Constraining and Watermarking PRFs from Milder Assumptions

Chris Peikert[1]* and Sina Shiehian[2]**

[1] Computer Science and Engineering, University of Michigan, Ann Arbor, USA
[2] Computer Science Department, University of Maryland, College Park, USA

Abstract. *Constrained* pseudorandom functions (C-PRFs) let the possessor of a secret key delegate the ability to evaluate the function on certain authorized inputs, while keeping the remaining function values pseudorandom. A *constraint-hiding* constrained PRF (CHC-PRF) additionally conceals the predicate that determines which inputs are authorized. These primitives have a wealth of applications, including watermarking schemes, symmetric deniable encryption, and updatable garbled circuits.

Recent works have constructed (CH)C-PRFs from rather aggressive parameterizations of Learning With Errors (LWE) with *subexponential* modulus-noise ratios, even for relatively simple "puncturing" or NC^1 circuit constraints. This corresponds to strong lattice assumptions and inefficient constructions, and stands in contrast to LWE-based unconstrained PRFs and fully homomorphic encryption schemes, which can be based on quasi-polynomial or even (nearly) polynomial modulus-noise ratios.

In this work we considerably improve the LWE assumptions needed for building (constraint-hiding) constrained PRFs and watermarking schemes. In particular, for CHC-PRFs and related watermarking schemes we improve the modulus-noise ratio to $\lambda^{O((d+\log\lambda)\log\lambda)}$ for depth-d circuit constraints, which is merely quasi-polynomial for NC^1 circuits and closely related watermarking schemes. For (constraint-revealing) C-PRFs for NC^1 we do even better, obtaining a *nearly polynomial* $\lambda^{\omega(1)}$ ratio. These improvements are partly enabled by slightly modifying the definition of C-PRFs, in a way that is still compatible with many of their applications.

Finally, as a contribution of independent interest we build CHC-PRFs for special constraint classes from *generic*, weaker assumptions: we obtain bit-fixing constraints based on the minimal assumption of one-way functions, and hyperplane-membership constraints based on key-homomorphic PRFs.

* This material is based upon work supported by the National Science Foundation under CAREER Award CCF-1054495 and CNS-1606362. The views expressed are those of the authors and do not necessarily reflect the official policy or position of the National Science Foundation or the Sloan Foundation.

** Work was done while at the University of Michigan.

1 Introduction

Constrained pseudorandom functions (C-PRFs), introduced concurrently and independently by [26,11,12], are PRFs in which the holder of the secret key can delegate *constrained* keys that let one evaluate the function on certain *authorized* inputs, while keeping the function values on all other inputs pseudorandom. Constrained PRFs for various constraint classes have been constructed under different assumptions, including ones where the set of authorized inputs can be specified by an arbitrary boolean circuit. (See below for details.)

In the original conception and constructions of C-PRFs, a constrained key can and does reveal the constraint that determines whether an input is authorized. Boneh, Lewi, and Wu [10] introduced the notion of *constraint-hiding* constrained PRFs (CHC-PRFs), also known as private constrained PRFs, in which constrained keys conceal their underlying constraints. In particular, they considered CHC-PRFs for "punctured" constraints that authorize all but a single input. They also defined *privately programmable* PRFs (PP-PRFs), which allow the constrained key to be "programmed" to output a desired value at the punctured input, and showed that PP-PRFs can be used to build *watermarkable* PRFs [19].

1.1 Constructions and Assumptions

By now there are many constructions of constrained PRFs and their descendants, under various assumptions. The original works of [26,11,12] constructed (constraint-revealing) punctured PRFs based on the minimal assumption that one-way functions exist. Additionally, Boneh and Waters [11] constructed C-PRFs for constraints represented by arbitrary polynomial-sized circuits, under the strong assumption that cryptographic multilinear maps exist. Subsequently, Brakerski and Vaikuntanathan [17] gave a construction based on the Learning With Errors (LWE) assumption, but for which security holds only for a single constrained key. More recently, Attrapadung *et al.* [4] built C-PRFs for NC^1 constraints under number-theoretic assumptions, specifically, DDH and L-DDHI.

Moving now to constraint-hiding constrained PRFs, Boneh *et al.* [10] constructed them for arbitrary (polynomial-sized) constraining circuits, under the strong assumption that indistinguishability obfuscation (iO) exists [7,35]. LWE-based constructions soon followed, first for puncturing constraints [9], then for NC^1 circuits [18], then for all polynomial-sized circuits [15,32]. Like [17], all these LWE-based constructions are secure only for a single constrained key. However, this is an inherent limitation of CHC-PRFs for NC^1 circuits, because security for even two keys implies iO [18].

For privately programmable PRFs and watermarking schemes, the original constructions from [19,10] were based on iO. Later, Kim and Wu [27] built LWE-based watermarking schemes through a different but conceptually similar approach related to programming PRFs. Subsequently, Peikert and Shiehian [32] actually constructed LWE-based privately programmable PRFs.

1.2 LWE Error Rate

An important parameter in the LWE problem, which is related to both its concrete hardness and its connection to lattice problems, is the *error rate* α, or equivalently, the modulus-to-noise ratio $q/r = 1/\alpha$, where q is the modulus and r is the "width" of the (Gaussian) error distribution. In more detail, dimension-n LWE with error width $r \geq 2\sqrt{n}$ is at least as hard as (quantumly) approximating various worst-case lattice problems to within $\tilde{O}(n/\alpha) = \tilde{O}(q\sqrt{n})$ factors on n-dimensional lattices [34,31]. Therefore, using a smaller modulus q (equivalently, a larger error rate α) yields both a stronger security guarantee and a more efficient scheme. More concretely, according to current lattice algorithms, obtaining λ bits of security requires using a dimension $n = \lambda \cdot \tilde{\Omega}(\log(1/\alpha))$, and representing elements of $\mathbb{Z}_q$ requires $\log q = \tilde{\Omega}(\log(1/\alpha))$ bits. Therefore, LWE-based cryptographic schemes using a small error rate α (i.e., large q) suffer from large parameters and key sizes that can be cubic, or even quartic, in $\log(1/\alpha)$.

While there are LWE-based (ordinary) PRFs where the modulus is quasi-polynomial $\lambda^{\mathrm{polylog}(\lambda)}$ [6] or even nearly polynomial $\lambda^{\omega(1)}$ [5], the current LWE-based *constrained* PRFs for punctured, NC^1, and arbitrary circuit constraints all require a *subexponential* $\exp(\mathrm{poly}(\lambda))$ modulus (unless the domain of the PRF is restricted to quasi-polynomially long strings). It is instructive to compare this state of affairs with fully homomorphic encryption (FHE) schemes, whose underlying techniques are used in the constrained PRFs.

Without bootstrapping, state-of-the-art "leveled" FHE schemes [13,22] require a modulus that is merely *exponential in the depth* of the supported circuit class of homomorphic computations. (Bootstrapping can bring the modulus down to quasi-polynomial [13,22] or even polynomial [16,1], independent of the depth of the supported circuits.) By contrast, for constrained PRFs the modulus is *always subexponential in λ, regardless of the circuit depth.* In particular, NC^1 circuits induce a subexponential modulus, instead of a quasi-polynomial one as we might hope. This seems to be an artifact unrelated to the main construction and proof techniques, and raises the following natural question:

Question 1. Can we construct LWE-based (constraint-hiding) constrained PRFs with smaller-than-subexponential modulus, e.g., exponential in the depth of the circuit class?

We also point out that all of the known LWE-based watermarkable PRFs [27,10] (excluding [33], which is not pseudorandom to the setup authority), where the latter is instantiated with LWE-based PP-PRFs [32], also need a subexponential modulus. Roughly speaking, these constructions are essentially built upon privately puncturable PRFs. This motivates the following question:

Question 2. Can we construct LWE-based watermarkable PRFs with a quasi-polynomial modulus?

1.3 Our Results

In this work, our main focus is on improving the LWE assumptions needed for constructing (single-key) constrained PRFs, including their constraint-hiding and privately programmable variants. As a contribution of independent interest, we also obtain (single-key) CHC-PRFs for bit-fixing and hyperplane-membership constraints from *generic* assumptions, namely, one-way functions and key-homomorphic PRFs, respectively.

Our main insight is that by slightly modifying the correctness requirement for constrained keys—but in a way that is still strong enough for most applications—we can construct C-PRFs using much larger LWE error rates, and hence much weaker assumptions and much smaller moduli and key sizes. In particular, we answer Question 1 in the affirmative. We also demonstrate that our new notion of correctness is sufficient for many of the applications of C-PRFs, including watermarking schemes and updatable garbled circuits; this positively answers Question 2 as well. We stress that the security level of our constructions scale proportional to the inverse of the LWE error rate; however, we note that this is a property shared by all current efficient (ordinary) lattice-based PRFs [6,5]. We now summarize our specific results.

Feasible correctness. We first observe that satisfying a strict correctness requirement for constrained keys is the main reason previous LWE-based C-PRFs needed a subexponential modulus. In a bit more detail, the prior definitions require that, given a constrained key, it is computationally hard (or even impossible) to find an authorized input where the constrained key yields a different output than the real key. We give an alternative definition, which says that no efficient adversary, even with oracle access to the function, can *find* an input x for which *there exists* a constrained key that authorizes x yet yields a different output than the real key. (However, *after* obtaining a constrained key, an adversary may be able to find such an input.) We call this new notion *feasible correctness.*

Feasibly correct PP-PRFs and watermarking PRFs from LWE with quasi-polynomial modulus. Our first construction under our new correctness notion is a key-injective, privately programmable PRF, based on LWE with only a quasi-polynomial modulus $q = \lambda^{O(\log^2 \lambda)}$. We plug this construction into the watermarking PRF construction of [10] and show that the resulting scheme satisfies all of the watermarking requirements presented in [27] (which are stronger than the definitions in [10]). This results in a watermarking scheme from LWE with quasi-polynomial modulus $q = \lambda^{O(\log^2 \lambda)}$, improving on the prior best of subexponential.

Feasibly correct CHC-PRFs from LWE with quasi-polynomial modulus. We next construct a feasibly correct CHC-PRF for arbitrary polynomial-sized circuit constraints, based on LWE with modulus $q = \lambda^{O((d+\log \lambda)\log \lambda)}$ where d is the depth of the supported circuit class. As an application, we instantiate the "message-embedding" construction of watermarkable PRFs from [33], which uses CHC-PRFs for log-depth constraints, with our feasibly correct CHC-PRF, thus reducing the modulus size from subexponential to quasi-polynomial $\lambda^{O(\log^2 \lambda)}$.

Feasibly correct C-PRFs from LWE with nearly polynomial modulus. Using the construction in the previous paragraph, feasibly correct CHC-PRFs for NC^1 circuits require a quasi-polynomial modulus $q = \lambda^{O(\log^2 \lambda)}$. Trivially, this construction is also a feasibly correct C-PRF for NC^1. However, we go farther by constructing such a C-PRF from LWE with only a *nearly polynomial* modulus $q = \lambda^{\omega(1)}$, by building upon the branching-program techniques of [16]. As an application, we show that we can replace regular C-PRFs with feasibly correct ones in the updatable garbled circuits construction of [2], thus reducing the modulus size from subexponential to nearly polynomial.

Bit-fixing and hyperplane-membership PRFs from generic assumptions. As results of independent interest, we build CHC-PRFs for specific constraint classes based on *generic*, weaker assumptions than prior constructions. We consider the class of constraints that authorize inputs that lie in a specified hyperplane. For such constraints we build (feasibly correct) CHC-PRFs generically from key-homomorphic PRFs. Using the key-homomorphic PRFs of [5], we can base the security of our construction on LWE with nearly polynomial modulus $q = \lambda^{\omega(1)}$, which is significantly smaller than the quasi-polynomial $q = \lambda^{O(\log^2 \lambda)}$ that we would get by naïvely using our feasibly correct CHC-PRF for NC^1 circuits.

Lastly, for bit-fixing constraints, i.e., constraints that authorize strings matching a specified pattern in $\{0, 1, \star\}^*$ (where $\star$ denotes the wildcard symbol), we build (fully correct) CHC-PRFs based on the minimal assumption that one-way functions exist. Previously, bit-fixing PRFs were only known based on LWE with subexponential modulus [18,15,32], DDH [4], or multilinear maps [11], although the latter can securely issue more than one constrained key.

1.4 Concurrent and Independent Works

In a concurrent and independent work, Kim and Wu [28] construct watermarking PRFs from LWE with quasi-polynomial (nearly polynomial) modulus. While their security model for watermarking PRFs is an interesting strengthening of the model in [33], however, similar to [33] their PRFs do not offer full pseudorandomness in the presence of the setup authority. Furthermore, to make the LWE modulus quasi-polynomial (nearly polynomial), they have to limit the input domain of their PRFs to polylogarithmically (nearly logarithmically) long bit-strings. In comparison, both of our watermarking constructions support polynomially long bit-strings as inputs, with one satisfying the authority pseudorandom model of [27] and the other satisfying the [33] model.

