Private Puncturable PRFs From Standard Lattice Assumptions

Dan Boneh
Stanford University
dabo@cs.stanford.edu

Sam Kim*
Stanford University
skim13@cs.stanford.edu

Hart Montgomery
Fujitsu Laboratories of America
hmontgomery@us.fujitsu.com

Abstract

A puncturable pseudorandom function (PRF) has a master key k that enables one to evaluate the PRF at all points of the domain, and has a punctured key k_x that enables one to evaluate the PRF at all points but one. The punctured key k_x reveals no information about the value of the PRF at the punctured point x. Punctured PRFs play an important role in cryptography, especially in applications of indistinguishability obfuscation. However, in previous constructions, the punctured key k_x completely reveals the punctured point x: given k_x it is easy to determine x. A *private* puncturable PRF is one where k_x reveals nothing about x. This concept was defined by Boneh, Lewi, and Wu, who showed the usefulness of private puncturing, and gave constructions based on multilinear maps. The question is whether private puncturing can be built from a standard (weaker) cryptographic assumption.

We construct the first privately puncturable PRF from standard lattice assumptions, namely from the hardness of learning with errors (LWE) and 1 dimensional short integer solutions (1D-SIS), which have connections to worst-case hardness of general lattice problems. Our starting point is the (non-private) PRF of Brakerski and Vaikuntanathan. We introduce a number of new techniques to enhance this PRF, from which we obtain a privately puncturable PRF. In addition, we also study the simulation based definition of private constrained PRFs for general circuits, and show that the definition is not satisfiable.

1 Introduction

A pseudorandom function (PRF) [GGM86] is a function $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ that can be computed by a deterministic polynomial time algorithm: on input $(k, x) \in \mathcal{K} \times \mathcal{X}$ the algorithm outputs $F(k, x) \in \mathcal{Y}$. The PRF F is said to be a constrained PRF [BW13, KPTZ13, BGI14] if one can derive constrained keys from the master PRF key k. Each constrained key k_g is associated with a predicate $g\colon \mathcal{X} \to \{0, 1\}$, and this k_g enables one to evaluate $F(k, x)$ for all $x \in \mathcal{X}$ for which $g(x) = 1$, but at no other points of $\mathcal{X}$. A constrained PRF is secure if given constrained keys for predicates $g_1, \ldots, g_Q$, of the adversary's choosing, the adversary cannot distinguish the PRF from a random function at points not covered by the given keys, namely at points x where $g_1(x) = \cdots = g_Q(x) = 0$. We review the precise definition in Section 3.

The simplest constraint, called a *puncturing constraint*, is a constraint that enables one to evaluate the PRF at all points except one. For $x \in \mathcal{X}$ we denote by k_x a *punctured key* that lets one evaluate the PRF at all points in $\mathcal{X}$, except for the punctured point x. Given the key k_x, the adversary should be unable to distinguish $F(k, x)$ from a random element in $\mathcal{Y}$. Puncturable PRFs have found numerous applications in cryptography [BW13, KPTZ13, BGI14], most notably in applications of indistinguishability obfuscation ($i\mathcal{O}$) [SW14]. Note that two punctured keys, punctured at two different points, enable the evaluation of the PRF at all points in the domain $\mathcal{X}$, and are therefore equivalent to the master PRF key k. For this reason,

*Part of this work was done as an intern at Fujitsu Laboratories of America.

for puncturing constraints, we are primarily interested in settings where the adversary is limited to obtaining at most a single punctured key, punctured at a point of its choice. At the punctured point, the adversary should be unable to distinguish the value of the PRF from random.

PRFs supporting puncturing constraints can be easily constructed from the tree-based PRF of [GGM86], as discussed in [BW13, KPTZ13, BGI14]. Notice, however, that a punctured key k_x completely reveals what the point x is. An adversary that is given k_x can easily tell where the key was punctured.

Private puncturing. Can we construct a PRF that can be punctured *privately*? The adversary should learn nothing about x from the punctured key k_x. More generally, Boneh, Lewi, and Wu [BLW17] define private constrained PRFs, where a constrained key k_g reveals nothing about the predicate g. They present applications of private constraint PRFs to constructing software watermarking [CHN$^+$16], deniable encryption [CDNO97], searchable encryption, and more. They also construct private constrained PRFs from powerful tools, such as multilinear maps and $i\mathcal{O}$.

Several of the applications for private constraints in [BLW17] require only private puncturing. Here we describe one such application, namely the connection to distributed point functions (DPF) [GI14, BGI15] and 2-server private information retrieval (PIR) [CKGS98]. In a DPF, the key generation algorithm is given a point $x^* \in \mathcal{X}$ and outputs two keys k_0 and k_1. The two keys are equivalent, except at the point x^*. More precisely, $F(k_0, x) = F(k_1, x)$ for all key $x \neq x^*$ and $F(k_0, x^*) \neq F(k_1, x^*)$. A DPF is secure if given one of k_0 or k_1, the adversary learns nothing about x^*. In [GI14] the authors show that DPFs give a simple and efficient 2-server PIR scheme. They give an elegant DPF construction from one-way functions.

A privately puncturable PRF is also a DPF: set k_0 to be the master PRF key k, and set the key k_1 to be the punctured key k_{x^*}, punctured at x^*. The privacy property ensures that this is a secure DPF. However, there is an important difference between a DPF and a privately puncturable PRF. DPF key generation takes the punctured point x^* as input, and generates the two keys k_0, k_1. In contrast, private puncturing works differently: one first generates the master key k, and at some time later asks for a punctured key k_{x^*} at some point x^*. That is, the punctured point is chosen *adaptively* after the master key is generated. This adaptive capability gives rise to a 2-server PIR scheme that has a surprising property: one of the servers can be offline. In particular, one of the servers does its PIR computation *before* the PIR query is chosen, sends the result to the client, and goes offline. Later, when the client chooses the PIR query, it only talks to the second server.

Our contribution. We construct the first *privately* puncturable PRF from the learning with errors problem (LWE) [Reg09] and the one-dimensional short integer solution problem (1D-SIS) [Ajt96, BV15], which are both related to worst-case hardness of general lattice problems. We give a brief overview of the construction here, and give a detailed overview in Section 2.

Our starting point is the elegant LWE-based PRF of Brakerski and Vaikuntanathan [BV15], which is a constrained PRF for general circuits, but is only secure if at most one constrained key is published (publishing two constrained keys reveals the master key). This PRF is not private because the constraint is part of the constrained key and is available in the clear. As a first attempt, we try to make this PRF private by embedding in the constrained key, an FHE encryption of the constraint, along with an encryption of the FHE decryption key (a similar structure is used in the predicate encryption scheme of [GVW15b]). Now the constraint is hidden, but PRF evaluation requires an FHE decryption, which is a problem. We fix this in a number of steps, as described in the next section. To prove security, we introduce an additional randomizing component as part of the FHE plaintext to embed an LWE instance in the challenge PRF evaluation.

We prove security of our private puncturable PRF in the selective setting, where the adversary commits ahead of time to the punctured point x where it will be challenged. To obtain adaptive security, where the punctured point is chosen adaptively, we use standard complexity leveraging [BB04].

In addition to our punctured PRF construction, we show in Section 6 that, for general function constraints, a simulation based definition of privacy is impossible. This complements [BLW17] who show that a game-based definition of privacy is achievable assuming the existence of $i\mathcal{O}$. To prove the impossibility, we show that even for a single key, a simulation-secure privately constrained PRF for general functions, implies a simulation secure functional encryption for general functions, which was previously shown to be impossible [BSW11, AGVW13].

Finally, our work raises a number of interesting open problems. First, our techniques work well to enable private puncturing, but do not seem to generalize to arbitrary circuit constraints. It would be a significant achievement if one could use LWE/SIS to construct a *private* constrained PRF for arbitrary circuits, even in the single-key case. Also, can we construct an LWE/SIS-based *adaptively* secure private puncturable PRF, without relying on complexity leveraging? We discuss these questions in more detail in Section 7.

1.1 Related Work

PRFs from LWE. The first PRF construction from the learning with errors assumption was given by Banerjee, Peikert, and Rosen in [BPR12]. Subsequent PRF constructions from LWE gave the first key-homomorphic PRFs [BLMR13, BP14]. The constructions of [BV15, BFP$^+$15] generalized the previous works to the setting of constrained PRFs.

Constrained PRFs. The notion of constrained PRFs was first introduced in three independent works [BW13, KPTZ13, BGI14] and since then, there have been a number of constructions from different assumptions. We briefly survey the state of the art. The standard GGM tree [GGM86] gives PRFs for simple constraints such as prefix-fixing or puncturing [BW13, KPTZ13, BGI14]. Bilinear maps give left/right constraints but in the random oracle model [BW13]. LWE gives general circuit constraints, but only when a single constrained key is released [BV15]. Multilinear maps and indistinguishability obfuscation provide general circuit constraints, and even for constraints represented as Turing machines with unbounded inputs [BW13, BZ14, BFP$^+$15, CRV14, AFP16, DKW16], as well as constrained verifiable random functions [Fuc14]. Several works explore how to achieve adaptive security [FKPR14, BV15, HKW15, HKKW14].

Private constrained PRFs were introduced by Boneh, Lewi, and Wu [BLW17]. They construct a privately constrained PRF for puncturing and bit-fixing constraints from multilinear maps, and for circuit constraints using indistinguishability obfuscation.

ABE and PE from LWE. The techniques used in this work build upon a series of works in the area of *attribute-based encryption* [SW05] and *predicate encryption* [BW07, KSW08] from LWE. These include constructions of [ABB10, GVW15a, BGG$^+$14, GV15, BV16, BCTW16], and predicate encryption constructions of [AFV11, GMW15, GVW15b].[1]

Concurrent Work. In an independent and concurrent work, Canetti and Chen [CC17] construct a single-key privately constrained PRF for general NC^1 circuits from LWE. Their techniques are very different from the ones used in this work as their construction relies on instances of the graph-induced multilinear maps construction by Gentry, Gorbunov, and Halevi [GGH15] that can be reduced to LWE. They also analyze their construction with respect to a simulation-based definition. We note that the simulation-based definition that we consider in this work is much stronger than their definition and therefore, the impossibility that we show does not apply to their definition.

2 Overview of the Main Construction

In this section, we provide a general overview of our main construction. The complete construction and proof of security are provided in Section 5.1.

Recall that the LWE assumption states that for a uniform vector $\mathbf{s} \in \mathbb{Z}_q^n$ and a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ for an appropriately chosen n, m, q, it holds that $(\mathbf{A},\ \mathbf{s}^T\mathbf{A} + \mathbf{e}^T)$ is indistinguishable from uniform where $\mathbf{e}$ is sampled from an appropriate low-norm error distribution. We present the outline ignoring the precise generation or evolution of $\mathbf{e}$ and just refer to it as noise.

Embedding Circuits into Matrices. Our starting point is the single-key constrained PRF of [BV15], which builds upon the ABE construction of [BGG$^+$14] and the PRF of [BP14]. At a high level, the ABE of [BGG$^+$14] encodes an attribute vector $\mathbf{x} \in \{0,1\}^\ell$ as a vector

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

[1] We note that LWE based predicate encryption constructions satisfy a weaker security property often referred to as *weak attribute-hiding* than as is defined in [BW07, KSW08].

for public matrices $\mathbf{A}_1, ..., \mathbf{A}_\ell$ in $\mathbb{Z}_q^{n\times m}$, a secret random vector $\mathbf{s}$ in $\mathbb{Z}_q^n$, and a specific fixed "gadget matrix" $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$. This encoding allows for fully homomorphic operations on the attributes, while keeping the noise small. In particular, given $\mathbf{x}$ and a poly-size circuit $f : \{0,1\}^\ell \to \{0,1\}$, one can compute from (2.1), the vector

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

where the matrix $\mathbf{A}_f$ depends only on the function f, and not on the underlying attribute $\mathbf{x}$. This implies a homomorphic operation on the matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\ell$ defined as $\mathsf{Eval}_{\mathsf{pk}}(f, \mathbf{A}_1, \ldots, \mathbf{A}_\ell) \to \mathbf{A}_f$.

This homomorphic property leads to the following puncturable PRF. Let $\mathsf{eq}(\mathbf{x}^*, \mathbf{x})$ be the equality check circuit (represented as NAND gates) defined as follows:

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

For $\mathbf{x} = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$ define the PRF as:

$$\mathsf{PRF}_{\mathbf{s}}(\mathbf{x}) := \lfloor \mathbf{s}^T \cdot \mathbf{A}_{\mathsf{eq}} \rceil_p \ \ \in \mathbb{Z}_p^m \quad \text{where} \quad \mathbf{A}_{\mathsf{eq}} := \mathsf{Eval}_{\mathsf{pk}}(\mathsf{eq}, \mathbf{B}_1, ..., \mathbf{B}_\ell, \mathbf{A}_{x_1}, ..., \mathbf{A}_{x_\ell}).$$

Here $\mathbf{s} \in \mathbb{Z}_q^n$ is the master secret key, and the matrices $\mathbf{A}_0, \mathbf{A}_1, \mathbf{B}_1, ..., \mathbf{B}_\ell$ are random public matrices in $\mathbb{Z}_q^{n\times m}$ chosen at setup. Note that $\mathbf{A}_{\mathsf{eq}}$ is a function of $\mathbf{x}$. The operation $\lfloor\cdot\rceil_p$ is component-wise rounding that maps an element in $\mathbb{Z}_q$ to an element in $\mathbb{Z}_p$ for an appropriately chosen p, where $p < q$.

Next, define the punctured key at the point $\mathbf{x}^* = (x_1^*, \ldots, x_\ell^*) \in \{0,1\}^\ell$ as:

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

To use this key to evaluate the PRF at a point $\mathbf{x} \in \{0,1\}^\ell$, the user homomorphically evaluates the equality check circuit $\mathsf{eq}(\mathbf{x}^*, \mathbf{x})$, as in (2.2), to obtain the vector $\mathbf{s}^T\big(\mathbf{A}_{\mathsf{eq}} + \mathsf{eq}(\mathbf{x}^*, \mathbf{x})\cdot\mathbf{G}\big) + \mathsf{noise}$. Rounding this vector gives the correct PRF value whenever $\mathsf{eq}(\mathbf{x}^*, \mathbf{x}) = 0$, namely $\mathbf{x} \neq \mathbf{x}^*$, as required. A security argument as in [BV15] proves that with some minor modifications, this PRF is a secure (non-private) puncturable PRF, assuming that the LWE problem is hard.

FHE to hide puncture point. The reason why the construction above is not private is because to operate on the ABE encodings, one needs the description of the attributes. Therefore, the punctured key must include the point $\mathbf{x}^*$ in the clear, for the evaluator to run the equality check circuit on the punctured key (2.3).

Our plan to get around this limitation is to first encrypt the attributes $(x_1^*, \ldots, x_\ell^*)$ using a fully homomorphic encryption (FHE) scheme before embedding it as the attributes. In particular, we define our punctured key to be

$$\begin{aligned} k_{\mathbf{x}^*} = \big(\mathsf{ct}, \quad \mathbf{s}^T \cdot \big(\mathbf{A}_0 + 0\cdot\mathbf{G} \mid \mathbf{A}_1 + 1\cdot\mathbf{G} \quad \big| \quad & \mathbf{B}_1 + \mathsf{ct}_1\cdot\mathbf{G} \mid \cdots \mid \mathbf{B}_z + \mathsf{ct}_z\cdot\mathbf{G} \\ & \big| \quad \mathbf{C}_1 + \mathsf{sk}_1\cdot\mathbf{G} \mid \cdots \mid \mathbf{C}_t + \mathsf{sk}_t\cdot\mathbf{G}\big) + \mathsf{noise} \ \ \big), \end{aligned}$$

where $\mathsf{ct} \in \mathbb{Z}_q^z$ is an FHE encryption of the punctured point $\mathbf{x}^*$, and $\mathsf{sk} \in \mathbb{Z}_q^t$ is the FHE secret key. While it is not clear how to use this key to evaluate the PRF, at least the punctured point $\mathbf{x}^*$ is not exposed in the clear. One can show that the components of $k_{\mathbf{x}^*}$ that embed the secret key sk do not leak information about sk.

Now, given $\mathbf{x} \in \{0,1\}^\ell$, one can now run the equality check operation inside the FHE ciphertext, which gives the *encrypted* result of the equality check circuit. The question is how the evaluator can extract this result from the ciphertext. To do this, we take advantage of another property of the underlying ABE: to homomorphically multiply two attributes, one requires knowledge of just one of the attributes, not both. This means that even without the knowledge of the FHE secret key sk, the evaluator can compute the inner product of sk and ct. Recall that for lattice-based FHE schemes (e.g.,[GSW13]), the decryption operation is the rounding of the inner product of the ciphertext with the FHE secret key. This technique was also used in the lattice-based predicate encryption scheme of [GVW15b].

Rounding away FHE noise. The problem with the approach above is that we cannot compute the full FHE decryption. We can only compute the first decryption step, the inner product. The second step, rounding, cannot be done while keeping the FHE decryption key secret. Computing just the inner product produces the FHE plaintext, but offset by some small additive error term $e \in \mathbb{Z}_q$. More specifically, the homomorphic evaluation of $\mathsf{eq}(\mathbf{x}^*, \mathbf{x})$ followed by the inner product with sk, results in the vector

$$\mathbf{s}^T \left(\mathbf{A}_{\mathsf{fhe,eq}} + \left(\frac{q}{2} \cdot \mathsf{eq}(\mathbf{x}^*, \mathbf{x}) + e \right) \cdot \mathbf{G} \right) + \mathsf{noise} \quad \in \mathbb{Z}_q^m,$$

where $\mathbf{A}_{\mathsf{fhe,eq}}$ is the result of homomorphically computing the FHE equality test circuit, along with the inner product with the secret key, on the public matrices. Here $e \in \mathbb{Z}_q$ is some offset term. Even when $\mathsf{eq}(\mathbf{x}^*, \mathbf{x}) = 0$, the rounding of this vector will not produce the correct evaluation due to this offset term e. Moreover, the term e contains information about the original plaintext and therefore, to ensure private puncturing, we must somehow allow for correct computation without revealing the actual value of e. Resolving this issue seems difficult. It is precisely the reason why the predicate encryption scheme of [GVW15b] cannot be converted to a *fully-attribute hiding* predicate encryption scheme (and therefore a full-fledged functional encryption scheme). However, in our context, the problem of *noisy decryption* has an elegant solution.

