathbb{Z}_q$. Since $\hat{\mathbf{A}}$ is sampled uniformly at random, the relation $\hat{\mathbf{s}}^T\hat{\mathbf{A}} = -\hat{\mathbf{s}}^T\mathbf{B}' + \mathbf{e}^T$ is satisfied for some $\mathbf{e} \in [-B', B']^n$ with probability at most $(2B'/q)^m = (1/p)^m$. The claim then follows if we take a union bound over the $(4B)^n$ possible vectors $\hat{\mathbf{s}} \in [-2B, 2B]^n$. ☐

We conclude that for any x, with overwhelming probability over the choice of $\hat{\mathbf{A}}$, there does not exist a pair of keys $\mathbf{s}_1$ and $\mathbf{s}_2$ such that $\mathbf{s}_1^T(\hat{\mathbf{A}} + \mathbf{B}')$ and $\mathbf{s}_2^T(\hat{\mathbf{A}} + \mathbf{B}')$ are close. In particular this means that with overwhelming probability, $\mathsf{TPRF.Eval}(\mathsf{msk}_1, x) \neq \mathsf{TPRF.Eval}(\mathsf{msk}_2, x)$ for any $x \in \{0,1\}^\rho$. Thus, Π_{TPRF} satisfies key-injectivity. ☐

B Watermarking Correctness and Security Analysis

In this section, we give the formal correctness and security analysis of the watermarking scheme described in Construction 6.13.

B.1 Proof of Theorem 6.14

Take any message $m \in \{0,1\}^t$. Let $\mathsf{msk} = (\mathsf{pp}, \mathsf{tk}, h_1, \ldots, h_d, \mathsf{k}^*) \leftarrow \mathsf{WM.Setup}(1^\lambda)$ be the master secret key for the watermarking scheme. Take $k \leftarrow \mathsf{TPRF.SampleKey}(\mathsf{pp})$ and let $C \leftarrow \mathsf{WM.Mark}(\mathsf{msk}, k, m)$ be the watermarked key. By construction $C(\cdot) = \mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk}_S, \cdot)$ where $\mathsf{sk}_S = \mathsf{TPRF.Constrain}(\mathsf{pp}, k, S)$ and $S \subseteq \{0,1\}^n$ is a set of $2^n - t$ points. We now show each of the requirements separately:

- **Functionality-preserving:** Let $T \subseteq S$ be the set of points x where $C(x) \neq \mathsf{TPRF.Eval}(\mathsf{pp}, k, x)$. By evaluation correctness of Π_{TPRF}, no efficient adversary is able to find any such $x \in T$, except with negligible probability. In particular, this means that $|T|/2^n = \mathsf{negl}(\lambda)$. Finally, since $C(\cdot)$ can differ from $\mathsf{TPRF.Eval}(\mathsf{pp}, k, \cdot)$ on at most $|T| + t$ points and $t = \mathsf{poly}(\lambda)$, we conclude that $C(\cdot)$ agrees with $\mathsf{TPRF.Eval}(\mathsf{pp}, k, \cdot)$ on all but a negligible fraction of points.

- **Extraction correctness:** First, define $\mathbf{x} = \left(x_1^{(0)}, x_1^{(1)}, \ldots, x_t^{(0)}, x_t^{(1)}\right)$ as in $\mathsf{WM.Mark}$ (that is, as the output of $\mathsf{PRF.Eval}(\mathsf{k}^*, \cdot)$). By construction, $\{0,1\}^n \setminus S \subset \{x_1^{(0)}, x_1^{(1)}, \ldots, x_t^{(0)}, x_t^{(1)}\}$. Since Π_{PRF} is secure and $n = \omega(\log \lambda)$, it follows that $\Pr[x_i^{(b)} = h_j] = \mathsf{negl}(\lambda)$ for all $j \in [d]$, $i \in [t]$, and $b \in \{0,1\}$. Since $d, t = \mathsf{poly}(\lambda)$, we conclude via a union bound that with overwhelming probability, $h_j \neq x_i^{(b)}$ for all $j \in [d]$, $i \in [t]$, and $b \in \{0,1\}$. Equivalently, $h_1, \ldots, h_d \in S$ with overwhelming probability. Since $h_1, \ldots, h_d$ are chosen uniformly over $\{0,1\}^n$ and independently of all other parameters, we invoke evaluation correctness of Π_{TPRF} and Remark 4.6 to conclude that with overwhelming probability, $C(h_j) = \mathsf{TPRF.Eval}(\mathsf{pp}, k, h_j)$ for each $j \in [d]$. Since $d = \mathsf{poly}(\lambda)$, we apply a union bound to conclude that with overwhelming probability, the extraction algorithm $\mathsf{WM.Extract}$ will derive the same tuple $\mathbf{x}$ as $\mathsf{WM.Mark}$. The claim now follows from verification correctness of Π_{TPRF}.

B.2 Proofs of Theorem 6.15 and 6.16

Our unremovability and unforgeability proofs for our watermarking scheme follow a similar structure as the proofs in [BLW17, Appendix I], who construct a watermarkable family of PRFs from private programmable PRFs. However, we require a more intricate argument to handle adversarial marking oracle queries (where the adversary is allowed to choose the key to be watermarked) as well. Moreover, relying on private translucent t-puncturable PRFs rather than private programmable PRFs (the former provides a much weaker programmability property) also require modifying the hybrid structure in [BLW17].

Our security proofs consist of a sequence of hybrid experiments between a challenger and an adversary $\mathcal{A}$. In each experiment, the adversary $\mathcal{A}$ is given access to a marking oracle and a challenge oracle. We now define our initial hybrid experiment, denoted H_0, which is identical to the watermarking experiment $\mathsf{Expt}_{\Pi_{\mathsf{WM}},\mathcal{A}}$ (Definition 6.7). Note that we isolate this particular hybrid because it will be useful in both the proofs of Theorem 6.15 as well as Theorem 6.16. In this section, for a hybrid experiment H, we write $\mathsf{H}(\mathcal{A})$ to denote the output distribution of H when interacting with an adversary $\mathcal{A}$.

Definition B.1 (Hybrid H_0)**.** Fix a security parameter λ. Let $\Pi_{\mathsf{WM}} = (\mathsf{WM.Setup}, \mathsf{WM.Mark}, \mathsf{WM.Extract})$ be the watermarking scheme from Construction 6.13, and let $\mathcal{A}$ be a watermarking adversary. Hybrid $\mathsf{H}_0(\mathcal{A})$ corresponds to the watermarking experiment $\mathsf{Expt}_{\Pi_{\mathsf{WM}},\mathcal{A}}(\lambda)$. For clarity, we describe the experiment with respect to the concrete instantiation described in Construction 6.13.

