Constraint-Hiding Constrained PRFs for NC^1 from LWE[*]

Ran Canetti[†] Yilei Chen[‡]

December 30, 2019

Abstract

Constraint-hiding constrained PRFs (CHCPRFs), initially studied by Boneh, Lewi, and Wu [PKC 2017], are constrained PRFs where the constrained key hides the description of the constraint. Envisioned with powerful applications such as searchable encryption, private-detectable watermarking, and symmetric deniable encryption, the only known candidates of CHCPRFs are based on indistinguishability obfuscation or multilinear maps with strong security properties.

In this paper, we construct CHCPRFs for all NC^1 circuits from the Learning with Errors assumption. The construction draws heavily from the graph-induced multilinear maps by Gentry, Gorbunov, and Halevi [TCC 2015], as well as the existing lattice-based PRFs. Our construction gives an instance of the GGH15 applications with a security reduction from LWE.

We also show how to build from CHCPRFs reusable garbled circuits (RGC), or equivalently private-key function-hiding functional encryptions with 1-key security. This provides a different approach to constructing RGC from that of Goldwasser et al. [STOC 2013].

[*]Supported by the NSF MACS project, NSF grants CNS1012798, CNS1012910, and ISF grant 1523/14.

[†]Boston University, Tel Aviv University and the Check Point Institute for Information Security. `canetti@bu.edu` .

[‡]Boston University. `cheny1@bu.edu`. Part of the research conducted while visiting Tel Aviv University and IST Austria.

Contents

1 Introduction

Constrained PRFs [BW13, KPTZ13, BGI14] are pseudorandom functions with a special mode that outputs a constrained key defined by a predicate C. The constrained key CK_C preserves the functionality on every input x s.t. $C(x) = 1$, while randomizing the output value on every input x s.t. $C(x) = 0$. In the standard formulation of a constrained PRF, the constrained key is not required to hide the predicate C. In fact, many constructions of constrained PRFs do reveal the constraint. A quintessential example is GGM's puncturable PRF [GGM86] where CK explicitly reveals the punctured points.

The notion of a *constraint-hiding constrained PRF* (CHCPRF), also known as a private constrained PRF, is proposed by Boneh, Lewi and Wu [BLW17]. It additionally requires that the constraint predicate C remains hidden, even given the constrained key. Such a property allows the primitive to provide fairly natural constructions of searchable encryption, watermarking, deniable encryption, and others. However, they only propose candidates of CHCPRFs based on strong assumptions, like indistinguishability obfuscation (iO) [BGI$^+$12] or assumptions on candidate multilinear maps (multilinear-DDH or subgroup elimination) [BS03].

This work. We further investigate the notion of a CHCPRF, propose constructions based on standard cryptographic assumptions, and demonstrate more applications.

We first propose an alternative, simulation-based definition for CHCPRF. While for the cases addressed in our constructions the new style is (almost) equivalent to the indistinguishability-based one from [BLW17], the new formulation provides a different viewpoint on the primitive.

Our main result is a construction of CHCPRF for all NC^1 circuit constraints based on the Learning with Errors (LWE) assumption [Reg09]:

Theorem 1.1. *Assuming the intractability of LWE, there are CHCPRFs with 1-key simulation-based security, for all constraints recognizable by* NC^1 *circuits.*

The construction combines the graph-induced multilinear maps by Gentry, Gorbunov and Halevi [GGH15], their candidate obfuscator, and the lattice-based PRFs of [BPR12, BLMR13]. At the heart of our technical contribution is identifying a restricted (yet still powerful) variant of the GGH15 maps, whose security can be reduced to LWE. This involves formulating new "LWE-hard" secret distributions that handle the permutation matrices underlying Barrington's construction.

In addition, we construct function-hiding private-key functional encryptions (equivalently, reusable garbled circuits [GKP$^+$13]) from CHCPRFs. This gives a construction of reusable garbled circuits from LWE that is very different from that of [GKP$^+$13]:

Theorem 1.2. *For a circuit class $\mathcal{C}$, assuming 1-key simulation-based CHCPRFs for constraints in $\mathcal{C}$, and CPA secure private-key encryption whose decryption circuit is in $\mathcal{C}$, there exist 1-key secure reusable garbled circuits for $\mathcal{C}$.*

1.1 CHCPRFs, functional encryption and obfuscation

We propose a simulation-based definitional approach for CHCPRF, and compare this approach to the indistinguishability-based approach of Boneh et al. [BLW17].

Defining CHCPRFs. A constrained PRF consists of three algorithms: Master secret key generation, constrained key generation, and function evaluation. We first note that to have hope to hide the constraint, the function evaluation algorithm should return a random-looking value v even if evaluated on a constrained input x, as opposed to returning $\perp$ as in the standard formulation. Furthermore, we require that the value of the original function on x remains pseudorandom even given the constrained key and the value v.

The definition of CHCPRF is aimed at capturing three requirements: (1) the constrained keys preserve functionality on inputs that do not match the constraint; (2) the function values at constrained points remain pseudorandom given the constrained key; (3) the constrained key does not reveal any information on the constraining function.

Boneh et al. [BLW17] give a number of indistinguishability-based definitions that vary in strength, depending on the level of adaptivity of the adversary in choosing the constraints and evaluation points, as well as on the number of constrained keys that the adversary is allowed to see. We take an alternative approach and give a simulation-based definition. We also compare the definitions, and show equivalence and derivations in a number of cases.

Here is a sketch of the non-adaptive single-key variant of our simulation-based definition. The definition captures all three requirements via a single interaction: We require that, for any polytime adversary, there exists a polytime simulator such that the adversary can distinguish between the outcomes of the following two experiments only with negligible probability:

- In the real experiment, the system first generates a master secret key K. The adversary can then query a constraint circuit C and many inputs $x^{(1)}, ..., x^{(t)}$. In return, it obtains $\mathsf{CK}_C, x^{(1)}, ..., x^{(t)}, y^{(1)}, ..., y^{(t)}$, where CK_C is a key constrained by C, and $y^{(i)}$ is the result of evaluating the original, unconstrained function with key K at point $x^{(i)}$. (This is so regardless of whether $x^{(i)}$ meets the constraint or not.)

- In the ideal experiment, the simulator samples a master secret key K^S. Once received a constraint query, the simulator obtains only the description length of C and creates a simulated constrained key CK^S. Once received input queries $x^{(1)}, ..., x^{(t)}$, the simulator also obtains t indicator bits $d^{(1)}, ..., d^{(t)}$ where $d^{(i)} := C(x^{(i)})$, and generates simulated values $y^{(1)S}, ..., y^{(t)S}$. If $d^{(i)} = 0$, then the simulated $y^{(i)S}$ is uniformly random. The output of the experiment is $\mathsf{CK}^S, x^{(1)}, ..., x^{(t)}, y^{(1)S}, ..., y^{(t)S}$.

Secret-key functional encryption from simulation-based CHCPRFs. Functional encryption schemes [BSW11] allow the system to derive a function-specific decryption key f_{SK}. Given f_{SK}, the user can learn the value of the function f applied to encrypted data without learning anything else. From CHCPRF, it is rather simple to construct a private-key functional encryption scheme that is both function-private and input-private. We sketch our construction:

- Key generation: The master key for the scheme is a key K for a CHCPRF, and a key SK for a CPA-secure symmetric encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$.

- Encrypt a message m: $\mathsf{CT} = (c, t)$, where $c = \mathsf{Enc}_{\mathsf{SK}}(m)$, and $t = \mathsf{CHCPRF}_K(c)$.

- Functional decryption key: The functional decryption key for a binary function f is a constrained-key $\mathsf{CK}_{\hat{f}}$ for the function $\hat{f}(c) = f(\mathsf{Dec}_{\mathsf{SK}}(c))$. That is, $\hat{f}$ has SK hardwired; it decrypts its input c and applies f to the plaintext.

- Functional decryption: Given ciphertext $\mathsf{CT} = (c, t)$ and the constrained decryption key $\mathsf{CK}_{\hat{f}}$, output 1 if $t = \mathsf{CHCPRF}_{\mathsf{CK}_{\hat{f}}}(c)$, 0 otherwise.

Correctness of decryption follows from the functionality of the CHCPRF, and secrecy follows from the constraint-hiding property.

Our construction is completely different from the initial construction of Goldwasser et al. [GKP$^+$13] and the recent construction of Agrawal and Rosen [AR17]. In particular, the size of the ciphertext (for functions with 1-bit output) is the size of a symmetric encryption ciphertext $|\mathsf{Sym.CT}(m)|$ plus the security parameter, independent of the depth of the circuit. (For functions with τ-bit outputs, a straightforward batch mode achieves ciphertext size $|\mathsf{Sym.CT}(m)| + \tau\lambda$, less than simply making τ copies.)

While our scheme is still not compact enough to imply iO through the bootstrapping techniques from functional encryption [AJ15, BV15a, LPST16, BNPW16], it provides another starting point for future attempts.

Two-key CHCPRFs imply obfuscation. It is natural to consider an extension of the CHCPRF definition to the case where the adversary may obtain multiple constrained keys derived from the same master key. Indeed in [BLW17] some applications of this extended notion are presented.

We observe that this extended notion in fact implies full fledged program obfuscation. To obfuscate a circuit C, choose a master secret key K for a CHCPRF, and output two constrained keys: The constrained key $\mathsf{CK}[C]$, and the constrained key $\mathsf{CK}[I]$, where I is the circuit that always outputs 1. To evaluate $C(x)$ check whether $\mathsf{CHCPRF}_{\mathsf{CK}[C]}(x) = \mathsf{CHCPRF}_{\mathsf{CK}[I]}(x)$.

Again, correctness of evaluation follows from the functionality of the CHCPRF. The level of security for the obfuscation depends on the definition of CHCPRF in use. Specifically, the natural extension of the above simulation-based definition to the two-key setting implies that the obfuscator above satisfies the strong VBB definition considered in [Had00], which is impossible to achieve for a large class of functions. The indistinguishability-based definition of [BLW17] implies that the obfuscator above is iO.

1.2 Overview of our construction

Our construction of CHCPRFs draws heavily from the multilinear maps by Gentry, Gorbunov and Halevi [GGH15], and the lattice-based PRFs of Banerjee, Peikert and Rosen, and others [BPR12, BLMR13, BP14, BV15b, BFP$^+$15]. We thus start with a brief review of the relevant parts of these works.

Recap GGH15. The GGH15 multilinear encoding is depicted by a DAG that defines the rule of homomorphic operations and zero-testing. For our purpose it is sufficient to consider the following special functionality (which corresponds to a graph of ℓ nodes and two parallel edges from node i to node $i+1$, see Figure 1.1a): We would like to encode 2ℓ secrets $s_1^0, s_1^1, ..., s_\ell^0, s_\ell^1$ over some finite group $\mathbb{G}$, in such a way that an evaluator who receives the encodings can test, for any given $x, y \in \{0,1\}^\ell$, whether $\prod_{i=1}^{\ell} s_i^{x_i} = \prod_{i=1}^{\ell} s_i^{y_i}$; and at the same time the encodings hide "everything else" about the secrets. (Indeed, "everything else" might have different meanings in different contexts.)

To do that, GGH15 embeds the secrets in a ring R_q, where R denotes the base ring (typical choices include $R = \mathbb{Z}^{n\times n}$ or $R = \mathbb{Z}[x]/(\Phi_n(x))$, where n is a parameter related to the lattice dimension, and Φ_n is the n^{th} cyclotomic polynomial), and q is the modulus. The encoder samples $\ell + 1$ hard Ajtai-type matrices $\{\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_\ell, \mathbf{A}_{\ell+1} \leftarrow R_q^{1\times m}\}$ with trapdoors [Ajt99, AP11, MP12], and associates each matrix with the corresponding node of the graph. These matrices and their trapdoors are treated as (universal) public and secret parameters, respectively. We refer to the indices $1 \ldots \ell + 1$ as *levels*.

The 2ℓ secrets are associated with the 2ℓ edges of the graph in the natural way. Encoding a secret s_i^b is done in two steps: First create an LWE sample for the secret s_i^b under the matrix $\mathbf{A}_{i+1}$, namely $\mathbf{Y}_i^b = s_i^b \mathbf{A}_{i+1} + \mathbf{E}_i^b$.

$$\begin{tikzcd}
\mathbf{A}_1 \arrow[r, bend left, "s_1^1"] \arrow[r, bend right, "s_1^0"'] & \mathbf{A}_2 \arrow[r, bend left, "s_2^1"] \arrow[r, bend right, "s_2^0"'] & \mathbf{A}_3 \arrow[r, bend left, "s_3^1"] \arrow[r, bend right, "s_3^0"'] & \mathbf{A}_4
\end{tikzcd}$$

(a) The normal mode (i.e. $\forall x,\ C(x) = 1$)

$$\begin{tikzcd}
\mathbf{A}_1 \arrow[r, bend left, "s_1^1"] \arrow[r, bend right, "s_1^0"'] & \mathbf{A}_2 \arrow[r, bend left, "t_2^1"] \arrow[r, bend right, "s_2^0"'] & \mathbf{A}_3 \arrow[r, bend left, "s_3^1"] \arrow[r, bend right, "s_3^0"'] & \mathbf{A}_4
\end{tikzcd}$$

(b) The bit-fixing constraint $\star 0 \star$ (i.e. $C(x) = 1$ iff $x_2 = 0$)

$$\begin{tikzcd}
\mathbf{A}_J \arrow[r, bend left, "\mathbf{R} \otimes s_1^1"] \arrow[r, bend right, "\mathbf{I} \otimes s_1^0"'] & \mathbf{A}_2 \arrow[r, bend left, "\mathbf{I} \otimes s_2^1"] \arrow[r, bend right, "\mathbf{Q} \otimes s_2^0"'] & \mathbf{A}_3 \arrow[r, bend left, "\mathbf{R}^{-1} \otimes s_3^1"] \arrow[r, bend right, "\mathbf{I} \otimes s_3^0"'] & \mathbf{A}_4 \arrow[r, bend left, "\mathbf{I} \otimes s_4^1"] \arrow[r, bend right, "\mathbf{Q}^{-1} \otimes s_4^0"'] & \mathbf{A}_5
\end{tikzcd}$$

(c) The NC^1 constraint for point $x = 10$ (i.e. $C(x) = 0$ iff $x_1 = 1 \wedge x_2 = 0$)

Figure 1.1: Examples of the GGH15-based PRFs

Next, sample a preimage $\mathbf{D}_i^b$ of $\mathbf{Y}_i^b$ under the matrix $\mathbf{A}_i$, using the trapdoor of $\mathbf{A}_i$. That is, $\mathbf{A}_i \mathbf{D}_i^b = \mathbf{Y}_i^b$ and $\mathbf{D}_i^b$ is sampled from discrete Gaussian distribution of small width. The encoder then lets $\mathbf{D}_i^b$ be the encoding of s_i^b. The values $\mathbf{A}_1, \mathbf{D}_1^0, \mathbf{D}_1^1, ..., \mathbf{D}_\ell^0, \mathbf{D}_\ell^1$ are given to the evaluator. Given $x, y \in \{0,1\}^\ell$, the evaluator computes $F(x) = \mathbf{A}_1 \prod_{i=1}^{\ell} \mathbf{D}_i^{x_i}$ and $F(y) = \mathbf{A}_1 \prod_{i=1}^{\ell} \mathbf{D}_i^{y_i}$, and checks whether $F(x) - F(y)$ is a matrix with small entries.

To see why this works out functionality-wise consider the following equation:

$$F(x) = \mathbf{A}_1 \prod_{i=1}^{\ell} \mathbf{D}_i^{x_i} = \prod_{i=1}^{\ell} s_i^{x_i} \mathbf{A}_{\ell+1} + \underbrace{\sum_{i=1}^{\ell} \left(\prod_{j=1}^{i-1} s_j^{x_j} \cdot \mathbf{E}_i^{x_i} \cdot \prod_{k=i+1}^{\ell} \mathbf{D}_k^{x_k} \right)}_{\mathbf{E}_x} \pmod{q}. \tag{1}$$

If the secrets s_i^b are set with small norm, then the entire $\mathbf{E}_x$ term can be viewed as a small error term, so the dominant factor, $\prod_{i=1}^{\ell} s_i^{x_i} \mathbf{A}_{\ell+1}$, is purely determined by the multiplicative relationship of the secrets. As for security, observe that the encoding $\mathbf{D}_i^b$ of each secret s_i^b amounts to an LWE encoding of s_i^b, and furthermore the encoding of $\prod_{i=1}^{\ell} s_i^{x_i}$ is also in the form of an LWE instance $\mathbf{A}_{\ell+1}$, $\prod_{i=1}^{\ell} s_i^{x_i} \mathbf{A}_{\ell+1} + \mathbf{E}_x \pmod{q}$. However, it is not clear how to translate this observation to a concrete security property that is based on LWE. We discuss this point further below.

The power and danger in the GGH15 approach. The GGH15 encoding embeds the plaintext s into the *secret* term of the LWE instance, unlike in other LWE-based systems (e.g. Regev [Reg09] or dual-Regev [GPV08]) where the plaintext is associated with the error term or the $\mathbf{A}$ matrix. While the graph structure and trapdoor sampling mechanism enables homomorphic evaluations on the LWE secrets, analyzing the security becomes tricky. Unlike the traditional case where the LWE secrets s are independent and random, here the LWE secrets, representing plaintexts, are taken from distributions that are potentially structured or correlated with each other.

Such dependencies make it hard to prove security of the trapdoor sampling: Recall that the encoding $\mathbf{D}_i$ of some secret $\hat{s}_i$ (possibly obtained from an evaluation over correlated secrets) is the preimage of $\mathbf{Y}_i := \hat{s}_i \mathbf{A}_{i+1} + \mathbf{E}$ sampled by the trapdoor of $\mathbf{A}_i$. For instance, in the extreme case where $\hat{s}_i = 0$, then the public encoding $\mathbf{D}_i$ becomes a "weak trapdoor" of $\mathbf{A}_i$, which endangers the secrets encoded on the edges heading to $\mathbf{A}_i$ [GGH15].