In another concurrent work, Davidson, Katsumata, Nishimaki and Yamada [21] construct bit-fixing PRFs from one-way functions. Their construction is very similar to what we present in Construction 6. Later, Tsabary [36] makes an observation essentially identical to our Remark 1 and uses it to build LWE-based adaptively secure attribute based encryption for constant-width CNFs.

1.5 Techniques

Achieving feasible correctness. We start by reviewing why the correctness definition in current LWE-based C-PRFs leads to a subexponential modulus. In these constructions, to compute the function on an input x, first a value $\mathbf{y}_x \in \mathbb{Z}_q^m$ is computed and the final output is $\lfloor \mathbf{y}_x \rceil_p$. Using a constrained key for a depth-d constraint C that authorizes x, one can obtain $\mathbf{y}_x + \mathbf{e}_x$ where $\mathbf{e}_x$ is a B-bounded error vector for some $B = \lambda^{\tilde{O}(d)}$. For correctness we need $\lfloor \mathbf{y}_x + \mathbf{e} \rceil_p = \lfloor \mathbf{y}_x \rceil_p$, i.e., the coordinates of $\mathbf{y}_x$ should not be in the *border* interval $\frac{q}{p}(\mathbb{Z} + \frac{1}{2}) + [-B, B]$. Unfortunately, to guarantee that, given a constrained key, an adversary cannot find an input x such that $\mathbf{y}_x$ has a coordinate in the border interval, we currently do not know any solution other than making q subexponential. This is because we need to rely on the hardness of the "1-dimensional SIS problem" over $\mathbb{Z}_q$, as originally used in [17], or use a union bound over the subexponential PRF domain, as in [15].

We observe that if we can make $\mathbf{y}_x$ a pseudorandom function of x then an alternative "feasible correctness" property can be achieved. Namely, an adversary without a constrained key, but with oracle access to the PRF functionality, can only produce an x for which $\mathbf{y}_x$ has a coordinate in the border interval with probability at most $(Bp/q) \cdot \mathrm{poly}(\lambda)$. Setting $q = Bp \cdot \lambda^{\omega(1)} = \lambda^{\tilde{O}(d)}$ yields feasible correctness.

Interestingly, we observe that the the notion of feasible correctness is compatible with many applications of C-PRFs. Most notably, the watermarking schemes based on C-PRFs maintain all of their requisite properties, because correctness only requires agreement between marked and unmarked PRFs on an overwhelming fraction of inputs. More generally (and somewhat informally), as long as a C-PRF is used in a context where it is evaluated on inputs that do not depend on the constrained key, then feasible correctness can substitute for full correctness.

Making $\mathbf{y}_x$ a pseudorandom function of x can be done by adding an independent PRF value in the computation of $\mathbf{y}_x$. In more detail, we generate a key $\kappa \leftarrow \mathsf{PRF.KG}(1^\lambda)$ for an arbitrary PRF with the same input domain as our PRF and with range $\mathbb{Z}_q^m$, and output it as part of both the master secret key and the constrained key. When evaluating the PRF on input x we compute $\mathbf{y}_x$ as before, then compute $\mathbf{y}'_x = \mathbf{y}_x + \mathsf{PRF.Eval}(\kappa, x)$ and output $\lfloor \mathbf{y}'_x \rceil_p$. Evaluation using a constrained key is similar. Using an LWE-based PRF which only requires a nearly polynomial modulus [5], we achieve feasible correctness from LWE with modulus $q = \lambda^{\tilde{O}(d)}$.

Constrained PRFs for NC^1 constraints. We now describe how we construct feasibly correct (constraint-revealing) C-PRFs for NC^1 constraints from LWE with a *nearly polynomial* modulus $q = \lambda^{\omega(1)}$. Conceptually, our construction is similar to the one of [15], however we also use the technique from [16] of representing computations as branching programs. For each input x we denote by $\mathbf{M}_x$ the efficiently computable public binary matrix constructed as in [5]. These matrices have the property that for a uniformly random $\mathbf{s}$ over $\mathbb{Z}_q$, the randomized procedure which, on input x, samples a sufficiently wide Gaussian error $\mathbf{e}_x$ and outputs $\mathbf{s}\mathbf{M}_x + \mathbf{e}_x$, is pseudorandom. Our construction also crucially relies on the [16] procedure for homomorphically evaluating branching programs.

In our construction, a constrained key for a circuit C consists of the circuit C, a key κ for an auxiliary PRF, and several LWE samples $\mathbf{a}_i = \mathbf{s}(\mathbf{A}_i + C_i \cdot G) + \mathbf{e}_i$, where the C_is are the individual bits of C. Evaluating the PRF at an input x is done as follows:

- We use the "gadget homomorphisms" for branching programs with asymmetric noise growth [16] to homomorphically evaluate $U_x(\cdot)$ on the LWE samples $\mathbf{a}_i$, where $U_x(C) = C(x)$ is the branching program for a depth-universal circuit for NC^1. The result is

$$\mathbf{a}_x = \mathbf{s} \cdot \mathbf{A}_x + C(x) \cdot \mathbf{s} \cdot \mathbf{G} + \mathbf{e}_x, \tag{1}$$

  where $\mathbf{A}_x$ does not depend on C and $\mathbf{e}_x$ is polynomially bounded.
- Next, we multiply $\mathbf{a}_x$ by $\mathbf{G}^{-1}(\mathbf{A} \cdot \mathbf{M}_x)$, where $\mathbf{A}$ is a public uniformly random matrix over $\mathbb{Z}_q$, to get

$$\mathbf{b}_x = \mathbf{s} \cdot \mathbf{A}_x \cdot \mathbf{G}^{-1}(\mathbf{A} \cdot \mathbf{M}_x) + C(x) \cdot \mathbf{s} \cdot \mathbf{A} \cdot \mathbf{M}_x + \mathbf{e}', \tag{2}$$

  where $\mathbf{e}'$ is also polynomially bounded.
- Finally, we define the value of the PRF at input x to be

$$\lfloor \mathbf{s} \cdot \mathbf{A}_x \cdot \mathbf{G}^{-1}(\mathbf{A} \cdot \mathbf{M}_x) + \mathsf{PRF.Eval}(\kappa, x) \rceil_p, \tag{3}$$

  and the constrained value at x to be

$$\lfloor \mathbf{b}_x + \mathsf{PRF.Eval}(\kappa, x) \rceil_p. \tag{4}$$

Because $\mathbf{e}'$ is B-bounded for $B = \mathrm{poly}(\lambda)$, a nearly polynomial modulus $q = B \cdot \lambda^{\omega(1)} = \lambda^{\omega(1)}$ is sufficient for feasible correctness. To show that the C-PRF at unauthorized inputs x (i.e., where $C(x) = 1$) remains pseudorandom, we have to argue that the extra term $\mathbf{s} \cdot \mathbf{A} \cdot \mathbf{M}_x + \mathbf{e}'$ completely masks $\mathbf{s} \cdot \mathbf{A}_x \cdot \mathbf{G}^{-1}(\mathbf{A} \cdot \mathbf{M}_x)$. The high-level idea here is that, because $\mathbf{M}_x$ has small entries, $\mathbf{s} \cdot \mathbf{A} \cdot \mathbf{M}_x + \mathbf{e}'$ and $(\mathbf{s} \cdot \mathbf{A} + \mathbf{e}'') \cdot \mathbf{M}_x + \mathbf{e}'$ are very close to each other. Then by LWE, $(\mathbf{s} \cdot \mathbf{A} + \mathbf{e}'')$ and a uniformly chosen $\mathbf{s}'$ are indistinguishable. But as already noted, $\mathbf{s}' \cdot \mathbf{M}_x + \mathbf{e}'$ is pseudorandom, as desired.

Generic CHC-PRFs for hyperplane membership constraints. We give a brief overview of our generic feasibly correct CHC-PRF construction for the class of hyperplane membership predicates. This construction can be based on any (noisy) key-homomorphic PRF. Here, for simplicity we only consider constraints of dimension 1, i.e., each constraint consists of a pair $(\alpha_0 \in \mathbb{Z}_q, \alpha_1 \in \mathbb{Z}_q)$ for some modulus q and only authorizes inputs $x \in \mathbb{Z}_q$ such that $\alpha_0 + \alpha_1 \cdot x = 0 \pmod{q}$. Generalizing for higher dimensions is straightforward. Let KHPRF be a key-homomorphic PRF. The master secret key consists of two keys (k_0, k_1) for KHPRF. To evaluate the PRF on an input x, we output $\mathsf{KHPRF.Eval}(k_0, x) +$

$\mathsf{KHPRF.Eval}(x \cdot k_1, x)$. To produce a constrained key for constraint (α_0, α_1), we first sample a KHPRF key d and then output $(k_0 - \alpha_0 d, k_1 - \alpha_1 d)$ as the constrained key. It is straightforward to observe that the constrained-key (perfectly) hides (α_0, α_1). The correctness and pseudorandomness property follow from the key-homomorphism and the pseudorandomness of KHPRF. This is because

$$\begin{aligned}\mathsf{Eval}(k_0 - \alpha_1 d, x) + \mathsf{Eval}(x \cdot (k_1 - \alpha_1 d), x) = \mathsf{Eval}(k_0, x) + \\ \mathsf{Eval}(x \cdot k_1, x) - (\alpha_0 + \alpha_1 \cdot x)\mathsf{Eval}(d, x),\end{aligned} \tag{5}$$

and in particular if $\alpha_0 + \alpha_1 \cdot x \neq 0 \pmod q$ then the last term computationally hides the true value of the PRF.

2 Preliminaries

We denote row vectors by lower-case bold letters, e.g., $\mathbf{a}$. We denote matrices by upper-case bold letters, e.g., $\mathbf{A}$. The Kronecker product $\mathbf{A} \otimes \mathbf{B}$ of two matrices (or vectors) $\mathbf{A}$ and $\mathbf{B}$ is obtained by replacing each entry $a_{i,j}$ of $\mathbf{A}$ with the block $a_{i,j}\mathbf{B}$.

Depth-universal circuits. We use *depth-universal* circuits, i.e., universal circuits with depth $O(d)$ where d is the depth of the simulated circuit class. The construction of Cook and Hover [20] is depth-universal and has size $O(\sigma^3 \cdot d/\log d)$ for circuits of size σ and depth d.

Learning with errors. For a positive integer dimension n and modulus q, and an error distribution χ over $\mathbb{Z}$, the LWE distribution and decision problem are defined as follows. For an $\mathbf{s} \in \mathbb{Z}^n$, the LWE distribution $A_{\mathbf{s},\chi}$ is sampled by choosing a uniformly random $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ and an error term $e \leftarrow \chi$, and outputting $(\mathbf{a}, b = \langle \mathbf{s}, \mathbf{a} \rangle + e) \in \mathbb{Z}_q^{n+1}$.

Definition 1. *The decision-LWE$_{n,q,\chi}$ problem is to distinguish, with non-negligible advantage, between any desired (but polynomially bounded) number of independent samples drawn from $A_{\mathbf{s},\chi}$ for a single $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, and the same number of* uniformly random *and independent samples over $\mathbb{Z}_q^{n+1}$.*

A standard instantiation of LWE is to let χ be a *discrete Gaussian* distribution (over $\mathbb{Z}$) with parameter $r = 2\sqrt{n}$. A sample drawn from this distribution has magnitude bounded by, say, $r\sqrt{n} = \Theta(n)$ except with probability at most 2^{-n}. For this parameterization, it is known that LWE is at least as hard as *quantumly* approximating certain "short vector" problems on n-dimensional lattices, in the worst case, to within $\tilde{O}(q\sqrt{n})$ factors [34,31]. Classical reductions are also known for different parameterizations [30,14].

3 Definitions

Here we recall prior definitions of constrained PRFs [32], then relax them to our new notion of *feasible* correctness.