1. **Setup phase:** The challenger begins by sampling $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{TPRF.Setup}(1^\lambda)$, a tuple $(h_1, \dots, h_d) \xleftarrow{\text{R}} (\{0,1\}^n)^d$ and a PRF key $\mathsf{k}^* \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$. It sets $\mathsf{msk} = (\mathsf{pp}, \mathsf{tk}, h_1, \dots, h_d, \mathsf{k}^*)$ and gives pp to the adversary.
2. **Query phase:** The adversary can now make queries to a marking oracle or a challenge oracle. The challenger responds to the oracle queries as follows:
   - **Marking oracle:** On input a message $m \in \{0,1\}^t$ and a PRF key $k \in \mathcal{K}$ to be marked, the challenger computes $y_j \leftarrow \mathsf{TPRF.Eval}(\mathsf{pp}, k, h_j)$ for each $j \in [d]$. Next, it sets $\mathbf{y} = (y_1, \dots, y_d)$, and computes $\mathbf{x} = \big(x_1^{(0)}, x_1^{(1)}, \dots, x_t^{(0)}, x_t^{(1)}\big) \leftarrow \mathsf{PRF.Eval}(\mathsf{k}^*, \mathbf{y})$. Then, it constructs the t-punctured key $\mathsf{sk}_S \leftarrow \mathsf{TPRF.Constrain}(\mathsf{pp}, k, S)$ where $S = \{x \in \{0,1\}^n : x \neq x_i^{(m_i)} \ \forall i \in [t]\}$. Finally, it replies with the circuit C to the adversary, where $C(\cdot) = \mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk}_S, \cdot)$.
   - **Challenge oracle:** On input a message $\hat{m} \in \{0,1\}^t$, the challenger samples a key $\hat{k} \leftarrow \mathsf{TPRF.SampleKey}(\mathsf{pp})$. Next, for each $j \in [d]$, it computes $\hat{y}_j \leftarrow \mathsf{TPRF.Eval}(\mathsf{pp}, \hat{k}, h_j)$, sets $\hat{\mathbf{y}} = (\hat{y}_1, \dots, \hat{y}_d)$, and computes $\hat{\mathbf{x}} = \big(\hat{x}_1^{(0)}, \hat{x}_1^{(1)}, \dots, \hat{x}_t^{(0)}, \hat{x}_t^{(1)}\big) \leftarrow \mathsf{PRF.Eval}(\mathsf{k}^*, \hat{\mathbf{y}})$. Then, it constructs the t-punctured key $\mathsf{sk}_{\hat{S}} \leftarrow \mathsf{TPRF.Constrain}(\mathsf{pp}, \hat{k}, \hat{S})$ where $\hat{S} = \{x \in \{0,1\}^n : x \neq \hat{x}_i^{(\hat{m}_i)} \ \forall i \in [t]\}$. It replies with $\hat{C}$ to the adversary where $\hat{C}(\cdot) = \mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk}_{\hat{S}}, \cdot)$.
3. **Challenge phase:** The adversary outputs a circuit $\tilde{C}$.
4. **Extraction phase:** The challenger first computes the tuple $\tilde{\mathbf{y}} = (\tilde{C}(h_1), \dots, \tilde{C}(h_d))$. Then, it sets $\tilde{\mathbf{x}} = \big(\tilde{x}_1^{(0)}, \tilde{x}_1^{(1)}, \dots, \tilde{x}_t^{(0)}, \tilde{x}_t^{(1)}\big) \leftarrow \mathsf{PRF.Eval}(\mathsf{k}^*, \tilde{\mathbf{y}})$. For each $i \in [t]$ and $b \in \{0,1\}$, the challenger computes $\tilde{z}_i^{(b)} = \mathsf{TPRF.Test}\big(\mathsf{pp}, \mathsf{tk}, \tilde{C}(\tilde{x}_i^{(b)})\big)$. If there exists some $i \in [t]$ for which $\tilde{z}_i^{(0)} = \tilde{z}_i^{(1)}$, the experiment outputs $\perp$. Otherwise, for each $i \in [t]$, the challenger sets $\tilde{m}_i = 0$ if $\tilde{z}_i^{(0)} = 1$, and $\tilde{m}_i = 1$ otherwise. Finally, the experiment outputs $\tilde{m} \in \{0,1\}^t$.

B.2.1 Proof of Theorem 6.15

We begin by defining our sequence of hybrid experiments:

- **Hybrid H_1:** Same as H_0 (Definition B.1), except the challenger begins by choosing a random function $f : (\{0,1\}^m)^d \to (\{0,1\}^n)^{2t}$ during the setup phase. Then, whenever the challenger needs to evaluate $\mathsf{PRF.Eval}(\mathsf{k}^*, \cdot)$ in the remainder of the experiment, it instead evaluates $f(\cdot)$.

- **Hybrid H_2:** Same as H_1, except at the beginning of the game, the challenger initializes a table $T \leftarrow \varnothing$ to maintain mappings of the form $\mathcal{K} \to (\{0,1\}^n)^{2t}$, where $\mathcal{K}$ is the key-space of the PRF. Then, in the query phase, the challenger responds to the oracle queries as follows:
  - **Marking oracle:** Same as H_1, except on input a message $m \in \{0,1\}^t$ and a PRF key $k \in \mathcal{K}$, if k is already present in T, then the challenger sets $\mathbf{x} = T[k]$ and proceeds as in H_1. Otherwise, it samples $\mathbf{x} \xleftarrow{\text{R}} (\{0,1\}^n)^{2t}$, add the mapping $(k \mapsto \mathbf{x})$ to T. The remainder of the query processing is handled as in H_1.
  - **Challenge oracle:** On input a message $\hat{m} \in \{0,1\}^t$, the challenger first samples a key $\hat{k} \leftarrow \mathsf{TPRF.SampleKey}(\mathsf{pp})$. It then checks to see if $\hat{k}$ is already present in T. If so, the challenger sets $\hat{\mathbf{x}} = T[\hat{k}]$ and proceeds as in H_1. Otherwise, it samples $\hat{\mathbf{x}} \xleftarrow{\text{R}} (\{0,1\}^n)^{2t}$, adds the mapping $(\hat{k} \mapsto \hat{\mathbf{x}})$ to T and continues as in H_1.

  Let Q be the number of marking and challenge queries the adversary makes, $\mathbf{y}_1, \ldots, \mathbf{y}_Q$ be the vectors $\mathbf{y}$ (and $\hat{\mathbf{y}}$) the challenger computes when responding to the marking and challenger oracles during the query phase, and let $k_1, \ldots, k_\ell$ be the keys the adversary provided to the marking oracle (or sampled by the challenge oracle) in those queries. During the extraction phase, if there are distinct indices $\ell_1, \ell_2 \in [Q]$ such that $k_{\ell_1} \neq k_{\ell_2}$, but $\mathbf{y}_{\ell_1} = \mathbf{y}_{\ell_2}$, then the challenger aborts the experiment and outputs Bad_1. Otherwise, the challenger computes $\tilde{\mathbf{y}}$ as in H_1. Then, if $\tilde{\mathbf{y}} = \mathbf{y}_\ell$ for some $\ell \in [Q]$, the challenger sets $\tilde{\mathbf{x}} = T[k_\ell]$. Otherwise, it samples $\tilde{\mathbf{x}} \xleftarrow{\text{R}} (\{0,1\}^n)^{2t}$. The rest of the extraction step is unchanged.