Consequently, to safely use the GGH15 encoding, one has to consider the joint distribution of all the LWE secrets s_i^b, and prove that the trapdoor sampling algorithm remains secure with respect to these secrets. We demonstrate how to prove such a statement in a specific setting, where it is possible to simulate the encodings without knowing the secrets or the trapdoors.

LWE-based PRFs. The example of the "subset product" type encoding may remind the readers of the lattices-based pseudorandom functions [BPR12, BLMR13]. Indeed, recall the basic construction of Banerjee et al. [BPR12, Section 5.1]. For modulus $2 \leq p < q$ chosen such that q/p is exponential in the input length ℓ. The secret keys of the PRF are exactly 2ℓ LWE secrets $s_1^0, s_1^1, ..., s_\ell^0, s_\ell^1$ and a uniform matrix $\mathbf{A}$ over R_q. To evaluate, compute $F(x) = \left\lfloor \prod_{i=1}^{\ell} s_i^{x_i} \mathbf{A} \right\rceil_p$ where $\lfloor v \rceil_p$ means multiplying v by p/q and rounding to the nearest integer. Rounding plays a crucial role in the security proof, since it allows to add fresh small noise terms without changing the functionality whp, hence one can inductively obtain fresh LWE instances on any level.

Although not required for understanding this paper, we would like to mention that the existing lattice-based PRFs enjoy several nice properties, such as supporting low-depth evaluation in $\mathsf{TC}^0 \subseteq \mathsf{NC}^1$, and being additively key-homomorphic. Our CHCPRFs inherit these properties. In terms of the constraining ability, the construction from [BV15b] is the only known constrained PRF for all circuits based on LWE, but it is not constraint-hiding; neither is it secure given multiple constrained keys.

Our construction for bit-fixing constraints. A bit-fixing constraint is specified by a string $\mathbf{c} \in \{0, 1, \star\}^\ell$, where 0 and 1 are the matching bits and $\star$ denotes the wildcards. The constrain predicate C outputs 1 if the input matches $\mathbf{c}$. The combination of GGH15 and lattice-based PRFs inspires us to construct CHCPRFs for bit-fixing constraints. In fact, after rounding $F(x)$ in Equation (1), the functionality of $\lfloor F(x) \rceil$ is equivalent to (up to the rounding error) both the PRF from [BPR12, Section 5.1] and a variant of the PRF from [BLMR13, Section 5.1]. If we take the 2ℓ LWE secrets $s_1^0, s_1^1, ..., s_\ell^0, s_\ell^1$ as master secret key, the encodings $\mathbf{A}_1, \mathbf{D}_1^0, \mathbf{D}_1^1, ..., \mathbf{D}_\ell^0, \mathbf{D}_\ell^1$ as the evaluation key in the normal mode. An intuitive constraining algorithm is simply replacing the LWE secret of the constrained bit with an independent random element t, and reproduce its encoding $\mathbf{D}_t$. As an example, Figure 1.1a and Figure 1.1b illustrate the normal mode and constrained mode of a bit-fixing PRF.

We show that the key and the outputs from both the normal mode and the constrained mode (both modes use trapdoor sampling) are indistinguishable from an oblivious sampling procedure without using the trapdoors. The proof proceeds level-by-level (from level ℓ to level 1). Within each level i, there are two steps. The first step uses the computational hardness of LWE: observe that the LWE samples associated on $\mathbf{A}_{i+1}$ are with independent secrets, and $\mathbf{A}_{i+1}$ is trapdoor-free in that hybrid distribution by induction, so the LWE samples are indistinguishable from uniformly random. The second step uses a statistical sampling lemma by Gentry, Peikert and Vaikuntanathan [GPV08], which says the preimage of uniform outputs can be sampled without using the trapdoor of $\mathbf{A}_i$. The proof strategy is first illustrated by Brakerski et al. where they construct an evasive conjunction obfuscator from GGH15 [BVWW16].

Our construction and analysis imply that a variant of the PRF from [BLMR13] also satisfies 1-key bit-fixing constraint hiding. Although the PRF from [BLMR13] does not involve the trapdoor sampling procedure and is much simpler as a bit-fixing CHCPRF, understanding the GGH15-based version is beneficial for understanding the CHCPRF for NC^1 coming next.

Embedding a general constraint in the PRF keys. We move on towards embedding a general constraint in the key. Consider in particular the task of puncturing the key at a single point without revealing the point, which is essential to the applications like watermarking and deniable encryption mentioned in [BLW17]. Indeed, even that simple function seems to require some new idea.

To preserve the graph structure while handling general constraints, Barrington's Theorem [Bar86] comes into the picture. Recall that Barrington's Theorem converts any depth-d Boolean circuits into an oblivious branching program of length $z \leq 4^d$ composed of permutation matrices $\{\mathbf{B}_i^b\}_{b\in\{0,1\},i\in[z]}$ of dimension w (by default $w = 5$). Evaluation is done via multiplying the matrices selected by input bits, with the final output $\mathbf{I}^{w\times w}$ or a w-cycle $\mathbf{P}$ recognizing 1 or 0 respectively.

To embed permutation matrices in the construction, we set the secret term for the normal mode as $\mathbf{S}_i^b = \mathbf{I}^{w\times w} \otimes s_i^b = \begin{bmatrix} s_i^b & & 0 \\ & \ddots & \\ 0 & & s_i^b \end{bmatrix}$ (where $\otimes$ is the tensor product operator); in the constrained mode as $\mathbf{S}_i^b = \mathbf{B}_i^b \otimes s_i^b$. This provides the functionality of constraining all NC^1 circuits. See Figure 1.1c for an example of 2-bit point constraint $x_1x_2 \in \{0,1\}^2$, where x_1 controls the 1^{st} and 3^{rd} branches, x_2 controls the 2^{nd} and 4^{th} branches, $\mathbf{Q}$ and $\mathbf{R}$ represent different w-cycles.

We then analyze whether the permutation matrix structures are hidden in the constrained key, and whether the constrained outputs are pseudorandom. The first observation is that the tensor product of a permutation matrix $\mathbf{B}$ and any hard LWE secret distribution s forms a hard LWE distribution, i.e. $\mathbf{A}$, $(\mathbf{B} \otimes s) \cdot \mathbf{A} + \mathbf{E}$ is indistinguishable from uniformly random. This means both the secret and the permutation matrices are hidden in the constrained key.

Still, the rounded constrained output $\left\lfloor (\mathbf{P} \otimes \prod_{i=1}^{\ell} s_i^{x_i}) \cdot \mathbf{A}_{z+1} \right\rceil$ is a fixed permutation of the original value, so the adversary can left-multiply $\mathbf{P}^{-1}$ to obtain the original output. To randomize the constrained outputs, we adapt the "bookend" idea from the GGH15 candidate obfuscator. That is, we multiply the output on the left by a small random vector $\mathbf{J} \in R^{1\times w}$. By a careful reduction to standard LWE, one can show that $\mathbf{A}$, $\mathbf{J}\mathbf{A} + \mathbf{E}$, $\mathbf{J}\,(\mathbf{P} \otimes 1_R)\,\mathbf{A} + \mathbf{E}'$ is indistinguishable from uniformly random.

With these two additional hard LWE distributions in the toolbox, we can base NC^1 CHCPRF on LWE via the same two-step proof strategy (i.e. LWE+GPV in each level) used in the bit-fixing construction.

1.3 More on related work

More background on multilinear maps and the implication of this work. The notion of cryptographic multilinear maps was introduced by Boneh and Silverberg [BS03]. Currently there are three main candidates [GGH13, CLT13, GGH15], with a number of variants. However, what security properties hold for the candidates remains unclear. In particular, none of the candidates is known to satisfy the multilinear DDH or subgroup elimination assumptions that are sufficient for the CHCPRFs by Boneh et al. [BLW17] (see [GGH13, CHL+15, HJ16, CLLT16] for the attacks on these assumptions).

Note that even our result fails to formalize a general secret distribution that is safe for GGH15, it merely demonstrates a safe setting. Indeed, a central task in the study of the existing candidate multilinear maps is to identify settings where they can be used based on standard cryptographic assumptions [Hal15].

Relations to the GGH15 candidate program obfuscator. Our construction for NC^1 constraints is strongly reminiscent of the candidate obfuscator from GGH15 [GGH15, Section 5.2]. In particular, the "secrets" in the CHCPRF corresponds to the "multiplicative bundling scalars" from the GGH15 obfuscator. Under the restriction of releasing only 1 branch (either the functional branch or the dummy branch), our result implies that the "scalars" and permutation matrices can be hidden (without using additional safeguards such as the Kilian-type randomization and padded randomness on the diagonal).

In contrast, the recent cryptanalysis of the GGH15 obfuscator [CGH17] shows that when releasing both the functional key and the dummy key, one can extract the bundling scalars even if the obfuscator is equipped with all the safeguards.

It might be instructive to see where our reduction to LWE fail if one attempts to apply our proof technique to the two-key setting. The point is that when given two keys derived from the same master secret key, the adversary obtains LWE samples $\mathbf{Y}, \mathbf{Y}'$ with correlated secrets; Therefore it is not clear how to simulate the Gaussian samples of $\mathbf{D}$ conditioned on $\mathbf{AD} = \mathbf{Y}$ or of $\mathbf{D}'$ conditioned on $\mathbf{A}'\mathbf{D}' = \mathbf{Y}'$, without knowing the trapdoors of $\mathbf{A}$ and $\mathbf{A}'$.

1.4 Concurrent and follow-up work on CHCPRFs

The concurrent work of Boneh et al. [BKM17] and the follow-up work of Brakerski et al. [BTVW17] construct CHCPRFs for puncturing and for poly-size circuit constraints based on LWE. Their common methodology is to combine the existing lattice-based FHE [BV11, GSW13], ABE [GVW13, BGG$^+$14], and (non-constraint-hiding) constrained PRFs [BP14, BV15b, BFP$^+$15]. Such a methodology is originated from [GVW15]. It is plausible that there are some connections between the ABE+FHE approach and the GGH15 approach. Finding out such a connection is an interesting open problem.

To achieve a private-detectable watermarking scheme following the construction in [BLW17] requires a private programmable PRF, which is not immediately implied by an arbitrary constraint-hiding puncturable PRF, but still follows one with a randomizable puncturing algorithm. The approach is folklore, as mentioned in the work of Kim and Wu [KW17]. In Section 7, we formalize this approach.

Peikert and Shiehian [PS18] propose an direct construction of private programmable PRF, different from the folklore generic transformation. The direct construction supports programmability on polynomially many inputs, whereas the generic transformation from rerandomization can only program logarithmically many inputs. They also give a simulation-based definition for programmability. An interesting proposal from their paper is the abstraction of a "shift-hiding shift function", which provides a different view of a CHCPRF.

Chen, Vaikuntanathan and Wee [CVW18] extend the safe mode of GGH15 so as to cover non-permutation branching programs. As a result (among others), it gives a very simple constraint-hiding puncturable PRF that uses read-once branching program, whereas the construction that uses permutation branching programs requires an $O(\ell^2)$-step branching program for an ℓ-bit PRF.

Boneh, Kim and Wu [BKW17] investigate constrained PRFs for invertible functions, with a motivation of achieving puncturable pseudorandom permutations. We would like to mention that, even without concerning the ability of inverting, constructing a puncturable pseudorandom permutation remains an open problem.

Finally, we would like to mention that Attrapadung et al. [AMN$^+$18] propose the first constrained PRF for NC^1 circuit constraints from group theoretic assumptions (without assuming LWE, multilinear maps or iO). They also provide a constraint-hiding bit-fixing PRF.

2 Preliminaries

Notations and terminology. Let $\mathbb{R}, \mathbb{Z}, \mathbb{N}$ be the set of real numbers, integers and positive integers. The notation R is often used to denote some base ring. The concrete choices of R are $\mathbb{Z}^{n\times n}$ (the integer matrices) and $\mathbb{Z}[x]/(x^n+1)$ (where n is a power of 2). We denote $R/(qR)$ by R_q. The rounding operation $\lfloor a \rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$ is defined as multiplying a by p/q and rounding the result to the nearest integer.

For $n \in \mathbb{N}$, $[n] := \{1, ..., n\}$. A vector in $\mathbb{R}^n$ is represented in column form, and written as a bold lower-case letter, e.g. $\mathbf{v}$. For a vector $\mathbf{v}$, the i^{th} component of $\mathbf{v}$ will be denoted by v_i. A matrix is written as a bold capital letter, e.g. $\mathbf{A}$. The i^{th} column vector of $\mathbf{A}$ is denoted $\mathbf{a}_i$.

The length of a vector is the ℓ_p-norm $\|\mathbf{v}\|_p = (\sum v_i^p)^{1/p}$. The length of a matrix is the norm of its longest column: $\|\mathbf{A}\|_p = \max_i \|\mathbf{a}_i\|_p$. By default we use ℓ_2-norm unless explicitly mentioned. When a vector or matrix is called "small" (or "short"), we refer to its norm (resp. length). The thresholds of "small" will be precisely parameterized in the article and are not necessary negligible functions.

In cryptography, the security parameter (denoted as λ) is a variable that is used to parameterize the computational complexity of the cryptographic algorithm or protocol, and the adversary's probability of breaking security. An algorithm is "efficient" if it runs in (probabilistic) polynomial time over λ.

Many experiments and probability statements in this paper contain randomized algorithms (such as adversaries) within them. The probability of success of an experiment is always taken over the random coins used by the relevant randomized algorithms; therefore, we do not mention these coins explicitly. We use $\approx_s$ and $\approx_c$ as the abbreviation for statistically close and computationally indistinguishable.

2.1 Matrix branching programs

Definition 2.1 (Matrix branching programs)**.** *A width-w, length-z matrix branching program over ℓ-bit inputs consists of an index-to-input map, a sequence of pairs of matrices $\mathbf{B}_i^b$, and a non-identity matrix $\mathbf{P}$ representing 0:* $\mathsf{BP} = \{\iota : [z] \to [\ell], \{\mathbf{B}_i^b \in \{0,1\}^{w\times w}\}_{i\in[z],b\in\{0,1\}},\ \mathbf{P} \in \{0,1\}^{w\times w} \setminus \{\mathbf{I}\}\}$. *The program computes the function $f_{\mathsf{BP}} : \{0,1\}^\ell \to \{0,1\}$, defined as*

$$f_{\mathsf{BP}}(x) = \begin{cases} 1 & \text{if } \prod_{i\in[z]} \mathbf{B}_i^{x_{\iota(i)}} = \mathbf{I} \\ 0 & \text{if } \prod_{i\in[z]} \mathbf{B}_i^{x_{\iota(i)}} = \mathbf{P} \\ \perp & \text{elsewhere} \end{cases}$$

A set of branching programs $\{\mathsf{BP}\}$ is called **oblivious** if all the programs in the set have the same index-to-input map ι.

Theorem 2.2 (Barrington's theorem [Bar86])**.** *For $d \in \mathbb{N}$, and for any set of depth-d fan-in-2 Boolean circuits $\{C\}$, there is an oblivious set of width-5 length-4^d branching programs $\{\mathsf{BP}\}$ with a index-to-input map ι, where each BP is composed of permutation matrices $\{\mathbf{B}_i^b \in \{0,1\}^{5\times 5}\}_{i\in[z],b\in\{0,1\}}$, a 5-cycle $\mathbf{P}$, and ι.*

2.2 Lattices

An n-dimensional lattice Λ is a discrete additive subgroup of $\mathbb{R}^n$. Given n linearly independent basis vectors $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n \in \mathbb{R}^n\}$, the lattice generated by $\mathbf{B}$ is $\Lambda(\mathbf{B}) = \Lambda(\mathbf{b}_1, ..., \mathbf{b}_n) = \{\sum_{i=1}^{n} x_i \cdot \mathbf{b}_i, x_i \in \mathbb{Z}\}$. We have the quotient group $\mathbb{R}^n/\Lambda$ of cosets $\mathbf{c} + \Lambda = \{\mathbf{c} + \mathbf{v}, \mathbf{v} \in \Lambda\}$, $\mathbf{c} \in \mathbb{R}^n$. Let $\tilde{\mathbf{B}}$ denote the Gram-Schmidt orthogonalization of $\mathbf{B}$.

Gaussian on lattices. For any $\sigma > 0$, define the Gaussian function on $\mathbb{R}^n$ centered at $\mathbf{c}$ with parameter σ:

$$\forall \mathbf{x} \in \mathbb{R}^n,\ \rho_{\sigma,\mathbf{c}}(\mathbf{x}) = e^{-\pi\|\mathbf{x}-\mathbf{c}\|^2/\sigma^2}$$

For any $\mathbf{c} \in \mathbb{R}^n$, $\sigma > 0$, and n-dimensional lattice Λ, define the discrete Gaussian distribution over Λ as:

$$\forall \mathbf{x} \in \Lambda,\ D_{\Lambda+\mathbf{c},\sigma}(\mathbf{x}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{x})}{\rho_{\sigma,\mathbf{c}}(\Lambda)}$$

Lemma 2.3 ([PR06, MR07]). *Let $\mathbf{B}$ be a basis of an m-dimensional lattice Λ, and let $\sigma \geq \|\tilde{\mathbf{B}}\| \cdot \omega(\log n)$, then $\Pr_{\mathbf{x}\leftarrow D_{\Lambda,\sigma}}[\|\mathbf{x}\| \geq \sigma \cdot \sqrt{m} \vee \mathbf{x} = \mathbf{0}] \leq \mathsf{negl}(n)$.*

Gentry, Peikert and Vaikuntanathan [GPV08] show how to sample statistically close to discrete Gaussian distribution in polynomial time for sufficiently large σ (the algorithm is first proposed by Klein [Kle00]). The sampler is upgraded in [BLP$^+$13] so that the output is distributed exactly as a discrete Gaussian.

Lemma 2.4 ([GPV08, BLP$^+$13]). *There is a p.p.t. algorithm that, given a basis $\mathbf{B}$ of an n-dimensional lattice $\Lambda(\mathbf{B})$, $\mathbf{c} \in \mathbb{R}^n$, $\sigma \geq \|\tilde{\mathbf{B}}\| \cdot \sqrt{\ln(2n+4)/\pi}$, outputs a sample from $D_{\Lambda+\mathbf{c},\sigma}$.*

We then present the trapdoor sampling algorithm and the corollary of GPV lemma in the general ring R.