The idea is to "shorten" the vector $(\mathbf{s}^T \cdot e \cdot \mathbf{G})$ so that it is absorbed into noise, and disappears as we round to obtain the PRF value at x. Towards this goal, we sample the secret vector $\mathbf{s}$ from the LWE noise distribution, which does not change the hardness of LWE [ACPS09]. Next, we observe that although the gadget matrix $\mathbf{G}$ is not a short matrix as a whole, it does contain a number of short column vectors. For instance, a subset of the columns vectors of the gadget matrix consist of elementary basis vectors $\mathbf{u}_i \in \mathbb{Z}_q^n$ with the ith entry set to 1 and the rest set to 0. More precisely, for $1 \leq i \leq n$, let the vector $\mathbf{v}_i \in \mathbb{Z}_q^m$ be an m dimensional basis vectors with its $i \cdot \lfloor \log q - 1 \rfloor$th entry set to 1 and the rest set to 0. Then, $\mathbf{G} \cdot \mathbf{v}_i = \mathbf{u}_i$.

With this observation, we can simply define the PRF with respect to these short column positions in the gadget matrix. For instance, consider defining the PRF with respect to the first column position as follows

$$\mathsf{PRF}_{\mathbf{s}}(\mathbf{x}) := \lfloor \mathbf{s}^T \cdot \mathbf{A}_{\mathsf{fhe,eq}} \cdot \mathbf{v}_1 \rceil_p \quad \in \mathbb{Z}_p.$$

Since we are simply taking the first component of a pseudorandom vector, this does not change the pseudorandomness property of the PRF (to adversaries without a constrained key). However, for the evaluation with the punctured key, this allows the FHE error term to be "merged" with noise

$$\begin{aligned}
&\left(\mathbf{s}^T \; \left(\mathbf{A}_{\mathsf{fhe,eq}} + \left(\frac{q}{2} \cdot \mathsf{eq}(\mathbf{x}^*, \mathbf{x}) + e \right) \cdot \mathbf{G} \right) + \mathsf{noise} \right) \mathbf{v}_1 \\
&\quad = \mathbf{s}^T \mathbf{A}_{\mathsf{fhe,eq}} \mathbf{v}_1 + \mathbf{s}^T \left(\frac{q}{2} \cdot \mathsf{eq}(\mathbf{x}^*, \mathbf{x}) + e \right) \mathbf{u}_1 + \mathsf{noise}' \\
&\quad = \mathbf{s}^T \mathbf{A}_{\mathsf{fhe,eq}} \mathbf{v}_1 + \left(\frac{q}{2} \cdot \mathsf{eq}(\mathbf{x}^*, \mathbf{x}) + e \right) \langle \mathbf{s}, \mathbf{u}_1 \rangle + \mathsf{noise}' \\
&\quad = \mathbf{s}^T \mathbf{A}_{\mathsf{fhe,eq}} \mathbf{v}_1 + \frac{q}{2} \cdot \mathsf{eq}(\mathbf{x}^*, \mathbf{x}) \langle \mathbf{s}, \mathbf{u}_1 \rangle + \underbrace{e \cdot \langle \mathbf{s}, \mathbf{u}_1 \rangle + \mathsf{noise}'}_{\text{short}}.
\end{aligned}$$

When $\mathsf{eq}(\mathbf{x}^*, \mathbf{x}) = 0$, then the rounding of the vector above results in the correct PRF evaluation since the final noise $e \cdot \langle \mathbf{s}, \mathbf{u}_1 \rangle + \mathsf{noise}'$ is small and will disappear with the rounding.

Pseudorandomness at punctured point. The remaining problem is to make the PRF evaluation at the punctured point look random to an adversary who only holds a punctured key. Note that if the adversary evaluates the PRF at the punctured point x^* using its punctured key, the result is the correct PRF output, but offset by the term $(\frac{q}{2} + e) \cdot s_1 + \mathsf{noise}'$, which is clearly distinguishable from random. To fix this, we make the following modifications. First, we include a uniformly generated vector $\mathbf{w} = (w_1, ..., w_n) \in \mathbb{Z}_q^n$ as part of the public parameters. Then, we modify the FHE homomorphic operation such that after evaluating the equality check circuit, we multiply the resulting message with one of the w_i's such that decryption outputs $w_i \cdot \mathsf{eq}(\mathbf{x}^*, \mathbf{x}) + e$, instead of $\frac{q}{2} \cdot \mathsf{eq}(\mathbf{x}^*, \mathbf{x}) + e$. Then, we define the PRF evaluation as the vector

$$\mathsf{PRF}_{\mathbf{s}}(\mathbf{x}) = \left\lfloor \sum_i \mathbf{s}^T \cdot \mathbf{A}_{\mathsf{fhe,eq},i} \cdot \mathbf{v}_i \right\rceil_p .$$

where $\mathbf{v}_1, ..., \mathbf{v}_n \in \mathbb{Z}_q^m$ are elementary basis vectors such that $\mathbf{G} \cdot \mathbf{v}_i = \mathbf{u}_i \in \mathbb{Z}_q^n$. Here, the matrix $\mathbf{A}_{\mathsf{fhe},\mathsf{eq},i}$ represents the matrix encoding the equality check circuit operation, followed by scalar multiplication by w_i. Now, evaluating the PRF with the punctured key at the punctured point results in the vector

$$\begin{aligned}\sum_i \big(\mathbf{s}^T \ (\mathbf{A}_{\mathsf{fhe},\mathsf{eq},i} + (w_i \cdot \mathsf{eq}(\mathbf{x}^*, \mathbf{x}) + e) \cdot \mathbf{G}) + \mathsf{noise}\big)\, \mathbf{v}_i \\ &= \sum_i \mathbf{s}^T \mathbf{A}_{\mathsf{fhe},\mathsf{eq},i} \cdot \mathbf{v}_i + \sum_i \mathbf{s}^T \left(w_i \cdot \mathsf{eq}(\mathbf{x}^*, \mathbf{x}) + e_i\right) \mathbf{u}_i + \mathsf{noise}' \\ &= \sum_i \mathbf{s}^T \mathbf{A}_{\mathsf{fhe},\mathsf{eq},i} \cdot \mathbf{v}_i + \sum_i (\mathsf{eq}(\mathbf{x}^*, \mathbf{x}) + e_i) \langle \mathbf{s}, w_i \cdot \mathbf{u}_i \rangle \cdot + \mathsf{noise}'. \\ &= \sum_i \mathbf{s}^T \mathbf{A}_{\mathsf{fhe},\mathsf{eq},i} \cdot \mathbf{v}_i + \mathsf{eq}(\mathbf{x}^*, \mathbf{x}) \langle \mathbf{s}, \mathbf{w} \rangle + \mathsf{noise}''.\end{aligned}$$

We note that when $\mathsf{eq}(\mathbf{x}^*, \mathbf{x}) = 1$, then the offset term is a noisy inner product on the secret vector $\mathbf{s}$. This allows us to embed an LWE sample in the offset term and show that the evaluation indeed looks uniformly random to an adversary with a punctured key.

3 Preliminaries

Basic Notations. For an integer n, we write $[n]$ to denote the set $\{1, ..., n\}$. For a finite set S, we write $x \xleftarrow{\$} S$ to denote sampling x uniformly at random from S. We use bold lowercase letters (*e.g.*, $\mathbf{v}, \mathbf{w}$) to denote column vectors and bold uppercase letters (*e.g.*, $\mathbf{A}, \mathbf{B}$) to denote matrices. For a vector or matrix $\mathbf{s}, \mathbf{A}$, we use $\mathbf{s}^T, \mathbf{B}^T$ to denote their transpose. We write λ for the security parameter. We say that a function $\epsilon(\lambda)$ is negligible in λ, if $\epsilon(\lambda) = o(1/\lambda^c)$ for every $c \in \mathbb{N}$, and we write $\mathsf{negl}(\lambda)$ to denote a negligible function in λ. We say that an event occurs with *negligible probability* if the probability of the event is $\mathsf{negl}(\lambda)$, and an event occurs with *overwhelming probability* if its complement occurs with negligible probability.

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

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

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

Norm for Vectors and Matrices. Throughout this work, we will always use the infinity norm for vectors and matrices. This means that for a vector $\mathbf{x}$, the norm $\|\mathbf{x}\|$ is the maximal absolute value of an element in $\mathbf{x}$. Similarly, for a matrix $\mathbf{A}$, $\|\mathbf{A}\|$ is the maximal absolute value of any of its entries. If $\mathbf{x}$ is n-dimensional and $\mathbf{A}$ is $n \times m$, then $\left\|\mathbf{x}^T \mathbf{A}\right\| \leq n \cdot \|\mathbf{x}\| \cdot \|\mathbf{A}\|$.

3.1 Private Constrained PRFs

We first review the definition of a pseudorandom function (PRF) [GGM86].

Definition 3.1 (Pseudorandom Function [GGM86])**.** Fix a security parameter λ. A keyed function $F \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ with keyspace $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ is pseudorandom if for all efficient algorithms $\mathcal{A}$,

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

Sometimes, a PRF is defined more naturally with respect to a pair of algorithms $\Pi_{\mathsf{PRF}} = (\mathsf{PRF.Setup}, \mathsf{PRF.Eval})$ where $\mathsf{PRF.Setup}$ is a randomized algorithm that samples the PRF key k in $\mathcal{K}$ and $\mathsf{PRF.Eval}$ computes the keyed function $F(k, \cdot)$.

In a constrained PRF [BW13, KPTZ13, BGI14], an authority with a master secret key msk for the PRF can create a *restricted key* sk_f associated with some function f that allows one to evaluate the PRF only at inputs $x \in \mathcal{X}$ for which $f(x) = 0$.[2]

Definition 3.2 (Constrained PRF [BW13, KPTZ13, BGI14])**.** A constrained PRF consists of a tuple of algorithms $\Pi_{\mathsf{pPRF}} = (\mathsf{cPRF.Setup}, \mathsf{cPRF.Constrain}, \mathsf{cPRF.ConstrainEval}, \mathsf{cPRF.Eval})$ over domain $\mathcal{X}$, range $\mathcal{Y}$, and circuit class $\mathcal{C}$ is defined as follows:

- $\mathsf{cPRF.Setup}(1^\lambda) \to \mathsf{msk}$: On input the security parameter λ, the setup algorithm outputs the master secret key msk.
- $\mathsf{cPRF.Constrain}(\mathsf{msk}, f) \to \mathsf{sk}_f$: On input the master secret key msk, and a circuit $f \in \mathcal{C}$, the constrain algorithm outputs a constrained key sk_f.
- $\mathsf{cPRF.ConstrainEval}(\mathsf{sk}, x) \to y$: On input a constrained key sk, and an input $x \in \mathcal{X}$, the puncture evaluation algorithm evaluates the PRF value $y \in \mathcal{Y}$.
- $\mathsf{cPRF.Eval}(\mathsf{msk}, x) \to y$: On input the master secret key msk and an input $x \in \mathcal{X}$, the evaluation algorithm evaluates the PRF value $y \in \mathcal{Y}$.

Algorithms $\mathsf{cPRF.Setup}$ and $\mathsf{cPRF.Constrain}$ are randomized, while algorithms $\mathsf{cPRF.ConstrainEval}$ and $\mathsf{cPRF.Eval}$ are always deterministic.

Correctness. A constrained PRF is correct if for all $\lambda \in \mathbb{N}$, $\mathsf{msk} \leftarrow \mathsf{cPRF.Setup}(1^\lambda)$, for every circuit $C \in \mathcal{C}$, and input $x \in \mathcal{X}$ for which $f(x) = 0$, we have that

$$\mathsf{cPRF.ConstrainEval}(\mathsf{cPRF.Constrain}(\mathsf{msk}, f), x) = \mathsf{cPRF.Eval}(\mathsf{msk}, x)$$

with overwhelming probability.

Security. We require two security properties for constrained PRFs: pseudorandomness and privacy. The first property states that given constrained PRF keys, an adversary cannot distinguish the PRF evaluation at the points where it is not allowed to compute, from a randomly sampled point from the range.

Definition 3.3 (Pseudorandomness)**.** Fix a security parameter λ. A constrained PRF scheme $\Pi_{\mathsf{cPRF}} = (\mathsf{cPRF.Setup}, \mathsf{cPRF.Constrain}, \mathsf{cPRF.ConstrainEval}, \mathsf{cPRF.Eval})$ is pseudorandom if for all PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there is a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{rand}}_{\Pi_{\mathsf{cPRF}},\mathcal{A}}(\lambda) = \left| \Pr[\mathsf{Expt}^{(0)}_{\Pi_{\mathsf{cPRF}},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{(1)}_{\Pi_{\mathsf{cPRF}},\mathcal{A}}(\lambda) = 1] \right| \leq \mathsf{negl}(\lambda)$$

where for each $b \in \{0, 1\}$ and $\lambda \in \mathbb{Z}$, the experiment $\mathsf{Expt}^{(b)}_{\Pi_{\mathsf{cPRF}},\mathcal{A}}(\lambda)$ is defined as follows:

1. $\mathsf{msk} \leftarrow \mathsf{cPRF.Setup}(1^\lambda)$
2. $(x^*, \mathsf{state}_1) \leftarrow \mathcal{A}_1^{\mathsf{cPRF.Constrain}(\mathsf{msk},\cdot),\mathsf{cPRF.Eval}(\mathsf{msk},\cdot)}(1^\lambda)$
3. $y_0 \leftarrow \mathsf{cPRF.Eval}(\mathsf{msk}, x^*)$
4. $y_1 \xleftarrow{\$} \mathcal{Y}$
5. $b' \leftarrow \mathcal{A}_2^{\mathsf{cPRF.Constrain}(\mathsf{msk},\cdot),\mathsf{cPRF.Eval}(\mathsf{msk},\cdot)}(y_b, \mathsf{state}_1)$
6. Output b'

To prevent the adversary from trivially winning the game, we require that for any query f that $\mathcal{A}$ makes to the $\mathsf{cPRF.Constrain}(\mathsf{msk}, \cdot)$ oracle, it holds that $f(x^*) = 1$, and for any query x that $\mathcal{A}$ makes to the $\mathsf{cPRF.Eval}(\mathsf{msk}, \cdot)$ oracle, it holds that $x \neq x^*$.

[2] We adopt the convention that $f(x) = 0$ signifies the ability to evaluate the PRF. This is opposite of the standard convention, and is done purely for convenience in the technical section.

The security games as defined above is the *fully adaptive* game. One can also define a *selective* variant of the games above where the adversary commits to the challenge point before the game starts. We do so in Definition 3.6 below.

Next, we require that a constrained key sk_f not leak information about the constraint function f as in the setting of private constrained PRFs of [BLW17].

Definition 3.4 (Privacy)**.** Fix a security parameter $\lambda \in \mathbb{N}$. A constrained PRF scheme $\Pi_{\mathsf{cPRF}} = (\mathsf{cPRF.Setup}, \mathsf{cPRF.Constrain}, \mathsf{cPRF.ConstrainEval}, \mathsf{cPRF.Eval})$ is private if for all PPT adversary $\mathcal{A}$, there is a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{priv}}_{\Pi_{\mathsf{cPRF}},\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Expt}^{(0)}_{\Pi_{\mathsf{cPRF}},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{(1)}_{\Pi_{\mathsf{cPRF}},\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

where the experiments $\mathsf{Expt}^{(b)}_{\Pi_{\mathsf{cPRF}},\mathcal{A}}$ are defined as follows:

1. $\mathsf{msk} \leftarrow \mathsf{cPRF.Setup}(1^\lambda)$.
2. $b' \leftarrow \mathcal{A}^{\mathsf{cPRF.Constrain}_b(\mathsf{msk},\cdot,\cdot),\mathsf{cPRF.Eval}(\mathsf{msk},\cdot)}(1^\lambda)$.
3. Output b'

where the oracle $\mathsf{cPRF.Constrain}_b(\cdot,\cdot,\cdot)$ is defined as follows

- $\mathsf{cPRF.Constrain}_b(\mathsf{msk}, f_0, f_1)$: On input the master secret key msk, and a pair of constraint functions f_0, f_1, outputs $sk_{f,b} \leftarrow \mathsf{cPRF.Constrain}(f_b)$.

In the experiment above, we require an extra admissibility condition on the adversary to prevent it from trivially distinguishing the two experiments. For a circuit $f \in \mathcal{C}$, define the set $S(f) \subseteq \mathcal{X}$ where $\{x \in \mathcal{X} : f(x) = 0\}$. Let d be the number of queries that $\mathcal{A}$ makes to $\mathsf{cPRF.Constrain}_b(\mathsf{msk},\cdot,\cdot)$ and let $(f_0^{(i)}, f_1^{(i)})$ for $i \in [d]$ denote the ith pair of circuits that the adversary submits to the constrain oracle. Then we require that

1. For every query x that $\mathcal{A}$ makes to the evaluation oracle, $f_0^{(i)}(x) = f_1^{(i)}(x)$.
2. For every pair of distinct indices $i, j \in [d]$,

$$S\left(f_0^{(i)}\right) \cap S\left(f_0^{(j)}\right) = S\left(f_1^{(i)}\right) \cap S\left(f_1^{(j)}\right).$$

Justification for the second admissibility condition is discussed in [BLW17, Remark 2.11].