- **Hybrid H_3:** Same as H_2 except when simulating the challenge oracle, the challenger always samples $\hat{\mathbf{x}} \xleftarrow{\text{R}} (\{0,1\}^n)^{2t}$. Moreover, the challenger only adds the mapping $(\hat{k} \mapsto \hat{\mathbf{x}})$ to T at the beginning of the extraction phase (rather than the query phase).

- **Hybrid H_4:** Same as H_3 except during the extraction phase, if there exists some $j \in [d]$ where $\hat{C}(h_j) \neq \mathsf{TPRF.Eval}(\mathsf{pp}, \hat{k}, h_j)$, then the challenger aborts and outputs Bad_2.

- **Hybrid H_5:** Same as H_4 except during the extraction phase, the challenger aborts the experiment and outputs Bad_3 if there exists $j \in [d]$ where $\tilde{C}(h_j) \neq \mathsf{TPRF.Eval}(\mathsf{pp}, \hat{k}, h_j)$. Otherwise, the challenger sets $\tilde{\mathbf{x}} = \hat{\mathbf{x}}$ and continues with the extraction phase as in H_4.

- **Hybrid H_6:** Same as H_5 except during the extraction phase, the challenger also checks (after checking for Bad_1, Bad_2, and Bad_3) whether $\tilde{C}\big(\hat{x}_i^{(b)}\big) = \hat{C}\big(\hat{x}_i^{(b)}\big)$ for all $i \in [t]$ and $b \in \{0,1\}$. If the check passes, the challenger aborts and outputs $\hat{m}$. Otherwise, it follows the same extraction phase of H_5.

- **Hybrid H_7:** Same as H_6 except when the challenger responds to the challenge oracle, it first chooses d distinct random points $\alpha_1, \ldots, \alpha_d \xleftarrow{\text{R}} \{0,1\}^n$ and then sets $\hat{S} = \{\alpha_1, \ldots, \alpha_d\}$ when generating the constrained key $\mathsf{sk}_{\hat{S}}$.

We now proceed in a sequence of lemmas to show that for each consecutive pair of hybrid experiments $\mathsf{H}_\ell, \mathsf{H}_{\ell+1}$, $|\Pr[\mathsf{H}_\ell(\mathcal{A}) \neq \hat{m}] - \Pr[\mathsf{H}_{\ell+1}(\mathcal{A}) \neq \hat{m}]| = \mathsf{negl}(\lambda)$, where $\mathcal{A}$ is an efficient adversary for the unremovability game (Definition 6.8) and $\hat{m} \in \{0,1\}^t$ is the message the adversary submits to the challenge oracle. In the final hybrid H_7, we show that $\Pr[\mathsf{H}_7(\mathcal{A}) \neq \hat{m}] = \mathsf{negl}(\lambda)$, which proves the theorem. Recall that in the unremovability game, the adversary makes exactly one challenge query during the query phase.

Lemma B.2. *If Π_{PRF} is secure, then for all efficient adversaries $\mathcal{A}$,*

$$|\Pr[\mathsf{H}_0(\mathcal{A}) \neq \hat{m}] - \Pr[\mathsf{H}_1(\mathcal{A}) \neq \hat{m}]| = \mathsf{negl}(\lambda).$$

Proof. The only difference between H_0 and H_1 is that invocations of $\mathsf{PRF.Eval}(\mathsf{k}^*, \cdot)$ where $\mathsf{k}^* \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$ are replaced by invocations of $f(\cdot)$ where $f \xleftarrow{\text{R}} \mathrm{Funs}[(\{0,1\}^m)^d, (\{0,1\}^n)^{2t}]$. The claim follows immediately by security of Π_{PRF}. Specifically, any distinguisher for the distributions $\mathsf{H}_0(\mathcal{A})$ and $\mathsf{H}_1(\mathcal{A})$ can be used to distinguish the outputs of the PRF from those of a truly random function. ☐

Lemma B.3. *If* Π_{TPRF} *is key-injective (Definition 4.15), then for all adversaries* $\mathcal{A}$*,*

$$|\Pr[\mathsf{H}_1(\mathcal{A}) \neq \hat{m}] - \Pr[\mathsf{H}_2(\mathcal{A}) \neq \hat{m}]| = \mathsf{negl}(\lambda).$$

Proof. It is easy to see that as long as the vectors $\mathbf{y}$ and $\hat{\mathbf{y}}$ are unique (for distinct keys) in the marking and challenge queries, then H_1 and H_2 are identically distributed (in this case, the procedure in H_2 just corresponds to a *lazy sampling* of the random function f). To show that the vectors $\mathbf{y}$ and $\hat{\mathbf{y}}$ for different keys are unique with overwhelming probability, we define a sequence of intermediate hybrid experiments:

- **Hybrid** $\mathsf{H}_{1,0}$**:** Same as H_1.
- **Hybrid** $\mathsf{H}_{1,\ell}$**:** Same as H_1 except at the beginning of the game, the challenger initializes a table $T \leftarrow \varnothing$ to maintain mappings of the form $\mathcal{K} \to (\{0,1\}^n)^{2t}$. For the first ℓ marking or challenge queries, the challenger responds according to the specification in H_2 (updating the table T accordingly). Let $k_1, \ldots, k_\ell$ be the keys appearing in the first ℓ queries, and let $(\mathbf{y}_1, \mathbf{x}_1), \ldots, (\mathbf{y}_\ell, \mathbf{x}_\ell)$ be the vectors the challenger uses to answer the first ℓ queries. When answering all of the subsequent queries and in the extraction phase, after the challenger computes $\mathbf{y}$ (analogously, $\hat{\mathbf{y}}$ or $\tilde{\mathbf{y}}$), it first checks to see if $\mathbf{y} = \mathbf{y}_{\ell^*}$ for some $\ell^* \in [\ell]$ (choosing one arbitrarily if there are multiple). If so, it sets $\mathbf{x} = \mathbf{x}_{\ell^*}$ when answering the query. Otherwise, it sets $\mathbf{x} = f(\mathbf{y})$ as in H_1. If there exist distinct $\ell_1, \ell_2 \in [\ell]$ where $k_{\ell_1} \neq k_{\ell_2}$, but $\mathbf{y}_{\ell_1} = \mathbf{y}_{\ell_2}$, then during the extraction phase, the challenger aborts and outputs Bad_1.