Lemma 2.5 ([Ajt99, AP11, MP12]). *There is a p.p.t. algorithm $\mathsf{TrapSam}(R, 1^n, 1^m, q)$ that, given the base ring R, modulus $q \geq 2$, lattice dimension n, and width parameter m (under the condition that $m = \Omega(\log q)$ if $R = \mathbb{Z}^{n\times n}$, $m = \Omega(n \log q)$ if $R = \mathbb{Z}[x]/(x^n+1)$), outputs $\mathbf{A} \leftarrow U(R_q^{1\times m})$ with a trapdoor τ.*

Lemma 2.6 ([GPV08]). *There is a p.p.t. algorithm $\mathsf{PreimgSam}(\mathbf{A}, \tau, \mathbf{y}, \sigma)$ that with all but negligible probability over $(\mathbf{A}, \tau) \leftarrow \mathsf{TrapSam}(R, 1^n, 1^m, q)$, for sufficiently large $\sigma = \Omega(\sqrt{n \log q})$, the following distributions are statistically close:*

$$\{\mathbf{A}, \mathbf{x}, \mathbf{y} : \mathbf{y} \leftarrow U(R_q), \mathbf{x} \leftarrow \mathsf{PreimgSam}(\mathbf{A}, \tau, \mathbf{y}, \sigma)\} \approx_s \{\mathbf{A}, \mathbf{x}, \mathbf{y} : \mathbf{x} \leftarrow \gamma_\sigma, \mathbf{y} = \mathbf{A}\mathbf{x}\}$$

where γ_σ represents $D^{1\times n}_{\mathbb{Z}^{nm},\sigma}$ if $R = \mathbb{Z}^{n\times n}$; represents $D_{R^m,\sigma}$ if $R = \mathbb{Z}[x]/(x^n+1)$.

When the image is a matrix $\mathbf{Y} = [\mathbf{y}_1||...||\mathbf{y}_\ell]$, we abuse the notation for the preimage sampling algorithm, use $\mathbf{D} \leftarrow \mathsf{PreimgSam}(\mathbf{A}, \tau, \mathbf{Y}, \sigma)$ to represent the concatenation of ℓ samples from $\mathbf{d}_i \leftarrow \mathsf{PreimgSam}(\mathbf{A}, \tau, \mathbf{y}_i, \sigma)_{i\in[\ell]}$.

2.3 General learning with errors problems

The learning with errors (LWE) problem, formalized by Regev [Reg09], states that solving noisy linear equations, in certain rings and for certain error distributions, is as hard as solving some worst-case lattice problems. The two typical forms used in cryptographic applications are (standard) LWE and RingLWE. The latter is introduced by Lyubashevsky, Peikert and Regev [LPR13a].

We formulate them as the General learning with errors problems similar to those of [BGV12], with more flexibility in the secret distribution and the base ring.

Definition 2.7 (General learning with errors problem). *The (decisional) general learning with errors problem (GLWE) is parameterized by the base ring R, dimension parameters k, ℓ, m for samples, dimension parameter n for lattices, modulus q, the secret distribution η over $R^{k\times\ell}$, and the error distribution χ over $R^{\ell\times m}$. The $\mathsf{GLWE}_{R,k,\ell,m,n,q,\eta,\chi}$ problem is to distinguish the following two distributions: (1) LWE samples $s \leftarrow \eta$, $\mathbf{A} \leftarrow U(R_q^{\ell\times m})$, $\mathbf{E} \leftarrow \chi^{k\times m}$, output $(\mathbf{A}, s\mathbf{A}+\mathbf{E}) \in (R_q^{\ell\times m} \times R_q^{k\times m})$; (2) uniform distributions $U(R_q^{\ell\times m} \times R_q^{k\times m})$.*

We define $\mathsf{GLWE}_{R,k,\ell,m,n,q,\eta,\chi}$-hardness for secret distributions. The subscripts are dropped if they are clear from the context.

Definition 2.8. *A secret distribution η is called $\mathsf{GLWE}_{R,k,\ell,m,n,q,\eta,\chi}$-hard if no p.p.t. adversary distinguishes the two distributions in the $\mathsf{GLWE}_{R,k,\ell,m,n,q,\eta,\chi}$ problem with $1/2$ plus non-negligible probability.*

Here are the connections of decisional LWE/RingLWE to the worst-case lattice problems, in the language of GLWE-hardness. For the LWE problem we present the version where the secret is a square matrix.

Lemma 2.9 (LWE [Reg09, Pei09, BLP+13])**.** *Let n be an integer, $R = \mathbb{Z}^{n\times n}$. q be an integer modulus, $0 < \sigma < q$ such that $\sigma > 2\sqrt{n}$. If there exists an efficient (possibly quantum) algorithm that breaks $\mathsf{GLWE}_{R,1,1,m,n,q,U(R_q),D^{n\times n}_{\mathbb{Z},\sigma}}$, then there exists an efficient (possibly quantum) algorithm for approximating SIVP and GapSVP in the ℓ_2 norm, in the worst case, to within $\tilde{O}(nq/\sigma)$ factors.*

Lemma 2.10 (RingLWE [LPR13a, DD12, LS15])**.** *Let n be a power of 2, $R = \mathbb{Z}[x]/(x^n+1)$. Let q be a prime integer s.t. $q \equiv 1 \pmod{n}$. $0 < \sigma < q$, $\sigma > \omega(\sqrt{\log(n)})$, $\sigma' > n^{3/4}m^{1/4}\sigma$. If there exists an efficient (possibly quantum) algorithm that breaks $\mathsf{GLWE}_{R,1,1,m,n,q,U(R_q),D_{R,\sigma'}}$, then there exists an polynomial time quantum algorithm for solving SVP for ideal-lattices over R, in the worst case, to within $\tilde{O}(\sqrt{n}q/\sigma)$ factors.*

For proper choices of parameters, error distributions of small norm can be used as hard secret distribution (usually called Hermit-normal-form LWE).

Lemma 2.11 (HNF-LWE [ACPS09, BLP+13])**.** *For R, m, n, q, σ chosen as was in Lemma 2.9, $\mathsf{GLWE}_{R,1,1,m',n,q,D^{n\times n}_{\mathbb{Z},\sigma},D^{n\times n}_{\mathbb{Z},\sigma}}$ is as hard as $\mathsf{GLWE}_{R,1,1,m,n,q,U(R_q),D^{n\times n}_{\mathbb{Z},\sigma}}$ for $m' \le m - (16n + 4\log\log q)$.*

Lemma 2.12 (HNF-RingLWE [LPR13b])**.** *For $R, m, n, q, \sigma, \sigma'$ chosen as in Lemma 2.10, $\mathsf{GLWE}_{R,1,1,m-1,n,q,D_{R,\sigma'},D_{R,\sigma'}}$ is as hard as $\mathsf{GLWE}_{R,1,1,m,n,q,U(R_q),D_{R,\sigma'}}$.*

Pseudorandom functions based on GLWE. We adapt theorems from the PRF construction of Boneh, Lewi, Montgomery, and Raghunathan [BLMR13, Theorems 4.3, 5.1]. The result was originally stated for LWE. We observe that it holds for general rings under proper choices of parameters.

Lemma 2.13 (Adapted from [BLMR13])**.** *Let $\ell \in \mathbb{N}$ be the bit-length of the input. $m, n, q, p \in \mathbb{N}$, $\sigma, B \in \mathbb{R}$ s.t. $0 < \sigma < q$, $B \ge \sigma\sqrt{m}$, $q/p > B^\ell$. $\eta = U(R_q)$, γ_σ is a distribution over $R^{m\times m}$ parameterized by σ, χ_σ is a distribution over R parameterized by σ. $\|\gamma_\sigma\|, \|\chi_\sigma\| \le \sigma\sqrt{m}$.*

Consider the function $f : \{0,1\}^\ell \to R_p^{1\times m}$, $f_{\mathbf{U}}(x) = \left\lfloor \mathbf{U}\prod_{i=1}^{\ell}\mathbf{D}_i^{x_i}\right\rceil_p$, where $\mathbf{U} \leftarrow U(R_q^{1\times m})$ is the private parameter, $\{\mathbf{D}_i^b \leftarrow \gamma_\sigma\}_{b\in\{0,1\},i\in[\ell]}$ is the public parameter.

If there is an efficient algorithm that given input $\mathbf{A} \leftarrow U(R_q^{1\times m})$, outputs $(\mathbf{U} \in R_q^{1\times m}, \mathbf{D} \in R^{m\times m})$ that are statistically close to $(U(R_q^{1\times m}) \times \gamma_\sigma)$ and $\mathbf{UD} = \mathbf{A}$; then f is a PRF assuming the hardness of $\mathsf{GLWE}_{R,1,1,m,n,q,\eta,\chi_\sigma}$.

Proof sketch: The proof consists of two parts, both extended from [BLMR13, Theorems 4.3, 5.1]. The first part defines a variant of the GLWE problem called *Non-uniform GLWE* (hereafter NuGLWE). [BLMR13, Theorems 4.3] proves that for the integral matrix ring, NuLWE is implied by standard LWE. Below we describe the adaption of the first part (from [BLMR13, Theorem 4.3]) to the general rings. The second part which proves that f is a PRF assuming NuGLWE follows the immediate extension from [BLMR13, Theorem 5.1].

The NuGLWE problem asks to distinguish samples $(\mathbf{D}, \mathbf{KD}+\mathbf{E}) \in (R^{m\times m} \times R^{1\times m})$ from $(\gamma \times U(R_q^{1\times m}))$, where $\mathbf{D} \leftarrow \gamma$ is possibly non-uniform, $\mathbf{K} \leftarrow U(R_q^{1\times m})$, $\mathbf{E} \leftarrow \chi_\sigma^{1\times m}$. To reduce NuGLWE to GLWE with samples $(\mathbf{A}, \mathbf{Y}) \in (R_q^{1\times m} \times R_q^{1\times m})$ where $\mathbf{A}$ is uniform, let the GLWE attacker sample $(\mathbf{U} \in R_q^{1\times m}, \mathbf{D} \in$

$R^{m\times m}) \approx_s (U(R_q^{1\times m}) \times \gamma)$ s.t. $\mathbf{UD} = \mathbf{A}$, send $(\mathbf{D}, \mathbf{Y})$ to the NuGLWE distinguisher. If $(\mathbf{A}, \mathbf{Y})$ is a GLWE sample where $\mathbf{Y} = s\mathbf{A} + \mathbf{E}$, then take $\mathbf{K} := s\mathbf{U}$, and observe that $(\mathbf{D}, \mathbf{Y})$ is from the correct distribution of NuGLWE. If $(\mathbf{A}, \mathbf{Y})$ is from uniform, $(\mathbf{D}, \mathbf{Y})$ is from $(\gamma \times U(R_q^{1\times m}))$. □

The proof from [BLMR13, Theorem 4.3] shows the reduction above for LWE under proper parameter settings. To base it on RingLWE for $R = \mathbb{Z}[x]/(x^n + 1)$ where n is a power of 2, parameters can be set as $m \geq 2n \log q$, $\sigma = \omega(\sqrt{n \log q})$, $\gamma_\sigma = D^{1\times m}_{R^m,\sigma}$, $\chi_\sigma = D_{R,\sigma}$.

3 GLWE-hard distributions: extension package

We prove GLWE-hardness for the following "structural" secret distributions. They are used in the analysis of Construction 5.7.

Lemma 3.1. *Fix a permutation matrix $\mathbf{B} \in \{0,1\}^{w\times w}$. If a secret distribution η over R is $\mathsf{GLWE}_{R,1,1,w^2m,n,q,\eta,\chi}$-hard, then the secret distribution $\mathbf{B} \otimes \eta$ is $\mathsf{GLWE}_{R^{w\times w},1,1,m,n,q,\mathbf{B}\otimes\eta,\chi^{w\times w}}$-hard.*

Proof. For a permutation matrix $\mathbf{B} \in \{0,1\}^{w\times w}$, suppose there is a p.p.t. distinguisher for samples from

$$(\mathbf{A}, (\mathbf{B} \otimes s)\mathbf{A} + \mathbf{E}), \text{ where } \mathbf{A} \leftarrow U(R_q^{w\times wm}), s \leftarrow \eta, \mathbf{E} \leftarrow \chi^{w\times wm}$$

and samples from the uniform distribution $(U(R_q^{w\times wm}), U(R_q^{w\times wm}))$, then we build an attacker for $\mathsf{GLWE}_{R,1,1,w^2m,n,q,\eta,\chi}$. The attacker is given a $\mathsf{GLWE}_{R,1,1,w^2m,n,q,\eta,\chi}$ instance

$$(\mathbf{A}', \mathbf{Y}') = (\mathbf{A}_1||...||\mathbf{A}_w, \mathbf{Y}_1||...||\mathbf{Y}_w), \text{ where } \mathbf{A}_i, \mathbf{Y}_i \in R^{1\times wm}, \; i \in [w].$$

It then rearranges the blocks as $(\mathbf{U}, \mathbf{V}) \in R^{w\times wm} \times R^{w\times wm}$, where the i^{th} (blocked) row of $\mathbf{U}$ is $\mathbf{A}_i$, the i^{th} (blocked) row of $\mathbf{V}$ is $\mathbf{Y}_i$. The attacker then sends $(\mathbf{U}, (\mathbf{B} \otimes 1_R)\,\mathbf{V})$ to the distinguisher. Observe that $(\mathbf{U}, (\mathbf{B} \otimes 1_R)\,\mathbf{V})$ is from the $\mathbf{B} \otimes \eta$ secret distribution if $(\mathbf{A}', \mathbf{Y}')$ is from the η secret distribution, or from the uniform distribution if $(\mathbf{A}', \mathbf{Y}')$ is from the uniform distribution. Hence the attacker wins with the same probability as the distinguisher. □

Lemma 3.2. *Let $w \in [2, \infty) \cap \mathbb{Z}$. Fix a permutation matrix $\mathbf{C} \in \{0,1\}^{w\times w}$ that represents a w-cycle. If a secret distribution η over R is $\mathsf{GLWE}_{R,1,1,wm,n,q,\eta,\chi}$-hard, then the secret distribution $(\eta^{1\times w}, \eta^{1\times w} \times (\mathbf{C} \otimes 1_R))$ is $\mathsf{GLWE}_{R,2,w,m,n,q,(\eta^{1\times w},\eta^{1\times w}\times(\mathbf{C}\otimes 1_R)),\chi}$-hard.*

Proof. Let $\mathbf{H} = [h_1, h_2, ..., h_w]$ where $\{h_i \leftarrow \eta\}_{i\in[w]}$. Let

$$\mathcal{H} := \{(\mathbf{A}_j, \mathbf{Y}_{i,j} = h_i\mathbf{A}_j + \mathbf{E}_{i,j}) | \mathbf{A}_j \leftarrow U(R_q^{1\times m}), h_i \leftarrow \eta, \mathbf{E}_{i,j} \leftarrow \chi, i, j \in [w]\}$$

be the rearrangement of w independent GLWE samples from $\mathsf{GLWE}_{R,1,1,wm,n,q,\eta,\chi}$. $\mathcal{H}$ is indistinguishable from the uniform distribution $\mathcal{U} := \{(\mathbf{A}_j, \mathbf{Y}_{i,j}) | \mathbf{A}_j \leftarrow U(R_q^{1\times m}), \mathbf{Y}_{i,j} \leftarrow U(R_q^{1\times m}), i, j \in [w]\}$ due to standard GLWE.

We show that if there is an attacker D' that distinguishes

$$(\mathbf{A}, \mathbf{HA} + \mathbf{E}, \mathbf{H}\,(\mathbf{C} \otimes 1_R)\,\mathbf{A} + \mathbf{E}'),$$

where $\mathbf{E}, \mathbf{E}' \leftarrow \chi^{1\times m}$ from

$$U(R_q^{w\times m} \times R_q^{1\times m} \times R_q^{1\times m}),$$

then there is a distinguisher D for (a subset of) $\mathcal{H}$ and $\mathcal{U}$.

To do so, we simulate the $(\eta^{1\times w}, \eta^{1\times w} \times (\mathbf{C} \otimes 1_R))$ samples from $\mathcal{H}$ or $\mathcal{U}$ by setting $\mathbf{A} \in R^{w\times m}$ where the j^{th} row of $\mathbf{A}$ is $\mathbf{A}_j$, $\mathbf{Y} := \sum_{j\in[w]} \mathbf{Y}_{j,j}$, and $\mathbf{Z} := \sum_{j\in[w]} \mathbf{Y}_{\zeta(j),j}$, where $\zeta(j) : [w] \to [w]$ outputs the row number of the 1-entry in the j^{th} column of $\mathbf{C}$. Note that being a w-cycle indicates that the 1-entries in $\mathbf{C}$ disjoint with the 1-entries in $\mathbf{I}^{w\times w}$. Observe that the sample $(\mathbf{A}, \mathbf{Y}, \mathbf{Z})$ is from the secret distribution $(\eta^{1\times w}, \eta^{1\times w} \times (\mathbf{C} \otimes 1_R))$ if transformed from $\mathcal{H}$, or from the uniform distribution if transformed from $\mathcal{U}$. Hence the distinguisher D' wins with the same probability as the attacker D. $\square$

4 Constraint-hiding constrained PRFs

This section provides the definitions of constraint-hiding constrained PRFs. We first recall the indistinguishability-based definition from [BLW17], then give our simulation-based definition, and discuss the relations among these two definitions and program obfuscation.

4.1 The indistinguishability-based definition

We first recall the indistinguishability-based definition for CHCPRF from [BLW17].