3.2 Private puncturable PRFs

A puncturable PRF is a special case of constrained PRFs where one can only request constained keys for point functions. That is, each constraining circuit C_{x^*} is associated with a point $x^* \in \{0,1\}^n$, and $C_{x^*}(x) = 0$ if and only if $x \neq x^*$. Concretely, a puncturable PRF is specified by a tuple of algorithms $\Pi_{\mathsf{pPRF}} = (\mathsf{pPRF.Setup}, \mathsf{pPRF.Puncture}, \mathsf{pPRF.PunctureEval}, \mathsf{pPRF.Eval})$ with identical syntax as regular constrained PRFs, with the exception that the algorithm $\mathsf{pPRF.Puncture}$ takes in a point x to be punctured rather than a circuit f.

In the context of private puncturing, we require without loss of generality, that algorithm $\mathsf{pPRF.Puncture}$ be deterministic (see [BLW17, Remark 2.14]). If it were randomized, it could be de-randomized by generating its random bits using a PRF keyed by a part of msk, and given the point x as input.

We define a slightly weaker variant of correctness than as is defined above for constrained PRF called *computational functionality preserving* as in the setting of [BV15]. In words, this property states that it is computationally hard to find a point $x \neq x^*$ such that the result of the puncture evaluation differs from the actual PRF evaluation. This is essentially a relaxation of the *statistical* notion of correctness to the *computational* notion of correctness.

Definition 3.5 (Computational Functionality Preserving)**.** Fix a security parameter λ and let $\Pi_{\mathsf{pPRF}} = (\mathsf{pPRF.Setup}, \mathsf{pPRF.Puncture}, \mathsf{pPRF.PunctureEval}, \mathsf{pPRF.Eval})$ be a private-puncturable PRF scheme. For every adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, consider the following experiment where we choose $\mathsf{msk} \leftarrow \mathsf{pPRF.Setup}(1^\lambda)$, $(\mathsf{state}, x^*) \leftarrow \mathcal{A}_1(1^\lambda)$, and $\mathsf{sk}_{x^*} \leftarrow \mathsf{pPRF.Puncture}(\mathsf{msk}, x^*)$. Then, the private-puncturable PRF scheme Π_{pPRF} is *computational functionality preserving* if

$$\Pr\left[x \leftarrow \mathcal{A}_2^{\mathsf{pPRF.Eval}(\mathsf{msk},\cdot)}(\mathsf{state}, \mathsf{sk}_{x^*}) : \begin{array}{c} x \neq x^* \wedge \\ \mathsf{pPRF.Eval}(\mathsf{msk}, x) \neq \\ \mathsf{pPRF.PunctureEval}(\mathsf{sk}_{x^*}, x) \end{array} \right] \leq \mathsf{negl}(\lambda)$$

for some negligible function negl.

We next specialize the security definitions to the settings of puncturing constraints. For puncturable PRFs, the adversary in the pseudorandomness game is limited to making at most one key query to $\mathsf{pPRF.Puncture}$. If it made two key queries, for two distinct punctures, it would be able to evaluate the PRF on all points in the domain, and then cannot win the game. Therefore, we need only consider two types of adversaries in the pseudorandomness game:

- an adversary that makes evaluation queries, but no key queries during the game, and
- an adversary that makes exactly one key query.

The first adversary plays the regular PRF security game. We show in Appendix A that selective security against an adversary of the second type, implies security against an adversary of the first type. Therefore, when defining (selective) security, it suffices to only consider (selective) adversaries of the second type.

One technicality in defining pseudorandomness for puncturable PRFs that satisfy a computational notion of correctness is that the adversary must also be given access to an evaluation oracle. This is because given only a punctured key, the adversary cannot efficiently detect whether a point in the domain evaluates to the correct PRF evaluation with the punctured key without the evaluation oracle. Therefore, we define the following pseudorandomness definition.

Definition 3.6. Fix a security parameter λ. A puncturable PRF scheme $\Pi_{\mathsf{pPRF}} = (\mathsf{pPRF.Setup}, \mathsf{pPRF.Puncture}, \mathsf{pPRF.PunctureEval}, \mathsf{pPRF.Eval})$ is *selectively-pseudorandom* if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function negl such that for $\mathsf{msk} \leftarrow \mathsf{pPRF.Setup}(1^\lambda)$, $(x^*, \mathsf{state}) \leftarrow \mathcal{A}_1(1^\lambda)$, $\mathsf{sk}_{x^*} \leftarrow \mathsf{pPRF.Puncture}(\mathsf{msk}, x^*)$, $u \stackrel{\$}{\leftarrow} \mathcal{Y}$, we have that

$$\left| \Pr[\mathcal{A}_2^{\mathsf{pPRF.Eval}(\mathsf{msk},\cdot)}(\mathsf{state}, \mathsf{sk}_{x^*}, \mathsf{pPRF.Eval}(\mathsf{msk}, x^*)) = 1] - \Pr[\mathcal{A}_2^{\mathsf{pPRF.Eval}(\mathsf{msk},\cdot)}(\mathsf{state}, \mathsf{sk}_{x^*}, u) = 1] \right| \leq \mathsf{negl}(\lambda).$$

To prevent the adversary from trivially breaking the game, we require that the adversary $\mathcal{A}$ cannot query the evaluation oracle on x^*.

We next define the notion of privacy for puncturable PRFs. Again, since we rely on the computational notion of correctness, we provide the adversary access to an honest evaluation oracle (except for at the challenge points). As in the pseudorandomness game, we only consider selective adversaries that make a *single* key query, although that results in a slightly weaker notion of privacy than in Definition 3.4.[3]

Definition 3.7. Fix a security parameter λ. A puncturable PRF scheme $\Pi_{\mathsf{pPRF}} = (\mathsf{pPRF.Setup}, \mathsf{pPRF.Puncture}, \mathsf{pPRF.PunctureEval}, \mathsf{pPRF.Eval})$ is *selectively-private* if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function negl such that for $\mathsf{msk} \leftarrow \mathsf{pPRF.Setup}(1^\lambda)$, $(x^*, \mathsf{state}) \leftarrow \mathcal{A}_1(1^\lambda)$, $\mathsf{sk}_{x^*} \leftarrow \mathsf{pPRF.Puncture}(\mathsf{msk}, x^*)$, $\mathsf{sk}_{\mathbf{0}} \leftarrow \mathsf{pPRF.Puncture}(\mathsf{msk}, \mathbf{0})$, we have that

$$\left| \Pr[\mathcal{A}_2^{\mathsf{pPRF.Eval}(\mathsf{msk},\cdot)}(\mathsf{state}, \mathsf{sk}_{x^*}) = 1] - \Pr[\mathcal{A}_2^{\mathsf{pPRF.Eval}(\mathsf{msk},\cdot)}(\mathsf{state}, \mathsf{sk}_{\mathbf{0}}) = 1] \right| \leq \mathsf{negl}(\lambda).$$

[3] We note that the admissibility condition in Definition 3.4 allows an adversary to make two constrained key queries (see [BLW17] Remark 2.14). However, applications of privately puncturable PRFs require pseudorandomness property to be satisfied, which can only be achieved in the single-key setting. Therefore, the restriction of privacy to the single-key setting does not affect the applications of privately puncturable PRFs.

To prevent the adversary from trivially winning the game, we require that the adversary $\mathcal{A}$ cannot query the evaluation oracle on x^* or $\mathbf{0}$.

Remarks. We note that a *selectively*-secure privately constrained PRF can be shown to be fully secure generically through complexity leveraging. In particular, the selectivity of the definition does not hurt the applicability of privacy as it can be shown to be adaptively secure generically. Achieving adaptive security for any kind of constrained PRFs without complexity leveraging (with polynomial loss in the reduction) remains a challenging problem. For puncturable PRFs, for instance, the only known adaptively secure constructions rely on the power of indistinguishability obfuscation([HKW15, HKKW14]).[4]

We also note that since constrained PRF is a symmetric-key notion, the setup algorithm just returns the master secret key msk. However, one can also consider dividing the setup into distinct parameter generation algorithm and seed generation algorithm where the parameters can be generated once and can be reused with multiple seeds for the PRF. In fact, for our construction in Section 5.1, a large part of the master secret key component can be fixed once and made public as parameters for the scheme. However, we maintain our current definition for simplicity.

3.3 Fully-Homomorphic Encryption

Following the presentation of [GVW15b], we give a minimal definition of fully homomorphic encryption (FHE) which is sufficient for this work. Technically, in this work, we use a leveled homomorphic encryption scheme (LHE); however, we will still refer to it simply as FHE. A leveled homomorphic encryption scheme 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:

- $\mathsf{HE.KeyGen}(1^\lambda, 1^d, 1^k) \to \mathsf{sk}$: On input the security parameter λ, a depth bound d, and a message length k, 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 $\mu \in \{0,1\}^k$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{HE.Eval}(C, \mathsf{ct}) \to \mathsf{ct}'$: On input a circuit $C \colon \{0,1\}^k \to \{0,1\}$ of depth d and a ciphertext ct, the homomorphic evaluation algorithm outputs ciphertext ct'.
- $\mathsf{HE.Dec}(\mathsf{sk}, \mathsf{ct}') \to \mu'$: On input a secret key sk and a ciphertext ct', the decryption algorithm outputs a message $\mu' \in \{0,1\}$.

Correctness. We require that for all λ, d, k, $\mathsf{sk} \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^d, 1^k)$, $\mu \in \{0,1\}^k$, and boolean circuits $C \colon \{0,1\}^k \to \{0,1\}$ of depth at most d, we have that

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

where the probability is taken over $\mathsf{HE.Enc}$ and $\mathsf{HE.KeyGen}$.

Security. For security, we require standard semantic security. For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, and for all $d, k = \mathsf{poly}(\lambda)$, there exists a negligible function negl such that

$$\Pr\left[b = b' : \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^d, 1^k); \\ \mu \leftarrow \mathcal{A}(1^\lambda, 1^d, 1^k); \\ b \xleftarrow{\$} \{0,1\}; \\ \mathsf{ct}_0 \leftarrow \mathsf{HE.Enc}(\mathsf{sk}, 0^{|\mu|}); \\ \mathsf{ct}_1 \leftarrow \mathsf{HE.Enc}(\mathsf{sk}, \mu); \\ b' \leftarrow \mathcal{A}(\mathsf{ct}_b) \end{array} \right] - \frac{1}{2} \leq \mathsf{negl}(\lambda)$$

[4] There are other adaptively secure constrained PRF constructions for prefix fixing and bit-fixing constraints as in [FKPR14, Hof14]; however, they too either require superpolynomial loss in the security parameter or rely on random oracles. The construction of [BV15] achieves adaptive security for the challenge point, but is selective with respect to the constraint.

4 LWE, SIS, Lattice FHE, and Matrix Embeddings

In this section, we present a brief background on the average case lattice problems of the Learning with Errors problem (LWE) as well as the one-dimensional Short Integer Solutions problem (1D-SIS). We also discuss the instantiations of FHE from LWE and summarize the circuit matrix embedding technique of the lattice ABE constructions.

Gaussian Distributions. We let $D_{\mathbb{Z}^m,\sigma}$ to be the discrete Gaussian distribution over $\mathbb{Z}^m$ with parameter σ. For simplicity, we truncate the distribution, which means that we replace the output by $\mathbf{0}$ whenever the norm $\|\cdot\|$ exceeds $\sqrt{m} \cdot \sigma$.

The LWE Problem. Let n, m, q be positive integers and χ be some noise distribution over $\mathbb{Z}_q$. In the $\mathsf{LWE}(n, m, q, \chi)$ problem, the adversary's goal is to distinguish between the two distributions:

$$(\mathbf{A}, \mathbf{s}^T\mathbf{A} + \mathbf{e}^T) \quad \text{and} \quad (\mathbf{A}, \mathbf{u}^T)$$

where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \xleftarrow{\$} \chi^n$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^m$ are uniformly sampled.

Connection to Worst-Case. Let $B = B(n) \in \mathbb{N}$. A family of distributions $\chi = \{\chi_n\}_{n\in\mathbb{N}}$ is called B-bounded if

$$\Pr[\chi \in \{-B, -B+1..., B-1, B\}] = 1.$$

For certain B-bounded error distributions χ, including the discrete Gaussian distributions[5], the $\mathsf{LWE}(n, m, q, \chi)$ problem is as hard as approximating certain worst-case lattice problems such as GapSVP and SIVP on n-dimensional lattices to within $\tilde{O}(n \cdot q/B)$ factor [Reg09, Pei09, ACPS09, MM11, MP12, BLP+13].

The Gadget Matrix. Let $\tilde{N} = n \cdot \lceil \log q \rceil$ and define the "gadget matrix" $\mathbf{G} = \mathbf{g} \otimes \mathbf{I}_n \in \mathbb{Z}_q^{n\times\tilde{N}}$ where $\mathbf{g} = (1, 2, 4, ..., 2^{\lceil \log q\rceil -1})$. We define the inverse function $\mathbf{G}^{-1} : \mathbb{Z}_q^{n\times m} \to \{0,1\}^{\tilde{N}\times m}$ which expands each entry $a \in \mathbb{Z}_q$ of the input matrix into a column of size $\lceil \log q \rceil$ consisting of the bits of the binary representation of a. To simplify the notation, we always assume that $\mathbf{G}$ has width m, which we do so without loss of generality as we can always extend the width of $\mathbf{G}$ by adding zero columns. We have the property that for any matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, it holds that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$.

The 1D-SIS Problem. Following the technique of [BV15], we use a variant of the Short Integer Solution (SIS) problem of [Ajt96] called 1D-SIS problem to show correctness and security for our scheme. Let m, β be positive integers and let q be a product of n prime moduli $p_1 < p_2 < \ldots < p_n$, $q = \prod_{i\in[n]} p_i$. Then, in the 1D-SIS$_{m,q,\beta}$, the adversary is given a uniformly random vector $\mathbf{v} \in \mathbb{Z}_q^m$ and its goal is to find $\mathbf{z} \in \mathbb{Z}^m$ such that $\|\mathbf{z}\| \le \beta$ and $\langle \mathbf{v}, \mathbf{z}\rangle = 0 \mod q$. For $m = O(n \log q)$, $p_1 \ge \beta \cdot \omega(\sqrt{mn \log n})$, the 1D-SIS-R$_{m,q,p,\beta}$ problem is as hard as approximating certain worst-case lattice problems such as GapSVP and SIVP to within $\beta \cdot \tilde{O}(\sqrt{mn})$ factor [Reg04, BV15].

For this work, we will use another variant called 1D-SIS-R that we define as follows. Let m, β be positive integers. We let $q = p \cdot \prod_{i\in[n]} p_i$, where all $p_1 < p_2 < \cdots < p_n$ are all co-prime and co-prime with p as well. In the 1D-SIS-R$_{m,q,p,\beta}$ problem, the adversary is given a uniformly random vector $\mathbf{v} \in \mathbb{Z}_q^m$ and its goal is to find a vector $\mathbf{z} \in \mathbb{Z}^m$ such that $\|\mathbf{z}\| \le \beta$ and $\langle \mathbf{v}, \mathbf{z}\rangle \in [-\beta, \beta] + (q/p)(\mathbb{Z} + 1/2)$.[6] In Appendix B, we show that 1D-SIS-R$_{m,q,p,\beta}$ is as hard as 1D-SIS$_{m,q,\beta}$ and therefore, is as hard as certain worst-case lattice problems.

4.1 FHE from LWE

There are a number of FHE constructions from LWE [BV14a, BGV12, GSW13, BV14b, ASP14, CM15, MW16, BP16, PS16]. For this work, we use the fact that these constructions can support not just binary, but field operations.

[5]By discrete Gaussian, we always mean the truncated discrete Gaussian.

[6]The term $(q/p)(\mathbb{Z} + 1/2)$ is a slight abuse of notation when q/p is not even. In this case, we mean $(q/p)\cdot(\mathbb{Z}) + \lfloor (q/p)\cdot(1/2)\rfloor$.

Specifically, given an encryption of a message $\mathbf{x} \in \{0,1\}^\ell$, a circuit $C\colon \{0,1\}^\ell \to \{0,1\}$, and any field element $w \in \mathbb{Z}_q$, one can homomorphically compute the function

$$f_{C,w}(\mathbf{x}) = w \cdot C(\mathbf{x}) \in \mathbb{Z}_q$$

on the ciphertext. Here, we take advantage of the fact that the FHE homomorphic operations can support scalar multiplication by a field element without increasing the noise too much. Looking ahead, we will encrypt the punctured point $\mathbf{x}^*$, and homomorphically compute the equality predicate $\mathsf{eq}_{\mathbf{x}}(\mathbf{x}^*) = \begin{cases} 1 & \mathbf{x} = \mathbf{x}^* \\ 0 & \text{otherwise} \end{cases}$ on the ciphertext such that it decrypts to a random element $w_\gamma \in \mathbb{Z}_q$ only if the evaluation of the PRF a point $\mathbf{x}$ equals to the punctured point. This is simply evaluating the equality check circuit on the FHE ciphertext and scaling the result by w_γ.