Let Q be the number of marking or challenge queries the adversary makes. We now show that

$$\mathsf{H}_1(\mathcal{A}) \equiv \mathsf{H}_{1,0}(\mathcal{A}) \overset{s}{\approx} \mathsf{H}_{1,1}(\mathcal{A}) \overset{s}{\approx} \cdots \overset{s}{\approx} \mathsf{H}_{1,Q}(\mathcal{A}) \equiv \mathsf{H}_2(\mathcal{A})$$

By definition, $\mathsf{H}_1 \equiv \mathsf{H}_{1,0}$ and $\mathsf{H}_{1,Q} \equiv \mathsf{H}_2$, so it suffices to show that for all $\ell \in [Q]$, $\mathsf{H}_{1,\ell-1}(\mathcal{A}) \overset{s}{\approx} \mathsf{H}_{1,\ell}(\mathcal{A})$. First, we note that the behavior of $\mathsf{H}_{1,\ell-1}$ and $\mathsf{H}_{1,\ell}$ differ only on how the ℓ^{th} query is handled. In both experiments, the adversary's view after the first $\ell - 1$ queries is *independent* of the query points $h_1, \ldots, h_d$ (since the vectors $\mathbf{x}$ as well as $\hat{\mathbf{x}}$ that occur in the first $\ell - 1$ queries are chosen independently and uniformly of $h_1, \ldots, h_d$). Thus, in hybrids $\mathsf{H}_{1,\ell-1}$ and $\mathsf{H}_{1,\ell}$, the challenger can defer the sampling of $h_1, \ldots, h_d$ until *after* the adversary has committed to its ℓ^{th} query. Let $k_1, \ldots, k_\ell$ be the keys the adversary submits in its first ℓ queries. Since $h_1, \ldots, h_d$ are sampled after the adversary has chosen $k_1, \ldots, k_\ell$, we conclude that $h_1, \ldots, h_d$ are distributed uniformly and independently of $k_1, \ldots, k_\ell$ (as well as the public parameter pp). There are now two possibilities to consider

- If $k_\ell = k_{\ell^*}$ for some $\ell^* < \ell$, then $\mathbf{y}_\ell = \mathbf{y}_{\ell^*}$. In $\mathsf{H}_{1,\ell-1}$, the adversary sets $\mathbf{x} = \mathbf{x}_{\ell^*}$ when answering the query. Note that this holds only if there does not exist two indices $\ell_1, \ell_2 < \ell$ where $\mathbf{y}_\ell = \mathbf{y}_{\ell_1} = \mathbf{y}_{\ell_2}$, but $k_{\ell_1} \neq k_{\ell_2}$. If this were to happen, then both $\mathsf{H}_{1,\ell-1}$ and $\mathsf{H}_{1,\ell}$ output Bad_1. Otherwise in hybrid $\mathsf{H}_{1,\ell}$, since $k_\ell = k_{\ell^*}$, the challenger sets $\mathbf{x}_\ell = T[k_\ell] = \mathbf{x}_{\ell^*}$. In either case, the challenger's response to the ℓ^{th} query is identically distributed in $\mathsf{H}_{1,\ell-1}$ and $\mathsf{H}_{1,\ell}$.
- If $k_\ell \neq k_{\ell^*}$ for all $\ell^* \neq \ell$, then by key injectivity, for all $\ell^* < \ell$ and all $j \in [d]$,

  $$\Pr[\mathsf{TPRF.Eval}(k_\ell, h_j) = \mathsf{TPRF.Eval}(k_{\ell^*}, h_j)] = \mathsf{negl}(\lambda),$$

  where the probability is taken over the randomness used to sample the parameters in $\mathsf{WM.Setup}$. We conclude that for all $\ell^* < \ell$,

  $$\Pr[\forall j \in [d] \colon \mathsf{TPRF.Eval}(k_\ell, h_j) = \mathsf{TPRF.Eval}(k_{\ell^*}, h_j)] = \mathsf{negl}(\lambda).$$

  Union bounding over all $\ell - 1 \leq Q = \mathsf{poly}(\lambda)$ queries, we conclude that with overwhelming probability, $\mathbf{y}_\ell \neq \mathbf{y}_{\ell^*}$ for all $\ell^* < \ell$. This means that in $\mathsf{H}_{1,\ell-1}$, the vector $\mathbf{x}_\ell$ used to answer the ℓ^{th} query is given by the output of $f(\mathbf{y}_\ell)$, where f is a truly random function (and independent of the challenger's responses in all previous queries). Thus, $\mathbf{x}_\ell$ in $\mathsf{H}_{1,\ell-1}$ is independently and uniformly distributed over $(\{0,1\}^n)^{2t}$, which is precisely the distribution from which it is sampled in $\mathsf{H}_{1,\ell}$. Thus, the challenger's responses in the first ℓ queries are *identically* distributed in $\mathsf{H}_{1,\ell-1}$ and $\mathsf{H}_{1,\ell}$.

Finally, we note that the probability that $\mathsf{H}_{1,\ell}$ outputs Bad_1 can only be *negligibly* greater than that in $\mathsf{H}_{1,\ell-1}$. To see this, observe that if there exists $\ell_1, \ell_2 \in [\ell-1]$ such that $\mathbf{y}_{\ell_1} = \mathbf{y}_{\ell_2}$, then both $\mathsf{H}_{1,\ell-1}$ and $\mathsf{H}_{1,\ell}$ output Bad_1. The only scenario where $\mathsf{H}_{1,\ell}$ outputs Bad_1 (and $\mathsf{H}_{1,\ell-1}$ does not) is if $\mathbf{y}_\ell = \mathbf{y}_{\ell^*}$ and $k_\ell \neq k_{\ell^*}$ for some $\ell^* < \ell$. But by the key-injectivity argument above, this happens with negligible probability. Conditioned on Bad_1 not happening, the outputs of experiments $\mathsf{H}_{1,\ell-1}$ and $\mathsf{H}_{1,\ell}$ are identically distributed. We conclude that $\mathsf{H}_{1,\ell-1}(\mathcal{A}) \stackrel{s}{\approx} \mathsf{H}_{1,\ell}(\mathcal{A})$ for all $\ell \in [Q]$. This proves the claim. $\square$

Lemma B.4. *If Π_{TPRF} satisfies selective constrained pseudorandomness (Definition 4.10), then for all efficient adversaries $\mathcal{A}$,*

$$|\Pr[\mathsf{H}_2(\mathcal{A}) \neq \hat{m}] - \Pr[\mathsf{H}_3(\mathcal{A}) \neq \hat{m}]| = \mathsf{negl}(\lambda).$$