Definition 4.1 (Indistinguishability-based CHCPRF [BLW17]). *Consider a family of functions $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbf{N}}$ where $\mathcal{F}_\lambda = \{F_k : D_\lambda \to R_\lambda\}_{\lambda\in\mathbb{N}}$, along with a triple of efficient functions (*Gen, $\mathsf{Constrain}$, Eval*). For a constraint family $\mathcal{C} = \{C_\lambda : D_\lambda \to \{0,1\}\}_{\lambda\in\mathbb{N}}$; the key generation algorithm* $\mathsf{Gen}(1^\lambda)$ *generates the master secret key* MSK*, the constraining algorithm* $\mathsf{Constrain}(1^\lambda, \mathsf{MSK}, C)$ *takes the master secret key* MSK*, a constraint C, outputs the constrained key* CK*; the evaluation algorithm* $\mathsf{Eval}(k, x)$ *takes a key k, an input x, outputs $F_k(x)$.*

We say that $\mathcal{F}$ is an **indistinguishability-based CHCPRF for** *$\mathcal{C}$ if it satisfies the following properties:*

Functionality preservation over unconstrained inputs. *For input $x \in D_\lambda$ s.t. $C(x) = 1$,* $\Pr[\mathsf{Eval}(\mathsf{MSK}, x) = \mathsf{Eval}(\mathsf{CK}, x)] \geq 1 - \mathsf{negl}(\lambda)$*, where the probability is taken over the randomness in algorithms* Gen *and* $\mathsf{Constrain}$.

Pseudorandomness for constrained inputs. *Consider the following experiment between a challenger and an adversary. The adversary can ask 3 types of oracle queries: constrained key oracle, evaluation oracle, and challenge oracle. For $b \in \{0,1\}$, the challenger responds to each oracle query in the following manner:*

- *Constrained key oracle. Given a circuit $C \in \mathcal{C}$, the challenger outputs a constrained key* $\mathsf{CK} \leftarrow \mathsf{Constrain}(1^\lambda, \mathsf{MSK}, C)$.
- *Evaluation oracle. Given an input $x \in D_\lambda$, the challenger outputs $y \leftarrow$* $\mathsf{Eval}(\mathsf{MSK}, x)$.
- *Challenge oracle. Given an input $x_c \in D_\lambda$, the challenger outputs $y \leftarrow$* $\mathsf{Eval}(\mathsf{MSK}, x_c)$ *if $b = 1$; outputs $y \leftarrow U(R_\lambda)$ if $b = 0$.*

The queries from the adversary satisfy the conditions that $C(x_c) = 0$, and x_c is not sent among evaluation queries. At the end of the experiment, the adversary chooses b' and wins if $b' = b$. The scheme satisfies the pseudorandomness property if the winning probability of any p.p.t. adversary is bounded by $1/2 + \mathsf{negl}(\lambda)$.

Indistinguishability-based constraint-hiding. *Consider the following experiment between a challenger and an adversary. The adversary can ask 2 types of oracle queries: constrained key oracle or evaluation oracle. For $b \in \{0,1\}$, the challenger responds to each oracle query in the following manner:*

- *Constrained key oracle. Given a pair of circuits $C_0, C_1 \in \mathcal{C}$, the challenger outputs a constrained key for C_b:* $\mathsf{CK} \leftarrow \mathsf{Constrain}(1^\lambda, \mathsf{MSK}, C_b)$.
- *Evaluation oracle. Given an input $x \in D_\lambda$, the challenger outputs $y \leftarrow$* $\mathsf{Eval}(\mathsf{MSK}, x)$.

For a circuit $C \in \mathcal{C}$, denote $S(C) := \{x \in D_\lambda : C(x) = 1\}$. Suppose the adversary asks h pairs of circuit constraints $\{C_0^{(g)}, C_1^{(g)}\}_{g\in[h]}$, the queries are **admissible** *if (1) $\forall i \neq j \in [h]$, $S(C_0^{(i)}) \cap S(C_0^{(j)}) = S(C_1^{(i)}) \cap S(C_1^{(j)})$; (2) for all input evaluation queries x, for all $g \in [h]$, $C_0^{(g)}(x) = C_1^{(g)}(x)$.*
At the end of the experiment, the adversary chooses b' and wins if $b' = b$. The scheme satisfies the constraint-hiding property if the winning probability of any p.p.t. adversary is bounded by $1/2 + \mathsf{negl}(\lambda)$.

4.2 The simulation-based definition

Next we give the simulation-based definition. We first present a definition that is central to the discussions and constructions in the paper, then mention its variants.

Definition 4.2 (Simulation-based CHCPRF)**.** *Consider a family of functions $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbf{N}}$ with the same syntax as in Definition 4.1. We say that $\mathcal{F}$ is* **simulation-based CHCPRF for family $\mathcal{C}$ of circuits** *if for any polytime stateful algorithm* Adv*, there is a polytime stateful algorithm* Sim *such that:*

$$\{\textit{Experiment REAL}_{\mathsf{Adv}}(1^\lambda)\}_{\lambda\in\mathbb{N}} \approx_c \{\textit{Experiment IDEAL}_{\mathsf{Adv},\mathsf{Sim}}(1^\lambda)\}_{\lambda\in\mathbb{N}} .$$

The ideal and real experiments are defined as follows for algorithms Adv *and* Sim*:*

```
Experiment REAL_Adv(1^λ)
$\mathsf{MSK} \leftarrow \mathsf{Gen}(1^\lambda)$,
Repeat :
  $\mathsf{Adv} \to (x, d_x);\ y = \mathsf{Eval}(\mathsf{MSK}, x)$
  $\mathsf{Adv} \leftarrow y$
$\mathsf{Adv} \to C$;
 if $d_x \neq C(x)$ for some $x$ then Output $\perp$
 else $Adv \leftarrow \mathsf{Constrain}(\mathsf{MSK}, C)$
Repeat :
  $\mathsf{Adv} \to x;\ y = \mathsf{Eval}(\mathsf{MSK}, x)$
  $\mathsf{Adv} \leftarrow y$
$\mathsf{Adv} \to b$; Output $b$
```

```
Experiment IDEAL_Adv,Sim(1^λ)
$\mathsf{Sim} \leftarrow 1^\lambda$
Repeat :
  $\mathsf{Adv} \to (x, d_x);\ y = \mathsf{Sim}(x, d_x)$
  if $d_x = 0$ then $y = U(R)$; $\mathsf{Adv} \leftarrow y$
$\mathsf{Adv} \to C$;
 if $d_x \neq C(x)$ for some $x$ then Output $\perp$
 else $\mathsf{Adv} \leftarrow \mathsf{Sim}(1^{|C|})$
Repeat :
  $\mathsf{Adv} \to x;\ y = \mathsf{Sim}(x, C(x))$
  if $C(x) = 0$ then $y = U(R)$; $\mathsf{Adv} \leftarrow y$
$\mathsf{Adv} \to b$; Output $b$
```

That is, in the experiments the adversary can ask a single constraint query and polynomially many input queries, in any order. For input queries x made before the circuit query, Adv *is expected to provide a bit d_x indicating whether $C(x) = 1$. In the real experiment* Adv *obtains the unconstrained function value at x. In the ideal experiment* Sim *learns the indicator bit d_x; if $d_x = 1$ then* Adv *gets a value generated by* Sim*, and if $d_x = 0$ then* Adv *obtains a random value from the range R of the function. Once* Adv *makes the constraint query $C \in \mathcal{C}_\lambda$, both experiments verify the consistency of the indicator bits d_x for all the inputs x queried by* Adv *so far. If any inconsistency is found then the experiment halts.*

Next, in the real experiment Adv *obtains the constrained key generated by the constraining algorithm; in the ideal experiment* Adv *obtains a key generated by* Sim*, whereas* Sim *is given only the size of* C*. The handling of input queries made by* Adv *after the circuit query is similar to the ones before, with the exception that the indicator bit* d_x *is no longer needed and* Sim *obtains the value of* $C(x)$ *instead. The output of the experiment is the final output bit of* Adv.

Remark 4.3. *One may also consider a stronger definition than Definition 4.2 where the adversary is not required to provide the indicator bits* d_x *in the queries prior to providing the constraint. However we note that this stronger definition is unachievable if the number of input queries before the constraint query is unbounded, due to an "incompressibility" argument similar to the one from [AGVW13].*

Definition 4.4 (Selective security). *A function ensemble* $\mathcal{F}$ *is a* **selective simulation-secure CHCPRF** *if the adversary in the experiments of Definition 4.2 sends all the constraint and input queries at once.*

The simulation-based definition can be generalized to the setting where the adversary queries multiple constrained keys. We present some natural generalizations, and discuss the possibility of achieving them shortly.

Definition 4.5 (Multiple-key simulation security). *A function ensemble* $\mathcal{F}$ *is an* **h-key simulation-secure CHCPRF** *if the adversary in the experiments of Definition 4.2 can send* h *constrained key queries* $\{C_k\}_{k\in[h]}$*. The simulator's inputs are changed as follows: on an input query* x*, the simulator receives* x *and indicators* $\{C_k(x)\}_{k\in[h]}$*; on a constraint query for* C_k*, the simulator only receives the description length of* C_k*.*

Note that Definition 4.5 is unachievable when the adversary does not make evaluation queries. This is so since the view of the simulator is independent of whether the two constraints agree on some known input value (say, 1), so there is no hope of providing meaningful simulation.

Below we consider a weaker variant of the multi-key simulation security, where the simulator is given oracle access to the constrained circuits.

Definition 4.6 (Multiple-key weak simulation security). *A function ensemble* $\mathcal{F}$ *is an* **h-key weakly simulation-secure CHCPRF** *if in addition to Definition 4.5, the simulator has oracle access to the circuits* $\{C_k\}_{k\in[h]}$*.*

4.3 Relations among the definitions

We discuss the relation among the definitions of CHCPRF and program obfuscation.

Multiple-key CHCPRFs implies obfuscation. We show that the weakly simulation-based CHCPRF for 2 keys implies the strong virtual black-box obfuscation notion of [Had00] which is impossible to obtain for certain functionalities. For the indistinguishability-based definition proposed in [BLW17], achieving 2-key security implies indistinguishability obfuscation [BGI+12].

Recall the definitions for strong VBB obfuscation and indistinguishability obfuscation.

Definition 4.7 (Obfuscation [Had00, BGI+12]). *A probabilistic algorithm* O *is an obfuscator for a class of circuit* $\mathcal{C}$ *if the following conditions hold:*

- *(Preservation of the function) For all inputs* x, $\Pr[C(x) = O(C(x))] > 1 - \mathsf{negl}(\lambda)$.
- *(Polynomially slowdown) There is a polynomial* p *s.t.* $|O(C)| < p(|C|)$.
- *(Strong virtual black-box obfuscation) For any p.p.t. adversary* Adv*, there is a p.p.t. simulator* Sim *s.t. for all* C*,* $\{\mathsf{Adv}(1^\lambda, O(C))\} \approx_c \{\mathsf{Sim}^{C(\cdot)}(1^\lambda, |C|)\}$.

- *(Indistinguishability obfuscation) For functionally equivalent circuits C_0, C_1, $O(C_0) \approx_c O(C_1)$.*

Construction 4.8 (Obfuscator from 2-key CHCPRFs)**.** *Given a CHCPRF, we construct an obfuscator for C by create a constrained key $\mathsf{CK}[C]$, and a constrained key $\mathsf{CK}[I]$ where I is the circuit that always outputs 1. To evaluate $C(x)$, output 1 if $\mathsf{CHCPRF}_{\mathsf{CK}[C]}(x) = \mathsf{CHCPRF}_{\mathsf{CK}[I]}(x)$, 0 otherwise.*

Theorem 4.9. *If 2-key weakly simulation-secure CHCPRF from Definition 4.6 exists for circuit class $\mathcal{C}$, then Construction 4.8 is a strong VBB obfuscation for circuit class $\mathcal{C}$.*

Proof. The simulator for the VBB obfuscator $\mathsf{Sim}_{\mathsf{VBB}}$ runs the simulator for CHCPRF Sim_F. Once Sim_F makes indicator queries on x, $\mathsf{Sim}_{\mathsf{VBB}}$ queries its circuit oracle $C(\cdot)$ on x, sends the indicators $C(x)$, 1 to Sim_F. Sim_F outputs simulated constraint keys $\mathsf{CK}^S[C]$, $\mathsf{CK}^S[I]$, which are indistinguishable from the real constrained keys $\mathsf{CK}[C]$, $\mathsf{CK}[I]$ used in the obfuscator. □

Corollary 4.10 ([Had00, BGI+12])**.** *There exists a class of constraints for which 2-key weakly simulation-secure CHCPRF is impossible to obtain.*

Theorem 4.11. *If 2-key indistinguishability-based CHCPRF exists for circuit class $\mathcal{C}$, then Construction 4.8 is an indistinguishability obfuscator for circuit class $\mathcal{C}$.*

Proof. For a circuit C, the obfuscator outputs $\mathsf{CK}[C]$, $\mathsf{CK}[I]$. For functionally equivalent circuits C_0 and C_1, $S(C_0) \cap S(I) = S(C_1) \cap S(I)$. By indistinguishability constraint-hiding, $(\mathsf{CK}[C_0], \mathsf{CK}[I]) \approx_c (\mathsf{CK}[C_1], \mathsf{CK}[I])$. □

Simulation and indistinguishability-based definitions for CHCPRF. Next we discuss the relation of the simulation and indistinguishability-based definitions for CHCPRF, under 1-key security. The two definitions are equivalent in the selective setting. Below we state the theorems for the selective versions of the definitions, then discuss when the implications hold in the adaptive setting.

Theorem 4.12. *If a CHCPRF satisfies the selective simulation-based definition, then it satisfies the selective indistinguishability-based definition.*

Proof. Correctness follows directly from the simulation-based definition. For constraint-hiding, we prove by a hybrid argument, that the constrained key for either C_0 or C_1 is indistinguishable from an intermediate simulated constrained key. Formally, for CHCPRF $\mathcal{F}$, suppose there is an distinguisher D that violates Definition 4.1, we build a distinguisher D' between the real and simulated distributions in Definition 4.2. For circuits C_0, C_1 and input queries $\{x^{(k)}\}_{k\in[t]}$ that are admissible for Definition 4.1, for $b \in \{0,1\}$, D' obtains the constrained keys and outputs either from the real $\mathcal{R}_b := (\mathsf{CK}[C_b], \{x^{(k)}, y^{(k)}\}_{k\in[t]})$ or the simulated distribution $\mathcal{S} := (\mathsf{CK}[C^S], \{x^{(k)}, y^{(k)S}\}_{k\in[t]})$, send it to D. Given that D is able to distinguish $\mathcal{R}_0$ from $\mathcal{R}_1$ with non-negligible advantage, D is also able to distinguish (at least) one of $\mathcal{R}_b$ from $\mathcal{S}$ with non-negligible advantage, $b \in \{0,1\}$.

Pseudorandomness of the constrained outputs can be shown via a similar hybrid argument. □

Using the same hybrid argument, one can show the implications hold for the adaptive settings. More precisely, the standard simulation definition from Definition 4.2 implies a variant of the indistinguishability definition from Definition 4.1 where the indicators on the input queries are fixed before the circuits are chosen; for the stronger simulation definition discussed in Remark 4.3, it implies the fully adaptive variant of Definition 4.1.

Theorem 4.13. *If a CHCPRF satisfies 1-key selective indistinguishability-based definition, it satisfies the 1-key selective simulation-based definition.*

Proof. For a CHCPRF $\mathcal{F}$ that satisfies Definition 4.1 for one constrained key query, we construct a simulator as per Definition 4.2. The simulator picks an all-1 circuit $C^S = I$ such that $I(x) = 1, \forall x \in D_\lambda$, and use the indistinguishability-secure constraining algorithm to derive a constrained key CK^S for C^S. Once the simulator obtains the inputs and the indicators $\{x^{(k)}, d^{(k)}\}_{k\in[t]}$, if $d^{(k)} = 1$, outputs $\mathsf{Eval}(\mathsf{CK}^S, x^{(k)})$; if $d^{(k)} = 0$, outputs $y \leftarrow U(R_\lambda)$.

We first prove constraint-hiding. Suppose there is an adversary A' that distinguishes the simulated distribution from the real distribution, we build an adversary A that breaks the indistinguishability definition for $\mathcal{F}$. A sends constrained circuit queries $C_0 = C$ and $C_1 = I$, obtains $\mathsf{CK}[C_b]$. Then A sends input queries. For $x^{(k)}$ s.t. $C(x^{(k)}) = I(x^{(k)}) = 1$, the output is $\mathsf{Eval}(\mathsf{CK}[C_b], x^{(k)})$; for $x^{(k)}$ s.t. $C(x^{(k)}) \neq I(x^{(k)})$, it is an inadmissible query so A samples an uniform random output on its own. Then A forwards $\mathsf{CK}[C_b]$, inputs and outputs to A'. The choice of A' for the real or the simulated distribution corresponds to $b = 0$ or 1, hence the advantage of A is equivalent to A'.

Next we prove pseudorandomness on the constrained inputs. For input x such that $C(x) = 0$, suppose there is an adversary A' that distinguishes the simulated output (which is uniformly random) from the real output, we build an adversary A that breaks the indistinguishability-based pseudorandomness property on the challenge input for $\mathcal{F}$. The reduction is the same as one above except that the adversary A forwards the challenge output as the reply of one of the real or simulated output. The choice of A' for the real or the simulated distribution corresponds to $b = 0$ or 1, hence the advantage of A is equivalent to A'. $\square$

The implication extends to the adaptive setting (namely, 1-key adaptive indistinguishability security implies 1-key adaptive simulation security) where the input queries are made after the constraint query.

5 The constructions

In Sections 5.1 and 5.2 we present the CHCPRFs for bit-fixing and NC^1 circuit constraints. The proofs of both constructions achieve a partially adaptive variant of Definition 4.2, where the input queries can only be made adaptively after the constraint query[1].

5.1 Bit-fixing CHCPRFs

Definition 5.1 (Bit-fixing constraint [BW13])**.** *A bit-fixing constraint is specified by a string* $\mathbf{c} \in \{0, 1, \star\}^\ell$*, where* 0 *and* 1 *are the fixing bits and* $\star$ *denotes the wildcards.* $C(x) = 1$ *if the input matches* $\mathbf{c}$*, namely* $((x_1 = c_1) \vee (c_1 = \star)) \wedge ... \wedge ((x_\ell = c_\ell) \vee (c_\ell = \star))$.

We start with a brief overview of the construction and then give the details. For a PRF with ℓ-bit input, the key-generation algorithm samples 2ℓ secrets from GLWE-hard distributions with small Euclidean norm $\{s_i^b \leftarrow \eta\}_{b\in\{0,1\},i\in[\ell]}$, places them in a chain of length ℓ and width 2, and uses the GGH15 methodology to encode the chain. The evaluation key consists of the resulting $\mathbf{A}_1$ matrix and the $\mathbf{D}$ matrices $\{\mathbf{D}_i^b\}_{b\in\{0,1\},i\in[\ell]}$.