We formally summarize the properties of FHE constructions from LWE below.[7]

Theorem 4.1 (FHE from LWE)**.** *Fix a security parameter λ and depth bound $d = d(\lambda)$. Let n, m, q, χ be LWE parameters where χ is a B-bounded error distribution and $q > B \cdot m^{O(d)}$. Then, there is an FHE scheme* $\Pi_{\mathsf{HE}} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Eval}, \mathsf{HE.Dec})$ *for circuits of depth bound d, with the following properties:*

- HE.KeyGen *outputs a secret key* $\mathsf{sk} \in \mathbb{Z}_q^n$
- HE.Enc *takes in a message* $\mathbf{m} \in \{0,1\}^k$ *and outputs a ciphertext* $\mathsf{ct} \in \{0,1\}^z$ *where* $z = \mathsf{poly}(\lambda, d, \log q, k)$.
- HE.Eval *takes in a circuit* $f_{C,w}$ *and a ciphertext* ct *and outputs ciphertexts* $\mathsf{ct}' \in \{0,1\}^n$.
- *For any boolean circuit C of depth d and scalar element $w \in \mathbb{Z}_q$,* $\mathsf{HE.Eval}(f_{C,w}, \cdot)$ *is computed by a boolean circuit of depth* $\mathsf{poly}(d, \log z)$.
- HE.Dec *on input* sk *and* ct*, when $C(\mathbf{m}) = 1$ we have that*

$$\sum_{i=1}^{t} \mathsf{sk}[i] \cdot \mathsf{ct}[i] \in [w - E, w + E].$$

  When $C(\mathbf{m}) = 0$ we have

$$\sum_{i=1}^{t} \mathsf{sk}[i] \cdot \mathsf{ct}[i] \in [-E, E]$$

  for some bound $E = B \cdot m^{O(d)}$.
- *Security relies on* $\mathsf{LWE}(n, m, q, \chi)$.

We note that in the predicate encryption construction of [GVW15b], the result of [BV14b] is used, which applies the *sequential* homomorphic multiplication of ciphertexts (through branching programs) to take advantage of the asymmetric noise growth of FHE. This allows the final noise from the FHE homomorphic operations to be bounded by $\mathsf{poly}(\lambda)$, but the depth of the FHE evaluation grows polynomially in the bit length of the FHE modulus. In our construction, this optimization is not needed because we will only be concerned with the equality check circuit which is already only logarithmic in the depth of the input length. Therefore, one can perform regular FHE homomorphic operations with depth logarithmic in the bit length of the FHE modulus.

4.2 Matrix Embeddings

In the ABE construction of [BGG+14], Boneh *et al.* introduced a method to embed circuits into LWE matrices and since then, the technique saw a number of applications in lattice-based constructions [BV15, GVW15b, GV15, BV16, BCTW16].

We provide an overview of this technique since our proof of security will rely on the specifics of this matrix encodings. Our, description will be informal, but we formally describe the properties that we need for the proofs below. We refer the readers to [BGG+14, GVW15b] for the formal treatment.

[7] We slightly abuse the FHE syntax in Section 3.3.

In the setting of [BGG+14], for the set of public matrix $\mathbf{A}_1, ...\mathbf{A}_\ell$, we encode a vector of field elements $\mathbf{x} \in \mathbb{Z}_q^t$ as an LWE sample as

$$\mathbf{a}_{x_i} = \mathbf{s}^T(\mathbf{A}_i + x_i \cdot \mathbf{G}) + \mathbf{e}_i$$

for $i = 1, ..., \ell$ where $\mathbf{s}$ and $\mathbf{e}_i$'s are sampled according to the standard LWE distribution. Then, given two encodings, $\mathbf{a}_{x_i}$, $\mathbf{a}_{x_j}$, we can add and multiply them as follows:

$$\begin{aligned}
\mathbf{a}_{x_i+x_j} &= \mathbf{a}_{x_i} + \mathbf{a}_{x_j} \\
&= \mathbf{s}^T(\mathbf{A}_i + x_i \cdot \mathbf{G}) + \mathbf{e}_i + \mathbf{s}^T(\mathbf{A}_j + x_j \cdot \mathbf{G}) + \mathbf{e}_j \\
&= \mathbf{s}^T([\mathbf{A}_i + \mathbf{A}_j] + [x_i + x_j] \cdot \mathbf{G}) + [\mathbf{e}_i + \mathbf{e}_j] \\
&= \mathbf{s}^T(\mathbf{A}_{+,i,j} + [x_i + x_j] \cdot \mathbf{G}) + \mathbf{e}_{+,i,j}
\end{aligned}$$

$$\begin{aligned}
\mathbf{a}_{x_i \times x_j} &= \mathbf{a}_{x_i} \cdot x_j - \mathbf{a}_{x_j}\mathbf{G}^{-1}(\mathbf{A}_i) \\
&= \mathbf{s}^T(x_j\mathbf{A}_i + x_i x_j \cdot \mathbf{G}) + x_j\mathbf{e}_j - \mathbf{s}^T(\mathbf{A}_j\mathbf{G}^{-1}(\mathbf{A}_i) + x_j\mathbf{A}_i) + \mathbf{e}_j\mathbf{G}^{-1}(\mathbf{A}_i) \\
&= \mathbf{s}^T([-\mathbf{A}_j\mathbf{G}^{-1}(\mathbf{A}_i)] + [x_i \cdot x_j] \cdot \mathbf{G}) + [x_j\mathbf{e}_i + \mathbf{e}_j\mathbf{G}^{-1}(\mathbf{A}_i) \\
&= \mathbf{s}^T(\mathbf{A}_{\times,i,j} + [x_i \cdot x_j] \cdot \mathbf{G}) + \mathbf{e}_{\times,i,j}
\end{aligned}$$

Correspondingly, we can define operations on the matrices

- $\mathbf{A}_{+,i,j} = \mathbf{A}_i + \mathbf{A}_j$
- $\mathbf{A}_{\times,i,j} = -\mathbf{A}_j\mathbf{G}^{-1}(\mathbf{A}_i)$

Using these operations, one can compute an arithmetic circuit F on the encodings gate-by-gate. In particular, restricting $\mathbf{x}$ to be a binary string, we can compute the NAND operation as

$$\mathbf{a}_{\neg(x_i \wedge x_j)} = \mathbf{a}_1 - \mathbf{a}_{x_i \times x_j}$$

$$\mathbf{A}_{\neg(x_i \wedge x_j)} = \mathbf{A}^* - \mathbf{A}_{\times,i,j}$$

where $\mathbf{a}_1 = \mathbf{s}^T(\mathbf{A}^* + \mathbf{G}) + \mathbf{e}^*$ is a fixed encoding of 1.

We note that in the description above, to compute a single multiplication on the encodings $\mathbf{a}_{x_i}$, $\mathbf{a}_{x_j}$, one must know one of x_i or x_j, but it is not required to know both. This means that computing operations such as inner products on two vector attributes can be done without the knowledge of one of the vectors. In particular, given the encodings of $(\mathbf{x}, \mathbf{w}) \in \{0,1\}^z \times \mathbb{Z}_q^t$, and a pair $(C, \mathbf{x})$ where $C : \{0,1\}^z \to \{0,1\}^t$ and $\mathbf{x} \in \{0,1\}^z$, one can derive an encoding of $(\mathsf{IP} \circ C)(\mathbf{x}, \mathbf{w}) = \langle C(\mathbf{x}), \mathbf{w} \rangle$.

Theorem 4.2 ([BGG+14, GVW15b])**.** *Fix a security parameter λ, and lattice parameters n, m, q, χ where χ is a B-bounded error distribution. Let C be a depth-d Boolean circuit on z input bits. Let $\mathbf{A}_1, ..., \mathbf{A}_z, \tilde{\mathbf{A}}_1, ..., \tilde{\mathbf{A}}_t \in \mathbb{Z}_q^{n \times m}$, $(x_1, \mathbf{b}_1), ..., (x_z, \mathbf{b}_z) \in \{0,1\} \times \mathbb{Z}_q^m$, and $(w_1, \tilde{\mathbf{b}}_1), ..., (w_t, \tilde{\mathbf{b}}_t) \in \mathbb{Z}_q \times \mathbb{Z}_q^m$ such that*

$$\left\|\mathbf{b}_i^T - \mathbf{s}^T(\mathbf{A}_i + x_i \cdot \mathbf{G})\right\| \leq B \quad \text{for } i = 1, ..., z$$

$$\left\|\tilde{\mathbf{b}}_j^T - \mathbf{s}^T(\tilde{\mathbf{A}}_j + w_j \cdot \mathbf{G})\right\| \leq B \quad \text{for } j = 1, ..., t$$

for some $\mathbf{s} \in \mathbb{Z}_q^n$. There exists the following pair of algorithms

- $\mathsf{Eval}_{\mathsf{pk}}((\mathsf{IP} \circ C), \mathbf{A}_1, ..., \mathbf{A}_z, \tilde{\mathbf{A}}_1, ..., \tilde{\mathbf{A}}_t) \to \mathbf{A}_{(\mathsf{IP} \circ C)}$*: On input a circuit* $(\mathsf{IP} \circ C)$ *for* $C : \{0,1\}^z \to \{0,1\}^t$ *and* $z + t$ *matrices* $\mathbf{A}_1, ..., \mathbf{A}_z, \tilde{\mathbf{A}}_1, ..., \tilde{\mathbf{A}}_t$*, outputs a matrix* $\mathbf{A}_{(\mathsf{IP} \circ C)}$.
- $\mathsf{Eval}_{\mathsf{ct}}((\mathsf{IP} \circ C), \mathbf{b}_1, ..., \mathbf{b}_z, \tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_t, \mathbf{x}) \to \mathbf{b}_{(\mathsf{IP} \circ C)}$*: On input a circuit* $(\mathsf{IP} \circ C)$ *for* $C : \{0,1\}^z \to \{0,1\}^t$*,* $z + t$ *vectors* $\mathbf{b}_1, ..., \mathbf{b}_z, \tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_t$*, and length* z *string* $\mathbf{x}$*, outputs a vector* $\mathbf{b}_{(\mathsf{IP} \circ C)}$

such that for $\mathbf{A}_{(\mathsf{IP}\circ C)} \leftarrow \mathsf{Eval}_{\mathsf{pk}}((\mathsf{IP}\circ C), \mathbf{A}_1, ..., \mathbf{A}_z, \tilde{\mathbf{A}}_1, ..., \tilde{\mathbf{A}}_t)$*, and* $\mathbf{b}_{(\mathsf{IP}\circ C)} \leftarrow \mathsf{Eval}_{\mathsf{ct}}((\mathsf{IP}\circ C), \mathbf{b}_1, ..., \mathbf{b}_z, \tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_t, \mathbf{x})$*, we have that*

$$\left\| \mathbf{b}^T_{(\mathsf{IP}\circ C)} - \mathbf{s}^T(\mathbf{A}_{(\mathsf{IP}\circ C)} + \langle C(\mathbf{x}), \mathbf{w}\rangle \cdot \mathbf{G}) \right\| \leq B \cdot m^{O(d)}.$$

Moreover, $\mathbf{b}_{(\mathsf{IP}\circ C)}$ *is a "low-norm" linear function of* $\mathbf{b}_1, ..., \mathbf{b}_z, \tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_z$*. That is, there are matrices* $\mathbf{R}_1, ..., \mathbf{R}_z, \tilde{\mathbf{R}}_1, ..., \tilde{\mathbf{R}}_t$ *such that* $\mathbf{b}^T_{(\mathsf{IP}\circ C)} = \sum_{i=1}^{z} \mathbf{b}_i^T \mathbf{R}_i + \sum_{j=1}^{t} \tilde{\mathbf{b}}_j^T \tilde{\mathbf{R}}_j$ *and* $\|\mathbf{R}_i\|, \|\tilde{\mathbf{R}}_j\| \leq m^{O(d)}$*.*

5 Main Construction

In this section, we present our private puncturable PRF. We first give a formal description of the construction followed by a sample instantiation of the parameters used in the construction. Then, we prove security followed by a correctness analysis. We conclude the section with some extensions.

5.1 Construction

Our construction uses a number of parameters and indices, which we list here for reference:

- (n, m, q, χ) - LWE parameters
- ℓ - length of the PRF input
- p - rounding modulus
- z - size of FHE ciphertext (indexed by i)
- t - size FHE secret key (indexed by j)
- d' - depth of the equality check circuit
- d - depth of the circuit that computes the FHE homomorphic operation of equality check
- γ - index for the randomizers $w_1, ..., w_n$

For $\gamma \in [n]$ we use $\mathbf{u}_\gamma$ to denote the n dimensional basis vector in $\mathbb{Z}_q^n$ with γth entry set to 1 and the rest set to 0. Also, for $\gamma \in [n]$, we denote by $\mathbf{v}_\gamma$ the m dimensional basis vector in $\mathbb{Z}_q^m$ with the $\gamma \cdot (\lceil \log q \rceil - 1)$th component set to 1 and the rest set to 0. By construction of $\mathbf{G}$ we have that $\mathbf{G} \cdot \mathbf{v}_\gamma = \mathbf{u}_\gamma$.

For the cleanest way to describe the construction, we slightly abuse notation and define the setup algorithm $\mathsf{pPRF.Setup}$ to also publish a set of public parameters pp along with the master secret key msk. One can view pp as a fixed set of parameters for the whole system that is available to each algorithms $\mathsf{pPRF.Puncture}$, $\mathsf{pPRF.PunctureEval}$, $\mathsf{pPRF.Eval}$, or it can be viewed as a component included in both the master secret key msk and the punctured key $\mathsf{sk}_{\mathbf{x}^*}$.

Fix a security parameter λ. We construct a privately puncturable PRF $\Pi_{\mathsf{pPRF}} = (\mathsf{pPRF.Setup}, \mathsf{pPRF.Puncture}, \mathsf{pPRF.PunctureEval}, \mathsf{pPRF.Eval})$ with domain $\{0,1\}^\ell$ and range $\mathbb{Z}_p$ as follows:

- $\mathsf{pPRF.Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm generates a set of uniformly random matrices in $\mathbb{Z}_q^{n\times m}$:
  - $\mathbf{A}_0, \mathbf{A}_1$ that will encode the input to the PRF
  - $\mathbf{B}_1, ..., \mathbf{B}_z$ that will encode the FHE ciphertext
  - $\mathbf{C}_1, ..., \mathbf{C}_t$ that will encode the FHE secret key

  Then, it generates a secret vector $\mathbf{s}$ from the error distribution $\mathbf{s} \leftarrow \chi^n$, and also samples a uniformly random vector $\mathbf{w} \in \mathbb{Z}_q^n$. It sets

  $$\mathsf{pp} = \left(\{\mathbf{A}_b\}_{b\in\{0,1\}},\ \{\mathbf{B}_i\}_{i\in[z]},\ \{\mathbf{C}_j\}_{j\in[t]},\ \mathbf{w}\right) \quad \text{and} \quad \mathsf{msk} = \mathbf{s}$$

- $\mathsf{pPRF.Eval}(\mathsf{msk}, \mathbf{x})$: On input the master secret key $\mathsf{msk} = \mathbf{s}$ and the PRF input $\mathbf{x}$, the evaluation algorithm first computes

  $$\tilde{\mathbf{B}}_\gamma \leftarrow \mathsf{Eval}_{\mathsf{pk}}(C_\gamma, \mathbf{B}_1, ..., \mathbf{B}_z, \mathbf{A}_{x_1}, ..., \mathbf{A}_{x_\ell}, \mathbf{C}_1, ..., \mathbf{C}_t)$$

where for $\gamma \in [n]$ the circuit C_γ is defined as $C_\gamma(\cdot) = \mathsf{IP} \circ \mathsf{HE.Eval}(\mathsf{eq}_{w_\gamma}, \cdot)$ and the equality check function eq_{w_γ} is defined as:

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

The algorithm outputs the following as the PRF value:

$$\left\lfloor \sum_{\gamma \in [n]} \left\langle \mathbf{s}^T \tilde{\mathbf{B}}_\gamma, \mathbf{v}_\gamma \right\rangle \right\rfloor_p \in \mathbb{Z}_p.$$

- $\mathsf{pPRF.Puncture}(\mathsf{msk}, \mathbf{x}^*)$: given msk and the point to be punctured $\mathbf{x}^* = (x_1^*, ..., x_\ell^*) \in \{0,1\}^\ell$ as input, the puncturing algorithm generates an FHE key $\mathsf{he.sk} \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^{d'}, 1^\ell)$ and encrypts $\mathbf{x}^*$ as

  $$\mathsf{he.ct} \leftarrow \mathsf{HE.Enc}\big(\mathsf{he.sk},\ (x_1^*, ..., x_\ell^*)\big) \in \mathbb{Z}_q^z.$$

  Then, it samples an error vector $\mathbf{e} \leftarrow \chi^{2+z+t}$ from the error distribution and computes

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

  It outputs the punctured key $\mathsf{sk}_{\mathbf{x}^*} = \big(\{\mathbf{a}_b\}_{b \in \{0,1\}},\ \{\mathbf{b}_i\}_{i \in [z]},\ \{\mathbf{c}_j\}_{j \in [t]},\ \mathsf{he.ct}\big)$.

- $\mathsf{pPRF.PunctureEval}(\mathsf{sk}_{\mathbf{x}^*}, \mathbf{x})$: On input a punctured key $\mathsf{sk}_{\mathbf{x}^*} = \big(\{\mathbf{a}_b\}_{b \in \{0,1\}},\ \{\mathbf{b}_i\}_{i \in [z]},\ \{\mathbf{c}_j\}_{j \in [t]},\ \mathsf{he.ct}\big)$ and $\mathbf{x} \in \{0,1\}^\ell$, the puncture evaluation algorithm runs

  $$\tilde{\mathbf{b}}_\gamma \leftarrow \mathsf{Eval}_{\mathsf{ct}}\big(C_\gamma,\ \mathbf{b}_1, ..., \mathbf{b}_z,\ \mathbf{a}_{x_1}, ..., \mathbf{a}_{x_\ell},\ \mathbf{c}_1, ..., \mathbf{c}_t,\ (\mathsf{he.ct}, \mathbf{x})\ \big)$$

  for $\gamma = 1, ..., n$. Here C_γ is the circuit defined as in algorithm $\mathsf{pPRF.Eval}$. The puncture evaluation algorithm then outputs the PRF value:

  $$\left\lfloor \sum_{\gamma \in [n]} \left\langle \tilde{\mathbf{b}}_\gamma, \mathbf{v}_\gamma \right\rangle \right\rfloor_p \in \mathbb{Z}_p.$$

As discussed in Section 3.2, we can de-randomize algorithm $\mathsf{pPRF.Puncture}$ so that it always returns the same output when run on the same input.