Definition 2. *A* constrained function *for a constraint class $\mathcal{C}$ is given by a tuple of efficient algorithms* ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval}$) *having the following interfaces (where the domain $\mathcal{X}$, range $\mathcal{Y}$, and class $\mathcal{C}$ may depend on the security parameter):*

– $\mathsf{Setup}(1^\lambda)$*, given the security parameter λ, outputs public parameters pp.*
– $\mathsf{KeyGen}(pp)$*, given the public parameters pp, outputs a master secret key msk.*
– $\mathsf{Eval}(pp, msk, x)$*, given the master secret key and an input $x \in \mathcal{X}$, outputs some $y \in \mathcal{Y}$.*
– $\mathsf{Constrain}(pp, msk, C)$*, given the master secret key and a constraint $C \in \mathcal{C}$, outputs a constrained key sk_C.*
– $\mathsf{CEval}(pp, sk_C, x)$*, given a constrained key sk_C and an input $x \in \mathcal{X}$, outputs some $y \in \mathcal{Y}$.*

In some constructions there is no need for a Setup *algorithm, in which case the security parameter 1^λ takes the place of the public parameters pp.*

```
procedure $\mathsf{CHCPRFReal}_{\mathcal{A}}(1^\lambda)$
    $C \leftarrow \mathcal{A}(1^\lambda)$
    $pp \leftarrow \mathsf{Setup}(1^\lambda)$
    $msk \leftarrow \mathsf{KeyGen}(pp)$
    $sk_C \leftarrow \mathsf{Constrain}(pp, msk, C)$
    $sk_C \rightarrow \mathcal{A}$
    repeat
        $x \leftarrow \mathcal{A}$
        $\mathsf{Eval}(pp, msk, x) \rightarrow \mathcal{A}$
    until $\mathcal{A}$ halts
```

(a) The real experiment

```
procedure $\mathsf{CHCPRFIdeal}_{\mathcal{A},\mathcal{S}}(1^\lambda)$
    $C \leftarrow \mathcal{A}(1^\lambda)$
    $(pp, sk) \leftarrow \mathcal{S}(1^\lambda)$
    $sk \rightarrow \mathcal{A}$
    repeat
        $x \leftarrow \mathcal{A}$
        if $C(x) = \text{true}$ then
            $\mathsf{CEval}(pp, sk, x) \rightarrow \mathcal{A}$
        else
            $y \leftarrow \mathcal{Y}$; $y \rightarrow \mathcal{A}$
    until $\mathcal{A}$ halts
```

(b) The ideal experiment

Fig. 1: The real and ideal constraint-hiding constrained PRF experiments.

Definition 3. *A constrained function is a* constraint-hiding constrained PRF *(CHC-PRF) if there is a PPT simulator $\mathcal{S}$ such that, for any PPT adversary $\mathcal{A}$ (that without loss of generality never repeats an* Eval *query),*

$$\{\mathsf{CHCPRFReal}_{\mathcal{A}}(1^\lambda)\}_{\lambda \in \mathbb{N}} \overset{c}{\approx} \{\mathsf{CHCPRFIdeal}_{\mathcal{A},\mathcal{S}}(1^\lambda)\}_{\lambda \in \mathbb{N}}, \tag{6}$$

where $\mathsf{CHCPRFReal}$ *and* $\mathsf{CHCPRFIdeal}$ *are the respective views of $\mathcal{A}$ in the experiments defined in Figure 1.*

We now introduce an alternative (but still sufficient for applications) notion of correctness for constrained evaluation, which we call *feasible correctness.* This requires that CEval and Eval agree on any input x that the adversary outputs after having query access to Eval, but *before* obtaining a constrained key. Correspondingly, in the simulation-based security definitions we limit the adversary to query only *unauthorized* inputs. This is because after obtaining the constrained key, it may be able to find inputs x on which CEval and Eval differ, which would allow it to distinguish the real and ideal experiments. We still call this property *simulation security*, where the restriction to unauthorized inputs is clear by the context of feasible correctness.

Definition 4. *A constrained function is* feasibly correct *if for all PPT adversaries $\mathcal{A}$ and all pp in the support of* Setup *we have*

$$\Pr\left[\begin{array}{c}\exists C \in \mathcal{C} \text{ s.t. } C(x) = \mathit{true} \\ \exists sk_C \in \text{support of } \mathsf{Constrain}(pp, msk, C) \text{ s.t.} \\ \mathsf{CEval}(pp, sk_C, x) \neq \mathsf{Eval}(pp, msk, x)\end{array}\right] = \operatorname{negl}(\lambda), \tag{7}$$

where $msk \leftarrow \mathsf{KeyGen}(pp)$, $x \leftarrow \mathcal{A}^{\mathsf{Eval}(pp,msk,\cdot)}(pp)$, and the probability is taken over the random coins of the KeyGen *algorithm and the random coins of the adversary $\mathcal{A}$.*

We recall the notion of key-homomorphic PRFs.

Definition 5. *A PRF* $(\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Eval})$ *with domain $\mathcal{X}$, finite-group key space $\mathcal{K}$, and range $\mathbb{Z}_p^m$ for some integer modulus p and dimension m, is* noisy key homomorphic *with noise bound E if for every pp in the support of* Setup*, every two keys msk_1, msk_2 in the support of* $\mathsf{Keygen}(pp)$*, and every $x \in \mathcal{X}$, we have*

$$\mathsf{Eval}(pp, msk_1 + msk_2, x) = \mathsf{Eval}(pp, msk_1, x) + \mathsf{Eval}(pp, msk_2, x) + \mathbf{e} \tag{8}$$

for some noise vector $\mathbf{e}$ where each entry of $\mathbf{e}$ has magnitude at most E.

Theorem 1 ([5]). *Assuming the hardness of* LWE *with nearly polynomial modulus size $q = \lambda^{\omega(1)}$, there exists a noisy key-homomorphic PRF with key space $\mathbb{Z}_q^n$, range $\mathbb{Z}_p^m$ for any p for which $p/q = \operatorname{negl}(\lambda)$, and noise upper bound $E = 1$.*

4 Feasibly Correct Shift-Hiding Shiftable Functions

Recall the notion of shift-hiding shiftable functions (SHSFs) in [32] (a brief overview is available in Appendix A). Here we give a construction of what we call *feasibly correct shift-hiding shiftable functions* (FC-SHSFs) which are essentially SHSFs which satisfy *shift hiding* and *approximate shift correctness* but instead of *border avoiding* they satisfy a new notion which we call *feasible border avoiding.* Interestingly, the main building block for our FC-SHSFs is a SHSF.

4.1 Notation

Let $\mathsf{PRF} = (\mathsf{KG}, \mathsf{Eval})$ be a PRF. We can instantiate PRF based on various assumptions. In particular, the instantiation in [5] which is based on LWE with a nearly polynomial modulus, incurs no additional cost to later applications of our FC-SHSF construction. Let $U(H, x) = H(x)$ denote a depth-universal circuit for boolean circuits $H \colon \{0,1\}^\ell \to \{0,1\}^k$ of size σ and depth d, and let $U_x(\cdot) = U(\cdot, x)$. Denote the SHSFs constructed in [32] by $\mathsf{PSSHSF} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Shift}, \mathsf{SEval}, \mathcal{S})$ and recall that in PSSHSF the noise growth is $\lambda^{O(d \log \lambda)}$.

4.2 Construction

Here we give the tuple of algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Shift}, \mathsf{SEval}, \mathcal{S})$ that make up our SHSF. For security parameter λ, constraint circuit size σ, and constraint circuit depth d the algorithms are parameterized by some dimension n and modulus q, and $m = n\lceil \lg q \rceil$.

Construction 1. Let $\mathcal{X} = \{0,1\}^\ell$ and $\mathcal{Y} = \mathbb{Z}_q^m$. Define:

- $\mathsf{Setup}(1^\lambda, 1^\sigma, 1^d)$: Generate PSSHSF public parameters $pp' \leftarrow \mathsf{PSSHSF.Setup}(1^\lambda, 1^\sigma, 1^d)$. Output $pp := pp'$.
- $\mathsf{KeyGen}(pp)$: Generate a PRF key $\kappa \leftarrow \mathsf{PRF.KG}(1^\lambda)$ and PSSHSF master secret key $msk' \leftarrow \mathsf{PSSHSF.KeyGen}(1^\lambda)$. Output (κ, msk').
- $\mathsf{Eval}(pp, msk, x \in \{0,1\}^\ell)$: output
$$\mathsf{PRF.Eval}(\kappa, x) + \mathsf{PSSHSF.Eval}(pp', msk', x) \tag{9}$$
- $\mathsf{Shift}(pp, msk, H)$: for a shift function $H \colon \{0,1\}^\ell \to \mathbb{Z}_q^m$ whose binary decomposition $H' \colon \{0,1\}^\ell \to \{0,1\}^k$ can be implemented by a circuit of size σ, compute $sk'_H \leftarrow \mathsf{SHSF.Shift}(pp', msk', H)$. Output
$$sk_H = (\kappa, sk'_H). \tag{10}$$
- $\mathsf{SEval}(pp, sk_H, x)$: On input $sk_H = (\kappa, sk'_H)$ and $x \in \{0,1\}^\ell$, output
$$\mathsf{PRF.Eval}(\kappa, x) + \mathsf{PSSHSF.SEval}(pp', sk'_H, x) \tag{11}$$
- $\mathcal{S}(1^\lambda, 1^\sigma, 1^d)$: Sample a PRF key $\kappa \leftarrow \mathsf{PRF.KG}(1^\lambda)$, a simulated PSSHSF key $(pp', sk') \leftarrow \mathsf{PSSHSF.Sim}(1^\lambda, 1^\sigma)$, and output $pp = (pp')$ and $sk = (\kappa, sk')$.

4.3 Properties

We now state the main properties of our construction that we will use in subsequent sections.

The following two lemmas follow directly from the shift-hiding and approximate shift correctness properties of PSSHSF. So, we omit the proofs of these lemmas.

```
procedure RealKey_A(1^λ, 1^σ, 1^d)
    $H \leftarrow \mathcal{A}(1^\lambda, 1^\sigma, 1^d)$
    $pp \leftarrow \mathsf{Setup}(1^\lambda, 1^\sigma, 1^d)$
    $msk \leftarrow \mathsf{KeyGen}(pp)$
    $sk \leftarrow \mathsf{Shift}(pp, msk, H)$
    $(pp, sk) \rightarrow \mathcal{A}$
```

(a) The real shifted key generation experiment

```
procedure IdealKey_A(1^λ, 1^σ, 1^d)
    $H \leftarrow \mathcal{A}(1^\lambda, 1^\sigma, 1^d)$
    $(pp, sk) \leftarrow \mathcal{S}(1^\lambda, 1^\sigma, 1^d)$
    $(pp, sk) \rightarrow \mathcal{A}$
```

(b) The random key generation experiment

Fig. 2: The real and random shifted key generation experiments.

Lemma 1 (Shift Hiding). *Assuming the hardness of $LWE_{n-1,q,\chi}$, for any PPT $\mathcal{A}$, any $\sigma = \sigma(\lambda) = \mathrm{poly}(\lambda)$ and any d,*

$$\{\mathsf{RealKey}_{\mathcal{A}}(1^\lambda, 1^\sigma, 1^d)\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \{\mathsf{IdealKey}_{\mathcal{A}}(1^\lambda, 1^\sigma, 1^d)\}_{\lambda\in\mathbb{N}}, \tag{12}$$

where RealKey *and* IdealKey *are the respective views of $\mathcal{A}$ in the experiments defined in Figure 2.*

Lemma 2 (Approximate Shift Correctness). *For any shift function $H\colon \{0,1\}^\ell \to \mathbb{Z}_q^m$ whose binary decomposition $H'\colon \{0,1\}^\ell \to \{0,1\}^k$ can be represented by a boolean circuit of size σ and depth d, and any $x \in \{0,1\}^\ell$, $pp \leftarrow \mathsf{Setup}(1^\lambda, 1^\sigma, 1^d)$, $msk \leftarrow \mathsf{KeyGen}(pp)$ and $sk_H \leftarrow \mathsf{Shift}(pp, msk, H)$, we have*

$$\mathsf{SEval}(pp, sk_H, x) \approx \mathsf{Eval}(pp, msk, x) + H(x) \tag{13}$$

where the approximation hides some $\lambda^{O(d\log\lambda)}$-bounded error vector.

Lemma 3 (Feasible Border Avoiding). *If* PRF *is a pseudorandom function, then for any polynomial-time adversary $\mathcal{A}$, $i \in [m]$, $\sigma = \mathrm{poly}(\lambda)$, $d \in \mathbb{N}$, large enough $B = \lambda^{O(d\log\lambda)} \in \mathbb{N}$, primes p and q such that $q = p \cdot B \cdot \lambda^{\omega(1)}$, pp in the support of* $\mathsf{Setup}(1^\lambda, 1^\sigma, 1^d)$*, and $\beta \in \mathbb{Z}_q$, we have*

$$\Pr_{msk\leftarrow\mathsf{KeyGen}(pp)}\Big[\mathsf{Eval}(pp, msk, \mathcal{A}^{\mathsf{Eval}(pp,msk,\cdot)}(pp))_i \in \frac{q}{p}(\mathbb{Z} + \tfrac{1}{2}) + [\beta - B, \beta + B]\Big] \le \mathrm{negl}(\lambda). \tag{14}$$

The following is an immediate consequence of Lemma 2.