The evaluation algorithm selects the path according to the input, computes the product of $\mathbf{D}$ matrices along the path $\prod_{i=1}^{\ell} \mathbf{D}_i^{x_i}$, then multiplies $\mathbf{A}_1$ on the left. The unrounded version of the output $\mathbf{A}_1 \prod_{i=1}^{\ell} \mathbf{D}_i^{x_i}$ is

[1] In the previous version, we mistakenly claimed that “the proofs of both constructions directly apply to the adaptive setting, i.e. the constrained circuit and input queries can be chosen adaptively, as long as the simulator learns the indicator of each input”.

close to $\prod_{i=1}^{\ell} s_i^{x_i}\mathbf{A}_{\ell+1}$, where "close" hides the cumulated error terms. Finally, the resulting subset product is rounded by p where $2 \le p < q$, $q/p > B$ with B being the maximum error bound. Rounding is required for correctness and security.

Construction 5.2 (Bit-fixing CHCPRFs). *We construct a function family $\mathcal{F} = \{f : \{0,1\}^\ell \to R_p^{1\times m}\}$ equipped with algorithms* $(\mathsf{Gen}, \mathsf{Constrain}, \mathsf{Eval})$ *and a set of vectors $\mathcal{C} = \{\mathbf{c} \in \{0,1,\star\}^\ell\}$:*

- $\mathsf{Gen}(1^\lambda)$ *takes the security parameter λ, samples parameters q, p, σ, m, $\mathbf{A}_{\ell+1} \leftarrow U(R_q^m)$, $\{(\mathbf{A}_i, \tau_i) \leftarrow$* $\mathsf{TrapSam}(R, 1^n, 1^m, q)\}_{i\in[\ell]}$. *Then, sample 2ℓ independent small secrets from GLWE-hard distributions $\{s_i^b \leftarrow \eta\}_{b\in\{0,1\}, i\in[\ell]}$. Next, encode the secrets as follows: first compute $\{\mathbf{Y}_i^b = s_i^b\mathbf{A}_{i+1} + \mathbf{E}_i^b, \mathbf{E}_i^b \leftarrow \chi^m\}_{i\in[\ell], b\in\{0,1\}}$, then sample $\{\mathbf{D}_i^b \leftarrow \mathsf{PreimgSam}(\mathbf{A}_i, \tau_i, \mathbf{Y}_i^b, \sigma)\}_{i\in[\ell], b\in\{0,1\}}$*

  Set $\mathsf{MSK} := (\{\mathbf{A}_i\}_{i\in[1,\ell+1]}, \{\tau_i\}_{i\in[\ell]}, \{s_i^b, \mathbf{D}_i^b\}_{i\in[\ell], b\in\{0,1\}})$.

- $\mathsf{Constrain}(\mathsf{MSK}, \mathbf{c})$ *takes* MSK *and the bit-matching vector $\mathbf{c}$, for $i \in [\ell]$, if $c_i \neq \star$ (i.e. specified as 0 or 1), replaces the original $s_i^{1-c_i}$ by a fresh $t_i^{1-c_i} \leftarrow \eta$, then updates the encodings on these secrets: $\mathbf{Y}_i^{1-c_i} = t_i^{1-c_i}\mathbf{A}_{i+1} + \mathbf{E'}_i^{1-c_i}, \mathbf{E'}_i^{1-c_i} \leftarrow \chi^m$, samples $\mathbf{D}_i^{1-c_i} \leftarrow \mathsf{PreimgSam}(\mathbf{A}_i, \tau_i, \mathbf{Y}_i^{1-c_i}, \sigma)$.*

  Set $\mathsf{CK} := (\mathbf{A}_1, \{\mathbf{D}_i^b\}_{i\in[\ell], b\in\{0,1\}})$.

- $\mathsf{Eval}(k, x)$ *takes the key $k = (\mathbf{A}_1, \{\mathbf{D}_i^b\}_{i\in[\ell], b\in\{0,1\}})$ and the input x, outputs $\left\lfloor \mathbf{A}_1 \prod_{i=1}^{\ell} \mathbf{D}_i^{x_i} \right\rceil_p$.*

Remark 5.3. *We occasionally call $i \in \{1, 2, ..., \ell\}$ "levels", from low to high.*

Setting of parameters. Parameters shall be set to ensure both correctness (i.e. the preservation of functionality over unconstrained inputs) and security. Note that the approximation factors of the underlying worst-case (general or ideal) lattices problems are inherently exponential in ℓ.

Specifically, for $R = \mathbb{Z}^{n\times n}$, set $\eta = \chi = D_{\mathbb{Z},\sigma}^{n\times n}$, $\gamma = D_{\mathbb{Z}^{nm},\sigma}^{1\times nm}$. The parameters are set to satisfy $m \ge 2\log q$ due to Lemma 2.5; $q/p > (\sigma \cdot m)^\ell$ due to Lemma 2.3 for the correctness of rounding; $0 < \sigma < q$, $\sigma = 2\sqrt{n \log q}$, $nq/\sigma < 2^{\lambda^{1-\epsilon}}$ due to Lemmas 2.6, 2.9, 2.11, and 2.13. An example setting of parameters: $p = 2$, $\epsilon = 1/2$, $q = (32\ell n^2 \log n)^\ell$, $\lambda = n = (\log q)^2$.

For $R = \mathbb{Z}[x]/(x^n + 1)$, n being a power of 2, set $\eta = \chi = D_{R,\sigma}$, $\gamma = D_{R^m,\sigma}^{1\times m}$. The parameters are set to satisfy $m \ge 2 \cdot n \log q$ due to Lemma 2.5; $q/p > (\sigma \cdot n^{3/4}m^{5/4})^\ell$ due to Lemma 2.3 for the correctness of rounding; $0 < \sigma < q$, $\sigma = 2\sqrt{n\log q}$, $nq/\sigma < 2^{\lambda^{1-\epsilon}}$ due to Lemmas 2.6, 2.10, 2.12, and 2.13. An example setting of parameters against the state-of-art ideal SVP algorithms [BS16, CDPR16, CDW17]: $p = 2$, $\epsilon = 0.5001$, $q = (70\ell n^3 \log n)^\ell$, $\lambda = n = (\log q)^{2.1}$.

Theorem 5.4. *Assuming* $\mathsf{GLWE}_{R,1,1,m,n,q,\eta,\chi}$, *Construction 5.2 satisfies Definition 4.2 where the evaluation queries are made adaptively after the constrained-key query.*

Functionality preservation on the unconstrained inputs. The constraining algorithm does not change any secrets on the unconstrained paths. So the functionality is perfectly preserved.

Security proof overview. The aim is to capture two properties: (1) pseudorandomness on the constrained inputs (2) the constrained key is indistinguishable from an obliviously sampled one.

We construct a simulator as follows: the simulator samples a key composed of $\mathbf{A}$ matrices from uniform distribution and $\mathbf{D}$ matrices from discrete-Gaussian distribution of small width. For the input-output pairs queried by the adversary, if the functionality is preserved on that point, then the simulator, knowing the input x, simply outputs the honest evaluation on the simulated key. If the input is constrained, it means at some level i, the secret $t_i^{x_i}$ in the constrained key is sampled independently from the original secret key $s_i^{x_i}$. Therefore the LWE instance $s_i^{x_i}\mathbf{A}_{i+1}+\mathbf{E}_i^{x_i}$, in the expression of the constrained output, provides an fresh random mask $\mathbf{U}$. The reduction moves from level $\ell+1$ to level 1. At level 1, by the result of [BLMR13], the rounded output on x is pseudorandom if $C(x)=0$.

Note that the evaluation algorithm only needs $\mathbf{A}_1$ but not the rest of the $\mathbf{A}$ matrices. However, in the analysis we assume all the $\mathbf{A}$ matrices are public.

Proof. The simulator samples all the $\{\mathbf{A}_j\}_{j\in[1,\ell+1]}$ matrices from random and $\{\mathbf{D}_i^b\}_{b\in\{0,1\},i\in[\ell]}$ from γ, outputs the constrained key $(\mathbf{A}_1,\{\mathbf{D}_i^b\}_{i\in[\ell],b\in\{0,1\}})$. To respond the input queries, the simulator picks $\{y^{(k)}\}_{k\in[t]}$ according to $\{d^{(k)}\}_{k\in[t]}$: if $d^{(k)}=1$ (i.e. the functionality is preserved on the constraint key at $x^{(k)}$), then outputs $y^{(k)}=\left\lfloor \mathbf{A}_1\prod_{i=1}^{\ell}\mathbf{D}_i^{x_i^{(k)}}\right\rceil_p$ (the honest evaluation on the simulated key); otherwise $y^{(k)}\leftarrow U(R_p^{1\times m})$.

The proof consists of two parts. The first part (Lemma 5.5) shows that the real distribution is indistinguishable from a semi-simulated one, where all the $\mathbf{D}$ matrices on the constrained key are sampled obliviously without knowing the constraint and the trapdoors of $\mathbf{A}$ matrices, and all the outputs are derived from the simulated constrained key. The second part (Lemma 5.6) argues that on every input x such that $C(x)=0$, the output is indistinguishable from random.

In the first part, we define intermediate hybrid distributions $\{\mathsf{H}_v\}_{v\in[0,\ell]}$. H_ℓ corresponds to the real constrained key and outputs, H_0 corresponds to the simulated constrained key and the semi-simulated outputs. For $v\in\{1,...,v\}$, the intermediate simulator in H_v knows the constraint, and the level $w^{(k)}\in\{v,...,\ell\}$ where the input $x^{(k)}$ starts to deviate from the constraint vector.

Descriptions of H_v, $v\in[0,\ell]$: The simulator Sim_v in H_v

1. Preprocessing. Before receiving any queries, Sim_v samples $\{(\mathbf{A}_j,\tau_j)\leftarrow\mathsf{TrapSam}(R,1^n,1^m,q)\}_{j\in[v]}$ with trapdoors, samples $\{\mathbf{A}_{j'}\leftarrow U(R_q^m)\}_{j'\in[v+1,\ell+1]}$ from uniform. Samples the GLWE secrets $\{s_i^b\leftarrow\eta\}_{b\in\{0,1\},i\in[v]}$ below level v.

2. Responding to the constrained key query. Given the constrained key query $\mathbf{c}$. For $i\in[v]$, if $c_i\neq\star$, samples $t_i^{1-c_i}\leftarrow\eta$. Then, for $b\in\{0,1\}, i\in[v]$, if t_i^b is sampled in the previous step, samples $\mathbf{Y}_i^b:=t_i^b\mathbf{A}_{i+1}+\mathbf{E'}_i^b$; otherwise, $\mathbf{Y}_i^b:=s_i^b\mathbf{A}_{i+1}+\mathbf{E}_i^b$.

   Next, Sim_v samples $\{\mathbf{D}_i^b\leftarrow\mathsf{PreimgSam}(\mathbf{A}_i,\tau_i,\mathbf{Y}_i^b,\sigma)\}_{b\in\{0,1\},i\in[v]}$ as the constrained-key below level v. Samples the rest of the $\mathbf{D}$ matrices obliviously $\{\mathbf{D}_i^b\leftarrow\gamma\}_{b\in\{0,1\},i\in[v+1,\ell]}$. Outputs $\mathbf{A}_1$, $\{\mathbf{D}_i^b\leftarrow\gamma\}_{b\in\{0,1\},i\in[\ell]}$ as the constrained key.

3. Responding to the evaluation queries. To simulate the outputs, the simulator maintains a list $\mathcal{U}$ of $\mathbf{U}$ matrices (to be specified) initiated empty. For $k\in[t]$, if the constraint is known to deviate in the path of $x^{(k)}_{[v+1,\ell]}$ from level $w^{(k)}\in[v+1,\ell]$, then compute $y^{(k)}$ as $\left\lfloor\prod_{i=1}^{v}s_i^{x_i^{(k)}}\mathbf{U}^{x^{(k)}_{[v+1,w]}}\prod_{j=w^{(k)}}^{\ell}\mathbf{D}_j^{x_j^{(k)}}\right\rceil_p$ — here $\mathbf{U}^{x^{(k)}_{[v+1,w]}}$ is indexed by $x^{(k)}_{[v+1,w]}$; if it is not in the list $\mathcal{U}$, sample $\mathbf{U}^{x^{(k)}_{[v+1,w]}}\leftarrow U(R_q^m)$, include it in $\mathcal{U}$; otherwise, reuse the one in $\mathcal{U}$. If $x^{(k)}$ has not deviated above level v, then $y^{(k)}=\left\lfloor\prod_{i=1}^{v}s_i^{x_i^{(k)}}\mathbf{A}_{v+1}\prod_{j=v+1}^{\ell}\mathbf{D}_j^{x_j^{(k)}}\right\rceil_p$.

Lemma 5.5. $\mathsf{H}_v\approx_c\mathsf{H}_{v-1}$, *for* $v\in\{\ell,...,1\}$.

Proof. The difference of H_v and H_{v-1} lies in the sampling of $\mathbf{D}_v^0$, $\mathbf{D}_v^1$ and the outputs $\{y^{(k)}\}$. We first analyze the difference of the outputs between H_v and H_{v-1} by classifying the input queries into 3 cases:

1. For input $x^{(k)}$ that matches the partial constraint vector $\mathbf{c}_{[v,\ell]}$, observe that $\left\lfloor \prod_{i=1}^{v-1} s_i^{x_i^{(k)}} \mathbf{A}_v \prod_{j=v}^{\ell} \mathbf{D}_j^{x_j^{(k)}} \right\rceil_p = \left\lfloor \prod_{i=1}^{v-1} s_i^{x_i^{(k)}} (s_v^{x_v^{(k)}} \mathbf{A}_{v+1} + \mathbf{E}_v^{x_v^{(k)}}) \prod_{j=v+1}^{\ell} \mathbf{D}_j^{x_j^{(k)}} \right\rceil_p \approx_s \left\lfloor \prod_{i=1}^{v} s_i^{x_i^{(k)}} \mathbf{A}_{v+1} \prod_{j=v+1}^{\ell} \mathbf{D}_j^{x_j^{(k)}} \right\rceil_p$, where $\approx_s$ is due to the small norm of $\prod_{i=1}^{v-1} s_i^{x_i^{(k)}} \mathbf{E}_v^{x_v^{(k)}} \prod_{j=v+1}^{\ell} \mathbf{D}_j^{x_j^{(k)}}$. Hence the output is statistically close in H_{v-1} and H_v.
2. For the input $x^{(k)}$ that is preserving above level v but deviated at level v, the fresh LWE secret $t_v^{x_v^{(k)}}$ sampled in the constrained key is independent from the original key $s_v^{x_v^{(k)}}$. So $s_v^{x_v^{(k)}} \mathbf{A}_{v+1} + \mathbf{E}_v^{x_v^{(k)}}$ and $t_v^{x_v^{(k)}} \mathbf{A}_{v+1} + \mathbf{E'}_v^{x_v^{(k)}}$ are treated as independent LWE instances w.r.t. $\mathbf{A}_{v+1}$.
3. For $x^{(k)}$ that has deviated above level v, the output can be written as
$$y^{(k)} = \left\lfloor \prod_{i=1}^{v} s_i^{x_i^{(k)}} \mathbf{U}^{x^{(k)}_{[v+1,w]}} \prod_{j=w^{(k)}}^{\ell} \mathbf{D}_j^{x_j^{(k)}} \right\rceil_p \approx_s \left\lfloor \prod_{i=1}^{v-1} s_i^{x_i^{(k)}} (s_v^{x_v^{(k)}} \mathbf{U}^{x^{(k)}_{[v+1,w]}} + \mathbf{E}') \prod_{j=w^{(k)}}^{\ell} \mathbf{D}_j^{x_j^{(k)}} \right\rceil_p,$$
where $\mathbf{U}^{x^{(k)}_{[v+1,w]}}$ is uniform by induction.

To summarize, there are less than $3(|\mathcal{U}|+1)$ matrices that are from the GLWE distribution in H_v and being uniform in H_{v-1}. The GLWE samples involve 3 independent secrets: s_v^0, s_v^1 and $t_v^{1-c_v}$ if $c_v \neq \star$. $t_v^{1-c_v}$ is only masked by $\mathbf{A}_{v+1}$; $\{s_v^b\}_{b\in\{0,1\}}$ are masked by $\mathbf{A}_{v+1}$ (in the constrained key and the outputs of cases (1) and (2)) and the uniform matrices in the list $\mathcal{U}$ (the outputs of case (3)); all the samples are associated with independently sampled noises.

If there is an attacker A' that distinguishes H_v and H_{v-1} with non-negligible probability ζ, we can build an attacker A who distinguishes (a subset among the $3(|\mathcal{U}|+1)$) GLWE samples

$$\{[\mathbf{A}_{v+1}, \mathbf{U}^1, \mathbf{U}^2, ..., \mathbf{U}^{|\mathcal{U}|}], [s_v^0, s_v^1, t_v^{1-c_i}]^T \cdot [\mathbf{A}_{v+1}, \mathbf{U}^1, \mathbf{U}^2, ..., \mathbf{U}^{|\mathcal{U}|}] + \tilde{\mathbf{E}}\},$$

where $\tilde{\mathbf{E}} \leftarrow \chi^{3\times(|\mathcal{U}|+1)m}$ from

$$\{U(R_q^{(|\mathcal{U}|+1)m} \times R_q^{3\times(|\mathcal{U}|+1)m})\}$$

To do so, once A obtains the samples, it places the samples under mask $\mathbf{A}_{v+1}$ in the constrained key and the outputs of cases (1) and (2); places the samples under masks $\mathbf{U}^1, ..., \mathbf{U}^{|\mathcal{U}|}$ in the outputs of cases (3). Then samples $\{\mathbf{A}_j\}_{j\in[v]}$ with trapdoors, GLWE secrets $\{s_i^b \leftarrow \eta\}_{b\in\{0,1\}, i\in[v]}$. Then samples $\{\mathbf{D}_i^b \leftarrow \mathsf{PreimgSam}(\mathbf{A}_i, \tau_i, \mathbf{Y}_i^b, \sigma)\}_{b\in\{0,1\}, i\in[v]}$ as the constrained-key below level v. Samples the rest of the $\mathbf{D}$ matrices obliviously $\{\mathbf{D}_i^b \leftarrow \gamma\}_{b\in\{0,1\}, i\in[v+1,\ell]}$.