5.2 Parameters

The parameters can be instantiated such that breaking correctness or security translates to solving worst-case lattice problems to $2^{\tilde{O}(n^{1/c})}$ for some constant c. We set the parameters to account for the noise of both (a) the FHE decryption and (b) the homomorphic computation on the ABE encodings. The former will be bounded largely by $B \cdot m^{O(d')}$ and the latter by $B \cdot m^{O(d)}$. Here, d' is the depth of the equality check circuit and d is the depth of the *FHE operation* of the equality check circuit. We want to set the modulus of the encodings q to be big enough to account for these bounds. Furthermore, for the 1D-SIS-R assumption, we need q to be the product of coprime moduli $p_1, ..., p_\lambda$ such that the smallest of these primes exceeds these bounds.

Sample instantiations: We first set the PRF input length $\ell = \mathsf{poly}(\lambda)$. The depth of the equality check circuit is then $d' = O(\log \ell)$. We set $n = \lambda^{2c}$. We define q to be the product of λ coprime moduli $p, p_1, ..., p_\lambda$ where we set $p = \mathsf{poly}(\lambda)$ and for each $i \in [\lambda]$, $p_i = 2^{O(n^{1/2c})}$ such that $p_1 < \ldots < p_\lambda$. The noise distribution χ is set to be the discrete Gaussian distribution $D_{\mathbb{Z}, \sqrt{n}}$. Then the FHE ciphertext size z and the secret key size t is determined by q. Set $m = \Theta(n \log q)$. The depth of the FHE equality check circuit is $d = \mathsf{poly}(d', \log z)$.

5.3 Security

We next prove security of the construction. We start by describing a set of auxiliary algorithms that we will use in the description of the hybrids in the security proof.

5.3.1 Auxiliary Algorithms

We define the following set of auxiliary algorithms.[8]

- $\mathsf{Setup}^*(1^\lambda, \mathbf{x}^*) \to (\mathsf{pp}^*, \mathsf{msk}^*)$: On input a point to be punctured $\mathbf{x}^* \in \{0,1\}^\ell$, the setup algorithm first generates $\mathsf{he.sk} \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^d, 1^\ell)$ and encrypts $\mathsf{he.ct} \leftarrow \mathsf{HE.Enc}(\mathsf{he.sk}, (x_1^*, ..., x_\ell^*))$. It sets the public matrices

$$\begin{aligned} \mathbf{A}_b &= \mathbf{A}'_b - b \cdot \mathbf{G} && \text{where} && \mathbf{A}'_b \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, && \forall b \in \{0,1\} \\ \mathbf{B}_i &= \mathbf{B}'_i - \mathsf{he.ct}_i \cdot \mathbf{G} && \text{where} && \mathbf{B}'_i \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, && \forall i \in [z] \\ \mathbf{C}_j &= \mathbf{C}'_j - \mathsf{he.sk}_j \cdot \mathbf{G} && \text{where} && \mathbf{C}'_j \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, && \forall j \in [t]. \end{aligned}$$

  It samples a secret vector $\mathbf{s}$ from the error distribution $\mathbf{s} \leftarrow \chi^n$ and also samples a uniformly random vector $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_q^n$. It sets

$$\mathsf{pp}^* = \big(\{\mathbf{A}_b\}_{b\in\{0,1\}},\ \{\mathbf{B}_i\}_{i\in[z]},\ \{\mathbf{C}_j\}_{j\in[t]},\ \mathbf{w}\big) \quad \text{and} \quad \mathsf{msk}^* = \big(\mathbf{s}, \mathsf{he.sk}, \mathsf{he.ct}, \mathbf{x}^*\big)$$

  and outputs $(\mathsf{pp}^*, \mathsf{msk}^*)$.

- $\mathsf{Puncture}_1^*(\mathsf{msk}^*) \to \mathsf{sk}_{\mathbf{x}^*}$: On input the master secret key $\mathsf{msk}^* = \big(\mathbf{s}, \mathsf{he.sk}, \mathsf{he.ct}, \mathbf{x}^*\big)$ the puncture algorithm samples error vectors $\mathbf{e}_k \leftarrow \chi^{2+z+t}$ from the error distributions and computes

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

  It then sets $\mathsf{sk}_{\mathbf{x}^*} = \big(\{\mathbf{a}_b\}_{b\in\{0,1\}},\ \{\mathbf{b}_i\}_{i\in[z]},\ \{\mathbf{c}_j\}_{j\in[t]},\ \mathsf{he.ct}\big)$.

- $\mathsf{Puncture}_2^*(\mathsf{msk}^*) \to \mathsf{sk}_{\mathbf{x}^*}$: On input the master secret key $\mathsf{msk}^* = (\mathbf{s}, \mathsf{he.sk}, \mathsf{he.ct}, \mathbf{x}^*)$, the puncture algorithm instantiates $\{\mathbf{a}_b\}_{b\in\{0,1\}}, \{\mathbf{b}_i\}_{i\in[z]}, \{\mathbf{c}_j\}_{j\in[t]}$ with uniformly random vectors in $\mathbb{Z}_q^m$. It sets $\mathsf{sk}_{\mathbf{x}^*} = \big(\{\mathbf{a}_b\}_{b\in\{0,1\}}, \{\mathbf{b}_i\}_{i\in[z]}, \{\mathbf{c}_j\}_{j\in[t]}, \mathsf{he.ct}\big)$.

- $\mathsf{Eval}^*(\mathsf{msk}^*, \mathsf{sk}_{\mathbf{x}^*}, \mathbf{x}) \to \tilde{\mathbf{y}}$: On input the master secret key $\mathsf{msk}^* = \big(\mathbf{s}, \mathsf{he.sk}, \mathsf{he.ct}, \mathbf{x}^*\big)$, $\mathsf{sk}_{\mathbf{x}^*} = \big(\{\mathbf{a}_b\}_{b\in\{0,1\}}, \{\mathbf{b}_i\}_{i\in[z]}, \{\mathbf{c}_j\}_{j\in[t]}, \mathsf{ct}^*\big)$, and $\mathbf{x}$, the evaluation algorithm runs

$$\tilde{\mathbf{b}}_\gamma \leftarrow \mathsf{Eval}_{\mathsf{ct}}\big(C_\gamma, \mathbf{b}_1, ..., \mathbf{b}_z, \mathbf{a}_{x_1}, ..., \mathbf{a}_{x_\ell}, \mathbf{c}_1, ..., \mathbf{c}_t, (\mathsf{he.ct}, \mathbf{x})\ \big)$$

  for $\gamma = 1, ..., n$. Then, if $\mathbf{x} \neq \mathbf{x}^*$, it returns

$$\tilde{y} = \left\lfloor \sum_{\gamma\in[n]} \left\langle \tilde{\mathbf{b}}_\gamma, \mathbf{v}_\gamma \right\rangle \right\rceil_p \in \mathbb{Z}_p.$$

  If $\mathbf{x} = \mathbf{x}^*$, then it samples an error term $e_{\tilde{w}} \leftarrow \chi$, computes $\tilde{w} = \langle \mathbf{s}, \mathbf{w} \rangle + e_{\tilde{w}} \in \mathbb{Z}_q$ and outputs

$$\tilde{y} = \left\lfloor \sum_{\gamma\in[n]} \left\langle \tilde{\mathbf{b}}_\gamma, \mathbf{v}_\gamma \right\rangle - \tilde{w} \right\rceil_p \in \mathbb{Z}_p.$$

[8] As was the case in the description of the main construction, we assume that all algorithms excluding the setup implicitly takes in pp^* as part of its input.

5.3.2 Security Proof

We start with showing the pseudorandomness property of our construction.

Theorem 5.1. *The puncturable PRF from Section 5.1 with parameters instantiated as in Section 5.2 is* selectively-pseudorandom *as defined in Definition 3.6 assuming the hardness of $\mathit{LWE}_{n,m',q,\chi}$ and 1D-SIS-$R_{q,p,\beta,m'}$ for $\beta = B \cdot m^{\tilde{O}(d)}$ and $m' = m \cdot (2 + z + t) + 1$.*

Proof. Recall that in the selective pseudorandomness game (Definition 3.6), the adversary first commits to a puncture point $\mathbf{x}^*$. Then, for $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{pPRF.Setup}(1^\lambda)$ and $\mathsf{sk}^*_{\mathbf{x}} \leftarrow \mathsf{pPRF.Puncture}(\mathsf{msk}, \mathbf{x}^*)$, the adversary is provided with pp, $\mathsf{sk}^*_{\mathbf{x}}$, and an oracle access to the real evaluation algorithm $\mathsf{pPRF.Eval}(\mathsf{msk}, \cdot)$ on $\mathcal{X} \backslash \{\mathbf{x}^*\}$, along with a challenge $y^* \in \mathcal{Y}$.[9] Its goal is to distinguish whether the challenge y^* was generated using the real evaluation algorithm $y^* \leftarrow \mathsf{pPRF.Eval}(\mathsf{msk}, \mathbf{x}^*)$ or sampled uniformly $y^* \xleftarrow{\$} \mathbb{Z}_p$.

We proceed through a series of hybrid experiments where we start with the real experiment H_0 where the challenger provides the adversary with a real PRF evaluation at the punctured point. The final hybrid experiment H_5 is the ideal experiment where the challenger gives the adversary a randomly sampled element in the range for the PRF evaluation at the punctured point. We start with the precise hybrid descriptions.

Hybrid descriptions. For the hybrid descriptions, we use the auxiliary algorithms defined in the previous subsection.

- **Hybrid H_0:** This is the real experiment. The challenger first receives the commitment to the challenge point $\mathbf{x}^*$ from the adversary $\mathcal{A}$. It generates the public parameters pp and the master secret key msk as in the real scheme $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{pPRF.Setup}(1^\lambda)$. Using msk, it generates the punctured key $\mathsf{sk}_{\mathbf{x}^*} \leftarrow \mathsf{pPRF.Puncture}(\mathsf{msk}, \mathbf{x}^*)$ and the challenge PRF evaluation $y^* \leftarrow \mathsf{pPRF.Eval}(\mathsf{msk}, \mathbf{x}^*)$. The honest PRF evaluation queries are answered using the real evaluation algorithm $\mathsf{pPRF.Eval}(\mathsf{msk}, \cdot)$.

- **Hybrid H_1:** In this experiment, we change the setup and the puncturing algorithm by precomputing the point to be punctured during setup. More precisely, given the commitment to the challenge point $\mathbf{x}^*$ from the adversary $\mathcal{A}$, the challenger runs the auxiliary setup $(\mathsf{pp}^*, \mathsf{msk}^*) \leftarrow \mathsf{Setup}^*(1^\lambda, \mathbf{x}^*)$ and generates the punctured key with the auxiliary puncture algorithm $\mathsf{sk}_{\mathbf{x}^*} \leftarrow \mathsf{Puncture}^*_1(\mathsf{msk}^*)$. It generates the challenge PRF evaluation and answers the regular PRF evaluation queries still using the real evaluation algorithm $\mathsf{pPRF.Eval}(\mathsf{msk}, \cdot)$, where msk is the secret vector s that is part of msk^*.

- **Hybrid H_2:** This experiment is the same as H_1, but we change the way the challenger computes the PRF evaluations. Instead of evaluating the PRF with the master secret key, the challenger computes the PRF evaluation with the punctured key. More precisely, for the challenge PRF evaluation, the challenger computes $y^* \leftarrow \mathsf{Eval}^*(\mathsf{msk}^*, \mathsf{sk}_{\mathbf{x}^*}, \mathbf{x}^*)$ and for each of the regular PRF evaluation queries $\mathbf{x}$ from the adversary, the challenger returns $y_{\mathbf{x}} \leftarrow \mathsf{Eval}^*(\mathsf{msk}^*, \mathsf{sk}_{\mathbf{x}^*}, \mathbf{x})$.

- **Hybrid H_3:** This experiment is the same as H_2, but now, the challenger provides the adversary with a randomly sampled element in the range as the challenge PRF evaluation. Also, for the punctured key, the challenger replaces the "ABE component" of the punctured key with random strings. More precisely, for the punctured key, the challenger generates $\mathsf{sk}_{\mathbf{x}^*} \leftarrow \mathsf{Puncture}^*_2(\mathsf{msk}^*)$. For the challenge PRF evaluation, it samples a uniform element $y^* \xleftarrow{\$} \mathcal{Y}$ and provides it to the adversary. It answers the regular PRF evaluation queries in the same way as in H_2.

- **Hybrid H_4:** From this experiment, we start "unrolling back" the changes that we made from the previous hybrid experiments except for the challenge PRF evaluation. In this experiment, we change the way we generate the punctured key. Instead of using the auxiliary puncturing algorithm $\mathsf{Puncture}^*_2(\mathsf{msk}^*)$, the challenger generates the punctured key $\mathsf{sk}_{\mathbf{x}^*} \leftarrow \mathsf{Puncture}^*_1(\mathsf{msk}^*)$.

[9] Recall that we assume pp as a public component that is part of both msk and $\mathsf{sk}^*_{\mathbf{x}}$.

- **Hybrid H_5:** This experiment is the same as H_4 except that for the regular PRF evaluation queries, the challenger uses the real evaluation $\mathsf{pPRF.Eval}(\mathsf{msk}, \cdot)$. The way the challenger generates the challenge queries remains unchanged.
- **Hybrid H_6:** This experiment is the same as H_5 except now, instead of using the auxiliary setup and puncture algorithms, the challenger uses the honest setup $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{pPRF.Setup}(1^\lambda)$ and the honest puncture $\mathsf{sk}_{\mathbf{x}^*} \leftarrow \mathsf{pPRF.Puncture}(\mathsf{msk}, \mathbf{x}^*)$. We note that this experiment is also identical to H_0 except for the challenge evaluation query, which the challenger samples uniformly $y^* \xleftarrow{\$} \mathcal{Y}$.

Analysis. Now we proceed in arguing the indistinguishability of each of the hybrids.

Lemma 5.2. *The hybrid experiments* H_0 *and* H_1 *are perfectly indistinguishable.*

Proof. It is easy to see that in both experiments H_0 and H_1, the public parameters pp are distributed uniformly. Also, in H_0, the challenger generates the FHE ciphertext when it receives the point $\mathbf{x}^*$ to be punctured whereas in H_1, the challenger computes the FHE ciphertext at the point to be punctured before it sets pp. Since the challenger is only precomputing the ciphertext, the distribution of the punctured key remains unchanged. □

We now consider the indistinguishability of H_1 and H_2. In H_1, the challenger evaluates the PRF with the real evaluation algorithm $\mathsf{pPRF.Eval}(\mathsf{msk}^*, \cdot)$ and in H_2, the challenger uses the auxiliary evaluation algorithm $\mathsf{Eval}^*(\mathsf{msk}^*, \mathsf{sk}_{\mathbf{x}^*}, \cdot)$. We show that the adversary's view in both experiments is identical, provided that a certain "bad event" does not occur. To define this event formally, we first recall the definitions of $\mathsf{Eval}^*(\mathsf{msk}^*, \mathsf{sk}_{\mathbf{x}^*}, \cdot)$ and $\mathsf{pPRF.Eval}(\mathsf{msk}^*, \cdot)$.