Proof. By construction, hybrids H_2 and H_3 are identical experiments as long as the adversary never queries the marking oracle on the key $\hat{k}$ (the key the challenger samples during the challenge phase). Thus, the only possible way the adversary can obtain a nonzero advantage in distinguishing hybrids H_2 and H_3 is if it is able to query the marking oracle on $\hat{k}$, or equivalently, "guess" the master secret key sampled by the challenger given only the public parameters and the constrained key. Certainly, this completely breaks (selective) constrained pseudorandomness. More formally, let $\mathcal{A}$ be an efficient adversary that is able to distinguish H_2 from H_3 with some non-negligible probability ε. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks the (selective) constrained pseudorandomness of Π_{TPRF} with advantage ε/Q where Q is the number of marking oracle queries $\mathcal{A}$ makes during the query phase. Algorithm $\mathcal{B}$ works as follows:

1. At the beginning of the game, $\mathcal{B}$ samples t points $\hat{x}_1, \ldots, \hat{x}_t \xleftarrow{\text{R}} \{0,1\}^n$. It sends the t-puncturing set $\hat{S} = \{x \in \{0,1\}^n : x \neq \hat{x}_i \; \forall i \in [t]\}$ to the selective constrained pseudorandomness challenger. The constrained pseudorandomness challenger then samples the public parameters $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{TPRF.Setup}(1^\lambda)$ and a secret key $\mathsf{msk} \leftarrow \mathsf{TPRF.SampleKey}(\mathsf{pp})$. It constructs the constrained key $\mathsf{sk}_{\hat{S}} \leftarrow \mathsf{TPRF.Constrain}(\mathsf{pp}, \mathsf{msk}, \hat{S})$. The challenger gives pp and $\mathsf{sk}_{\hat{S}}$ to $\mathcal{B}$.

2. Algorithm $\mathcal{B}$ starts running $\mathcal{A}$ and starts simulating hybrids H_2 and H_3 for $\mathcal{A}$. In the setup phase, $\mathcal{B}$ gives pp to $\mathcal{A}$. The other components of the setup phase are simulated exactly as described in H_2 and H_3. Note that simulating the evaluations of the truly random function f can be done by lazily sampling the outputs of f on an as-needed basis.

3. During the query phase, whenever $\mathcal{A}$ makes an marking oracle query, $\mathcal{B}$ simulates the response exactly as described in H_3. This is possible because answering the marking queries only requires knowledge of the public parameters pp. When $\mathcal{A}$ makes it challenge query, $\mathcal{B}$ response with the constrained key $\mathsf{sk}_{\hat{S}}$ it received from the constrained pseudorandomness challenger.

4. Let $k_1, \ldots, k_Q \in \mathcal{K}$ be the keys $\mathcal{A}$ submitted to the marking oracle during the query phase. At the end of the query phase, $\mathcal{B}$ chooses an index $i \xleftarrow{\text{R}} [Q]$, and computes $y = \mathsf{TPRF.Eval}(\mathsf{pp}, k_i, \hat{x}_1)$. In addition, it makes a challenge oracle query to the constrained pseudorandomness challenger at the punctured point $\hat{x}_1$. The constrained pseudorandomness challenger responds with a value $\hat{y}$. Finally, $\mathcal{B}$ outputs 1 if $y = \hat{y}$ and 0 otherwise.

By construction, $\mathcal{B}$ *perfectly* simulates H_3 for $\mathcal{A}$. Here, the key msk sampled by the constrained pseudorandomness challenger plays the role of the key sampled by the challenger in response to the challenge oracle in H_3. Now, as stated above, H_2 and H_3 are identical experiments unless the adversary queries the marking oracle on msk during the query phase. Since $\mathcal{A}$ is able to distinguish H_2 from H_3 with probability ε, it must be the case that with probability ε, on one of its marking oracle queries, it submits msk. Moreover, up until making this query, $\mathcal{B}$ perfectly simulates *both* H_2 and H_3 for $\mathcal{A}$. This means that with probability ε, one of the keys $k_1, \ldots, k_Q$ that appears in the marking oracle queries of $\mathcal{A}$ is actually msk. We consider two cases, depending on whether the constrained pseudorandomness challenger replies with the real value of the PRF or a random value in response to the challenger queries:

- Suppose the constrained pseudorandomness challenger replies with the value $\mathsf{TPRF.Eval}(\mathsf{pp}, \mathsf{msk}, \hat{x}_1)$. With probability ε/Q, we have that $k_i = \mathsf{msk}$, in which case $y = \hat{y}$, and $\mathcal{B}$ outputs 1. Thus, in this case, $\mathcal{B}$ outputs 1 with probability at least ε/Q.

- If the constrained pseudorandomness challenger replies with a random value $\hat{y} \xleftarrow{\mathrm{R}} \{0,1\}^m$, then $y = \hat{y}$ with probability $1/2^m = \mathsf{negl}(\lambda)$.

Thus, $\mathcal{B}$ is able to break constrained pseudorandomness of Π_{TPRF} with advantage $\varepsilon/Q - \mathsf{negl}(\lambda)$. Since ε is non-negligible and $Q = \mathsf{poly}(\lambda)$, this is non-negligible. The claim follows. $\square$

Lemma B.5. *If Π_{TPRF} satisfies (selective) evaluation correctness, then for all adversaries $\mathcal{A}$,*

$$|\Pr[\mathsf{H}_3(\mathcal{A}) \neq \hat{m}] - \Pr[\mathsf{H}_4(\mathcal{A}) \neq \hat{m}]| = \mathsf{negl}(\lambda).$$

Proof. We show that for all adversaries $\mathcal{A}$, $\mathsf{H}_4(\mathcal{A})$ outputs Bad_2 with negligible probability. By definition, we have that $\hat{C}(\cdot) = \mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk}_{\hat{S}}, \cdot)$. In H_3 and H_4, the points $h_1, \ldots, h_d$ are sampled uniformly from the domain $\{0,1\}^n$ of Π_{TPRF} and independently of all other parameters. By evaluation correctness of Π_{TPRF} and Remark 4.6, we conclude that for all $j \in [d]$, $\Pr[\hat{C}(h_j) \neq \mathsf{TPRF.Eval}(\mathsf{pp}, \hat{k}, h_j)] = \mathsf{negl}(\lambda)$. Since $d = \mathsf{poly}(\lambda)$, we conclude that H_4 outputs Bad_2 with negligible probability. Since H_3 and H_4 are identical experiments with the only exception being H_4 could output Bad_2, we conclude that $\mathsf{H}_3(\mathcal{A})$ and $\mathsf{H}_4(\mathcal{A})$ in the two experiments are statistically indistinguishable, and the claim follows. $\square$

Lemma B.6. *For all unremoving-admissible adversaries $\mathcal{A}$ (Definition 6.8),*

$$|\Pr[\mathsf{H}_4(\mathcal{A}) \neq \hat{m}] - \Pr[\mathsf{H}_5(\mathcal{A}) \neq \hat{m}]| = \mathsf{negl}(\lambda).$$

Proof. We show that the output distributions $\mathsf{H}_4(\mathcal{A})$ and $\mathsf{H}_5(\mathcal{A})$ are statistically indistinguishable. Since the conditions for outputting Bad_1 and Bad_2 are identical in H_4 and H_5, it suffices to only reason about the case where Bad_1 and Bad_2 are not set. Our proof consists of two pieces.