With these matrices the attacker A is able to simulate the outputs, sends the outputs and constrained key to A'. If the samples are from GLWE, then it corresponds to H_v; if the samples are uniform, then the matrices $\{\mathbf{D}_v^b\}_{b\in\{0,1\}}$ sampled via $\{\mathbf{D}_v^b \leftarrow \mathsf{PreimgSam}(\mathbf{A}_v, \tau_v, \mathbf{Y}_v^b, \sigma)\}_{b\in\{0,1\}}$ are statistically close to the obliviously sampled ones due to Lemma 2.6, so it is statistically close to H_{v-1}. Hence A breaks GLWE with probability more than $\zeta/(3(t+1))$, which contradicts to Lemma 2.11. □

Lemma 5.6. *If $C(x^{(k)}) = 0$, then the output $y^{(k)}$ in H_0 is pseudorandom.*

Proof. A constrained output $y^{(k)}$ can be expressed as $\left\lfloor \mathbf{U}^{x^{(k)}_{[1,w]}} \prod_{j=w^{(k)}}^{\ell} \mathbf{D}_j^{x_j^{(k)}} \right\rceil_p$, where the secret $\mathbf{U}^{(k)}_{[1,w^{(k)}]}$ is uniform; the public $\mathbf{D}$ matrices are sampled from discrete-Gaussian distribution γ. By Lemma 2.13 $y^{(k)}$ is pseudorandom. □

The proof completes by combining Lemma 5.5 and Lemma 5.6. □

5.2 Constraint-hiding for NC^1 circuits

Next we present the CHCPRF for NC^1 circuit constraints. For circuits of depth d, use Barrington's Theorem [Bar86] to convert them into a set of oblivious branching program $\{\mathsf{BP}\}$ with the same index-to-input map $\iota : [z] \to [\ell]$, the same w-cycle $\mathbf{P}$ that represents the 0 output (by default $w = 5$). Let $\{\mathbf{B}_i^b \in \{0,1\}^{w\times w}\}_{i\in[z],b\in\{0,1\}}$ be the permutation matrices in each BP.

The master secret key for the CHCPRF consists of $2z$ secrets from GLWE-hard distributions η over R with small Euclidean norm $\{s_i^b \leftarrow \eta\}_{b\in\{0,1\},i\in[z]}$, together with a vector $\mathbf{J} \in R^{1\times w}$. To generate an evaluation key, in the normal setting, let $\mathbf{S}_i^b := \mathbf{I}^{w\times w} \otimes s_i^b \in \{0,1\}^{w\times w} \otimes_R R$; in the constrained setting for a constraint recognized by BP, let $\mathbf{S}_i^b := \mathbf{B}_i^b \otimes s_i^b \in \{0,1\}^{w\times w} \otimes_R R$. For both settings, places $\{\mathbf{S}_i^b\}_{b\in\{0,1\},i\in[z]}$ in a chain of length z and width 2, places $\mathbf{J}$ on the left end of the chain, and uses the GGH15 methodology to encode the chain. The encoding of $\mathbf{J}$ is merged into $\mathbf{A}_1$, denote the resulting matrix as $\mathbf{A}_J$. The evaluation key consists of $\mathbf{A}_J$ and the matrices $\{\mathbf{D}_i^b\}_{b\in\{0,1\},i\in[z]}$.

To evaluate on x, output $\left\lfloor \mathbf{A}_J \prod_{i=1}^{z} \mathbf{D}_i^{x_{\iota(i)}} \right\rceil_p$. To elaborate the functionality, for x s.t. $C(x) = 1$,

$$\left\lfloor \mathbf{A}_J \prod_{i=1}^{z} \mathbf{D}_i^{x_{\iota(i)}} \right\rceil_p \approx_s \left\lfloor \mathbf{J}(\mathbf{I}^{w\times w} \otimes \prod_{i=1}^{z} s_i^{x_{\iota(i)}})\mathbf{A}_{z+1} \right\rceil_p ;$$

for x s.t. $C(x) = 0$,

$$\left\lfloor \mathbf{A}_J \prod_{i=1}^{z} \mathbf{D}_i^{x_{\iota(i)}} \right\rceil_p \approx_s \left\lfloor \mathbf{J}(\mathbf{P} \otimes \prod_{i=1}^{z} s_i^{x_{\iota(i)}})\mathbf{A}_{z+1} \right\rceil_p .$$

As a reminder, the permutation matrix $\mathbf{P}$ that represent the w-cycle is not a secret to the construction, so the use of the left-bookend $\mathbf{J}$ is essential for security.

Note that our construction inherently reveals the length of the branching program which determines the upper bound of the depth of the constraint circuit.

Construction 5.7 (CHCPRFs for NC^1 circuits). *We construct a function family $\mathcal{F} = \{f : \{0,1\}^\ell \to R_p^{1\times wm}\}$ equipped with 3 algorithms* ($\mathsf{Gen}, \mathsf{Constrain}, \mathsf{Eval}$)*, associated with a set of oblivious branching programs $\{\mathsf{BP}\}$ of length z obtained by applying Lemma 2.2 on all the NC^1 circuits.*

- $\mathsf{Gen}(1^\lambda)$ *samples parameters q, p, σ, m, z, $\{(\mathbf{A}_i, \tau_i) \leftarrow \mathsf{TrapSam}(R^{w\times w}, 1^n, 1^m, q)\}_{i\in[z]}$, $\mathbf{A}_{z+1} \leftarrow U(R_q^{w\times wm})$. Samples $2z$ independent small secrets from GLWE-hard distributions $\{s_i^b \leftarrow \eta\}_{b\in\{0,1\},i\in[z]}$, sets the secret matrices to be $\mathbf{S}_i^b = \mathbf{I}^{w\times w} \otimes s_i^b$.*

  Next, encode the secrets as follows: first compute $\{\mathbf{Y}_i^b = \mathbf{S}_i^b\mathbf{A}_{i+1} + \mathbf{E}_i^b, \mathbf{E}_i^b \leftarrow \chi^{w\times wm}\}_{i\in[z],b\in\{0,1\}}$; then, sample $\{\mathbf{D}_i^b \leftarrow \mathsf{PreimgSam}(\mathbf{A}_i, \tau_i, \mathbf{Y}_i^b, \sigma)\}_{i\in[z],b\in\{0,1\}}$. Additionally, sample a small secret $\mathbf{J} \leftarrow \eta^{1\times w}$ as the left-bookend. Compute $\mathbf{A}_J := \mathbf{J}\mathbf{A}_1 + \mathbf{E}_J$ where $\mathbf{E}_J \leftarrow \chi^{1\times wm}$.

  Set $\mathsf{MSK} := (\{\mathbf{A}_i\}_{i\in[1,z+1]}, \{\tau_i\}_{i\in[z]}, \mathbf{A}_J, \{s_i^b, \mathbf{D}_i^b\}_{i\in[z],b\in\{0,1\}})$.

- $\mathsf{Constrain}(\mathsf{MSK}, \mathsf{BP})$ *takes* MSK*, and a matrix branching program* $\mathsf{BP} = \{\mathbf{B}_i^b \in R^{w\times w}\}_{i\in[z],b\in\{0,1\}}$*.* *For* $i \in [z]$*,* $b \in \{0,1\}$*, compute* $\mathbf{Y}_i^b = \left(\mathbf{B}_i^b \otimes s_i^b\right)\mathbf{A}_{i+1} + \mathbf{E}'^b_i, \mathbf{E}'^b_i \leftarrow \chi^{w\times wm}$*, samples* $\mathbf{D}_i^b \leftarrow \mathsf{PreimgSam}(\mathbf{A}_i, \tau_i, \mathbf{Y}_i^b, \sigma)$*.*

  Set the constrained key $\mathsf{CK} := (\mathbf{A}_J, \{\mathbf{D}_i^b\}_{i\in[z],b\in\{0,1\}})$*.*

- $\mathsf{Eval}(k, x)$ *takes the input* x *and the key* $k = (\mathbf{A}_J, \{\mathbf{D}_i^b\}_{i\in[z],b\in\{0,1\}})$*, outputs* $\left\lfloor \mathbf{A}_J \prod_{i=1}^{z} \mathbf{D}_i^{x_{\iota(i)}} \right\rceil_p$*.*

Setting of parameters. Settings of the distributions and their dimensions: For $R = \mathbb{Z}^{n\times n}$, set $\eta = \chi = D_{\mathbb{Z},\sigma}^{n\times n}$, $\gamma = D_{\mathbb{Z}^{nwm},\sigma}^{1\times nwm}$. For $R = \mathbb{Z}[x]/(x^n+1)$, n being a power of 2, set $\eta = \chi = D_{R,\sigma}$, $\gamma = D_{R^{wm},\sigma}^{1\times wm}$. The restriction on the parameters are analogous to the settings in the bit-fixing construction.

Theorem 5.8. *Assuming* $\mathsf{GLWE}_{R,1,1,w^2m,n,q,\eta,\chi}$*, Construction 5.7 satisfies Definition 4.2 where the evaluation queries are made adaptively after the constrained-key query.*

Proof overview. The simulation algorithm and the overall proof strategy are similar to the ones for the bit-fixing constraints. Namely, we close the trapdoors for $\mathbf{A}$ matrices from level z to level 1. Within each level v, there are several GLWE instance associated with the public matrix $\mathbf{A}_{v+1}$ whose trapdoor is closed in the previous hybrid. The additional complexity comes from dealing with secrets with permutation matrix structures. This is handled by the new GLWE-friendly packages from Section 3.

Proof. The simulator samples $\{\mathbf{A}_i \leftarrow U(R_q^{w\times wm})\}_{i\in[1,z+1]}$, and $\{\mathbf{D}_i^b \leftarrow \gamma\}_{b\in\{0,1\},i\in[z]}$. It also samples $\mathbf{J} \leftarrow \eta^{1\times w}$, computes $\mathbf{A}_J := \mathbf{J}\mathbf{A}_1 + \mathbf{E}_J$ where $\mathbf{E}_J \leftarrow \chi^{1\times wm}$. Outputs the constrained key $(\mathbf{A}_J, \{\mathbf{D}_i^b\}_{i\in[z],b\in\{0,1\}})$. The simulator responds the input queries by picking $\{y^{(k)}\}_{k\in[t]}$ according to $\{d^{(k)}\}_{k\in[t]}$: if $d^{(k)} = 1$, then outputs $y^{(k)} = \left\lfloor \mathbf{A}_J \prod_{i=1}^{z} \mathbf{D}_i^{x_i^{(k)}} \right\rceil_p$; otherwise $y^{(k)} \leftarrow U(R_p^{1\times wm})$.

The proof consists of two parts. The first part (Lemma 5.9) shows that the real distribution is indistinguishable from a semi-simulated one, where the $\mathbf{D}$ matrices in the constrained key are sampled without knowing the constraint or the trapdoors of the $\mathbf{A}$ matrices, the outputs are computed from the constrained key. The second part (Lemma 5.10) argues that on every input x such that $C(x) = 0$, the output is indistinguishable from random.

In the first part, we define intermediate hybrid distributions $\{\mathsf{H}_v\}_{v\in[0,z]}$. H_z corresponds to the real constrained key and output distributions, H_0 corresponds to the simulated constrained key and the semi-simulated outputs. The simulators in H_z, H_{z-1}, ..., H_1 know the full description of the constraint $\mathsf{BP} = \{\mathbf{B}_i^b\}_{i\in[z],b\in\{0,1\}}$; the simulator in H_0 does not have to know the constraint.

Descriptions of H_v, $v \in [0, z]$: The simulator Sim_v in H_v proceeds as follows:

1. Preprocessing. Before receiving any queries, Sim_v samples $\{(\mathbf{A}_j, \tau_j) \leftarrow \mathsf{TrapSam}(R^{w\times w}, 1^n, 1^m, q)\}_{j\in[v]}$ with trapdoors; samples $\{\mathbf{A}_{j'} \leftarrow U(R_q^{w\times wm})\}_{j'\in[v+1,z+1]}$ uniformly random. Samples the GLWE secrets $\{s_i^b \leftarrow \eta\}_{b\in\{0,1\},i\in[v]}$ below level v; and a bookend vector $\mathbf{J} \leftarrow \eta^{1\times w}$. Computes $\mathbf{A}_J := \mathbf{J}\mathbf{A}_1 + \mathbf{E}_J$.

2. Responding to the constrained key query. Given a constraint $\{\mathbf{B}_i^b\}_{b\in\{0,1\},i\in[z]}$, Sim_v samples $\mathbf{Y}_i^b := \left(\mathbf{B}_i^b \otimes s_i^b\right)\mathbf{A}_{i+1} + \mathbf{E}'^b_i$. Then, it computes $\{\mathbf{D}_i^b \leftarrow \mathsf{PreimgSam}(\mathbf{A}_i, \tau_i, \mathbf{Y}_i^b, \sigma)\}_{b\in\{0,1\},i\in[v]}$ as the constrained-key below level v. Samples the rest of the $\mathbf{D}$ matrices obliviously $\{\mathbf{D}_i^b \leftarrow \gamma\}_{b\in\{0,1\},i\in[v+1,z]}$.

3. Responding to the evaluation queries[2]. For $k \in [t]$, computes $y^{(k)}$ as

$$y^{(k)} = \left\lfloor \mathbf{J} \times \left(\left(\prod_{j=v+1}^{z} \mathbf{B}_j^{x^{(k)}_{\iota(j)}} \right)^{-1} \otimes \prod_{i=1}^{v} s_i^{x^{(k)}_{\iota(i)}} \right) \times \mathbf{A}_{v+1} \prod_{j=v+1}^{z} \mathbf{D}_j^{x^{(k)}_{\iota(j)}} \right\rceil_p \tag{2}$$

Lemma 5.9. $\mathsf{H}_v \approx_c \mathsf{H}_{v-1}$, *for* $v \in [z]$.

Proof. The difference of H_v and H_{v-1} lies in the sampling of $\mathbf{D}_v^0, \mathbf{D}_v^1$ and the outputs $\{y^{(k)}\}$. We first examine the outputs. For $k \in [t]$, we express the output $y^{(k)}$, starting from the expression in H_v to the one in H_{v-1}:

$$\begin{aligned}
y^{(k)} &= \left\lfloor \mathbf{J} \times \left(\left(\prod_{j=v+1}^{z} \mathbf{B}_j^{x^{(k)}_{\iota(j)}} \right)^{-1} \otimes \prod_{i=1}^{v} s_i^{x^{(k)}_{\iota(i)}} \right) \times \mathbf{A}_{v+1} \prod_{j=v+1}^{z} \mathbf{D}_j^{x^{(k)}_{\iota(j)}} \right\rceil_p \\
&= \left\lfloor \mathbf{J} \times \left(\left(\prod_{j=v+1}^{z} \mathbf{B}_j^{x^{(k)}_{\iota(j)}} \right)^{-1} \otimes \prod_{i=1}^{v-1} s_i^{x^{(k)}_{\iota(i)}} \right) \times \left(\mathbf{I}^{w \times w} \otimes s_v^{x^{(k)}_{\iota(v)}} \right) \mathbf{A}_{v+1} \prod_{j=v+1}^{z} \mathbf{D}_j^{x^{(k)}_{\iota(j)}} \right\rceil_p \\
&= \left\lfloor \mathbf{J} \times \left(\left(\prod_{j=v+1}^{z} \mathbf{B}_j^{x^{(k)}_{\iota(j)}} \right)^{-1} \otimes \prod_{i=1}^{v-1} s_i^{x^{(k)}_{\iota(i)}} \right) \times \left(\mathbf{B}_v^{x^{(k)}_{\iota(v)}} \otimes 1_R \right)^{-1} \times \left(\mathbf{B}_v^{x^{(k)}_{\iota(v)}} \otimes s_v^{x^{(k)}_{\iota(v)}} \right) \mathbf{A}_{v+1} \prod_{j=v+1}^{z} \mathbf{D}_j^{x^{(k)}_{\iota(j)}} \right\rceil_p \\
&\approx_s \left\lfloor \mathbf{J} \times \left(\left(\prod_{j=v}^{z} \mathbf{B}_j^{x^{(k)}_{\iota(j)}} \right)^{-1} \otimes \prod_{i=1}^{v-1} s_i^{x^{(k)}_{\iota(i)}} \right) \times \left[\left(\mathbf{B}_v^{x^{(k)}_{\iota(v)}} \otimes s_v^{x^{(k)}_{\iota(v)}} \right) \mathbf{A}_{v+1} + \mathbf{E}' \right] \prod_{j=v+1}^{z} \mathbf{D}_j^{x^{(k)}_{\iota(j)}} \right\rceil_p \\
&= \left\lfloor \mathbf{J} \times \left(\left(\prod_{j=v}^{z} \mathbf{B}_j^{x^{(k)}_{\iota(j)}} \right)^{-1} \otimes \prod_{i=1}^{v-1} s_i^{x^{(k)}_{\iota(i)}} \right) \times \mathbf{Y}_v^{x^{(k)}_{\iota(v)}} \prod_{j=v+1}^{z} \mathbf{D}_j^{x^{(k)}_{\iota(j)}} \right\rceil_p \\
&= \left\lfloor \mathbf{J} \times \left(\left(\prod_{j=v}^{z} \mathbf{B}_j^{x^{(k)}_{\iota(j)}} \right)^{-1} \otimes \prod_{i=1}^{v-1} s_i^{x^{(k)}_{\iota(i)}} \right) \times \mathbf{A}_v \prod_{j=v}^{z} \mathbf{D}_j^{x^{(k)}_{\iota(j)}} \right\rceil_p
\end{aligned} \tag{3}$$

where $\mathbf{Y}_v^{x^{(k)}_{\iota(v)}} = \mathbf{A}_v \mathbf{D}_v^{x^{(k)}_{\iota(v)}}$. The correctness of this equation is a routine check. The implication is that the difference of H_v and H_{v-1} fully lies in the sampling of $\mathbf{Y}_v^0, \mathbf{Y}_v^1$ (being GLWE samples in H_v or uniform in H_{v-1}) and their preimages $\mathbf{D}_v^0, \mathbf{D}_v^1$ sampled by the trapdoor of $\mathbf{A}_v$.