- The auxiliary evaluation algorithm $\mathsf{Eval}^*(\mathsf{msk}^*, \mathsf{sk}_{\mathbf{x}^*}, \mathbf{x})$ takes the encodings $\{\mathbf{a}_b\}_{b\in\{0,1\}}, \{\mathbf{b}_i\}_{i\in[z]}, \{\mathbf{c}_j\}_{j\in[t]}$ from the punctured key and computes the vectors $\{\tilde{\mathbf{b}}_\gamma\}_{\gamma\in[n]}$ defined as

$$\tilde{\mathbf{b}}_\gamma \leftarrow \mathsf{Eval}_{\mathsf{ct}}\big(C_\gamma, \mathbf{b}_1, ..., \mathbf{b}_z, \mathbf{a}_{x_1}, \ldots, \mathbf{a}_{x_\ell}, \mathbf{c}_1, ..., \mathbf{c}_t, (\mathsf{he.ct}, \mathbf{x})\ \big) \tag{5.1}$$

  and then returns the auxiliary PRF output as

$$\tilde{y} = \left\lfloor \sum_{\gamma\in[n]} \left\langle \tilde{\mathbf{b}}_\gamma, \mathbf{v}_\gamma \right\rangle - \mathsf{eq}(\mathbf{x}, \mathbf{x}^*) \cdot \tilde{w} \right\rceil_p \quad \text{where} \quad \tilde{w} = \langle \mathbf{s}, \mathbf{w} \rangle + e_{\tilde{w}}. \tag{5.2}$$

- The real evaluation algorithm $\mathsf{pPRF.Eval}(\mathsf{msk}, \cdot)$ first computes

$$\tilde{\mathbf{B}}_\gamma \leftarrow \mathsf{Eval}_{\mathsf{pk}}\big(C_\gamma, \mathbf{B}_1, ..., \mathbf{B}_z, \mathbf{A}_{x_1}, ..., \mathbf{A}_{x_\ell}, \mathbf{C}_1, ..., \mathbf{C}_t\big)$$

  and returns the value

$$\tilde{y}' = \left\lfloor \sum_{\gamma\in[n]} \left\langle \mathbf{s}^T \tilde{\mathbf{B}}_\gamma, \mathbf{v}_\gamma \right\rangle \right\rceil_p .$$

We can express the quantity $\tilde{y}'$ using the vectors $\{\tilde{\mathbf{b}}_\gamma\}_{\gamma\in[n]}$ defined in (5.1), and the quantity $\tilde{w} \in \mathbb{Z}_q$ defined in (5.2). First, by the correctness of FHE (Theorem 4.1) and the correctness of the ABE encodings (Theorem 4.2), the vector $\tilde{\mathbf{b}}_\gamma$ satisfies:

$$\tilde{\mathbf{b}}_\gamma^T = \mathbf{s}^T \left(\tilde{\mathbf{B}}_\gamma + (w_\gamma \cdot \mathsf{eq}(\mathbf{x}, \mathbf{x}^*) + \epsilon_\gamma) \cdot \mathbf{G} \right) + \mathbf{e}_\gamma^T \quad \text{for } \gamma \in [n].$$

for some FHE error term ϵ_γ with $\|\epsilon_\gamma\| \leq B \cdot m^{O(d')}$ and ABE error term $\mathbf{e}_\gamma$ with $\|\mathbf{e}_\gamma\| \leq B \cdot m^{O(d)}$. Therefore, $\tilde{y}'$ can be written as:

$$\begin{aligned}
\tilde{y}' &= \left\lfloor \sum_{\gamma \in [n]} \left\langle \mathbf{s}^T \tilde{\mathbf{B}}_\gamma, \mathbf{v}_\gamma \right\rangle \right\rceil_p \\
&= \left\lfloor \sum_{\gamma \in [n]} \left\langle \mathbf{s}^T (\tilde{\mathbf{B}}_\gamma + (w_\gamma \cdot \mathsf{eq}(\mathbf{x}, \mathbf{x}^*) + \epsilon_\gamma) \cdot \mathbf{G}) + \mathbf{e}_\gamma - \mathbf{s}^T (w_\gamma \cdot \mathsf{eq}(\mathbf{x}, \mathbf{x}^*) + \epsilon_\gamma) \cdot \mathbf{G} - \mathbf{e}_\gamma,\ \mathbf{v}_\gamma \right\rangle \right\rceil_p \\
&= \left\lfloor \sum_{\gamma \in [n]} \left\langle \mathbf{s}^T (\tilde{\mathbf{B}}_\gamma + (w_\gamma \cdot \mathsf{eq}(\mathbf{x}, \mathbf{x}^*) + \epsilon_\gamma) \cdot \mathbf{G}) + \mathbf{e}_\gamma, \mathbf{v}_\gamma \right\rangle - \mathsf{eq}(\mathbf{x}, \mathbf{x}^*) \left(\sum_{\gamma \in [n]} \langle \mathbf{s}^T \cdot w_\gamma \cdot \mathbf{G}, \mathbf{v}_\gamma \rangle + e_{\tilde{w}} \right) - \tilde{e} \right\rceil_p \\
&= \left\lfloor \sum_{\gamma \in [n]} \left\langle \tilde{\mathbf{b}}_\gamma, \mathbf{v}_\gamma \right\rangle - \mathsf{eq}(\mathbf{x}, \mathbf{x}^*) \left(\sum_{\gamma \in [n]} \langle \mathbf{s}^T \cdot w_\gamma \cdot \mathbf{G}, \mathbf{v}_\gamma \rangle + e_{\tilde{w}} \right) - \tilde{e} \right\rceil_p \\
&= \left\lfloor \sum_{\gamma \in [n]} \left\langle \tilde{\mathbf{b}}_\gamma, \mathbf{v}_\gamma \right\rangle - \mathsf{eq}(\mathbf{x}, \mathbf{x}^*)(\langle \mathbf{w}, \mathbf{s} \rangle + e_{\tilde{w}}) - \tilde{e} \right\rceil_p \\
&= \left\lfloor \underbrace{\sum_{\gamma \in [n]} \left\langle \tilde{\mathbf{b}}_\gamma, \mathbf{v}_\gamma \right\rangle - \mathsf{eq}(\mathbf{x}, \mathbf{x}^*) \cdot \tilde{w}}_{\xi_{\mathbf{x}}} - \tilde{e} \right\rceil_p
\end{aligned}$$

where $\tilde{e} = \sum_{\gamma \in [n]} \langle \mathbf{e}_\gamma, \mathbf{v}_\gamma \rangle - \mathsf{eq}(\mathbf{x}, \mathbf{x}^*) \cdot e_{\tilde{w}}$ and $e_{\tilde{w}}$ is the noise associated with the term $\tilde{w}$. We note that the error term $\tilde{e}$ is bounded $|\tilde{e}| \leq E$ for some $E = B \cdot m^{\tilde{O}(d)}$.

Therefore, as long as the term $\tilde{e}$ disappears with the rounding operation $\lfloor \cdot \rceil_p$, the auxiliary PRF evaluation $\mathsf{Eval}^*(\mathsf{msk}^*, \mathsf{sk}_{\mathbf{x}^*}, \cdot)$ and the honest PRF evaluation $\mathsf{pPRF.Eval}(\mathsf{msk}, \cdot)$ coincide.

To this end, we define the event $\mathsf{Borderline}_{\mathbf{x}}$ as the event where the adversary submits an input $\mathbf{x}$ to its challenge or regular evaluation queries such that

$$\xi_{\mathbf{x}} \in [-E, E] + (q/p) \cdot (\mathbb{Z} + 1/2).$$

Namely, this is the event that the puncture evaluation excluding the noise is close to being rounded in the wrong direction. By the definition of rounding, if $\neg\mathsf{Borderline}_{\mathbf{x}}$, then we have that $\mathsf{pPRF.Eval}(\mathsf{msk}, \mathbf{x}) = \mathsf{Eval}^*(\mathsf{msk}^*, \mathsf{sk}_{\mathbf{x}^*}, \mathbf{x})$. Therefore, we have the following lemma

Lemma 5.3. *We have that* $\mathsf{Adv}_{\mathsf{H}_1,\mathsf{H}_2}(\mathcal{A}) \leq \Pr_{\mathsf{H}_2}[\mathsf{Borderline}_{\mathbf{x}^*}]$.

Now, instead of bounding the probability $\Pr_{\mathsf{H}_2}[\mathsf{Borderline}_{\mathbf{x}^*}]$ directly, we first show that the hybrid experiments H_2 and H_3 are computationally indistinguishable. This means that the probability of the event $\mathsf{Borderline}_{\mathbf{x}}$ in the two hybrid experiments are negligible

$$\left| \Pr_{\mathsf{H}_2}[\mathsf{Borderline}_{\mathbf{x}}] - \Pr_{\mathsf{H}_3}[\mathsf{Borderline}_{\mathbf{x}}] \right| \leq \mathsf{negl}(\lambda).$$

Then, we show that the event $\mathsf{Borderline}_{\mathbf{x}}$ in H_3 is negligible, thus showing that the probability of the event $\mathsf{Borderline}_{\mathbf{x}}$ in H_2 is negligible.

Lemma 5.4. *The hybrid experiments* H_2 *and* H_3 *are computationally indistinguishable assuming the hardness of* $LWE_{n,m,q,\chi}$.

Proof. We construct a simulator $\mathcal{B}$ that breaks LWE given an attacker $\mathcal{A}$ that distinguishes between H_2 and H_3. $\mathcal{B}$ first receives $\{(\mathbf{A}'_b, \mathbf{a}'_b)\}_{b \in \{0,1\}}$, $\{(\mathbf{B}'_i, \mathbf{b}'_i)\}_{i \in [z]}$, and $\{(\mathbf{C}'_j, \mathbf{c}'_j)\}_{j \in [t]}$ from the LWE challenger. It also receives one additional LWE challenge $(\mathbf{w}, \tilde{w})$. Upon receiving $\mathbf{x}^*$ from $\mathcal{A}$, the simulator $\mathcal{B}$ runs $\mathsf{Setup}^*(1^\lambda, \mathbf{x}^*)$ with these LWE challenge matrices, instead of choosing the matrices at random.

Now, the simulator $\mathcal{B}$ provides the LWE challenge vectors $\{\mathbf{a}'_b\}_{b \in \{0,1\}}$, $\{\mathbf{b}'_i\}_{i \in [z]}$ $\{\mathbf{c}'_j\}_{j \in [t]}$ along with ct^* as the punctured key to $\mathcal{A}$. We note that if the challenge vectors are valid LWE samples, then the distribution is identical to the key generated by $\mathsf{Puncture}_1^*$ and if they are truly random samples, then the distribution is identical to the key generated by $\mathsf{Puncture}_2^*$.

Finally, the simulator $\mathcal{B}$ runs $\mathsf{Eval}^*(\mathsf{msk}^*, \mathsf{sk}_{\mathbf{x}^*}, \mathbf{x}^*)$ to generate the challenge PRF evaluation using the punctured key along with the challenge sample $\tilde{w}$. If $\tilde{w}$ that was provided by the LWE challenger is a valid LWE sample $\tilde{w} = \langle \mathbf{s}, \mathbf{w} \rangle + \mathbf{e}_{\mathbf{w}}$, then it is easy to see that this is a perfect simulation of Eval^*. If $\tilde{w}$ is a uniformly random element in $\mathbb{Z}_q$, then the evaluation

$$\tilde{y} = \left\lfloor \sum_{\gamma \in [n]} \left\langle \tilde{\mathbf{b}}_\gamma, \mathbf{v}_\gamma \right\rangle - \tilde{w} \right\rceil_p .$$

is a uniformly random vector in $\mathbb{Z}_p$ as q is a multiple of p. Therefore, with the distinguishing advantage of $\mathcal{A}$, the simulator $\mathcal{B}$ solves the LWE problem, which proves the lemma. □

Now we show that the event $\mathsf{Borderline}_{\mathbf{x}}$ in the hybrid experiment H_3 is negligible.

Lemma 5.5. *Under the 1D-SIS-*$\mathrm{R}_{q,p,\beta,m'}$ *assumption, it holds that* $\Pr_{\mathsf{H}_3}[\mathsf{Borderline}_{\mathbf{x}^*}] = \mathsf{negl}(\lambda)$ *where* $m' = m \cdot (2 + z + t) + 1$*, and* $\beta = B \cdot m^{\tilde{O}(d)}$.

Proof. Let $\mathbf{z} \in \mathbb{Z}_q^{(2+z+t)\cdot m+1}$ be an instance of 1D-SIS-$\mathrm{R}_{q,p,\beta,m'}$. We parse $\mathbf{z}$ into $2 + z + t$ m-dimension vectors and a single 1-dimension element. We define $\{\mathbf{a}_b\}_{b \in \{0,1\}}$, $\{\mathbf{b}_i\}_{i \in [z]}$, $\{\mathbf{c}_j\}_{j \in [t]}$ and $\tilde{w}$ in the $\mathsf{Puncture}_2^*$ and Eval^* algorithms to correspond to these parsed components of $\mathbf{z}$. We construct a simulator $\mathcal{B}$ that runs the experiment H_3 and solves 1D-SIS-$\mathrm{R}_{q,p,\beta,m'}$ if the event $\mathsf{Borderline}_{\mathbf{x}}$ occurs.

Assume that $\mathsf{Borderline}_{\mathbf{x}}$ occurs. Then, by how we defined $\mathsf{Borderline}_{\mathbf{x}}$, we found an $\mathbf{x}$ such that

$$\sum_{\gamma \in [n]} \left\langle \tilde{\mathbf{b}}_\gamma, \mathbf{v}_\gamma \right\rangle - \tilde{w} \in [-E, E] + (q/p) \cdot (\mathbb{Z} + 1/2)$$

where $E = B \cdot m^{O(d)}$. By Theorem 4.2, it follows that

$$\tilde{\mathbf{b}}_\gamma^T = \sum_{b \in \{0,1\}} \mathbf{a}_b^T \mathbf{R}_{1,b,\gamma} + \sum_{i \in [z]} \mathbf{b}_i^T \mathbf{R}_{2,i,\gamma} + \sum_{i \in [t]} \mathbf{c}_j^T \mathbf{R}_{3,j,\gamma}$$

for some matrices $\{\mathbf{R}_{1,b,\gamma}\}, \{\mathbf{R}_{2,i,\gamma}\}, \{\mathbf{R}_{3,j,\gamma}\}$ where $\|\mathbf{R}_{1,b,\gamma}\|, \|\mathbf{R}_{2,i,\gamma}\|, \|\mathbf{R}_{3,j,\gamma}\| \leq m^{O(d)}$. By combining these matrices, we get an efficiently computable matrix $\mathbf{R}_\gamma$ such that

$$\sum_{\gamma \in [n]} \langle \mathbf{z}, \mathbf{R}_\gamma \cdot \mathbf{v}_\gamma \rangle \in [-E, E] + (q/p) \cdot (\mathbb{Z} + 1/2)$$

with $\mathbf{R}_\gamma \cdot \mathbf{v}_\gamma \leq m^{O(d)}$. Therefore, the vector $\sum_{\gamma \in [n]} \mathbf{R}_\gamma \cdot \mathbf{v}_\gamma$ is a valid solution to 1D-SIS-$\mathrm{R}_{q,p,\beta,m'}$ with bound $\beta = B \cdot m^{\tilde{O}(d)}$ and this proves the lemma. □

Finally, since the event $\mathsf{Borderline}_{\mathbf{x}^*}$ is negligible in H_3 and the experiments H_2 and H_3 are indistinguishable, we have the following corollary.

Corollary 5.6. *We have that* $\Pr_{\mathsf{H}_2}[\mathsf{Borderline}_{\mathbf{x}^*}] = \mathsf{negl}(\lambda)$ *assuming the hardness of* $LWE_{n,m,q,\chi}$ *and 1D-SIS-*$R_{q,p,\beta,m'}$ *for* $\beta = B \cdot m^{\tilde{O}(d)}$ *and* $m' = (2 + z + t) \cdot m + 1$.

What remains are the indistinguishability of the sequence of hybrid experiments from H_3 to H_6. We note, however, that in each of these hybrid experiments, we simply unroll back the changes that we made in the previous hybrid experiments. Therefore, their indistinguishability follows from the symmetric argument used to show the indistinguishability of the previous hybrid experiments.

Lemma 5.7. *The hybrid experiments* H_3 *and* H_4 *are computationally indistinguishable assuming the hardness of* $LWE_{n,m,q,\chi}$.

Lemma 5.8. *The hybrid experiments* H_4 *and* H_5 *are computationally indistinguishable assuming the hardness of* $LWE_{n,m,q,\chi}$ *and* 1D-SIS-$R_{q,p,\beta,m'}$ *for* $\beta = B \cdot m^{\tilde{O}(d)}$ *and* $m' = m \cdot (2 + z + t) + 1$.

Lemma 5.9. *The hybrid experiments* H_5 *and* H_6 *are perfectly indistinguishable.*

This concludes the proof of Theorem 5.1.

□

Now, we proceed in showing that our construction satisfies the privacy of the punctured point. Since the arguments used to show privacy follows from the arguments used to show pseudorandomness in Theorem 5.1 in a straightforward way, we simply provide a general proof overview.

Theorem 5.10. *Let* Π_{HE} *be a secure leveled homomorphic encryption scheme with parameters instantiated as in Section 5.2. The puncturable PRF from Section 5.1 with parameters instantiated as in Section 5.2 satisfies the security property of a private puncturable PRF as defined in Definition 3.6 assuming the hardness of* $LWE_{n,m',q,\chi}$ *and* 1D-SIS-$R_{q,p,\beta,m'}$ *for* $\beta = B \cdot m^{\tilde{O}(d)}$ *and* $m' = m \cdot (2 + z + t) + 1$.

Proof Overview. Intuitively, we want to reduce the privacy of the punctured point to the semantic security of the underlying FHE scheme since the point is encrypted. The only obstacle that prevents a straightforward reduction to semantic security is that the FHE secret key $\mathsf{he.sk}$ is released as part of the ABE encodings in the punctured key. Therefore, we must argue that the ABE encodings in the punctured key computationally hides $\mathsf{he.sk}$ such that we can remove it altogether from the simulation. In fact, this is precisely what we do in the hybrid experiments $\mathsf{H}_0 \to \mathsf{H}_3$ in the proof of Theorem 5.1. In H_3, the ABE encodings of the punctured key are simply uniformly random strings. Therefore, following the experiment H_3, we can rely on the semantic security of FHE to replace the encryption of the punctured point to an encryption of the all zeros string. As in the proof of Theorem 5.1, we can now unroll back the generation of the public parameters and the punctured keys. This concludes the proof overview. □

5.4 Correctness

We now show the correctness of our construction. The correctness proof also follows in a straightforward manner as in the proof of Theorem 5.1 and therefore, we provide a general overview of the proof.

Theorem 5.11. *The puncturable PRF from Section 5.1 with parameters instantiated as in Section 5.2 satisfies the correctness property of Definition 3.5 assuming the hardness of* $LWE_{n,m,q,\chi}$ *and* 1D-SIS-$R_{q,p,\beta,m'}$ *for* $\beta = B \cdot m^{\tilde{O}(d)}$ *and* $m' = m \cdot (2 + z + t) + 1$.