Corollary 1. *Fix the same notation as in Lemma 2. Let $\mathbf{c} \in \mathbb{Z}_q^m$ be a fixed vector. If for all $i \in [m]$ we have*

$$\mathsf{Eval}(pp, sk, x)_i \notin \frac{q}{p}(\mathbb{Z} + \tfrac{1}{2}) + [\mathbf{c}_i - B, \mathbf{c}_i + B], \tag{15}$$

then

$$\lfloor \mathsf{SEval}(pp, sk, x) - H(x) - \mathbf{c} \rceil_p = \lfloor \mathsf{Eval}(pp, msk, x) - \mathbf{c} \rceil_p. \tag{16}$$

5 Feasibly Correct Privately Programmable PRFs

In this section we formally define feasibly correct privately programmable PRFs (PP-PRFs) and give a construction based on our feasibly correct shiftable PRFs from Section 4. Our construction satisfies an additional key-injectivity property which would later be useful in watermarking constructions.

5.1 Definitions

We start by giving a variety of definitions related to "programmable functions" and privately programmable PRFs.

Definition 6. *A* programmable function *is a tuple* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Program}, \mathsf{PEval})$ *of efficient algorithms having the following interfaces (where the domain $\mathcal{X}$ and range $\mathcal{Y}$ may depend on the security parameter):*

- $\mathsf{Setup}(1^\lambda)$*, given the security parameter λ outputs public parameters pp.*
- $\mathsf{KeyGen}(pp)$*, given the public parameters pp, outputs a master secret key msk.*
- $\mathsf{Eval}(pp, msk, x)$*, given the master secret key and an input $x \in \mathcal{X}$, outputs some $y \in \mathcal{Y}$.*
- $\mathsf{Program}(pp, msk, (x^*, y^*))$*, given the master secret key msk and $(x^*, y^*) \in \mathcal{X} \times \mathcal{Y}$, outputs a programmed key $sk_{\mathcal{P}}$.*
- $\mathsf{PEval}(pp, sk_{\mathcal{P}}, x)$*, given a programmed key $sk_{\mathcal{P}}$ and an input $x \in \mathcal{X}$, outputs some $y \in \mathcal{Y}$.*

Definition 7. *A programmable function is* statistically programmable *if for all $\lambda \in \mathbb{N}$ and all pairs $(x^*, y^*) \in \mathcal{X} \times \mathcal{Y}$ we have*

$$\Pr_{\substack{pp \leftarrow \mathsf{Setup}(1^\lambda) \\ msk \leftarrow \mathsf{KeyGen}(pp) \\ sk_{(x^*,y^*)} \leftarrow \mathsf{Program}(pp, msk, (x^*, y^*))}} [\mathsf{PEval}(pp, sk_{(x^*,y^*)}, x^*) \neq y^*] = \mathrm{negl}(\lambda). \tag{17}$$

Definition 8. *A programmable function is* feasibly correct *if for all PPT adversaries $\mathcal{A}$ and all pp in the support of* Setup *we have*

$$\Pr\left[\begin{array}{c} \exists (x^*, y^*) \in \mathcal{X} \backslash \{x\} \times \mathcal{Y} \\ \exists sk_{(x^*,y^*)} \in \text{ support of } \mathsf{Program}(pp, msk, (x^*, y^*)) \text{ s.t.} \\ \mathsf{PEval}(pp, sk_{(x^*,y^*)}, x) \neq \mathsf{Eval}(pp, msk, x) \end{array}\right] = \mathrm{negl}(\lambda), \tag{18}$$

where $msk \leftarrow \mathsf{KeyGen}(pp)$, $x \leftarrow \mathcal{A}^{\mathsf{Eval}(pp, msk, \cdot)}(pp)$, and the probability is taken over the random coins of the KeyGen *algorithm and the random coins of the adversary $\mathcal{A}$.*

Definition 9 (Key-Injectivity). *A programmable function is key-injective if*

$$\Pr_{pp \leftarrow \mathsf{Setup}(1^\lambda)}\left[\begin{array}{c} \exists \text{ distinct } msk_1, msk_2 \in \text{ support of } \mathsf{KeyGen}, x \in \{0,1\}^\ell : \\ \mathsf{Eval}(pp, msk_1, x) = \mathsf{Eval}(pp, msk_2, x) \end{array}\right] \tag{19}$$
$$\leq \mathrm{negl}(\lambda).$$

```
procedure $\mathsf{RealPPRF}_{\mathcal{A}}(1^\lambda)$
    $(x^*, y^*) \leftarrow \mathcal{A}(1^\lambda)$
    $pp \leftarrow \mathsf{Setup}(1^\lambda)$
    $msk \leftarrow \mathsf{KeyGen}(pp)$
    $sk_{(x^*,y^*)} \leftarrow \mathsf{Program}(pp, msk, (x^*, y^*))$
    $(pp, sk_{(x^*,y^*)}) \rightarrow \mathcal{A}$
    $\mathsf{Eval}(pp, msk, x^*) \rightarrow \mathcal{A}$
```

(a) The real experiment

```
procedure $\mathsf{IdealPPRF}_{\mathcal{A},\mathcal{S}}(1^\lambda)$
    $(x^*, y^*) \leftarrow \mathcal{A}(1^\lambda)$
    $(pp, sk_{(x^*,y^*)}) \leftarrow \mathcal{S}(1^\lambda, (x^*, y^*))$
    $(pp, sk_{(x^*,y^*)}) \rightarrow \mathcal{A}$
    $y \leftarrow \mathcal{Y}$; $y \rightarrow \mathcal{A}$
```

(b) The ideal experiment

Fig. 3: The real and ideal experiments

Definition 10. *A programmable function is* simulation secure *if there is a PPT simulator $\mathcal{S}$ such that for any PPT adversary $\mathcal{A}$,*

$$\{\mathsf{RealPPRF}_{\mathcal{A}}(1^\lambda)\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \{\mathsf{IdealPPRF}_{\mathcal{A},\mathcal{S}}(1^\lambda)\}_{\lambda\in\mathbb{N}}, \tag{20}$$

where $\mathsf{RealPPRF}$ *and* $\mathsf{IdealPPRF}$ *are the respective views of $\mathcal{A}$ in the procedures defined in Figure 3.*

```
procedure $\mathsf{RealPPRFPrivacy}_{\mathcal{A}}(1^\lambda)$
    $x^* \leftarrow \mathcal{A}(1^\lambda)$
    $y^* \leftarrow \mathcal{Y}$
    $pp \leftarrow \mathsf{Setup}(1^\lambda)$
    $msk \leftarrow \mathsf{KeyGen}(pp)$
    $sk \leftarrow \mathsf{Program}(pp, msk, (x^*, y^*))$
    $(pp, sk) \rightarrow \mathcal{A}$
```

(a) The real experiment

```
procedure
$\mathsf{IdealPPRFPrivacy}_{\mathcal{A},\mathcal{S}}(1^\lambda)$
    $x^* \leftarrow \mathcal{A}(1^\lambda)$
    $(pp, sk) \leftarrow \mathcal{S}(1^\lambda)$
    $(pp, sk) \rightarrow \mathcal{A}$
```

(b) The ideal experiment

Fig. 4: The real and ideal privacy experiments

Definition 11. *A programmable function is* privately programmable *if there is a PPT simulator $\mathcal{S}$ such that for any PPT adversary $\mathcal{A}$,*

$$\{\mathsf{RealPPRFPrivacy}_{\mathcal{A}}(1^\lambda)\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \{\mathsf{IdealPPRFPrivacy}_{\mathcal{A}}(1^\lambda)\}_{\lambda\in\mathbb{N}}, \tag{21}$$

where $\mathsf{RealPPRFPrivacy}$ *and* $\mathsf{IdealPPRFPrivacy}$ *are the respective views of $\mathcal{A}$ in the procedures defined in Figure 4.*

Definition 12. *A programmable function is a* feasibly correct privately programmable PRF *if it is statistically programmable, simulation secure, privately programmable, and feasibly correct.*

LWE-Based PRGs with Weak Seeds. In this construction we will need an LWE-based PRG $G\colon \mathbb{Z}_q^n \to \{0,1\}^{n_1}$ with the following property:

$$\{G(\mathbf{s}), \mathbf{s}\cdot\mathbf{A}+\mathbf{e}, \mathbf{A} : \mathbf{s},\mathbf{e} \leftarrow \chi^n; \mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}\} \overset{c}{\approx} \\ \{r, \mathbf{a}, \mathbf{A} : r \leftarrow \{0,1\}^{n_1}; \mathbf{a} \leftarrow \mathbb{Z}_q^m; \mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}\}. \tag{22}$$

Such PRGs can be built when q is superpolynomial by combining the techniques in [6] and [3].

5.2 Construction of Feasibly Correct Privately Programmable PRFs

In this section we construct a feasibly correct privately programmable PRF from our shiftable function of Section 4. We first define the auxiliary function that the construction will use. For $(x^*, \mathbf{w}) \in \{0,1\}^\ell \times \mathbb{Z}_q^m$ define the function $H_{(x^*,\mathbf{w})}\colon \{0,1\}^\ell \to \mathbb{Z}_q^m$ as

$$H_{(x^*,\mathbf{w})}(x) = \begin{cases} \mathbf{w} & \text{if } x = x^*, \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{23}$$

Notice that $H_{(x^*,\mathbf{w})}$ has circuit size upper bounded by some $\sigma' = \text{poly}(n, \log q)$ and depth at most $d' = O(\log q)$.

Construction 2. Our feasibly correct privately programmable PRF with input space $\mathcal{X} = \{0,1\}^\ell$ and output space $\mathcal{Y} = \mathbb{Z}_p^m$ where $p = \text{poly}(\lambda)$, uses the FC-SHSF from Section 4 with parameters $n = \tilde{O}(\lambda)$,$B = \lambda^{O(\log^2 \lambda)}$, $q = p \cdot B \cdot \lambda^{\omega(1)} = p \cdot \lambda^{O(\log^2 \lambda)}$, and is defined as follows:

- $\mathsf{Setup}(1^\lambda)$: First generate $pp' \leftarrow \mathsf{SHSF.Setup}(1^\lambda, 1^{\sigma'}, 1^{d'})$ then choose a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a uniformly random vector $\mathbf{r} \leftarrow \mathbb{Z}_p^m$, and output $pp := (pp', \mathbf{r}, \mathbf{A})$ we implicitly assume that pp contains the public parameters for a PRG with weak seeds which satisfies Equation (22).
- $\mathsf{KeyGen}(pp)$: Sample $\mathbf{s} \leftarrow \chi^n$, sample $msk' \leftarrow \mathsf{SHSF.KeyGen}(pp'; G(\mathbf{s}))$. Finally, output $msk := (\mathbf{s}, msk')$.
- $\mathsf{Eval}(pp, msk = (\mathbf{s}, msk'), x \in \{0,1\}^\ell)$: Compute $\mathbf{y}_x = \mathbf{s}\cdot\mathbf{A} + \mathsf{SHSF.Eval}(pp, msk', x)$ and output $\mathbf{r} + \lfloor \mathbf{y}_x \rceil_p$.
- $\mathsf{Program}(pp, msk, (x^*, \mathbf{y}^*))$: Given $(x^*, \mathbf{y}^*) \in \{0,1\}^\ell \times \mathbb{Z}_p^m$, compute $\mathbf{w}$ as follows: choose $\mathbf{y}' \leftarrow \mathbb{Z}_q^m$ uniformly at random conditioned on $\lfloor \mathbf{y}' \rceil_p = (\mathbf{y}^* - \mathbf{r})$, let $\mathbf{a} \leftarrow \mathbf{s}\cdot\mathbf{A} + \mathbf{e}$ where $\mathbf{e} \leftarrow \chi^m$, and set

$$\mathbf{w} = \mathbf{y}' - \mathsf{SHSF.Eval}(pp, msk', x^*) - \mathbf{a} \tag{24}$$

  Compute $sk'_{(x^*,y^*)} \leftarrow \mathsf{SHSF.Shift}(pp, msk', H_{(x^*,\mathbf{w})})$. Output $sk_{(x^*,y^*)} := (\mathbf{a}, sk'_{(x^*,y^*)})$.
- $\mathsf{PEval}(pp, sk_{(x^*,y^*)} = (\mathbf{a}, sk'_{(x^*,y^*)}), x)$: output $\mathbf{r} + \lfloor \mathbf{a} + \mathsf{SHSF.SEval}(pp, sk'_{(x^*,y^*)}, x) \rceil_p$.