- We first show that H_4 outputs Bad_3 with negligible probability assuming $\mathcal{A}$ is unremoving-admissible. Observe that in H_5, the challenger's behavior (and correspondingly, the adversary's view) during the query phase is independent of $h_1, \ldots, h_d$. Thus, in H_5, it is equivalent for the challenger to defer sampling $h_1, \ldots, h_d$ until the extraction phase, and in particular, *after* the adversary has output its challenge circuit $\tilde{C}$. By unremoving-admissibility, $\tilde{C} \sim_f \hat{C}$ where $1/f = \mathsf{negl}(\lambda)$. Since for all $j \in [d]$, h_j is sampled uniformly from $\{0,1\}^n$ and independent of both $\hat{C}$ and $\tilde{C}$, we have that $\Pr[\tilde{C}(h_j) \neq \hat{C}(h_j)] \leq 1/f = \mathsf{negl}(\lambda)$. Next, $d = \mathsf{poly}(\lambda)$, so we conclude via a union bound that for all $j \in [d]$, $\tilde{C}(h_j) = \hat{C}(h_j)$. Finally, since Bad_2 is not set, we have that $\hat{C}(h_j) = \mathsf{TPRF.Eval}(\mathsf{pp}, \hat{k}, h_j)$ for all $j \in [d]$, and so, H_5 outputs Bad_3 with negligible probability.

- To conclude the proof, we show that the distributions of outputs in the extraction phases of H_4 and H_5 are statistically indistinguishable. First, we note that the condition for outputting Bad_3 depends only on the adversary's output in the challenge phase. By construction, the adversary's outputs in the challenge phase of H_4 and H_5 are identically distributed. By our previous argument, the condition for outputting Bad_3 is satisfied with negligible probability in H_5, and so, the same condition is satisfied with negligible probability in H_4 (otherwise, the condition associated with Bad_3 can be used to distinguish the adversary's output in the challenge phase of H_4 and H_5). Thus, in H_4, $\tilde{C}(h_j) = \mathsf{TPRF.Eval}(\mathsf{pp}, \hat{k}, h_j)$ for all $j \in [d]$ with overwhelming probability. This means that for all $j \in [d]$,

$$\tilde{y}_j = \tilde{C}(h_j) = \mathsf{TPRF.Eval}(\mathsf{pp}, \hat{k}, h_j) = \hat{y}_j,$$

  or equivalently, $\tilde{\mathbf{y}} = \hat{\mathbf{y}}$. This means that in the extraction step of H_4, the challenger sets $\tilde{\mathbf{x}} = \hat{\mathbf{x}}$ (by assumption, it does not output Bad_1) with overwhelming probability. But this is precisely the behavior in H_5. Since the rest of the extraction step in H_4 and H_5 is the same, we conclude that the distribution of outputs in H_4 is statistically indistinguishable from that in H_5. $\square$

Lemma B.7. *If Π_{TPRF} satisfies (selective) verification correctness, then for all adversaries $\mathcal{A}$,*

$$|\Pr[\mathsf{H}_5(\mathcal{A}) \neq \hat{m}] - \Pr[\mathsf{H}_6(\mathcal{A}) \neq \hat{m}]| = \mathsf{negl}(\lambda).$$

Proof. We show that the distributions $\mathsf{H}_5(\mathcal{A})$ and $\mathsf{H}_6(\mathcal{A})$ are statistically indistinguishable. Hybrids H_5 and H_6 are identical experiments unless $\tilde{C}(\hat{x}_i^{(b)}) = \hat{C}(\hat{x}_i^{(b)})$ for all $i \in [t]$ and $b \in \{0,1\}$. We consider the output in H_5 when this is the case. Without loss of generality, we just consider the case where Bad_3 does not occur. In this case, the challenger sets $\tilde{\mathbf{x}} = \hat{\mathbf{x}}$. It follows that $\tilde{C}(\tilde{x}_i^{(b)}) = \tilde{C}(\hat{x}_i^{(b)}) = \hat{C}(\hat{x}_i^{(b)})$ for all $i \in [t]$ and $b \in \{0,1\}$. Then,

$$\tilde{z}_i^{(b)} = \mathsf{TPRF.Test}\big(\mathsf{pp}, \mathsf{tk}, \tilde{C}(\tilde{x}_i^{(b)})\big) = \mathsf{TPRF.Test}\big(\mathsf{pp}, \mathsf{tk}, \hat{C}(\hat{x}_i^{(b)})\big).$$

By definition, $\hat{C}(\hat{x}_i^{(b)}) = \mathsf{TPRF.ConstrainEval}\big(\mathsf{pp}, \mathsf{sk}_{\hat{S}}, \hat{x}_i^{(b)}\big)$. For each $i \in [t]$, we have that $\hat{x}_i^{(\hat{m}_i)} \in \hat{S}$, so by verification correctness of Π_{TPRF}, $\mathsf{TPRF.Test}\big(\mathsf{pp}, \mathsf{tk}, \tilde{C}(\tilde{x}_i^{(\hat{m}_i)})\big) = 1$ with overwhelming probability. On the other hand, since $\hat{x}_i^{(1-\hat{m}_i)} \notin \hat{S}$ (with overwhelming probability), and moreover, $\hat{x}_i^{(1-\hat{m}_i)}$ is independently and uniformly random over $\{0,1\}^n$, with overwhelming probability, $\mathsf{TPRF.Test}\big(\mathsf{pp}, \mathsf{tk}, \tilde{C}(\tilde{x}_i^{(1-\hat{m}_i)})\big) = 0$. Thus, with overwhelming probability, the challenger sets $\tilde{m}_i = \hat{m}_i$ in H_5. Since $t = \mathsf{poly}(\lambda)$, we have that with overwhelming probability $\tilde{m} = \hat{m}$. We conclude that the output of H_5 when the condition is satisfied is $\hat{m}$ with overwhelming probability. □

Lemma B.8. *If Π_{TPRF} is selectively-private (Definition 4.13), then for all efficient unremoving-admissible (Definition 6.8) adversaries $\mathcal{A}$,*

$$|\Pr[\mathsf{H}_6(\mathcal{A}) \neq \hat{m}] - \Pr[\mathsf{H}_7(\mathcal{A}) \neq \hat{m}]| = \mathsf{negl}(\lambda).$$

Moreover, under the same assumptions, $\Pr[\mathsf{H}_7(\mathcal{A}) \neq m] = \mathsf{negl}(\lambda)$.