Proof Overview. We note that in the correctness game, the adversary wins the game if it finds a point $\mathbf{x}$ such that $\mathsf{pPRF.Eval}(\mathsf{msk}, \mathbf{x}) \neq \mathsf{pPRF.PunctureEval}(\mathsf{sk}_{\mathbf{x}^*}, \mathbf{x})$ for $\mathbf{x} \neq \mathbf{x}^*$. By the correctness of $\mathsf{Eval}_{\mathsf{pk}}$ and $\mathsf{Eval}_{\mathsf{ct}}$, we have that $\mathsf{pPRF.Eval}(\mathsf{msk}, \mathbf{x}) \neq \mathsf{pPRF.PunctureEval}(\mathsf{sk}_{\mathbf{x}^*}, \mathbf{x})$ precisely when the event $\mathsf{Borderline}_{\mathbf{x}}$ occurs. Therefore, we can use the same argument in the proof of Theorem 5.1 to show that the adversary's advantage in forcing the event $\mathsf{Borderline}_{\mathbf{x}}$ is negligible. Specifically, we can first argue by LWE that the challenger can replace the "ABE encodings" of the punctured key with just random strings. Now, since the PRF evaluation consists of taking a low-norm linear combination of the encoding vectors, one can embed an instance of the 1D-SIS-R problem into the encoding vectors and therefore, solve 1D-SIS-R using an adversary that can force the event $\mathsf{Borderline}_{\mathbf{x}}$. This shows that the adversary's advantage is negligible assuming the hardness of 1D-SIS-R. This concludes the proof overview.

□

5.5 Extentions

We conclude this section with some high-level discussion on extending our scheme and how it relates to other lattice based PRF constructions.

Puncturing at Multiple Points. A private puncturable PRF can be combined to support a single-key private k-puncturable PRF generically where a constrained key can be punctured at k distinct points in the domain. One way of doing this is to simply define the PRF to be the xor of k independent instances of a 1-puncturable PRF. More precisely, let $\mathsf{msk}_i \leftarrow \mathsf{pPRF.Setup}(1^\lambda)$ for $i = 1, ..., k$. Then define the master secret key of the k-puncturable PRF to be the collection of these master secret keys $\mathsf{msk} = (\mathsf{msk}_1, ..., \mathsf{msk}_k)$. We define the evaluation of the PRF to be $F(\mathsf{msk}, \mathbf{x}) = F(\mathsf{msk}_1, \mathbf{x}) \oplus \ldots \oplus F(\mathsf{msk}_k, \mathbf{x})$. Then, to generate a punctured key at $S = \{\mathbf{x}_1, ..., \mathbf{x}_k\}$, we puncture each msk_i at point $\mathbf{x}_i$, to get punctured key $\mathsf{sk}_{\mathbf{x}_i} \leftarrow \mathsf{pPRF.Puncture}(\mathsf{msk}_i, \mathbf{x}_i)$, and then set $\mathsf{sk}_S = (\mathsf{sk}_{\mathbf{x}_1}, ..., \mathsf{sk}_{\mathbf{x}_k})$. It is easy to see that one can evaluate the PRF with the punctured key only at a point $\mathbf{x}$ in the domain $\mathbf{x} \notin S$. It is also straightforward to show that pseudorandomness and privacy follow from the security of the underlying 1-puncturable PRF.

Short constrained keys. In [BV15], Brakerski and Vaikuntanathan provide a way to achieve succinct constrained keys for their single-key constrained PRF, which also extends to our construction in Section 5.1. We provide a high level overview of this method.

In the constrained PRF construction of [BV15], a constrained key consists of the description of the constraint circuit along with the ABE encodings of the constraint circuit. To get succinct constrained keys, one can encrypt the bit encodings for each possible bits using an encryption scheme and publish it as part of the public parameters (just like in a garbling scheme). Then, as the constrained key, one can provide the decryption keys corresponding to the bit description of the constraint circuit. Now, using the attribute-based encryption construction of [BGG+14], which has short decryption keys, one can provide the ABE secret key that allows the decryption of the bits of the constraint circuit.

One difference with our construction is that we encode *field elements* in our ABE encodings for the FHE key. However, the FHE key stays the same for any punctured point. Therefore, we can garble just the bit positions corresponding to the encryption of the point to be punctured and publish the rest of the components in the clear. This allows the size of the public parameters to absorb the size of the constrained key.

Key homomomorphism. Our PRF construction has a similar structure as the other lattice-based PRF constructions and therefore, the master secret key (LWE vector) for which the PRF is defined can be added homomorphically from the PRF evaluations. However, we note that in our construction, the PRF key (secret vector) is from a short noise distribution χ. Although there are applications of key-homomorphic PRFs with short keys, for most applications of key-homomorphic PRFs, one requires a perfect secret sharing of the PRF key, which requires it to come from a uniform distribution over a finite group. We leave it as an open problem to extend the construction to the setting of key-homomorphic PRFs with uniform keys.

6 Impossibility of Simulation Based Privacy

In this section, we show that a simulation based privacy notion for constrained PRFs for general circuit constraints is impossible. More precisely, we show that even for the single-key setting where the adversary is given one single constrained key, a natural extension of the indistinguishability privacy definition (Definition 3.4) to a simulation based privacy definition cannot be satisfied. We do this rather indirectly by showing that a constrained PRF (for general circuits) satisfying the simulation based privacy definition implies a simulation secure functional encryption [SS10, BSW11, O'N10], which was shown to be impossible in [BSW11, AGVW13].

6.1 Definition

We begin with the definition of a simulation based privacy notion for constrained PRFs. The simulation based privacy requires that any adversary given a constrained key sk_f does not learn any more information about the constraint other than what can be implied by comparing the output of the real evaluation $\mathsf{cPRF.Eval}(\mathsf{msk}, \cdot)$

and $\mathsf{cPRF.ConstrainEval}(\mathsf{sk}_f, \cdot)$. The correctness and pseudorandomness properties stay the same as how it is defined in Section 3.

Definition 6.1 (Sim-Privacy)**.** Fix a security parameter $\lambda \in \mathbb{N}$. A constrained PRF scheme $\Pi_{\mathsf{cPRF}} = (\mathsf{cPRF.Setup}, \mathsf{cPRF.Constrain}, \mathsf{cPRF.ConstrainEval}, \mathsf{cPRF.Eval})$ is *simulation-private for single-key* if there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_{\mathsf{Eval}}, \mathcal{S}_{\mathsf{Constrain}})$ such that for all PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{priv}}_{\Pi_{\mathsf{cPRF}},\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Expt}^{\mathsf{REAL}}_{\Pi_{\mathsf{cPRF}},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{\mathsf{RAND}}_{\Pi_{\mathsf{cPRF}},\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

where the experiments $\mathsf{Expt}^{\mathsf{REAL}}_{\Pi_{\mathsf{cPRF}},\mathcal{A}}(\lambda)$ and $\mathsf{Expt}^{\mathsf{RAND}}_{\Pi_{\mathsf{cPRF}},\mathcal{A}}(\lambda)$ are defined as follows:

$\mathsf{Expt}^{\mathsf{REAL}}_{\Pi_{\mathsf{cPRF}},\mathcal{A}}(\lambda)$:

1. $\mathsf{msk} \leftarrow \mathsf{cPRF.Setup}(1^\lambda)$.
2. $(f^*, \mathsf{state}) \leftarrow \mathcal{A}^{\mathsf{cPRF.Eval}(\mathsf{msk},\cdot)}(1^\lambda)$.
3. $\mathsf{sk}_{f^*} \leftarrow \mathsf{cPRF.Constrain}(\mathsf{msk}, f^*)$.
4. $b \leftarrow \mathcal{A}(\mathsf{sk}_{f^*}, \mathsf{state})$.
5. Output b

$\mathsf{Expt}^{\mathsf{RAND}}_{\Pi_{\mathsf{cPRF}},\mathcal{A}}(\lambda)$:

1. $(f^*, \mathsf{state}_1) \leftarrow \mathcal{A}^{\mathcal{S}_{\mathsf{Eval}}(\cdot)}(1^\lambda)$.
2. $\mathsf{sk}_{f^*} \leftarrow \mathcal{S}_{\mathsf{Constrain}}()$.
3. $b \leftarrow \mathcal{A}(\mathsf{sk}_{f^*}, \mathsf{state}_1)$.
4. Output b.

Here, the algorithms $\mathcal{S}_{\mathsf{Eval}}$ and $\mathcal{S}_{\mathsf{Constrain}}$ share common state and the algorithm $\mathcal{S}_{\mathsf{Constrain}}$ is given the size $|f|$ and oracle access to the following set of mappings

$$\mathcal{C}_{\mathsf{constrain}} = \left\{ i \mapsto f^*(x^{(i)}) : i \in [Q] \right\}$$

where Q represents the number of times $\mathcal{A}$ queries the evaluation oracles $\mathcal{S}_{\mathsf{Eval}}$.

In words, the security definition above requires that an adversary cannot distinguish whether it is interacting with a real constrained PRF or it is interacting with a simulator that is not actually given the constraint f^* except for output of f^* applied to each of the adversary's queries to the evaluation oracle.

6.2 Functional Encryption

In this subsection, we define a simulation secure functional encryption for circuits. For simplicity, we consider functions with just binary outputs.

A (secret-key) functional encryption (FE) scheme is a tuple of algorithms $\Pi_{\mathsf{FE}} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Encrypt}, \mathsf{FE.Decrypt})$ defined over a message space $\mathcal{X}$, and a class of functions $\mathcal{F}_\lambda = \{f : \mathcal{X} \to \{0,1\}\}$ with the following properties:

- $\mathsf{FE.Setup}(1^\lambda) \to \mathsf{msk}$: On input the security parameter λ, the setup algorithm outputs the master secret key msk.
- $\mathsf{FE.KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$: On input the master secret key msk and a circuit f, the key generation algorithm outputs a secret key sk_f.
- $\mathsf{FE.Encrypt}(\mathsf{msk}, \mathbf{x}) \to \mathsf{ct}$: On input the master secret key msk, and a message $\mathbf{x}$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{FE.Decrypt}(\mathsf{ct}, \mathsf{sk}_f) \to \{0,1\}$: On input a ciphertext ct and a secret key sk_f, the decryption algorithm outputs a bit $y \in \{0,1\}$.

Correctness. A functional encryption scheme $\Pi_{\mathsf{FE}} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Encrypt}, \mathsf{FE.Decrypt})$ is correct if for all $\lambda \in \mathbb{N}$, $\mathsf{msk} \leftarrow \mathsf{FE.Setup}(1^\lambda)$, $f \in \mathcal{F}$, and $\mathsf{sk}_f \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}, f)$, we have that

$$\Pr[\mathsf{FE.Decrypt}(\mathsf{sk}_f, \mathsf{FE.Encrypt}(\mathsf{msk}, \mathbf{x})) = f(\mathbf{x})] = 1 - \mathsf{negl}(\lambda).$$

Security. For security, we require that any adversary given a secret key does not learn any more information about an encrypted message other than what can be deduced from an honest decryption.

Definition 6.2. Fix a security parameter $\lambda \in \mathbb{N}$. A functional encryption scheme $\Pi_{\mathsf{FE}} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Encrypt}, \mathsf{FE.Decrypt})$ is *simulation secure for single-key* if there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_{\mathsf{Encrypt}}, \mathcal{S}_{\mathsf{KeyGen}})$ such that for all PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{FE}}_{\Pi_{\mathsf{FE}},\mathcal{A}}(\lambda) = \left| \Pr[\mathsf{Expt}^{\mathsf{REAL}}_{\Pi_{\mathsf{FE}},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{\mathsf{RAND}}_{\Pi_{\mathsf{FE}},\mathcal{A}}(\lambda) = 1] \right| \leq \mathsf{negl}(\lambda)$$

where the experiments $\mathsf{Expt}^{\mathsf{REAL}}_{\Pi_{\mathsf{FE}},\mathcal{A}}(\lambda)$ and $\mathsf{Expt}^{\mathsf{RAND}}_{\Pi_{\mathsf{FE}},\mathcal{A}}(\lambda)$ are defined as follows:

$\mathsf{Expt}^{\mathsf{REAL}}_{\Pi_{\mathsf{FE}},\mathcal{A}}(\lambda)$:

1. $\mathsf{msk} \leftarrow \mathsf{FE.Setup}(1^\lambda)$.
2. $(f^*, \mathsf{state}) \leftarrow \mathcal{A}^{\mathsf{FE.Encrypt}(\mathsf{msk},\cdot)}(1^\lambda)$.
3. $\mathsf{sk}_{f^*} \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}, f^*)$.
4. $b \leftarrow \mathcal{A}(\mathsf{sk}_{f^*}, \mathsf{state})$.
5. Output b

$\mathsf{Expt}^{\mathsf{RAND}}_{\Pi_{\mathsf{FE}},\mathcal{A}}(\lambda)$:

1. $(f^*, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{S}_{\mathsf{Encrypt}}()}(1^\lambda)$
2. $\mathsf{sk}_{f^*} \leftarrow \mathcal{S}_{\mathsf{KeyGen}}(f^*)$.
3. $b \leftarrow \mathcal{A}(\mathsf{sk}_{f^*}, \mathsf{state})$.
4. Output b.

Here, the algorithms $\mathcal{S}_{\mathsf{Encrypt}}$ and $\mathcal{S}_{\mathsf{KeyGen}}$ share common state and the simulator $\mathcal{S}_{\mathsf{KeyGen}}$ is given oracle access to the set of mappings $\mathcal{C}_{\mathsf{msg}} = \{i \mapsto f^*(\mathbf{x}^{(i)}) : i \in [Q]\}$ where Q represents the number of queries that $\mathcal{A}$ makes to $\mathcal{S}_{\mathsf{Encrypt}}$.

It was shown in [BSW11, AGVW13] that a functional encryption scheme satisfying the security definition above is impossible to achieve.

6.3 FE from Constrained PRFs

In this subsection, we present our construction of functional encryption. Fix a security parameter λ. Let $\Pi_{\mathsf{cPRF}} = (\mathsf{cPRF.Setup}, \mathsf{cPRF.Constrain}, \mathsf{cPRF.ConstrainEval}, \mathsf{cPRF.Eval})$ be a constrained PRF with domain $\{0,1\}^{\lambda+\ell}$ and range $\{0,1\}^\lambda$ where ℓ is the size of the message in the functional encryption scheme. We also use an additional regular PRF, which we denote by $F_{\mathbf{k}} : \{0,1\}^\lambda \to \{0,1\}^\ell$. We construct $\Pi_{\mathsf{FE}} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Encrypt}, \mathsf{FE.Decrypt})$ as follows:

- $\mathsf{FE.Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm first samples a regular PRF key $\mathbf{k} \xleftarrow{\$} \{0,1\}^\lambda$. Then, it runs $\mathsf{cprf.msk} \leftarrow \mathsf{cPRF.Setup}(1^\lambda)$ and sets $\mathsf{msk} = (\mathsf{cprf.msk}, \mathbf{k})$.
- $\mathsf{FE.KeyGen}(\mathsf{msk}, f)$: On input the master secret key msk and a circuit f, the key generation algorithm generates a constrained PRF key $\mathsf{sk}_{C_{f,\mathbf{k}}} \leftarrow \mathsf{cPRF.Constrain}(\mathsf{cprf.msk}, C_{f,\mathbf{k}})$ where the circuit $C_{f,\mathbf{k}}$ is defined as follows:
$$C_{f,\mathbf{k}}(\mathbf{r}, \mathbf{y}) = \begin{cases} 0 & \text{if } f(F_{\mathbf{k}}(\mathbf{r}) \oplus \mathbf{y}) = 0 \\ 1 & \text{otherwise} \end{cases}.$$
It outputs $\mathsf{sk}_f = \mathsf{sk}_{C_{f,\mathbf{k}}}$.
- $\mathsf{FE.Encrypt}(\mathsf{msk}, \mathbf{x})$: On input the master secret key msk, and a message $\mathbf{x} \in \{0,1\}^\ell$, the encryption algorithm first samples encryption randomness $\mathbf{r} \xleftarrow{\$} \{0,1\}^\lambda$ and computes $\mathbf{y} = F_{\mathbf{k}}(\mathbf{r}) \oplus \mathbf{x}$. Then, it returns
$$\mathsf{ct} = (\mathbf{r}, \mathbf{y}, \mathsf{cPRF.Eval}(\mathsf{cprf.msk}, \mathbf{r}\|\mathbf{y}))$$
- $\mathsf{FE.Decrypt}(\mathsf{sk}_f, \mathsf{ct})$: On input a secret key $\mathsf{sk}_f = \mathsf{sk}_{C_{f,\mathbf{k}}}$ and $\mathsf{ct} = (\mathbf{r}, \mathbf{y}, \tilde{\mathsf{ct}})$, the decryption algorithm returns 0 if $\mathsf{cPRF.ConstrainEval}(\mathsf{sk}_{C_{f,\mathbf{k}}}, \mathbf{r}\|\mathbf{y}) = \tilde{\mathsf{ct}}$ and 1 otherwise.

Correctness. To show correctness, we note that the decryption algorithm simply evaluates the PRF using the constrained key $\mathsf{cPRF.Constrain}(\mathsf{sk}_{C_{f,\mathbf{k}}}, \mathbf{r}\|\mathbf{y})$ and returns 0 if the result equals $\tilde{\mathsf{ct}}$ and 1 otherwise. Since $\tilde{\mathsf{ct}}$ is precisely the PRF evaluation using the master secret key $\mathsf{cPRF.Eval}(\mathsf{cprf.msk}, \mathbf{r}\|\mathbf{y})$, the two evaluations coincide if $C_{f,\mathbf{k}}(\mathbf{r}, \mathbf{y}) = 0$. Also, if Π_{cPRF} satisfies the standard notion of pseudorandomness as in Definition 3.3, the PRF evaluation using the master secret key and the PRF evaluation using the constrained key differs with overwhelming probability if the constraint is not satisfied $C_{f,\mathbf{k}}(\mathbf{r}, \mathbf{y}) = 1$.

6.4 Security

In this section, we prove security of construction above.