Formally, suppose there is an attacker A' that distinguishes H_v and H_{v-1} with non-negligible probability ζ, we can build an attacker A who distinguishes:

$$\mathbf{A}_{v+1}, \{\mathbf{Y}_v^b = \left(\mathbf{B}_v^b \otimes s_v^b \right) \mathbf{A}_{v+1} + \mathbf{E}_v^b\}_{b \in \{0,1\}}$$

from

$$\{U(R_q^{w \times wm} \times R_q^{w \times wm} \times R_q^{w \times wm})\}$$

To do so, once A obtains the challenge samples, it samples $\{\mathbf{A}_j\}_{j \in [v]}$ with trapdoors, and produces the preimages $\{\mathbf{D}_v^b \leftarrow \mathsf{PreimgSam}(\mathbf{A}_v, \tau_v, \mathbf{Y}_v^b, \sigma)\}_{b \in \{0,1\}}$. Then places $\mathbf{A}_{v+1}$, $\mathbf{Y}_v^0$, $\mathbf{Y}_v^1$, $\mathbf{D}_v^0$, $\mathbf{D}_v^1$ in the constrained key and the outputs. It further samples GLWE secrets $\{s_i^b \leftarrow \eta\}_{b \in \{0,1\}, i \in [v]}$, $\{\mathbf{D}_i^b \leftarrow \mathsf{PreimgSam}(\mathbf{A}_i, \tau_i, \mathbf{Y}_i^b, \sigma)\}_{b \in \{0,1\}, i \in [v]}$ as the constrained-key below level v. Samples the rest of the $\mathbf{D}$ matrices obliviously $\{\mathbf{D}_i^b \leftarrow \gamma\}_{b \in \{0,1\}, i \in [v+1,z]}$.

With these matrices the attacker A is able to simulate the rest of the outputs, send the outputs and constrained key to A'. If the samples are from GLWE, then it corresponds to H_v; if the samples are uniform, then the matrices $\{\mathbf{D}_v^b\}_{b \in \{0,1\}}$ sampled via $\{\mathbf{D}_v^b \leftarrow \mathsf{PreimgSam}(\mathbf{A}_v, \tau_v, \mathbf{Y}_v^b, \sigma)\}_{b \in \{0,1\}}$ are statistically close to the obliviously sampled ones due to Lemma 2.6, so it is statistically close to H_{v-1}. Hence A breaks GLWE with probability more than $\zeta/2$, which contradicts to Lemma 3.1. □

[2]Here, the correct simulation of the output values depends on the branching program $\mathsf{BP} = \{\mathbf{B}_i^b\}_{i \in [z], b \in \{0,1\}}$ that recognizes the constraint. Such a proof strategy is limited to work only when the constrained key query is made first, followed by the adaptively chosen PRF evaluation queries. Similarly for the proof of Theorem 5.4.

Lemma 5.10. *If $C(x^{(k)}) = 0$, then the output $y^{(k)}$ in H_0 is pseudorandom.*

Proof. Following Eqn. (2), a constrained output $y^{(k)}$ in H_0 can be expressed as:

$$y^{(k)} = \left\lfloor \mathbf{J} \times \left(\mathbf{P}^{-1} \otimes 1_R\right) \times \mathbf{A}_1 \prod_{j=1}^{z} \mathbf{D}_j^{x^{(k)}_{\iota(j)}} \right\rceil_p \approx_s \left\lfloor \left(\mathbf{J} \times \left(\mathbf{P}^{-1} \otimes 1_R\right) \times \mathbf{A}_1 + \mathbf{E}\right) \prod_{j=1}^{z} \mathbf{D}_j^{x^{(k)}_{\iota(j)}} \right\rceil_p \tag{4}$$

For $\mathbf{J}\mathbf{A}_1 + \mathbf{E}_J$ as part of the constrained key, $\mathbf{J} \times \left(\mathbf{P}^{-1} \otimes 1_R\right) \times \mathbf{A}_1 + \mathbf{E}$ as part of the constrained output $y^{(k)}$, $(\mathbf{J}\mathbf{A}_1 + \mathbf{E}_J, \mathbf{J} \times \left(\mathbf{P}^{-1} \otimes 1_R\right) \times \mathbf{A}_1 + \mathbf{E})$ is indistinguishable from $U(R_q^{1\times wm}, R_q^{1\times wm})$ due to Lemma 3.2. This means each constrained output $y^{(k)}$ is indistinguishable from $\left\lfloor \mathbf{U} \prod_{j=1}^{z} \mathbf{D}_j^{x^{(k)}_{\iota(j)}} \right\rceil_p$ where $\mathbf{U} \leftarrow U(R_q^{1\times wm})$. Hence $y^{(k)}$ is pseudorandom if $C(x^{(k)}) = 0$ due to Lemma 2.13. □

The proof completes by combining the Lemmas 5.9 and 5.10. □

5.3 A variant of the NC^1 CHCPRF construction

We would like to mention a variant of the NC^1 CHCPRF construction that only differs in one place from Construction 5.7. The change preserves all the security properties from Construction 5.7, and makes it much easier to explain the underlying structural properties. The variant is due to Vinod Vaikuntanathan and Hoeteck Wee. We thank them for allowing us to include this variant in the current paper.

For $w \in \mathbb{N}$. Let $\mathbf{e}_i \in \{0,1\}^{1\times w}$ be the i^{th} unit vector, where the i^{th} bit is 1, the other bits are 0s. For a matrix $\mathbf{X} \in R^{w\times *}$, let $\mathbf{X}^{(i)} \in R^{1\times *}$ denote the i^{th} row of $\mathbf{X}$, i.e. $\mathbf{X}^{(i)} = (\mathbf{e}_i \otimes 1_R) \cdot \mathbf{X}$.

Following the notations of a matrix branching program from Defintion 2.1. For a width-w ($w \geq 5$) matrix branching program obtained from Barrington's theorem, the matrix $\mathbf{P} \in \{0,1\}^{w\times w}$ representing the w-cycle satisfies the following properties:

1. There exists an integer $\kappa \in \{2, ..., w\}$ s.t. $\mathbf{e}_1 \cdot \mathbf{P} = \mathbf{e}_\kappa$;
2. There exists an integer $\theta \in \{2, ..., w\}$ s.t. $\mathbf{e}_\theta \cdot \mathbf{P} = \mathbf{e}_1$.

Wlog we assume $\kappa = 2$ and $\theta = 3$ in the rest of this section.

Construction 5.11 (A variant of the CHCPRF for NC^1 circuits)**.** *The construction is the same as Construction 5.7, except that the left-bookend vector $\mathbf{J} \in R^{1\times w}$ is chosen as $\mathbf{e}_1 \otimes 1_R$ instead of a random vector from Gaussian distribution, and $\mathbf{A}_J$ is computed as $\mathbf{J} \cdot \mathbf{A}_1 = \mathbf{A}_1^{(1)}$.*

We define $\left\lfloor \prod_{i=1}^{z} s_i^{x_{\iota(i)}} \mathbf{A}_{z+1}^{(1)} \right\rceil_p$ as the original PRF output value.

To elaborate the functionality of the constrained key. For every input x such that $C(x) = 1$, the constrained key evaluates to

$$\left\lfloor \mathbf{A}_J \prod_{i=1}^{z} \mathbf{D}_i^{x_{\iota(i)}} \right\rceil_p \approx_s \left\lfloor \mathbf{J}(\mathbf{I}^{w\times w} \otimes \prod_{i=1}^{z} s_i^{x_{\iota(i)}})\mathbf{A}_{z+1} \right\rceil_p = \left\lfloor (\mathbf{e}_1 \otimes \prod_{i=1}^{z} s_i^{x_{\iota(i)}})\mathbf{A}_{z+1} \right\rceil_p = \left\lfloor \prod_{i=1}^{z} s_i^{x_{\iota(i)}} \mathbf{A}_{z+1}^{(1)} \right\rceil_p \tag{5}$$

For every input x s.t. $C(x) = 0$, the constrained key evaluates to

$$\left\lfloor \mathbf{A}_J \prod_{i=1}^{z} \mathbf{D}_i^{x_{\iota(i)}} \right\rceil_p \approx_s \left\lfloor \mathbf{J}(\mathbf{P} \otimes \prod_{i=1}^{z} s_i^{x_{\iota(i)}})\mathbf{A}_{z+1} \right\rceil_p = \left\lfloor (\mathbf{e}_2 \otimes \prod_{i=1}^{z} s_i^{x_{\iota(i)}})\mathbf{A}_{z+1} \right\rceil_p = \left\lfloor \prod_{i=1}^{z} s_i^{x_{\iota(i)}} \mathbf{A}_{z+1}^{(2)} \right\rceil_p \quad (6)$$

We claim that Construction 5.11 satisfies the partially adaptive security notion where the constrained-key is chosen before all the evaluation queries (i.e. the same security level for Construction 5.7 proven in Theorem 5.8). The proof for constraint-hiding is the same. To verify the proof for the pseudorandomness of PRF evaluation, observe that for every input query x such that $C(x) = 0$, the original PRF output value can be computed from the constrained key as follows:

$$\left\lfloor \prod_{i=1}^{z} s_i^{x_{\iota(i)}} \mathbf{A}_{z+1}^{(1)} \right\rceil_p = \left\lfloor \left((\mathbf{e}_3\mathbf{P}) \otimes \prod_{i=1}^{z} s_i^{x_{\iota(i)}} \right) \mathbf{A}_{z+1} \right\rceil_p \approx_s \left\lfloor (\mathbf{e}_3 \otimes 1_R) \cdot \mathbf{A}_1 \prod_{i=1}^{z} \mathbf{D}_i^{x_{\iota(i)}} \right\rceil_p = \left\lfloor \mathbf{A}_1^{(3)} \prod_{i=1}^{z} \mathbf{D}_i^{x_{\iota(i)}} \right\rceil_p \quad (7)$$

Then trigger Lemma 2.13 by treating $\mathbf{A}_1^{(3)}$ as the secret key of the BLMR PRF.

The analysis of the stronger adaptive security. To what extent can we argue about the full-fledged adaptive security of Construction 5.11 where the input queries can be made before the constrained-key query? We claim that under the polynomial hardness assumption on GLWE, Construction 5.11 satisfies a variant of adaptive security where $O(\log \lambda)$ input queries can be made before the constrained-key query.

Theorem 5.12. *Assuming* $\mathsf{GLWE}_{R,1,1,w^2m,n,q,\eta,\chi}$*, Construction 5.11 satisfies a variant of Definition 4.2 where at most* $O(\log \lambda)$ *input queries can be made before the constrained-key query, and arbitrarily polynomially many input queries can be made after.*

Proof. The simulator samples $\{\mathbf{A}_i \leftarrow U(R_q^{w\times wm})\}_{i\in[1,z+1]}$, and $\{\mathbf{D}_i^b \leftarrow \gamma\}_{b\in\{0,1\},i\in[z]}$. It then sets $\mathbf{J} = \mathbf{e}_1 \otimes 1_R$, computes $\mathbf{A}_J := \mathbf{J}\mathbf{A}_1$. The simulator responds the input queries (no matter if they arrives before or after the constrained key query) by picking $\{y^{(k)}\}_{k\in[t]}$ according to $\{d^{(k)}\}_{k\in[t]}$: if $d^{(k)} = 1$, then outputs $y^{(k)} = \left\lfloor \mathbf{A}_J \prod_{i=1}^{z} \mathbf{D}_i^{x_i^{(k)}} \right\rceil_p$; otherwise $y^{(k)} \leftarrow U(R_p^{1\times wm})$. Once given a constrained key query, the simulator outputs $(\mathbf{A}_J, \{\mathbf{D}_i^b\}_{i\in[z],b\in\{0,1\}})$.

We do require different descriptions of the intermediate hybrid distributions $\{\mathsf{H}_v\}_{v\in[0,z-1]}$ from those in the proof of Theorem 5.8. Looking ahead, the reduction from GLWE will try to produce H_v or H_{v-1}, depending on whether the underlying GLWE challenge is real or random.

Descriptions of H_v, $v \in [0, z-1]$: The intermediate simulator Sim_v in H_v proceeds as follows

1. The preprocessing phase. Before any input or constrained key queries arrive, Sim_v samples $\{(\mathbf{A}_j, \tau_j) \leftarrow \mathsf{TrapSam}(R^{w\times w}, 1^n, 1^m, q)\}_{j\in[v]}$ with trapdoors; samples $\{\mathbf{A}_{j'} \leftarrow U(R_q^{w\times wm})\}_{j'\in[v+1,z+1]}$ uniformly random. Set $\mathbf{A}_J := \mathbf{J}\mathbf{A}_1$. Then samples the GLWE secrets $\{s_i^b \leftarrow \eta\}_{b\in\{0,1\},i\in[v]}$; samples $\{\mathbf{D}_i^b \leftarrow \gamma\}_{b\in\{0,1\},i\in[v+1,z]}$.

2. To simulate the PRF evaluation queries before the constrained key query. For $k \in [t]$, guess a random row number $\rho(k) \in [w]$, computes $y^{(k)}$ as

$$y^{(k)} = \left\lfloor \prod_{i=1}^{v} s_i^{x_{\iota(i)}^{(k)}} \cdot \mathbf{A}_{v+1}^{(\rho(k))} \prod_{j=v+1}^{z} \mathbf{D}_j^{x_{\iota(j)}^{(k)}} \right\rceil_p \quad (8)$$

3. Given the constrained key query (represented by a branching program $\{\mathbf{B}_i^b\}_{i\in[z]}$), Sim_v checks for every input $x \in \{x^{(1)}, ..., x^{(t)}\}$ if

$$\mathbf{e}_1 \cdot \left(\prod_{j=v+1}^{z} \mathbf{B}_j^{x^{(k)}_{\iota(j)}}\right)^{-1} = \mathbf{e}_{\rho(k)}. \tag{9}$$

If any of the t guesses is wrong, Sim_v aborts; if all the guesses are correct, the simulation proceeds.

4. To simulate the constrained key given part of the branching program $\{\mathbf{B}_i^b\}_{i\in[v]}$, Sim_v computes $\mathbf{Y}_i^b := \left(\mathbf{B}_i^b \otimes s_i^b\right)\mathbf{A}_{i+1} + \mathbf{E'}_i^b$, for $i \in [v]$. Then computes $\{\mathbf{D}_i^b \leftarrow \mathsf{PreimgSam}(\mathbf{A}_i, \tau_i, \mathbf{Y}_i^b, \sigma)\}_{b\in\{0,1\}, i\in[v]}$. Outputs $\mathbf{A}_J$, $\{\mathbf{D}_i^b\}_{b\in\{0,1\}, i\in[z]}$.

5. To simulate the PRF evaluation queries after the constrained key query. For $k \in [t+1, t']$, Sim_v computes $y^{(k)}$ as

$$y^{(k)} = \left\lfloor \mathbf{J} \times \left(\left(\prod_{j=v+1}^{z} \mathbf{B}_j^{x^{(k)}_{\iota(j)}}\right)^{-1} \otimes \prod_{i=1}^{v} s_i^{x^{(k)}_{\iota(i)}} \right) \times \mathbf{A}_{v+1} \prod_{j=v+1}^{z} \mathbf{D}_j^{x^{(k)}_{\iota(j)}} \right\rceil_p \tag{10}$$

We argue that when there are less than $O(\log\lambda)$ many evaluation queries before the constrained key query, the reduction of GLWE to $\mathsf{H}_v \approx_c \mathsf{H}_{v-1}$ only loses a polynomial factor of success probability.

Lemma 5.13. *For $v \in [z]$, if there are less than $O(\log\lambda)$ many evaluation queries before the constrained key query, then $\mathsf{H}_v \approx_c \mathsf{H}_{v-1}$ assuming the polynomial hardness of GLWE.*

Proof. The difference from the proof of Lemma 5.9 lies in the treatment of pre-constrained-key input queries in Eqn. (8). Since the branching program description of the constrained key is not known at this moment, we cannot produce an expression like Eqn. (10). Nevertheless, Eqn. (10) can be written as

$$y^{(k)} = \left\lfloor \prod_{i=1}^{v} s_i^{x^{(k)}_{\iota(i)}} \cdot \left(\mathbf{e}_1 \left(\prod_{j=v+1}^{z} \mathbf{B}_j^{x^{(k)}_{\iota(j)}}\right)^{-1} \right) \otimes 1_R \cdot \mathbf{A}_{v+1} \prod_{j=v+1}^{z} \mathbf{D}_j^{x^{(k)}_{\iota(j)}} \right\rceil_p = \left\lfloor \prod_{i=1}^{v} s_i^{x^{(k)}_{\iota(i)}} \cdot \mathbf{A}_{v+1}^{(\rho(k))} \prod_{j=v+1}^{z} \mathbf{D}_j^{x^{(k)}_{\iota(j)}} \right\rceil_p \tag{11}$$

where $\rho(k) \in [w]$. Therefore we can anyway guess the row number $\rho(k)$ that corresponds to state of the partial evaluation of the branching program (note that this simulation strategy is not using the indicator $d^{(k)}$). For each evaluation, there is a chance of $1/w$ of guessing the ρ value correctly. So when $w = 5$, we can guess the correct rows for $O(\log\lambda)$ queries with non-negligible probability.

Formally, suppose there is an attacker A' that distinguishes H_v and H_{v-1} with non-negligible probability ζ, we build an attacker A that distinguishes

$$\mathbf{A}_{v+1}, \{\mathbf{Y}_{v,b} = \left(\mathbf{I}^{w\times w} \otimes s_v^b\right)\mathbf{A}_{v+1} + \mathbf{E}_v^b\}_{b\in\{0,1\}}$$

from

$$\{U(R_q^{w\times wm} \times R_q^{w\times wm} \times R_q^{w\times wm})\}.$$

A proceeds as follows:

1. Before any input or constrained key queries, A samples $\{(\mathbf{A}_j, \tau_j) \leftarrow \mathsf{TrapSam}(R^{w\times w}, 1^n, 1^m, q)\}_{j\in[v]}$ with trapdoors; samples $\{\mathbf{A}_{j'} \leftarrow U(R_q^{w\times wm})\}_{j'\in[v+2,z+1]}$ uniformly random. Set $\mathbf{A}_J := \mathbf{J}\mathbf{A}_1$. Then samples the GLWE secrets $\{s_i^b \leftarrow \eta\}_{b\in\{0,1\}, i\in[v-1]}$; samples $\{\mathbf{D}_i^b \leftarrow \gamma\}_{b\in\{0,1\}, i\in[v+1,z]}$.