Proof. First, we show that $\Pr[\mathsf{H}_7(\mathcal{A}) \neq \hat{m}] = \mathsf{negl}(\lambda)$. It suffices to consider the case where H_7 does not output one of the flags Bad_1, Bad_2, or Bad_3, since we previously showed in Lemmas B.3 through B.7 that each hybrid outputs these flags with negligible probability. In H_7, the setup and query phases are completely independent of the points $\hat{x}_i^{(b)}$ for all $i \in [t]$ and $b \in \{0,1\}$. Thus, it is equivalent to sample $\hat{x}_i^{(b)}$ at the extraction phase, *after* the adversary has output its challenge circuit $\tilde{C}$. Since $\hat{x}_i^{(b)}$ are sampled uniformly from $\{0,1\}^n$ and independently of $\tilde{C}$, by unremoving-admissibility of $\mathcal{A}$, we have that for all $i \in [t]$, $\Pr[\tilde{C}(\hat{x}_i^{(b)}) \neq \hat{C}(\hat{x}_i^{(b)})] = 1/f = \mathsf{negl}(\lambda)$. Since $t = \mathsf{poly}(\lambda)$, it follows that with overwhelming probability, $\tilde{C}(\hat{x}_i^{(b)}) = \hat{C}(\hat{x}_i^{(b)})$ for all $i \in [t]$ and $b \in \{0,1\}$. Thus, with overwhelming probability, H_7 outputs $\hat{m}$.

Now, suppose there exists an efficient adversary $\mathcal{A}$ such that $|\Pr[\mathsf{H}_6(\mathcal{A}) \neq \hat{m}] - \Pr[\mathsf{H}_7(\mathcal{A}) \neq \hat{m}]|$ is non-negligible. Since $\Pr[\mathsf{H}_7(\mathcal{A}) \neq \hat{m}] = \mathsf{negl}(\lambda)$, this must mean that $\Pr[\mathsf{H}_6(\mathcal{A}) \neq \hat{m}] = \varepsilon$ for some non-negligible ε. We now use $\mathcal{A}$ to build an efficient adversary $\mathcal{B}$ that can break the (selective) privacy of Π_{TPRF} with the same advantage ε. Algorithm $\mathcal{B}$ works as follows:

1. At the beginning of the game, $\mathcal{B}$ chooses values $\hat{x}_1, \ldots, \hat{x}_t \xleftarrow{R} \{0,1\}^n$ and $\alpha_1, \ldots, \alpha_t \xleftarrow{R} \{0,1\}^n$. It then constructs two sets $S_0 = \{x \in \{0,1\}^n : x \neq \hat{x}_i \ \forall i \in [t]\}$ and $S_1 = \{x \in \{0,1\}^n : x \neq \alpha_i \ \forall i \in [t]\}$. Algorithm $\mathcal{B}$ submits sets S_0 and S_1 to the challenger.
2. The privacy challenger replies to $\mathcal{B}$ with the public parameters pp for Π_{TPRF} and a constrained key sk_β where $\beta \in \{0,1\}$.
3. Algorithm $\mathcal{B}$ starts running $\mathcal{A}$. In the setup phase, $\mathcal{B}$ chooses the watermarking secret key components $h_1, \ldots, h_d \xleftarrow{R} \{0,1\}^n$ and $k \xleftarrow{R} \mathcal{K}$ for itself. It gives pp to $\mathcal{A}$ in the setup phase.
4. In the query phase, $\mathcal{B}$ answers the queries as follows:
   - **Marking oracle:** Algorithm $\mathcal{B}$ answers these queries exactly as in H_6 and H_7. This is possible since none of the queries depend on knowing tk, and algorithm $\mathcal{B}$ knows all of the other components of the watermarking secret key msk.

- **Challenge oracle:** On input the challenge message $\hat{m} \in \{0,1\}^t$, algorithm $\mathcal{B}$ sets $\hat{x}_i^{(\hat{m}_i)} = \hat{x}_i$ and samples $\hat{x}_i^{(1-\hat{m}_i)} \xleftarrow{\text{R}} \{0,1\}^n$. It replies with $\hat{C}(\cdot) = \mathsf{TPRF.ConstrainEval}(\mathsf{pp}, \mathsf{sk}_\beta, \cdot)$.

5. After the adversary finishes making its oracle queries, it outputs its challenge circuit $\tilde{C}$. Algorithm $\mathcal{B}$ then simulates the extraction phase as follows. First, it checks whether there exists $i \in [t]$ and $b \in \{0,1\}$ such that $\tilde{C}(\hat{x}_i^{(b)}) \neq \hat{C}(\hat{x}_i^{(b)})$. If so, $\mathcal{B}$ halts the experiment and outputs 1. Otherwise, $\mathcal{B}$ outputs 0.

First, observe that in the reduction, the values $\hat{x}_i$ play the role of $\hat{x}_i^{(\hat{m}_i)}$. We now consider the two cases $\beta = 0$ and $\beta = 1$.

- If $\beta = 0$, then $\mathcal{B}$ perfectly simulates H_6 for $\mathcal{A}$. In H_6, if $\tilde{C}(\hat{x}_i^{(b)}) = \hat{C}(\hat{x}_i^{(b)})$ for all $i \in [t]$ and $b \in \{0,1\}$, then by construction, $\mathsf{H}_6(\mathcal{A})$ outputs $\hat{m}$. Since $\Pr[\mathsf{H}_6(\mathcal{A}) \neq \hat{m}] = \varepsilon$, with probability at least ε, there exists some $i \in [t]$ and $b \in \{0,1\}$ for which $\tilde{C}(\hat{x}_i^{(b)}) \neq \hat{C}(\hat{x}_i^{(b)})$. Thus, with probability ε, $\mathcal{B}$ outputs 1.

- If $\beta = 1$, then $\mathcal{B}$ perfectly simulates H_7 for $\mathcal{A}$. We previously showed that in hybrid H_7, $\tilde{C}(\hat{x}_i^{(b)}) = \hat{C}(\hat{x}_i^{(b)})$ for all $i \in [t]$ and $b \in \{0,1\}$ with overwhelming probability. Thus, in this case, $\mathcal{B}$ outputs 1 with negligible probability.

We conclude that $\mathcal{B}$ is able to win the selective privacy game for Π_{TPRF} with advantage $\varepsilon - \mathsf{negl}(\lambda)$, which is non-negligible, as required. ☐

Combining Lemmas B.2 through B.8, we conclude that as long as Π_{PRF} is secure and Π_{TPRF} is a selectively-private translucent t-puncturable PRF that satisfies key injectivity, then the watermarking scheme Π_{WM} is unremovable. ☐

B.2.2 Proof of Theorem 6.16

We begin by defining our sequence of hybrid experiments:

- **Hybrid H_1:** This is the same hybrid as H_1 from the proof of Theorem 6.15.
- **Hybrid H_2:** This is the same hybrid as H_2 from the proof of Theorem 6.15.
- **Hybrid H_3:** Same as H_2 except in the extraction step, after computing the tuple $\tilde{\mathbf{y}} = (\tilde{C}(h_1), \dots, \tilde{C}(h_d))$, the challenger aborts the experiment and outputs Bad_2 if $\tilde{\mathbf{y}} \in \mathcal{Z}$ (where $\mathcal{Z}$ is the set of tuples $\mathbf{y}$ that appeared in a marking oracle query). Otherwise, it proceeds as in H_2.