The proof of the following theorem is deferred to the full version of this paper.

Theorem 2. *If* $\mathsf{LWE}_{n-1,q,\chi}$ *is hard and* PRF *is a pseudorandom function, Construction 2 is a key-injective, feasibly correct privately programmable PRF.*

5.3 Application

In the full version of this paper we show that by instantiating the watermarking scheme of [10] with Construction 2 PP-PRFs, we get a watermarking construction from LWE with quasi-polynomial modulus $q = \lambda^{O(\log^2 \lambda)}$. This watermarking scheme satisfies all the definitions in [27] and in particular it is authority pseudorandom.

6 Feasibly Correct Constraint-Hiding Constrained PRFs

Definition 13. *A constrained function is a* feasibly correct constraint-hiding constrained PRF *if it satisfies Definition 4, and there is a PPT simulator $\mathcal{S}$ such that, for any PPT adversary $\mathcal{A}$ that never queries its* Eval *oracle on an input x for which $C(x) =$ true (and without loss of generality never repeats an* Eval *query),*

$$\{\mathsf{CHCPRFReal}_{\mathcal{A}}(1^\lambda)\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \{\mathsf{CHCPRFIdeal}_{\mathcal{A},\mathcal{S}}(1^\lambda)\}_{\lambda\in\mathbb{N}}, \tag{25}$$

where $\mathsf{CHCPRFReal}$ *and* $\mathsf{CHCPRFIdeal}$ *are the respective views of $\mathcal{A}$ in the experiments defined in Figure 1.*

6.1 Construction

We now describe our construction of a feasibly correct CHC-PRF for domain $\mathcal{X} = \{0,1\}^\ell$ and range $\mathcal{Y} = \mathbb{Z}_p^m$, which handles constraining circuits of size σ and depth d. It uses the following components:

- A pseudorandom function $\mathsf{AuxPRF} = (\mathsf{AuxPRF.KG}, \mathsf{AuxPRF.Eval})$ having domain $\{0,1\}^\ell$ and range $\mathbb{Z}_q^m$, with key space $\{0,1\}^\kappa$.
- The feasibly correct shift hiding shiftable function
  $\mathsf{SHSF} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Shift}, \mathsf{SEval}, \mathsf{Sim})$ from Section 4, which has parameters q, B that appear in the analysis below.

For a boolean circuit C of size at most σ and depth upper-bound d and some $k \in \{0,1\}^\kappa$ define the function $H_{C,k} \colon \{0,1\}^\ell \to \mathbb{Z}_q^m$ as

$$\begin{aligned} H_{C,k}(x) &= C(x) \cdot \mathsf{AuxPRF.Eval}(k,x) \\ &= \begin{cases} \mathsf{AuxPRF.Eval}(k,x) & \text{if } U(C,x) = 1 \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{26}$$

Notice that the size of (the binary decomposition of) $H_{C,k}$ is upper bounded by

$$\sigma' = \sigma + s + \mathrm{poly}(n, \log q), \tag{27}$$

where s is the circuit size of (the binary decomposition of) $\mathsf{AuxPRF.Eval}(k,\cdot)$. And the depth of $H_{C,k}$ is upper-bounded by

$$d' = d + \delta + O(\log q), \tag{28}$$

where δ is the circuit depth of $\mathsf{AuxPRF.Eval}(k,\cdot)$. By instantiating AuxPRF with a log-depth PRF [6,5], we can set $n = \tilde{O}(\lambda)$, $B = \lambda^{O((d+\log\lambda)\log\lambda)}$, and $q = p \cdot B \cdot \lambda^{\omega(1)} = \lambda^{O((d+\log\lambda)\log\lambda)}$.

Construction 3. Our feasibly correct CHC-PRF with domain $\mathcal{X} = \{0,1\}^\ell$ and range $\mathcal{Y} = \mathbb{Z}_p^m$ is defined as follows:

– $\mathsf{Setup}(1^\lambda, 1^\sigma, 1^d)$: output $pp \leftarrow \mathsf{SHSF.Setup}(1^\lambda, 1^{\sigma'}, 1^{d'})$ where σ' and d' are defined as in Equations (27) and (28) respectively.
– $\mathsf{KeyGen}(pp)$: output $msk \leftarrow \mathsf{SHSF.KeyGen}(pp)$.
– $\mathsf{Eval}(pp, msk, x \in \{0,1\}^\ell)$: compute $\mathbf{y}_x = \mathsf{SHSF.Eval}(pp, msk, x)$ and output $\lfloor \mathbf{y}_x \rceil_p$.
– $\mathsf{Constrain}(pp, msk, C)$: on input a circuit C of size at most σ and depth at most d, sample a PRF key $k \leftarrow \mathsf{AuxPRF.KG}(1^\lambda)$ and output $sk_C \leftarrow \mathsf{SHSF.Shift}(pp, msk, H_{C,k})$.
– $\mathsf{CEval}(pp, sk_C, x)$: on input a constrained key sk_C and $x \in \{0,1\}^\ell$, output $\lfloor \mathsf{SHSF.SEval}(pp, sk_C, x) \rceil_p$.

6.2 Security Proof

In the full version of this paper we present the security proof of Construction 3.

Theorem 3. *Assuming that* $\mathsf{LWE}_{n-1,q,\chi}$ *is hard and* PRF *is a pseudorandom function, Construction 3 is a feasibly correct CHC-PRF.*

6.3 Applications

Watermarking in the alternative model of [33] from LWE with quasi-polynomial modulus Recently, [33] introduced an alternative model for watermarkable PRFs. In their model, the marking algorithm is *public* and roughly speaking, in their unremovability and correctness definition, the adversary also has access to an extraction oracle. Despite these advantages over the model considered in Section 5.3 , their model only guarantees pseudorandomness for adversaries that don't have the master secret key, i.e., they are not *authority pseudorandom.*

In their work, they present two constructions for their models. The first construction can be instantiated based solely on LWE with polynomial modulus but it does not support embedding messages in marked keys. The second construction does support embedding messages but needs private constrained PRFs for a circuit class consisting of an arbitrary PRF. As a consequence, instantiating the second construction with state of the art lattice-based CHC-PRFs needs LWE with subexponential modulus.

We observe that the message embedding scheme constructed in [33] can be instantiated with a feasibly correct CHC-PRF instead of a regular CHC-PRF. In more detail, in the security games of this scheme, the CHC-PRF is never evaluated on points that depend on the actual description of the constrained key. Indeed, in the correctness game, the challenge CHC-PRF constrained key is not given to the adversary and instead oracle access to Eval (and CEval) is provided. In the unremovability game, the oracles provided to the adversary evaluate the CHC-PRF using the challenge constrained key on points which don't depend on the constrained key, i.e., on points that can be sampled before any constrained key is generated. In all these scenarios feasible correctness would suffice to establish the security requirements.

Now, if we use the feasibly correct CHC-PRF constructed in Section 6.1, the underlying LWE assumption would have modulus size $\lambda^{O(d \log \lambda)}$ where d is the depth of the constraint circuit. If we use the log-depth LWE-based PRFs with modulus size $O(\lambda^{\log \lambda})$ constructed in [6,5] as our constraint circuits then, the message embedding scheme in [33] can be instantiated based solely on LWE with a quasi-polynomial modulus $q = O(\lambda^{\log^2 \lambda})$.

Symmetric Deniable Encryption Boneh, Lewi and Wu [10] showed that privately puncturable PRFs can be used to build a relaxed notion of symmetric deniable encryption. In more detail, their relaxed definition says that given a ciphertext encrypting an arbitrary plaintext it is possible to produce a fake secret key which decrypts the ciphertext to a random message but doesn't change the decryption of the rest of the ciphertexts. Using the current privately punctured PRFs [9,18,15,32], the deniable encryption scheme in [10] would have security based on the hardness of LWE with subexponential modulus. We observe that the construction in [10] evaluates CHC-PRFs only on random points. Consequently, in this application, we can replace CHC-PRFs with our feasibly correct CHC-PRFs. In particular, instantiating the [10] symmetric deniable encryption constriction with Construction 3 would give us a deniable symmetric encryption scheme based on LWE with quasi-polynomial modulus $q = \lambda^{O(\log^2 \lambda)}$.

7 Feasibly Correct C-PRFs for $\mathbf{NC}^1$ from LWE with Nearly Polynomial Modulus

7.1 Definitions

First, we recall the definition of C-PRFs and then we define feasibly correct C-PRFs. Compared to the constraint-hiding variant discussed in Section 6, feasibly correct C-PRFs are weaker in the sense that they do not necessarily hide the constraint.

Definition 14. *A constrained function is a* constrained PRF *(C-PRF) if there is a PPT simulator $\mathcal{S}$ such that, for any PPT adversary $\mathcal{A}$ (that without loss of generality never repeats an* Eval *query),*

$$\{\mathsf{CPRFReal}_{\mathcal{A}}(1^\lambda)\}_{\lambda \in \mathbb{N}} \overset{c}{\approx} \{\mathsf{CPRFIdeal}_{\mathcal{A},\mathcal{S}}(1^\lambda)\}_{\lambda \in \mathbb{N}}, \tag{29}$$

where $\mathsf{CPRFReal}$ *and* $\mathsf{CPRFIdeal}$ *are the respective views of $\mathcal{A}$ in the experiments defined in Figure 5.*

```
procedure CPRFReal_A(1^λ)
    $C \leftarrow \mathcal{A}(1^\lambda)$
    $pp \leftarrow \mathsf{Setup}(1^\lambda)$
    $msk \leftarrow \mathsf{KeyGen}(pp)$
    $sk_C \leftarrow \mathsf{Constrain}(pp, msk, C)$
    $(pp, sk_C) \rightarrow \mathcal{A}$
    repeat
        $x \leftarrow \mathcal{A}$
        $\mathsf{Eval}(pp, msk, x) \rightarrow \mathcal{A}$
    until $\mathcal{A}$ halts
```

(a) The real experiment

```
procedure CPRFIdeal_{A,S}(1^λ)
    $C \leftarrow \mathcal{A}(1^\lambda)$
    $(pp, sk_C) \leftarrow \mathcal{S}(1^\lambda, C)$
    $(pp, sk_C) \rightarrow \mathcal{A}$
    repeat
        $x \leftarrow \mathcal{A}$
        if $C(x) = \text{true}$ then
            $\mathsf{CEval}(pp, sk_C, x) \rightarrow \mathcal{A}$
        else
            $y \leftarrow \mathcal{Y}$; $y \rightarrow \mathcal{A}$
    until $\mathcal{A}$ halts
```

(b) The ideal experiment

Fig. 5: The real and ideal constrained PRF experiments.

Definition 15. *A constrained function is a* feasibly correct constrained PRF *if it satisfies Definition 4, and there is a PPT simulator $\mathcal{S}$ such that, for any PPT adversary $\mathcal{A}$ that never queries its* Eval *oracle on an input x for which $C(x) = true$ (and without loss of generality never repeats an* Eval *query),*

$$\{\mathsf{CPRFReal}_{\mathcal{A}}(1^\lambda)\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \{\mathsf{CPRFIdeal}_{\mathcal{A},\mathcal{S}}(1^\lambda)\}_{\lambda\in\mathbb{N}}, \tag{30}$$

where $\mathsf{CPRFReal}$ *and* $\mathsf{CPRFIdeal}$ *are the respective views of $\mathcal{A}$ in the experiments defined in Figure 5.*

7.2 Notation

Let $U(H, x) = H(x)$ denote a depth-universal circuit. We define $\mathsf{BP}_{U,x}$ to be the width 5 permutation branching program that on input a boolean circuit $H : \{0,1\}^\ell \rightarrow \{0,1\}$ of depth d and size σ, computes $U(H, x) = H(x)$. Observe that $\mathsf{BP}_{U,x}$ has length $O(4^d)$. Let χ' be a gaussian distribution with a nearly polynomial radius $n^{\omega(1)}$.

Gadgets and homomorphisms. Here we recall "gadgets" [29] over $\mathbb{Z}_q$ and several of their homomorphic properties, some of which were implicit in [22], and which were developed and exploited further
in [8,24,23,16,25]. For an integer modulus q, the gadget (or powers-of-two) vector over $\mathbb{Z}_q$ is defined as

$$\mathbf{g} = (1, 2, 4, \ldots, 2^{\lceil \lg q \rceil - 1}) \in \mathbb{Z}_q^{\lceil \lg q \rceil}. \tag{31}$$

The gadget matrix is defined as $\mathbf{G}_n = \mathbf{I}_n \otimes \mathbf{g} \in \mathbb{Z}_q^{n\times m}$, where $m = n\lceil \lg q \rceil$. There is an efficiently computable function $\mathbf{G}_n^{-1} : \mathbb{Z}_q^{n\times m} \rightarrow \{0,1\}^{m\times m}$ with the following property:

$$\forall \mathbf{A} \in \mathbb{Z}_q^{n\times m} : \mathbf{G}_n \cdot \mathbf{G}_n^{-1}(\mathbf{A}) = \mathbf{A}. \tag{32}$$

We often drop the subscript n when it is clear from context. We use algorithm $\mathsf{BranchEval}$ which has the following properties.