Theorem 6.3. *Let* $\Pi_{\mathsf{cPRF}} = (\mathsf{cPRF.Setup}, \mathsf{cPRF.Constrain}, \mathsf{cPRF.ConstrainEval}, \mathsf{cPRF.Eval})$ *be a constrained PRF scheme satisfying the security properties of Definition 6.1. Also, let* $F_{\mathbf{k}}$ *be a secure PRF. Then, the functional encryption scheme* Π_{FE} *constructed above satisfies the simulation based security notion of Definition 6.2.*

Proof. We proceed through a series of hybrid experiments where the first hybrid H_0 represents the real experiment $\mathsf{Expt}^{\mathsf{REAL}}_{\Pi_{\mathsf{FE}},\mathcal{A}}$ and the final hybrid H_3 represents the ideal simulation $\mathsf{Expt}^{\mathsf{RAND}}_{\Pi_{\mathsf{FE}},\mathcal{A}}$.

- **Hybrid** H_0: This is the *real* experiment. The challenger runs the real setup algorithm to generate msk. Then the adversary makes a number of encryption queries and a key generation query which the challenger answers using its msk.

- **Hybrid** H_1: In this experiment, the challenger runs the simulator for the constrained PRF to answer the adversary's queries. More precisely, given a constrained PRF simulator $\mathcal{S} = (\mathcal{S}_{\mathsf{Eval}}, \mathcal{S}_{\mathsf{Constrain}})$, the challenger first samples a PRF key $\mathbf{k} \xleftarrow{\$} \{0,1\}^\lambda$ as msk. Then for each encryption query $\mathbf{x}$ that the adversary makes, the challenger samples $\mathbf{r} \xleftarrow{\$} \{0,1\}^\lambda$, computes $\mathbf{y} \leftarrow F_{\mathbf{k}}(\mathbf{r}) \oplus \mathbf{x}$ and invokes the simulator to generate $\tilde{\mathsf{ct}} \leftarrow \mathcal{S}_{\mathsf{Eval}}(\mathbf{r}\|\mathbf{y})$. It provides $(\mathbf{r}, \mathbf{y}, \tilde{\mathsf{ct}})$ to the adversary as the encryption of $\mathbf{x}$. To answer the single key generation query on f^* from the adversary, the challenger invokes the simulator $\mathcal{S}_{\mathsf{Constrain}}()$ to generate the key. For the set of mappings $\mathcal{C}_{\mathsf{constrain}}$ that are to be provided to $\mathcal{S}_{\mathsf{Constrain}}$, the challenger computes $f^*(\mathbf{x}^{(i)})$ itself and feeds it to the simulator.

  By the assumption on the simulator $\mathcal{S} = (\mathcal{S}_{\mathsf{Eval}}, \mathcal{S}_{\mathsf{Constrain}})$, we have that the hybrids H_0 and H_1 are indistinguishable to the adversary. We note that in H_1, the challenger does not actually use the PRF key $\mathbf{k}$ to generate the secret keys.

- **Hybrid** H_2: In this experiment, the challenger replaces $F_{\mathbf{k}}(\cdot)$ with a random function. Namely, to answer an encryption query $\mathbf{x}$ by the adversary, the challenger ignores the message $\mathbf{x}$ and samples $\tilde{\mathbf{y}}$ randomly $\tilde{\mathbf{y}} \xleftarrow{\$} \{0,1\}^\ell$. It then invokes $\tilde{\mathsf{ct}} \leftarrow \mathcal{S}_{\mathsf{Eval}}(\mathbf{r}\|\tilde{\mathbf{y}})$ and sets $(\mathbf{r}, \tilde{\mathbf{y}}, \tilde{\mathsf{ct}})$ as the encryption of $\mathbf{x}$. The rest of the experiment remains unchanged from H_1.

  Note that in both hybrid experiments H_1 and H_2, the challenger does not use the PRF key $\mathbf{k}$ other than in evaluating the PRF $F_{\mathbf{k}}(\cdot)$ to encrypt. Therefore, by the PRF security of $F_{\mathbf{k}}(\cdot)$, the two experiments are indistinguishable to the adversary. We note that in H_2, the challenger does not use any information about the message $\mathbf{x}_i$ other than providing the simulator $\mathcal{S}_{\mathsf{Constrain}}$ with the values $f^*(\mathbf{x}^{(i)})$.

- **Hybrid** H_3: This experiment represents the *ideal* experiment where the challenger corresponds to the simulator for the functional encryption game. The simulator runs in exactly the same way as in the previous hybrid H_2. Namely, for each encryption query $\mathbf{x}$ that the adversary makes, it samples $\mathbf{r} \xleftarrow{\$} \{0,1\}^\lambda$, $\mathbf{y} \xleftarrow{\$} \{0,1\}^\ell$ and invokes $\tilde{\mathsf{ct}} \leftarrow \mathcal{S}_{\mathsf{Eval}}(\mathbf{r}\|\tilde{\mathbf{y}})$. It sets $(\mathbf{r}, \tilde{\mathbf{y}}, \tilde{\mathsf{ct}})$ as the encryption of $\mathbf{x}$. Note that to generate the ciphertext, it does not use any information about $\mathbf{x}$. For the single key query, the simulator invokes $\mathcal{S}_{\mathsf{Constrain}}()$. For the set of mappings $\mathcal{C}_{\mathsf{constrain}}$ that are to be provided to $\mathcal{S}_{\mathsf{Constrain}}$, it uses its own oracle $\mathcal{C}_{\mathsf{msg}}$ to provide the values $f^*(\mathbf{x}^{(i)})$.

  It is easy to see that the distribution of the experiments H_2 and H_3 are identical.

We have shown that the experiment H_0, which corresponds to $\mathsf{Expt}^{\mathsf{REAL}}_{\Pi_{\mathsf{FE}},\mathcal{A}}$ and the experiment H_3, which corresponds to $\mathsf{Expt}^{\mathsf{RAND}}_{\Pi_{\mathsf{FE}},\mathcal{A}}$ are indistinguishable. This concludes the proof. □

7 Conclusion and Open Problems

We constructed a privately puncturable PRF from worst-case lattice problems. Prior constructions of privately puncturable PRFs required heavy tools such as multilinear maps or $i\mathcal{O}$. This work provides the first privately puncturable PRF from a standard assumption. We also showed that for general functions, a natural simulation-based privacy definition for constrained PRFs is impossible to achieve.

Our PRF builds on the construction of [BV15], which supports circuit constraints. However, our construction does not extend to more general constraints, and it will be interesting to provide a private constrained PRF for a larger class of circuit constraints. For private puncturing, it will be interesting to give more constructions based on assumptions other than LWE.

Our construction satisfies the selective security game of private puncturable PRFs, and we rely on complexity leveraging for adaptive security. Recently, [HKW15] gave a way to achieve adaptively secure puncturable PRFs without complexity leveraging. Can we extend the result to *private* puncturable PRFs?

Finally, private constrained PRFs have a number of interesting applications, as explored here and in [BLW17]. It would be interesting to find further applications and relations to other cryptographic primitives.

Acknowledgements

We thank the anonymous Eurocrypt reviewers for their insightful comments which helped improve the paper. We also thank David Wu for his helpful comments on the definition of privately constrained PRFs. This work is supported by NSF, DARPA, the Simons foundation, and a grant from ONR. Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA. Part of the work was also done while the second author was visiting Fujitsu Laboratories of America as an intern.

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.

[AFP16] Hamza Abusalah, Georg Fuchsbauer, and Krzysztof Pietrzak. Constrained prfs for unbounded inputs. In *CT-RSA*, 2016.

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

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *CRYPTO*. 2013.

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

[ASP14] 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.

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

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *EUROCRYPT*, 2015.

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

[BLMR13] Dan Boneh, Kevin Lewi, Hart 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 Wu. Constraining pseudorandom functions privately. In *PKC*, 2017.

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

[BP16] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key fhe with short ciphertexts. CRYPTO, 2016.

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

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

[BV14a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *SIAM Journal on Computing*, 43(2):831–871, 2014.

[BV14b] 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. 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.

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

[CC17] Ran Canetti and Yilei Chen. Constraint-hiding constrained prfs for NC1 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.

[CKGS98] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

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

[CRV14] Nishanth Chandran, Srinivasan Raghuraman, and Dhinakaran Vinayagamurthy. Constrained pseudorandom functions: Verifiable and delegatable. *IACR Cryptology ePrint Archive*, 2014:522, 2014.

[DKW16] Apoorvaa Deshpande, Venkata Koppula, and Brent Waters. Constrained pseudorandom functions for unconstrained inputs. In *EUROCRYPT*, 2016.

[FKPR14] Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained prfs. In *ASIACRYPT*, 2014.

[Fuc14] Georg Fuchsbauer. Constrained verifiable random functions. In *SCN*, 2014.

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

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986.

[GI14] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In *EUROCRYPT*, 2014.

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

[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.

[GVW15a] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. *Journal of the ACM (JACM)*, 62(6):45, 2015.

[GVW15b] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In *CRYPTO*, 2015.

[HKKW14] Dennis Hofheinz, Akshay Kamath, Venkata Koppula, and Brent Waters. Adaptively secure constrained pseudorandom functions. *IACR Cryptology ePrint Archive*, 2014:720, 2014.

[HKW15] Susan Hohenberger, Venkata Koppula, and Brent Waters. Adaptively secure puncturable pseudorandom functions in the standard model. In *ASIACRYPT*, 2015.

[Hof14] Dennis Hofheinz. Fully secure constrained pseudorandom functions using random oracles. *IACR Cryptology ePrint Archive*, 2014:372, 2014.

[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.

[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.

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

[O'N10] Adam O'Neill. Definitional issues in functional encryption. *IACR Cryptology ePrint Archive*, 2010:556, 2010.

[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.

[Reg04] Oded Regev. Lattices in computer science-average case hardness. *Lecture Notes for Class (scribe: Elad Verbin)*, 2004.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *CCS*, 2010.

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

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

A Puncturable PRFs and regular PRFs

In this section, we show that a puncturable PRF satisfying the pseudorandomness notion of Definition 3.6 implies a secure PRF as defined in Definition 3.1.

Theorem A.1. *Let* $\Pi_{\mathsf{pPRF}} = (\mathsf{pPRF.Setup}, \mathsf{pPRF.Puncture}, \mathsf{pPRF.PunctureEval}, \mathsf{pPRF.Eval})$ *be a puncturable PRF satisfying Definition 3.6 with domain* $\mathcal{X}$ *and range* $\mathcal{Y}$*. Then* $\Pi_{\mathsf{PRF}} = (\mathsf{pPRF.Setup}, \mathsf{pPRF.Eval})$ *satisfies Definition 3.1.*

Proof. Fix an adversary $\mathcal{A}$ that, with oracle access to $\mathsf{pPRF.Eval}$, distinguishes $\mathsf{pPRF.Eval}$ from a random function as in Definition 3.2 making Q oracle calls to $\mathsf{pPRF.Eval}(\mathsf{msk}, \cdot)$. We define a series of hybrid experiments $\mathsf{H}_0, ..., \mathsf{H}_Q$ such that H_0 denotes the real experiment where $\mathcal{A}$ interacts with $\mathsf{pPRF.Eval}(\mathsf{msk}, \cdot)$ and in H_Q, $\mathcal{A}$ interacts with a random function. We define the intermediate hybrids as follows

- H_i: The challenger answers the adversary's first i evaluation queries with randomly sampled elements from the range $y^{(i)} \xleftarrow{\$} \mathcal{Y}$. Then, for the rest of the evaluation queries, the challenger answers the adversary with an honest evaluation using $\mathsf{pPRF.Eval}(\mathsf{msk}, \cdot)$.

We now show that the consecutive hybrids are indistinguishable.

Lemma A.2. *The hybrid experiments* H_i *and* H_{i+1} *are indistinguishable for* $i = 0, ..., Q-1$ *assuming that the puncturable PRF* Π_{pPRF} *satisfies Definition 3.6.*

Proof. For an adversary $\mathcal{A}$ that distinguishes the two hybrid experiments H_i and H_{i+1}, we construct a simulator $\mathcal{B}$ that breaks the security game of Definition 3.6. For each query that $\mathcal{A}$ makes to the evaluation oracle, $\mathcal{B}$ answers $\mathcal{A}$ as follows. For the first i evaluation queries $x^{(1)}, ..., x^{(i)}$, $\mathcal{B}$ simply answers $\mathcal{A}$ with randomly sampled elements from the range $y^{(1)}, ..., y^{(i)} \xleftarrow{\$} \mathcal{Y}$. Then, for the evaluation query $x^{(i+1)}$, the adversary submits $x^{(i+1)}$ as the commitment to the challenge point to the puncturable PRF challenger and receives the PRF key punctured at $x^{(i+1)}$, and the challenge evaluation $y^{(i+1)}$. Then, $\mathcal{B}$ answers $\mathcal{A}$ with $y^{(i+1)}$ as the $(i+1)$th evaluation. Now, for the subsequent evaluation queries $x^{(i+2)}, ..., x^{(Q)}$, the simulator $\mathcal{B}$ uses its own evaluation oracle to answer $\mathcal{A}$.

If the challenge evaluation $y^{(i+1)}$ is an honestly generated evaluation $y^{(i+1)} \leftarrow \mathsf{pPRF.Eval}(\mathsf{msk}, x^{(i+1)})$, then the view of the adversary is exactly the view of H_i. If $y^{(i+1)}$ is uniform over $\mathcal{Y}$, then the view of the adversary is exactly the view of H_{i+1}. Therefore, $\mathcal{B}$ breaks the security of Definition 3.6 with the distinguishing advantage of $\mathcal{A}$. ☐

☐

B Worst-Case Hardness of 1D-SIS-R

In this section, we give a reduction from 1D-SIS-R to 1D-SIS. For this, we first introduce an intermediate problem called 1D-SIS'.

Definition B.1 (1D-SIS')**.** Let m, β be positive integers and let q be a product of n prime moduli $p_1 < p_2 < \ldots < p_n$, $q = \prod_{i\in[n]} p_i$. Then the 1D-SIS'$_{m,q,\beta}$ problem is defined as follows. Given a uniformly random vector $\mathbf{v} \in \mathbb{Z}_q^m$, find $z \in \mathbb{Z}^m$ such that $\|\mathbf{z}\| \leq \beta$ and $\langle \mathbf{v}, \mathbf{z} \rangle = [-\beta, \beta] + q \cdot (\mathbb{Z} + 1/2)$.

We first show that the 1D-SIS' problem is at least as hard as 1D-SIS.

Lemma B.2. *Let* m, q, β *be positive integers. Then the 1D-SIS'*$_{m,q,\beta}$ *problem is at least as hard as 1D-SIS*$_{m+1,q,2(\beta+1)}$.

Proof. Let $\mathcal{A}$ be an adversary for 1D-SIS'$_{m,q,\beta}$ and let $\mathbf{v} = (v_1, ..., v_m) \in \mathbb{Z}_q^m$ be a random instance of the 1D-SIS$_{m,q,\beta}$ problem. If v_1 is not a unit in $\mathbb{Z}_q$, then the reduction aborts. Otherwise, we define the vector $\mathbf{v}' = (v_1^{-1} \cdot v_2, ..., v_1^{-1} \cdot v_2) \in \mathbb{Z}_q^{m-1}$ and provide $\mathbf{v}'$ to $\mathcal{A}$ to get a solution $\mathbf{z}' = (z'_1, ..., z'_{m-1}) \in \mathbb{Z}^{m-1}$ such that $\langle \mathbf{v}', \mathbf{z}' \rangle \in [-\beta, \beta] + q \cdot (\mathbb{Z} + 1/2)$. Then, consider the vector $\mathbf{z} = (-2 \langle \mathbf{v}', \mathbf{z}' \rangle, 2z'_1, ..., 2z'_{m-1})$. Since $\langle \mathbf{v}', \mathbf{z}' \rangle \in [-\beta, \beta] + q \cdot (\mathbb{Z} + 1/2)$, we have that $2 \cdot \langle \mathbf{v}', \mathbf{z}' \rangle \in [-2(\beta + 1), 2(\beta + 1)] + q\mathbb{Z}$, which shows that $\|\mathbf{z}\| \leq 2(\beta + 1)$. Also, it is easy to see that $\langle \mathbf{v}, \mathbf{z} \rangle = 2v_1(\langle \mathbf{v}', \mathbf{z}' \rangle - \langle \mathbf{v}', \mathbf{z}' \rangle) = 0 \mod q$. □

Now we show that the 1D-SIS-R problem is at least as hard as 1D-SIS', which follows in a straightforward way.

Lemma B.3. *Let m, β be positive integers and let q be a product of $n + 1$ prime modulus $p_1 < p_2 < \ldots < p_n$ and p, $q = p \prod_{i \in [n]} p_i$. Then, the 1D-SIS-R$_{m,q,p,\beta}$ is at least as hard as 1D-SIS'$_{m,q/p,\beta}$.*

Proof. Let $\mathcal{A}$ be an adversary for 1D-SIS-R$_{m,q,p,\beta}$. Given a random instance $\mathbf{v} \in \mathbb{Z}_{q/p}^m$ of 1D-SIS'$_{m,q/p,\beta}$, sample a random vector $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^m$ and compute $\mathbf{v}' \in \mathbb{Z}_q^m$ using the Chinese remainder theorem such that $\mathbf{v}' = \mathbf{v} \mod (p/q)$ and $\mathbf{v}' = \mathbf{r} \mod p$. Feeding $\mathcal{A}$ with $\mathbf{v}'$, we get a solution $\mathbf{z}'$ such that $\langle \mathbf{v}', \mathbf{z}' \rangle \in [-\beta, \beta] + (q/p)(\mathbb{Z} + 1/2)$. It is easy to see that by definition, $\|\mathbf{z}'\| \leq \beta$ and therefore, $\mathbf{z}'$ is a valid solution to 1D-SIS'$_{m,q/p,\beta}$. □