As in the proof of Theorem 6.15, we proceed in a sequence of lemmas and show that for each consecutive pair of hybrid experiments $\mathsf{H}_\ell, \mathsf{H}_{\ell+1}$, it is the case that $|\Pr[\mathsf{H}_\ell(\mathcal{A}) \neq \bot] - \Pr[\mathsf{H}_{\ell+1}(\mathcal{A}) \neq \bot]| = \mathsf{negl}(\lambda)$, where $\mathcal{A}$ is an efficient adversary for the δ-unforgeability game (Definition 6.9). Finally, in the final hybrid H_3, we show that $\Pr[\mathsf{H}_4(\mathcal{A}) \neq \bot] = \mathsf{negl}(\lambda)$, which proves the theorem. Recall that in the δ-unforgeability game, the adversary does not make any queries to the challenge oracle.

Lemma B.9. *If Π_{PRF} is a secure PRF, then for all efficient adversaries $\mathcal{A}$,*

$$|\Pr[\mathsf{H}_1(\mathcal{A}) \neq \bot] - \Pr[\mathsf{H}_2(\mathcal{A}) \neq \bot]| = \mathsf{negl}(\lambda).$$

Proof. Follows by the exact same argument as that given in the proof of Lemma B.2. ☐

Lemma B.10. *If Π_{TPRF} satisfies key injectivity (Definition 4.15), then for all adversaries $\mathcal{A}$,*

$$|\Pr[\mathsf{H}_1(\mathcal{A}) \neq \hat{m}] - \Pr[\mathsf{H}_2(\mathcal{A}) \neq \hat{m}]| = \mathsf{negl}(\lambda).$$

Proof. Follows by the exact same argument as that given in the proof of Lemma B.3. ☐

Lemma B.11. *If Π_{TPRF} satisfies evaluation correctness, then for all δ-unforging-admissible adversaries $\mathcal{A}$ (Definition 6.9) where $\delta = 1/\mathsf{poly}(\lambda)$,*

$$|\Pr[\mathsf{H}_2(\mathcal{A}) \neq \bot] - \Pr[\mathsf{H}_3(\mathcal{A}) \neq \bot]| = \mathsf{negl}(\lambda).$$

Proof. We show that the distributions $\mathsf{H}_2(\mathcal{A})$ and $\mathsf{H}_3(\mathcal{A})$ are statistically indistinguishable. By construction, the adversary's view in the setup and query phases of H_2 and H_3 are identically distributed. To show the lemma, it suffices to argue that H_3 does not output Bad_2 in the extraction phase. Let $Q = \mathsf{poly}(\lambda)$ be the number of marking queries the adversary made and for $\ell \in [Q]$, let k_ℓ be the PRF key the adversary submitted to the marking oracle on the ℓ^{th} query. For $\ell \in [Q]$, let T_ℓ be the set of points on which $\tilde{C}(\cdot)$ and $\mathsf{TPRF.Eval}(\mathsf{pp}, k_\ell, \cdot)$ differ, where $\tilde{C}$ is the circuit output by the adversary at the end of the challenge phase. Since $\mathcal{A}$ is δ-unforging-admissible, we have that $|T_\ell| / 2^n \geq \delta$. Next, we note that in H_3, the query phase does not depend on $h_1, \ldots, h_d$. Thus, we can defer the sampling of $h_1, \ldots, h_d$ until the extraction phase, *after* the adversary has output its challenge circuit $\tilde{C}$. Since each of the h_j is drawn uniformly and independently from $\{0,1\}^n$, we have for all $j \in [d]$ and $\ell \in [Q]$, $\Pr[h_j \in T_\ell] = |T_\ell| / 2^n \geq \delta$. It follows that for all $\ell \in [Q]$

$$\Pr[\forall j \in [d] : h_j \notin T_\ell] = \left(1 - \frac{|T_\ell|}{2^n}\right)^d \leq (1-\delta)^{\lambda/\delta} \leq e^{-\lambda} = \mathsf{negl}(\lambda),$$

where we have used the fact that $d = \lambda/\delta$ and $\delta = 1/\mathsf{poly}(\lambda)$. Since this holds for all $\ell \in [Q]$, we conclude that with overwhelming probability, it is the case that for all $\ell \in [Q]$, there exists some $j \in [d]$ such that $h_j \in T_\ell$, or equivalently, that $\tilde{C}(h_j) \neq \mathsf{TPRF.Eval}(\mathsf{pp}, k_\ell, h_j)$. By construction of the marking algorithm this means that $\tilde{\mathbf{y}} \neq \mathbf{y}_\ell$ for all $\ell \in [Q]$. We conclude that $\tilde{\mathbf{y}} \notin \mathcal{Z}$, and so H_3 outputs Bad_2 with negligible probability. ☐

Lemma B.12. *For all adversaries $\mathcal{A}$, $\Pr[\mathsf{H}_3(\mathcal{A}) \neq \bot] = \mathsf{negl}(\lambda)$.*

Proof. It suffices to consider the case where H_3 does not output Bad_1 and Bad_2 (as argued in Lemmas B.10 and B.11, these events occur with negligible probability). Conditioned on H_3 not outputting Bad_2, the test vector $\tilde{\mathbf{x}}$ is sampled uniformly and independently from $(\{0,1\}^n)^{2t}$ *after* the adversary has output its challenge circuit $\tilde{C}$. Now, for each $i \in [t]$ and $b \in \{0,1\}$, the extraction algorithm computes $\tilde{z}_i^{(b)} = \mathsf{TPRF.Test}(\mathsf{pp}, \mathsf{tk}, \tilde{C}(\tilde{x}_i^{(b)}))$. Since the test points $\tilde{x}_i^{(0)}$ and $\tilde{x}_i^{(1)}$ are chosen uniformly and independently from $\{0,1\}^n$ after the adversary has committed to $\tilde{C}$, we have that $\Pr[\tilde{z}_i^{(0)} \neq \tilde{z}_i^{(1)}] \leq 1/2$ for all $i \in [t]$, irrespective of $\tilde{C}$. Since $t = \omega(\log \lambda)$, with overwhelming probability, there exists some $i \in [t]$ where $\tilde{z}_i^{(0)} = \tilde{z}_i^{(1)}$, in which case, the extraction algorithm outputs $\bot$. ☐

Combining Lemmas B.9 through B.12, we conclude that as long as Π_{PRF} is secure, and Π_{TPRF} is a translucent t-puncturable PRF that satisfies key injectivity, the watermarking scheme Π_{WM} is δ-unforgeable. ☐