– $\mathsf{BranchEval}(\mathsf{BP}, x, \mathbf{A})$, given a width 5 permutation branching program $\mathsf{BP}\colon \{0,1\}^\ell \to \{0,1\}$ of length L, an $x \in \{0,1\}^\ell$, and some $\mathbf{A} \in \mathbb{Z}_q^{n\times(\ell+1)m}$, outputs an integral matrix $\mathbf{R}_{\mathsf{BP},x} \in \mathbb{Z}^{(\ell+1)m\times m}$ with poly(m)-bounded entries for which

$$(\mathbf{A} + (1,x) \otimes \mathbf{G}) \cdot \mathbf{R}_{\mathsf{BP},x} = \mathbf{A}_{\mathsf{BP}} + \mathsf{BP}(x) \cdot \mathbf{G}, \tag{33}$$

where $\mathbf{A}_{\mathsf{BP}} \in \mathbb{Z}_q^{n\times m}$ depends only on $\mathbf{A}$ and BP (and not on x).

Theorem 4 (Adapted From [5,15]). *Assuming the hardness of* $\mathsf{LWE}_{n,q,\chi}$*, there is a pair of polynomial time algorithms* $\mathsf{BPPRF} = (\mathsf{Setup}, \mathsf{Eval})$ *with the following interface*

– $\mathsf{Setup}(1^\lambda)$*: outputs public parameters pp.*
– $\mathsf{Eval}(pp, x)$*: on input $x \in \{0,1\}^\ell$ outputs a matrix $M_x \in \{0,1\}^{m\times m}$. This algorithm is deterministic.*

having the following property: the randomized functionality defined below,

– $\mathsf{P}(pp, \mathbf{s}, x)$*: P is a randomized functionality that on input pp in the range of* Setup*, $\mathbf{s} \in \mathbb{Z}_q^m$ and $x \in \{0,1\}^\ell$, first samples $\mathbf{e} \leftarrow (\chi')^m$ and then outputs* $\mathbf{s} \cdot \mathsf{Eval}(pp, x) + \mathbf{e} \in \mathbb{Z}_q^m$.

is pseudorandom. In other words, for any PPT adversary $\mathcal{A}$ we have

$$\Big|\Pr_{\substack{pp\leftarrow\mathsf{Setup}(1^\lambda)\\ \mathbf{s}\leftarrow\mathbb{Z}_q^n}}[\mathcal{A}^{P(pp,\mathbf{s},\cdot)}(pp) = 1] - \Pr_{pp\leftarrow\mathsf{Setup}(1^\lambda)}[\mathcal{A}^{U(\cdot)}(pp) = 1]\Big| = \mathrm{negl}(\lambda), \tag{34}$$

where $U\colon \{0,1\}^\ell \to \mathbb{Z}_q^m$ is a uniformly random function.

7.3 Construction

For security parameter λ, circuit depth d, the following construction is parametrized by some $n = \tilde{O}(\lambda)$ and $q = p \cdot B \cdot n^{\omega(1)}$ where $B = n^{\omega(1)}$ is an upper bound on the absolute value of the samples drawn from χ', and $m = n\lceil \log q\rceil = \mathrm{poly}(n)$. Setting $p = \mathrm{poly}(n)$ or even $p = n^{\omega(1)}$ makes $q = n^{\omega(1)} = \lambda^{\omega(1)}$. Let $\mathsf{PRF} = \{\mathsf{KG}, \mathsf{Eval}\}$ be a PRF with input domain $\mathbb{Z}_q^n$ and output range $\mathbb{Z}_q^m$.

Construction 4. Let $\mathcal{X} = \{0,1\}^\ell$ and $\mathcal{Y} = \mathbb{Z}_q^m$. Define:

– $\mathsf{Setup}(1^\lambda, 1^\sigma, 1^d)$: First sample public parameters $pp' \leftarrow \mathsf{BPPRF.Setup}(1^\lambda)$ for BPPRF. Next, sample uniformly random and independent matrices $\mathbf{A}_0 \in \mathbb{Z}_q^{n\times m}$, $\mathbf{A} \in \mathbb{Z}_q^{n\times(\sigma+1)m}$. Finally, output $pp = (pp', \mathbf{A}_0, \mathbf{A})$.
(The n-by-m chunks of $\mathbf{A}$ *will correspond to the σ bits of a circuit.)*
– $\mathsf{KeyGen}(pp)$: Generate a PRF key $\kappa \leftarrow \mathsf{PRF.KG}(1^\lambda)$. Sample $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and set and output the master secret key $msk = (\kappa, \mathbf{s})$.

– $\mathsf{Eval}(pp, msk, x \in \{0,1\}^\ell)$: compute

$$\mathbf{R}_0 = \mathsf{BranchEval}(\mathsf{BP}_{U,x}, 0^{\sigma+1}, \mathbf{A}) \in \mathbb{Z}^{(\sigma+1)m \times m} \tag{35}$$

and let

$$\mathbf{A}_x = (\mathbf{A} + (1, 0^{\overline{z}}) \otimes \mathbf{G}) \cdot \mathbf{R}_0 - \mathsf{BP}_{U,x}(0^{\sigma+1}) \cdot \mathbf{G} \in \mathbb{Z}_q^{n \times m}. \tag{36}$$

(Observe that by Equation (33), $\mathbf{A}_x = \mathbf{A}_{\mathsf{BP}}$ *for the branching program* $\mathsf{BP} = \mathsf{BP}_{U,x}$, *and does not depend on the "dummy" ciphertext* $0^{\sigma+1}$.*)*
Next, let

$$\mathbf{B}_x = \mathbf{A}_x \cdot \mathbf{G}^{-1}(\mathbf{A}_0 \cdot \mathsf{BPPRF.Eval}(pp', x)). \tag{37}$$

Finally, output

$$\lfloor \mathsf{PRF.Eval}(\kappa, x) + \mathbf{s} \cdot \mathbf{B}_x \rceil_p \tag{38}$$

– $\mathsf{Constrain}(pp, msk, C)$: for a circuit $C \colon \{0,1\}^\ell \to \{0,1\}$ of depth d and size σ, let

$$\mathbf{a} = \mathbf{s}(\mathbf{A} + (1, C) \otimes \mathbf{G}) + \mathbf{e} \tag{39}$$

where $\mathbf{e}$ is an error vector whose entries are sampled independently from χ. Output

$$sk_C = (\kappa, \mathbf{a}, C). \tag{40}$$

– $\mathsf{CEval}(pp, sk_C, x)$: On input $sk_C = (\kappa, \mathbf{a}, C)$ and $x \in \{0,1\}^\ell$, compute

$$\mathbf{R}_x = \mathsf{BranchEval}(\mathsf{BP}_{U,x}, C, \mathbf{A}) \tag{41}$$

$$\mathbf{a}_x = \mathbf{a} \cdot \mathbf{R}_x. \tag{42}$$

(By Equation (33), *we have* $\mathbf{a}_x \approx \mathbf{s}(\mathbf{A}_x + \mathsf{BP}_{U,x}(C) \cdot \mathbf{G})$, *where we recall that* $\mathsf{BP}_{U,x}(C) = C(x)$.*)*
Next, compute

$$\mathbf{b}_x = \mathbf{a}_x \cdot \mathbf{G}^{-1}(\mathbf{A}_0 \cdot \mathsf{BPPRF.Eval}(pp', x)) \tag{43}$$

Finally, output

$$\lfloor \mathsf{PRF.Eval}(\kappa, x) + \mathbf{b}_x \rceil_p. \tag{44}$$

We defer the proof of the following theorem to the full version of this paper.

Theorem 5. *If* $\mathsf{LWE}_{n,q,\chi}$ *is hard and* PRF *is a pseudorandom function, Construction 4 is a feasibly correct constrained PRF.*

7.4 Application

Updatable garbled circuits from LWE with superpolynomial modulus. Ananth, Cohen and Jain [2] used C-PRFs to build a cryptographic scheme that they call "updatable garbled circuits (UGC)". Specifically, they showed that a C-PRF which

– has range $\mathbb{Z}_p$ for a superpolynomially large p,
– supports point-function predicates as constraints,
– is *noisy constrained key-homomorphic*, i.e., for any two keys msk_1 and msk_2 and their respective constrained keys sk_{C_1} and sk_{C_2} and for any input x such that $C_1(x) = C_2(x) = 0$, $\mathsf{CEval}(sk_{C_1}, x_1) + \mathsf{CEval}(sk_{C_2}, x_2) = \mathsf{Eval}(msk_1 + msk_2, x) + e$ where e is bounded by some small constant,
– and has a KeyGen algorithm which simply outputs a random key from the key space,

can be used as a building block for UGCs. For their construction, they used the [17] C-PRF which satisfies all of the aforementioned properties. Since the C-PRF construction in [17] needs LWE with subexponential modulus, the security of [17] UGC construction also relies on hardness of LWE with subexponenial modulus.

We argue that in the UGC construction of [2], we can replace the [17] C-PRF with Construction 4. For this, we first notice that if we instantiate Construction 4 using [5] PRFs as the PRF (that is added before rounding) then, the resulting scheme is a noisy constrained key-homomorphic PRF. Furthermore, this instantiation has a KeyGen algorithm which samples a uniform key from its key space. Additionally, we notice that point-function predicates are in NC^1.

The last and most crucial observation is that, the UGC in [2] evaluates the C-PRF only on uniformly random inputs, both in the construction and the security definitions and games. Therefore, feasible correctness of the underlying C-PRF is enough for this UGC construction. So, we can instantiate the UGC using Construction 4. This will result in a UGC construction whose security is based on hardness of LWE with just nearly polynomial modulus $q = \lambda^{\omega(1)}$.

8 Constraint-Hiding PRFs for Hyperplane-Membership Predicates

In this section we construct constraint-hiding constrained PRFs for hyperplane-membership predicates, based solely on (approximate) key-homomorphic PRFs.

8.1 Construction

Construction 5. Let $\mathsf{KHPRF} = (\mathsf{Setup}, \mathsf{Eval})$ be a noisy key-homomorphic pseudorandom function having key space $\mathcal{K}$ (which is a finite group, with keys chosen uniformly at random), domain $\mathcal{X} = \{-D, \dots, D\}^\ell$ for some D, range $\mathcal{Y} = \mathbb{Z}_q^m$, and homomorphism error bound E.

Our constrained PRF has domain $\mathcal{X}$ and can be constrained to membership predicates for hyperplanes

$$H_{\boldsymbol{\alpha}} = \{x \in \mathcal{X} : \alpha_0 + \alpha_1 x_1 + \cdots + a_\ell x_\ell = 0\},$$

where $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \dots, \alpha_\ell) \in \{-A, \dots, A\}^{\ell+1}$. Define $B = (\ell+1)(AD+1)E$ and let $p = q/(B \cdot \lambda^{\omega(1)})$ be a divisor of q.

– $\mathsf{Setup}(1^\lambda)$: output $pp \leftarrow \mathsf{KHPRF.Setup}(1^\lambda)$.
– $\mathsf{KeyGen}(pp)$: sample KHPRF keys $k_i \leftarrow \mathcal{K}$ for $i = 0, \dots, \ell$ and output $msk := \{k_i\}_{i=0,\dots,\ell}$.
– $\mathsf{Eval}(pp, msk, x)$: on input $msk = \{k_i\}$ and $x \in \mathcal{X} = \{-D, \dots, D\}^\ell$, output $\lfloor y_x \rceil_p$ where

$$y_x = \mathsf{KHPRF.Eval}(pp, k_0, x) + \sum_{i \in [\ell]} \mathsf{KHPRF.Eval}(pp, x_i k_i, x). \tag{45}$$

– $\mathsf{Constrain}(pp, msk, H_{\boldsymbol{\alpha}})$: on input $msk = \{k_i\}$ and hyperplane $H_{\boldsymbol{\alpha}}$ where $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \dots, \alpha_\ell) \in \{-A, \dots, A\}^{\ell+1}$, first choose a KHPRF key $d \leftarrow \mathcal{K}$ and then for each $i = 0, \dots, \ell{+}1$ define $b_i := k_i - \alpha_i d$. Output $sk_{H_{\boldsymbol{\alpha}}} := \{b_i\}_{i=0,\dots,\ell}$.
– $\mathsf{CEval}(pp, sk_H, x)$: on input $sk_H = \{b_i\}$ and $x \in \mathcal{X}$, output $\lfloor y_x \rceil_p$ where

$$y_x = \mathsf{KHPRF.Eval}(pp, b_0, x) + \sum_{i \in [\ell]} \mathsf{KHPRF.Eval}(pp, x_i b_i, x). \tag{46}$$

8.2 Security Proof

Theorem 6. *If* KHPRF *is a noisy key-homomorphic pseudorandom function, and the smallest prime divisor of q is bigger than $(\ell+1)AD$, then Construction 5 is a feasibly correct and simulation-secure CHC-PRF for hyperplane-membership constraints.*

Proof. This follows from Theorem 7 and Theorem 8.

Theorem 7. *If* KHPRF *is a noisy key-homomorphic pseudorandom function, then Construction 5 is feasibly correct.*

Proof. Let $msk = \{k_i\}$ be a master secret key, let $sk_{H_{\boldsymbol{\alpha}}} = \{b_i = k_i - \alpha_i d\}$ be a constrained secret key for a hyperplane $H_{\boldsymbol{\alpha}}$, and let $x \in \mathcal{X}$ be an arbitrary input. Let y_x, y'_x be the "unrounded" values computed in Equations (45) and (46), respectively. If $x \in H_{\boldsymbol{\alpha}}$, i.e., $\alpha_0 + \sum_{i \in [\ell]} \alpha_i x_i = 0$, then

$$y'_x = \mathsf{KHPRF.Eval}(pp, b_0, x) + \sum_{i \in \ell} \mathsf{KHPRF.Eval}(pp, x_i b_i, x) \tag{47}$$

$$= \mathsf{KHPRF.Eval}(pp, k_0, x) + \sum_{i \in [\ell]} \mathsf{KHPRF.Eval}(pp, x_i k_i, x) \tag{48}$$

$$- (\alpha_0 + \sum_{i \in [\ell]} \alpha_i x_i)\mathsf{KHPRF.Eval}(pp, d, x) + \mathbf{e} \tag{49}$$

$$= \mathsf{KHPRF.Eval}(pp, k_0, x) + \sum_{i \in [\ell]} \mathsf{KHPRF.Eval}(pp, x_i k_i, x) + \mathbf{e} \tag{50}$$

$$= y_x + \mathbf{e}, \tag{51}$$

where the second equality is by key homomorphism, and $\mathbf{e}$ is an error vector whose entries have magnitudes at most B. Therefore, $\lfloor y'_x \rceil_p = \lfloor y_x \rceil_p$ (i.e., Eval

and CEval agree on x) unless some entry of y_x is in $\frac{q}{p}(\mathbb{Z} + \frac{1}{2}) + [-B, B]$. We next show that the probability of this event, for x output by any PPT algorithm $\mathcal{A}^{\mathsf{Eval}(pp,msk,\cdot)}$, is negligible (over the choice of msk and $\mathcal{A}$'s randomness). This follows straightforwardly from the pseudorandomness of KHPRF, and specifically the $\mathsf{KHPRF.Eval}(pp, k_0, x)$ term from Equation (45).

Formally, we construct an adversary $\mathcal{A}'$ against the pseudorandomness of KHPRF, which has access to an oracle $\mathcal{O}$ and runs as follows:

- given public parameters pp, choose $k_1, \ldots, k_\ell \leftarrow \mathcal{K}$;
- whenever $\mathcal{A}$ makes a query $\bar{x}$, respond with $\bar{y} = \mathcal{O}(x) + \sum_{i\in[\ell]} \mathsf{KHPRF.Eval}(pp, \bar{x}_i k_i, \bar{x})$
- when $\mathcal{A}$ finally outputs an x, accept if any of the entries of $y = \mathcal{O}(x) + \sum_{i\in[\ell]} \mathsf{KHPRF.Eval}(pp, x_i k_i, x)$ belong to $\frac{q}{p}(\mathbb{Z}+\frac{1}{2})+[-B, B]$, otherwise reject.

Clearly, if $\mathcal{O}$ is a uniformly random function, then all of the $\bar{y}$ and y are uniformly random, so by a union bound (which is needed because x might be one of the previously queried $\bar{x}$) $\mathcal{A}'$ accepts with probability at most $2Bpm \cdot \mathrm{poly}(\lambda)/q = \mathrm{negl}(\lambda)$. On the other hand, if $\mathcal{O}$ is $\mathsf{KHPRF.Eval}(\mathsf{pp}, \mathsf{k}_0, \cdot)$ (for some $k_0 \leftarrow \mathcal{K}$) then $\mathcal{A}'$ perfectly simulates the feasible correctness experiment, so the probability that $\mathcal{A}$ wins the feasible correctness game is at most the probability that $\mathcal{A}'$ accepts. By the pseudorandomness of KHPRF, the latter is negligible, as desired.

Theorem 8. *Under the hypotheses of Theorem 6, Construction 5 is simulation secure for the class of hyperplane-membership constraints.*

Proof. We need to build a simulator for adversaries that only submit unauthorized queries. The simulator $\mathcal{S}(1^\lambda, 1^\ell)$ for Construction 5, samples KHPRF keys $b_i \leftarrow \mathcal{K}$ for $i = 0, \ldots, \ell$ and outputs $\{b_i\}_{i=0,\ldots,\ell}$. Now let $\mathcal{A}$ be any polynomial-time adversary. To show that $\mathcal{S}$ satisfies Definition 3 we define a sequence of hybrid experiments and show that they are indistinguishable. Before defining the experiments in detail, we first define a particular "bad" event in all but one of them.

Definition 16. *In each of the following hybrid experiments except H_0, each query x is answered as $\lfloor y_x \rceil_p$ for some y_x that is computed in a certain way. Define* $\mathsf{Borderline}$ *to be the event that at least one such y_x has some coordinate in* $\frac{q}{p}(\mathbb{Z} + \frac{1}{2}) + [-B, B]$.

Hybrid H_0: This is the ideal experiment $\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}$.

Hybrid H_1: This is the same as H_0, except that on every (unauthorized) query x (i.e., where $\alpha_0 + \sum_{i\in[\ell]} \alpha_i x_i \neq 0$), instead of returning a uniformly random value from $\mathbb{Z}_p^m$, we choose $y_x \leftarrow \mathbb{Z}_q^m$ and output $\lfloor y_x \rceil_p$.

Hybrid H_2: This is the same as H_1, except that we abort the experiment if $\mathsf{Borderline}$ happens.

Hybrid H_3: This is the same as H_2, except that we initially choose a KHPRF key $\mathbf{d} \leftarrow \mathcal{K}$ and change how (unauthorized) queries x are handled. Specifically, for any query x we answer $\lfloor y_x \rceil_p$ where

$$\begin{aligned} y_x = \mathsf{KHPRF.Eval}(pp, b_0, x) + \sum_{i\in[\ell]} \mathsf{KHPRF.Eval}(pp, x_i b_i, x) \\ + (\alpha_0 + \sum_{i\in[\ell]} \alpha_i x_i)\mathsf{KHPRF.Eval}(pp, \mathbf{d}, x). \end{aligned} \tag{52}$$

Hybrid H_4: This is the same as H_3, except that (pp, sk) are generated as in the real experiment. Specifically, we sample $msk := \{k_i \leftarrow \mathcal{K}\}_{0\le i\le \ell}$, $\mathbf{d} \leftarrow \mathcal{K}$, and set $b_i := k_i - \alpha_i \mathbf{d}$ for $0 \le i \le \ell$.

Hybrid H_5: This is the same as H_4, except that we answer all (unauthorized) evaluation queries as in the Eval algorithm, i.e., we output $\lfloor y_x \rceil_p$ where

$$y_x = \mathsf{KHPRF.Eval}(pp, k_0, x) + \sum_{i\in[\ell]} \mathsf{KHPRF.Eval}(pp, x_i k_i, x). \tag{53}$$

Hybrid H_6: This is the same as H_5, except that we no longer abort when Borderline happens. Observe that this is exactly the real experiment $\mathsf{Real}_{\mathcal{A}}$.

We now prove that adjacent pairs of hybrid experiments are indistinguishable.

Claim. Experiments H_0 and H_1 are identical.

Proof. This follows immediately from the fact that p divides q.

Claim. Experiments H_1 and H_2 are statistically indistinguishable, in particular in H_1 the event Borderline happens with negligible probability.

Proof. This immediately follows by the fact that y_x is chosen uniformly at random from $\mathbb{Z}_q^m$ and $\frac{q}{pB} = n^{\omega(1)}$.

Claim. Assuming KHPRF is a noisy key-homomorphic PRF and the prime divisors of q are bigger than $(\ell+1)AD$, $H_2 \overset{c}{\approx} H_3$.

Proof. We need to show that in H_3, y_x is indistinguishable from uniform. Since $\mathsf{KHPRF.Eval}(pp, \mathbf{d}, x)$ is pseudorandom, all we need to prove is that $\alpha_0 + \sum_{i\in[\ell]} \alpha_i x_i$ is invertible in $\mathbb{Z}_q$. To see this recall that all queries are unauthorized and therefore $\alpha_0 + \sum_{i=1}^{\ell} \alpha_i x_i \neq 0$. On the other hand $|\alpha_0 + \sum_{i\in[\ell]} \alpha_i x_i| < (\ell+1)AD$. This proves that $\alpha_0 + \sum_{i\in[\ell]} \alpha_i x_i$ is invertible in $\mathbb{Z}_q$.

Claim. Experiments H_3 and H_4 are identical.

Proof. This follows from the fact that in both H_3 and H_4, the b_is and $\mathbf{d}$ have the same distribution, i.e., they are uniform elements in $\mathcal{K}$.

Claim. Assuming KHPRF is a noisy key-homomorphic PRF, H_4 and H_5 are identical.

Proof. We need to show that all (unauthorized) queries are answered identically in H_4 and H_5. Let y_x and y'_x be the unrounded answers to a query x in hybrids H_4 and H_5 respectively. We need to show $\lfloor y_x \rceil_p = \lfloor y'_x \rceil_p$. Since KHPRF is a noisy key-homomorphic PRF we have

$$\begin{aligned} y_x &= \mathsf{KHPRF.Eval}(pp, b_0, x) + \sum_{i \in [\ell]} \mathsf{KHPRF.Eval}(pp, x_i b_i, x) \\ &\quad + (\alpha_0 + \sum_{i \in [\ell]} \alpha_i x_i) \mathsf{KHPRF.Eval}(pp, \mathbf{d}, x) && (54) \\ &= \mathsf{KHPRF.Eval}(pp, k_0 - \alpha_0 \mathbf{d}, x) + \sum_{i \in [\ell]} \mathsf{KHPRF.Eval}(pp, x_i k_i - \alpha_i x_i \mathbf{d}, x) && (55) \\ &\quad + (\alpha_0 + \sum_{i \in [\ell]} \alpha_i x_i) \mathsf{KHPRF.Eval}(pp, \mathbf{d}, x) && (56) \\ &= \mathsf{KHPRF.Eval}(pp, k_0, x) + \sum_{i \in [\ell]} \mathsf{KHPRF.Eval}(pp, x_i k_i, x) + \mathbf{e} && (57) \\ &= y'_x + \mathbf{e}, && (58) \end{aligned}$$

where $\mathbf{e}$ is a vector with entries not bigger than $(\ell+1)(AD+1)E \leq B$. Since we abort when $\mathsf{Borderline}$ happens, the claim follows.

Claim. Assuming KHPRF is a noisy key-homomorphic PRF and the prime divisors of q are bigger than $(\ell+1)AD$, $H_5 \overset{s}{\approx} H_6$.

Proof. By previous claims, $\mathsf{Borderline}$ happens with negligible probability in H_5.

This completes the proof of Theorem 8.

9 Bit Fixing PRFs from Minimal Assumptions

The class of *bit-fixing* constraints for input space $\mathcal{X} = \{0,1\}^\ell$ is the set of constraints $\mathcal{C} = \{C_v : v \in \{0,1,\star\}^\ell\}$ where $C_v(x) = \text{true}$ if and only if $x_i = v_i$ for all $i \in [\ell]$ such that $v_i \neq \star$. In other words, x must match v at every position, where $\star$ is a "wildcard" that both 0 and 1 match.

Here we construct a constraint-hiding, bit-fixing PRF (for a single constrained-key query) from the minimal assumption that PRFs exist. Let
$\mathsf{PRF} = (\mathsf{PRF.KG}, \mathsf{PRF.Eval})$ be a pseudorandom function having key space $\mathcal{K}$, domain $\mathcal{X} = \{0,1\}^\ell$, and range $\mathcal{Y}$, which we assume to be a finite (additive) group.

Construction 6. Our bit-fixing PRF with domain $\mathcal{X} = \{0,1\}^\ell$ and range $\mathcal{Y}$ is defined as follows:

- $\mathsf{KeyGen}(1^\lambda)$: sample PRF keys $msk_i^b \leftarrow \mathsf{PRF.KG}(1^\lambda)$ for $i \in [\ell]$ and $b \in \{0,1\}$, and output $msk := \{msk_i^b\}$.
- $\mathsf{Eval}(1^\lambda, msk, x \in \{0,1\}^\ell)$: output $\sum_{i \in [\ell]} \mathsf{PRF.Eval}(msk_i^{x_i}, x)$.

– $\mathsf{Constrain}(1^\lambda, msk, v \in \{0,1,\star\})$: define and output $sk_v = \{sk_i^b\}_{i\in[\ell],b\in\{0,1\}}$, defined as follows:
  - if $v_i = b$ or $v_i = \star$ then let $sk_i^b := msk_i^b$,
  - otherwise (i.e., $v_i = 1-b$), let $sk_i^b \leftarrow \mathsf{PRF.KG}(1^\lambda)$ be a freshly sampled key for PRF.

– $\mathsf{CEval}(1^\lambda, sk_v = \{sk_i^b\}, x)$: output $\sum_{i\in[\ell]} \mathsf{PRF.Eval}(sk_i^{x_i}, x)$.

In words, the constrained key for pattern $v \in \{0,1,\star\}^\ell$ contains exactly those msk components msk_i^b for which b "matches" v_i, with fresh PRF keys taking the place of the other msk components. In particular, it is easy to see that this ensures correct constrained evaluation on authorized inputs. Also observe that the constrained key alone hides the pattern vector *perfectly*, i.e., the distribution of sk_v is the same for all v.

9.1 Security Proof

Theorem 9. *If* PRF *is a pseudorandom function, then Construction 6 is a constraint-hiding bit-fixing PRF according to Definition 3.*

Proof. The simulator $\mathcal{S}(1^\lambda)$ for Construction 6 simply samples PRF keys $r_i^b \leftarrow \mathsf{PRF.KG}(1^\lambda)$ for $i \in [\ell], b \in \{0,1\}$ and outputs $\{r_i^b\}$. Now let $\mathcal{A}$ be any polynomial-time adversary. To show that $\mathcal{S}$ satisfies Definition 3 we define a sequence of hybrid experiments and show that they are indistinguishable.

Hybrid H_0: This is the real experiment $\mathsf{Real}_\mathcal{A}$ (see Figure 1).
Hybrid H_1: This is the same as H_0 except that on every authorized query x (i.e., where x matches the pattern v output by $\mathcal{A}$) we answer it by $\mathsf{CEval}(pp, sk_C, x)$.
Hybrid H_2: This is the same as H_1 except that on every unauthorized query x (i.e., where x does not match the pattern v output by $\mathcal{A}$) we answer with a uniformly random element in $\mathcal{Y}$.
Hybrid H_3: This is the same as the ideal experiment $\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}$.

We now show that adjacent pairs of hybrid experiments are indistinguishable. First, it follows immediately that experiments H_0 and H_1 are identical, due to the way sk_C is constructed.

Claim. If PRF is a pseudorandom function, then $H_1 \overset{c}{\approx} H_2$.

Proof. Let $H_{1,0} = H_1$ and define hybrid $H_{1,i}$ for $i \in [\ell]$ as follows: it is the same as H_1 except that for every query x which does not match the pattern v *in one of the first i positions*, we answer with a uniformly random element in $\mathcal{Y}$.

Clearly $H_{1,\ell} = H_2$. We show that for every $i \in [\ell]$, $H_{1,i-1} \overset{c}{\approx} H_{1,i}$. Notice that if $v_i = \star$ then the two experiments are identical. So assume that $v_i = b \in \{0,1\}$. Let $\mathcal{A}$ be an adversary attempting to distinguish between $H_{1,i-1}$ and $H_{1,i}$. We build an efficient adversary $\mathcal{A}'$ against the security of PRF, which has access to an oracle $\mathcal{O}$ that is either a uniformly random function $U\colon \{0,1\}^\ell \to \mathcal{Y}$ or $\mathsf{PRF.Eval}(sk, \cdot)$ for $sk \leftarrow \mathsf{PRF.KG}(1^\lambda)$. $\mathcal{A}'$ interacts with $\mathcal{A}$ in the same way as

$H_{1,i-1}$, except that it does not sample msk_i^{1-b}, and on queries x for which i is the smallest index where x disagrees with the pattern v, it replies with $\sum_{j\in[\ell]\setminus\{i\}} \mathsf{PRF.Eval}(msk_j^{x_j}, x) + \mathcal{O}(x)$. Observe that if $\mathcal{O}$ is U then $\mathcal{A}$'s view is identical to $H_{1,i}$, otherwise the view is identical to $H_{1,i-1}$. Therefore, the advantage of $\mathcal{A}$ in distinguishing $H_{1,i-1}$ from $H_{1,i}$ is identical to the advantage of $\mathcal{A}'$ in attacking PRF, which by assumption is negligible, as desired.

Finally, we claim that experiments H_2 and H_3 are identical. This is because in both experiments the constrained key has the same distribution, i.e., it consists of 2ℓ independent keys for PRF.

Remark 1. We observe that any constraint-hiding bit-fixing PRF can be bootstrapped to support k-CNF formulas as constraints, where k is a constant. For input domain $\{0,1\}^\ell$, the construction is as follows:

- we generate parameters for a bit-fixing PRF with input length $(2\ell+1)^k$,
- to evaluate on an input $x \in \{0,1\}^\ell$, first, we convert x to $x' \in \{0,1\}^{(2\ell+1)^k}$ where the individual bits of x' are the result of evaluating all possible k-variable disjunctions involving the literals
  $F, x_1, \bar{x}_1, \cdots, x_\ell, \bar{x}_\ell$ in some specified order, then, we evaluate the bit-fixing PRF on x' and output the result,
- to generate a constrained-key for a k-CNF formula ϕ having $t \leq (2\ell+1)^k$ clauses, we generate a constrained key in the bit-fixing PRF for a pattern v where in v the positions corresponding to the clauses in ϕ are fixed to 1 and the rest of the positions are wildcard.

References

1. Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *CRYPTO*, pages 297–314, 2014.
2. Prabhanjan Ananth, Aloni Cohen, and Abhishek Jain. Cryptography with updates. In *EUROCRYPT*, pages 445–472, 2017.
3. Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618, 2009.
4. Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Constrained prfs for nc^1 in traditional groups. In *CRYPTO*, pages 543–574, 2018.
5. Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In *CRYPTO*, pages 353–370, 2014.
6. Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, pages 719–737, 2012.
7. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, 2012. Preliminary version in CRYPTO 2001.
8. 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*, pages 533–556, 2014.

9. Dan Boneh, Sam Kim, and Hart William Montgomery. Private puncturable PRFs from standard lattice assumptions. In *EUROCRYPT*, pages 415–445, 2017.
10. Dan Boneh, Kevin Lewi, and David J. Wu. Constraining pseudorandom functions privately. In *PKC*, pages 494–524, 2017.
11. Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT*, pages 280–300, 2013.
12. Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *PKC*, pages 501–519, 2014.
13. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.
14. Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, pages 575–584, 2013.
15. Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In *TCC*, pages 264–302, 2017.
16. Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In *ITCS*, pages 1–12, 2014.
17. Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In *TCC*, pages 1–30, 2015.
18. Ran Canetti and Yilei Chen. Constraint-hiding constrained PRFs for NC^1 from LWE. In *EUROCRYPT*, pages 446–476, 2017.
19. Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. In *STOC*, pages 1115–1127, 2016.
20. Stephen A. Cook and H. James Hoover. A depth-universal circuit. *SIAM J. Comput.*, 14(4):833–839, 1985.
21. Alex Davidson, Shuichi Katsumata, Ryo Nishimaki, and Shota Yamada. Constrained prfs for bit-fixing from owfs with constant collusion-resistance. Cryptology ePrint Archive, Report 2018/982, 2018. `https://eprint.iacr.org/2018/982`.
22. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, pages 75–92, 2013.
23. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In *CRYPTO*, pages 503–523, 2015.
24. Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In *STOC*, pages 469–477, 2015.
25. Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient ABE for branching programs. In *ASIACRYPT*, pages 550–574, 2015.
26. Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *CCS*, pages 669–684, 2013.
27. Sam Kim and David J. Wu. Watermarking cryptographic functionalities from standard lattice assumptions. In *CRYPTO*, pages 503–536, 2017.
28. Sam Kim and David J. Wu. Watermarking prfs from lattices: Stronger security via extractable PRFs. In *CRYPTO*, pages 335–366, 2019.
29. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2012.
30. Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342, 2009.
31. Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of Ring-LWE for any ring and modulus. In *STOC*, pages 461–473, 2017.

32. Chris Peikert and Sina Shiehian. Privately constraining and programming PRFs, the LWE way. In *PKC*, pages 675–701, 2018.
33. Willy Quach, Daniel Wichs, and Giorgos Zirdelis. Watermarking PRFs under standard assumptions: Public marking and security with extraction queries. In *TCC*, pages 669–698, 2018.
34. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):1–40, 2009. Preliminary version in STOC 2005.
35. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484, 2014.
36. Rotem Tsabary. Fully secure attribute-based encryption for t-cnf from LWE. In *CRYPTO*, pages 62–85, 2019.

A Shift Hiding Shiftable Functions and their Properties

Here we review the interface and relevant properties of *shift-hiding shiftable functions* introduced in [32]. For security parameter λ and constraint circuit size σ, shift-hiding shiftable functions are a tuple of algorithms parameterized by some $n = \mathrm{poly}(\lambda, \sigma)$ and $q = \lambda^{\mathrm{poly}(\lambda,\sigma)}$, with $m = n\lceil \lg q \rceil = \mathrm{poly}(\lambda, \sigma)$. These algorithms have the following interface.

- $\mathsf{Setup}(1^\lambda, 1^\sigma, 1^d)$: On input security parameter λ, shift circuit size σ, and shift circuit depth d, output public parameter pp.
- $\mathsf{KeyGen}(pp)$: Output the master secret key $msk = \mathbf{s}$.
- $\mathsf{Eval}(pp, msk, x)$: On input an ℓ bit string x, output $\mathbf{y} \in \mathbb{Z}_q^m$.
- $\mathsf{Shift}(pp, msk, H)$: On input a shift function $H \colon \{0,1\}^\ell \to \mathbb{Z}_q^m$, output a shifted key sk_H.
- $\mathsf{SEval}(pp, sk_H, x)$: On input a shifted key sk_H and input value $x \in \{0,1\}^\ell$, output $\mathbf{y} \in \mathbb{Z}_q^m$.
- $\mathcal{S}(1^\lambda, 1^\sigma, 1^d)$: On input security parameter λ, shift circuit size σ, and shift depth d, output simulated public paramters pp and simulated shifted key sk.

We use the following two properties of the shift-hiding shiftable functions construction of [32].

Property 1 (Shift Hiding). Assuming the hardness of $\mathrm{LWE}_{n-1,q,\chi}$ and CPA security of the GSW [22] encryption scheme, for any PPT $\mathcal{A}$, any $\sigma = \sigma(\lambda) = \mathrm{poly}(\lambda)$, and any $d = d(\lambda) = \mathrm{poly}(\lambda)$

$$\{\mathsf{RealKey}_{\mathcal{A}}(1^\lambda, 1^\sigma, 1^d)\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \{\mathsf{IdealKey}_{\mathcal{A}}(1^\lambda, 1^\sigma, 1^d)\}_{\lambda\in\mathbb{N}}, \tag{59}$$

where $\mathsf{RealKey}$ and $\mathsf{IdealKey}$ are the respective views of $\mathcal{A}$ in the experiments defined in Figure 2.

Property 2 (Approximate Shift Correctness). For any shift function $H \colon \{0,1\}^\ell \to \mathbb{Z}_q^m$ whose binary decomposition $H' \colon \{0,1\}^\ell \to \{0,1\}^k$ can be represented by a boolean circuit of size σ and depth d, and any $x \in \{0,1\}^\ell$, $pp \leftarrow \mathsf{Setup}(1^\lambda, 1^\sigma, 1^d)$, $msk \leftarrow \mathsf{KeyGen}(pp)$ and $sk_H \leftarrow \mathsf{Shift}(pp, msk, H)$, we have

$$\mathsf{SEval}(pp, sk_H, x) \approx \mathsf{Eval}(pp, msk, x) + H(x) \tag{60}$$

where the approximation hides some $\lambda^{O(d \log \lambda)}$-bounded error vector.