2. Once a PRF evaluation query before the constrained key query is given (say it is the k^{th} query), A guesses a random row number $\rho(k) \in [w]$, computes $y^{(k)}$ as

$$y^{(k)} = \left\lfloor \prod_{i=1}^{v-1} s_i^{x^{(k)}_{\iota(i)}} \cdot \mathbf{Y}^{(\rho(k))}_{v, x^{(k)}_{\iota(i)}} \prod_{j=v+1}^{z} \mathbf{D}_j^{x^{(k)}_{\iota(j)}} \right\rceil_p \approx_s \left\lfloor \prod_{i=1}^{v} s_i^{x^{(k)}_{\iota(i)}} \cdot \mathbf{A}^{(\rho(k))}_{v+1} \prod_{j=v+1}^{z} \mathbf{D}_j^{x^{(k)}_{\iota(j)}} \right\rceil_p \tag{12}$$

3. Given the constrained key query (represented by a branching program $\{\mathbf{B}_i^b\}_{i\in[z]}$), A checks for every input $x \in \{x^{(1)}, ..., x^{(t)}\}$ if

$$\mathbf{e}_1 \cdot \left(\prod_{j=v+1}^{z} \mathbf{B}_j^{x^{(k)}_{\iota(j)}} \right)^{-1} = \mathbf{e}_{\rho(k)}. \tag{13}$$

If any of the t guesses is wrong, A aborts; if all the guesses are correct, A proceeds.

4. To produce the constrained key, A computes $\mathbf{Y}_i^b := \left(\mathbf{B}_i^b \otimes s_i^b\right) \mathbf{A}_{i+1} + \mathbf{E}'^b_i$, for $i \in [v-1]$. Then computes $\{\mathbf{D}_i^b \leftarrow \mathsf{PreimgSam}(\mathbf{A}_i, \tau_i, \mathbf{Y}_i^b, \sigma)\}_{b\in\{0,1\}, i\in[v-1]}$, and $\{\mathbf{D}_v^b \leftarrow \mathsf{PreimgSam}(\mathbf{A}_v, \tau_v, (\mathbf{B}_v^b \otimes 1_R)\mathbf{Y}_{v,b}, \sigma)\}_{b\in\{0,1\}}$. Outputs $\mathbf{A}_J$, $\{\mathbf{D}_i^b\}_{b\in\{0,1\}, i\in[z]}$.

5. To produce the PRF evaluation queries after the constrained key query. For $k \in [t+1, t']$, A computes $y^{(k)}$ as

$$y^{(k)} = \left\lfloor \mathbf{J} \times \left(\left(\prod_{j=v+1}^{z} \mathbf{B}_j^{x^{(k)}_{\iota(j)}} \right)^{-1} \otimes \prod_{i=1}^{v-1} s_i^{x^{(k)}_{\iota(i)}} \right) \times \mathbf{Y}_{v, x^{(k)}_{\iota(i)}} \prod_{j=v+1}^{z} \mathbf{D}_j^{x^{(k)}_{\iota(j)}} \right\rceil_p \tag{14}$$

So if the GLWE samples are real, then the output distribution is statistically close to H_v; if the samples are uniform, then the output distribution is statistically close to H_{v-1} due to Lemma 2.6. Furthermore, the probability that A aborts is $(1/w)^t$, where t is the number of evaluation queries before the constrained key query. So A breaks GLWE with probability more than $\zeta \cdot (1/w)^t$. Therefore when $(1/w)^t$ is non-negligible, the indistinguishability of H_v and H_{v-1} follows the polynomial hardness of GLWE. □

Lemma 5.14. *If $C(x^{(k)}) = 0$, then the output $y^{(k)}$ in H_0 is pseudorandom.*

Proof. The proof follows the observation in Eqn. (7), and treats $\mathbf{A}_1^{(3)}$ as the secret key of the BLMR PRF (cf. Lemma 2.13). □

The proof completes by combining the Lemmas 5.13 and 5.14. □

6 Private-key functional encryption from CHCPRF

We construct private-key function-hiding functional encryptions for NC^1 circuits from (1) CHCPRFs for NC^1; (2) semantic secure private-key encryption schemes with decryption in NC^1. The scheme satisfies 1-key simulation-based security.

6.1 The definition of functional encryption

Definition 6.1 (Function-hiding private-key functional encryption [GKP+13]). *A functional encryption scheme for a class of functions $\mathcal{C}_\mu = \{C : \{0,1\}^\mu \to \{0,1\}\}$ is a tuple of p.p.t. algorithms* $(\mathsf{Setup}, \mathsf{FSKGen}, \mathsf{Enc}, \mathsf{Dec})$ *such that:*

- $\mathsf{Setup}(1^\lambda)$ *takes as input the security parameter 1^λ, outputs the master secret key* MSK.
- $\mathsf{FSKGen}(\mathsf{MSK}, C)$ *takes* MSK *and a function $C \in \mathcal{C}_\mu$, ouputs a functional decryption key* FSK_C.
- $\mathsf{Enc}(\mathsf{MSK}, m)$ *takes* MSK *and a message $m \in \{0,1\}^\mu$, outputs a ciphertext* CT_m.
- $\mathsf{Dec}(\mathsf{FSK}_C, \mathsf{CT}_m)$ *takes as input a ciphertext* CT_m *and a functional decryption key* FSK_C*, outputs (in the clear) the result $C(m)$ of applying the function on the message.*

We require that:

Correctness. *For every message $m \in \{0,1\}^\mu$ and function $C \in \mathcal{C}_\mu$ we have:*

$$\Pr\left[C(m) = \mathsf{Dec}(\mathsf{FSK}_C, \mathsf{Enc}(\mathsf{MSK}, m))\right] = 1 - \mathsf{negl}(\lambda),$$

where the probability is taken over the randomness of the algorithms $\mathsf{Setup}, \mathsf{FSKGen}, \mathsf{SKEnc}, \mathsf{Dec}$.

Security. *We require that for all polytime, stateful algorithm* Adv*, there is a polytime, stateful algorithm* Sim *such that:*

$$\{\textit{Experiment REAL}_{\mathsf{Adv}}(1^\lambda)\}_{\lambda\in\mathbb{N}} \approx_c \{\textit{Experiment IDEAL}_{\mathsf{Adv},\mathsf{Sim}}(1^\lambda)\}_{\lambda\in\mathbb{N}}$$

The real and ideal experiments of stateful algorithms $\mathsf{Adv}, \mathsf{Sim}$ *are as follow:*

```
Experiment REAL_Adv(1^λ)
$\mathsf{MSK} \leftarrow \mathsf{Gen}(1^\lambda)$,
$Repeat:$
  $\mathsf{Adv} \to (m, d_m);\ \mathsf{Adv} \leftarrow \mathsf{Enc}(\mathsf{MSK}, m);$
$\mathsf{Adv} \to C;$
  if $d_m \neq C(m)$ for some $m$ then Output $\perp$
  else $\mathsf{Adv} \leftarrow \mathsf{FSK}_C = \mathsf{FSKGen}(\mathsf{MSK}, C);$
$Repeat:$
  $\mathsf{Adv} \to m;\ \mathsf{Adv} \leftarrow \mathsf{Enc}(\mathsf{MSK}, m)$
$\mathsf{Adv} \to b;$ Output $b$
```

```
Experiment IDEAL_{Adv,Sim}(1^λ)
$\mathsf{Sim} \leftarrow 1^\lambda$
$Repeat:$
  $\mathsf{Adv} \to (m, d_m);\ \mathsf{Adv} \leftarrow \mathsf{Sim}(1^{|m|}, d_m);$
$\mathsf{Adv} \to C;$
  if $d_m \neq C(m)$ for some $m$ then Output $\perp$
  else $\mathsf{Adv} \leftarrow \mathsf{FSK}_S = \mathsf{Sim}(1^{|C|});$
$Repeat:$
  $\mathsf{Adv} \to m;\ \mathsf{Adv} \leftarrow \mathsf{Sim}(1^{|m|}, C(m))$
$\mathsf{Adv} \to b;$ Output $b$
```

That is, in the experiments Adv *can ask for a single functional decryption key and polynomially many input queries, in any order. For encryption queries m made before the decryption key query,* Adv *is expected to provide a bit d_x indicating whether $C(m) = 1$. In the real experiment* Adv *obtains the encryption of m. In the ideal experiment* Adv *obtains a value generated by* Sim*, whereas* Sim *is given only $1^{|m|}$ and d_m. Once* Adv *makes the functional key query for circuit $C \in \mathcal{C}_\lambda$, both experiments verify the consistency of the indicator bits d_m for all the encryption queries m made by* Adv *so far. If any inconsistency is found then the experiment halts. Next, in the real experiment* Adv *obtains the constrained key generated by the constraining algorithm; in the ideal experiment* Adv *obtains a key generated by* Sim*, whereas* Sim *is given only the size of C. The handling of encryption queries made by* Adv *after the circuit query is similar to the ones before, with the exception that the indicator bit d_m is no longer needed and* Sim *obtains the value of $C(m)$ instead. The output of the experiment is the final output bit of* Adv.

6.2 The construction

Theorem 6.2. *If there are 1-key secure constraint-hiding constraint PRFs for constraint class $\mathcal{C}$, and symmetric-key encryption schemes with decryption in the class $\mathcal{C}$, then there are 1-key secure private-key function-hiding functional encryptions for function class $\mathcal{C}$.*

Corollary 6.3. *Assuming the intractability of GLWE, there are 1-key secure private-key function-hiding functional encryptions for* NC^1.

Construction 6.4. *Given a CHCPRF* $(\mathsf{F.Gen}, \mathsf{F.Constrain}, \mathsf{F.Eval})$*, a semantic secure symmetric-key encryption scheme* $(\mathsf{Sym.Gen}, \mathsf{Sym.Enc}, \mathsf{Sym.Dec})$*, we build a private-key functional encryption* FE *as follows:*

- $\mathsf{FE.Setup}(1^\lambda)$ *takes as input the security parameter* 1^λ*, runs* $\mathsf{Sym.Gen}(1^\lambda) \to \mathsf{Sym.SK}$*,* $\mathsf{F.Gen}(1^\lambda) \to \mathsf{F.MSK}$*, outputs the master secret key* $\mathsf{FE.MSK} = (\mathsf{Sym.SK}, \mathsf{F.MSK})$.
- $\mathsf{FE.Enc}(\mathsf{FE.MSK}, m)$ *parses* $\mathsf{FE.MSK} = (\mathsf{Sym.SK}, \mathsf{F.MSK})$*, computes* $\mathsf{Sym.CT} = \mathsf{Sym.Enc}(m)$*,* $\mathsf{Tag} = \mathsf{F.Eval}(\mathsf{F.MSK}, \mathsf{Sym.CT})$. *Outputs* $\mathsf{FE.CT} = (\mathsf{Sym.CT}, \mathsf{Tag})$.
- $\mathsf{FE.FSKGen}(\mathsf{FE.MSK}, C)$ *parses* $\mathsf{FE.MSK} = (\mathsf{Sym.SK}, \mathsf{F.MSK})$*, outputs the functional decryption key* $\mathsf{FE.FSK}_C = \mathsf{F.Constrain}(\mathsf{F.MSK}, F[\mathsf{Sym.SK}, C])$*, where the functionality of* $F[\mathsf{Sym.SK}, C](\cdot)$ *is:*
  - *On input* x*, computes* $\mathsf{Sym.Dec}(\mathsf{Sym.SK}, x) \to m \in \{0,1\}^\mu \cap \perp$*;*
  - *if* $m = \perp$*, return 0; else, return* $C(m)$.
- $\mathsf{FE.Dec}(\mathsf{FE.FSK}_C, \mathsf{FE.CT})$ *parses* $\mathsf{FE.FSK}_C = \mathsf{F.CK}_F$*,* $\mathsf{FE.CT} = (\mathsf{Sym.CT}, \mathsf{Tag})$*, computes* $T = \mathsf{F.Eval}(\mathsf{F.CK}_F, \mathsf{Sym.CT})$. *Outputs* 1 *if* $T = \mathsf{Tag}$*,* 0 *if not.*

Correctness. Correctness follows the correctness of Sym and F.

Security. We build the FE simulator $\mathsf{FE.Sim}$ from the symmetric-key encryption simulator $\mathsf{Sym.Sim}$ and CHCPRF simulator $\mathsf{F.Sim}$:

1. Generates the simulated master secret-keys $\mathsf{Sym.SK}^S$ and $\mathsf{F.MSK}^S$
2. Given a function-decryption key query (for function C), $\mathsf{FE.Sim}$ runs $\mathsf{CK}^S \leftarrow \mathsf{F.Sim}_1(1^\lambda, 1^{|F[\mathsf{Sym.SK},C]|}, \mathsf{F.MSK}^S)$, outputs CK^S as $\mathsf{FE.FSK}^S$.
3. Given a ciphertext query and the output bit $C(m)$, $\mathsf{FE.Sim}$ runs $\mathsf{Sym.CT}^S \leftarrow \mathsf{Sym.Sim}(1^\lambda, 1^{|m|}, \mathsf{Sym.SK}^S)$ and $\mathsf{Tag}^S \leftarrow \mathsf{FSim}_2(\mathsf{F.MSK}^S, \mathsf{CK}^S, \mathsf{Sym.CT}^S, C(m))$, outputs $(\mathsf{Sym.CT}^S, \mathsf{Tag}^S)$ as $\mathsf{FE.CT}^S$.

To show that the simulated outputs are indistinguishable from the real outputs, consider an intermediate simulator $\mathsf{FE.Sim}'$ which is the same to $\mathsf{FE.Sim}$, except that it uses the real Sym ciphertexts in the ciphertext queries. Observe that the secret-key of Sym is not exposed in $\mathsf{FE.Sim}'$ or $\mathsf{FE.Sim}$, the output distributions of $\mathsf{FE.Sim}'$ and $\mathsf{FE.Sim}$ are indistinguishable following the security of Sym.

Next, assume there is a distinguisher D for the outputs of the real FE scheme and $\mathsf{FE.Sim}'$, we build an attacker A for the CHCPRF F. A samples a secret key for Sym, sends a constrained circuit query, obtains the real CK if it is the real distribution, or the simulated CK^S if it is the simulated distribution; then creates symmetric-key ciphertexts, sends as the input queries to the CHCPRF. It obtains the real outputs if it is the real case, or the simulated outputs if it is the simulated case. A treats the outputs as tags. A forwards the ciphertexts, tags and FSK to D. D's success probability transfers to the one for A.

Comparison. The garbled inputs and garbled circuits in reusable garbled circuits correspond to the ciphertexts and decryption keys in function-hiding functional encryptions. Compared to the reusable garbled circuits of Goldwasser et al. [GKP+13], our construction achieves smaller garbled input size. Recall that the size of the garbled input in the construction of Goldwasser et al. depends on the depth of the circuit to be garbled. In our construction, the size of the garbled input (for circuits with 1-bit output) is the size of the symmetric encryption ciphertext $|\mathsf{Sym.CT}(m)|$ plus the security parameter, independent of the depth of the circuit. For circuits with τ-bit outputs (represented by circuit $C(m,i) = C_i(m)$, $i \in [\tau]$), let the garbled circuit be $\mathsf{CK}[\mathsf{Sym.SK}, C](\mathsf{Sym.CT}, i)$. The garbled input is $\mathsf{Sym.CT}, \{\mathsf{Tag}_i = \mathsf{F.Eval}(\mathsf{F.MSK}, (\mathsf{Sym.CT}, i))\}_{i\in[\tau]}$ of size $|\mathsf{Sym.CT}(m)| + \tau\lambda$.

An advantage of the construction of Goldwasser et al. is that the time of producing the garbled input only depends on the depth of the circuit. For our scheme, the time of producing the garbled input depends on the length of the branching program (denoted as z), which is exponential in the depth of the circuit. One of the reasons behind the slow runtime is that the modulus of our construction is exponential in z, so even parsing the intermediate values in the computation takes time proportional to z.

7 Private programmable PRF from rerandomizable constraint-hiding puncturable PRF

A private programmable PRF, as defined in [BLW17], extends the functionality of a constraint-hiding puncturable PRF. On a the punctured point x^*, the programmability allows to embed a specific output value y^*, instead of just a random value as required in the normal puncturable PRF definition. We denote the resulting programmed key as $k\{x^*, y^*\}$. The existence of a private programmable PRF implies the existence of a privately detectable watermarking scheme [BLW17].

We show that private programmable PRFs are implied by constraint-hiding puncturable PRFs with rerandomization, and the constructions from the previous sections can be modified to support rerandomization. This approach was folklore (e.g. from the discussions in [KW17]). Here we formalize it.

7.1 Definition of a private programmable PRF

We recall the definition of a private programmable PRF from [BLW17] in the single-key setting.

Definition 7.1 (Private programmable PRF [BLW17])**.** *A family of private programmable PRFs $\mathcal{F} = \{F_k : D_\lambda \to R_\lambda\}_{\lambda\in\mathbb{N}}$ is specified by a tuple of efficient functions (*Gen, $\mathsf{Program}$, Eval, $\mathsf{Program.Eval}$*).*

- *The key generation algorithm* $\mathsf{Gen}(1^\lambda)$ *takes the security parameter λ, generates the master secret key* MSK.
- *The evaluation algorithm* $\mathsf{Eval}(\mathsf{MSK}, x)$ *takes* MSK*, an input x, outputs $F_{\mathsf{MSK}}(x)$.*
- *The programming algorithm* $\mathsf{Program}(1^\lambda, \mathsf{MSK}, x^*, y^*)$ *takes* MSK*, an input $x^* \in D_\lambda$, an output $y^* \in R_\lambda$, outputs a programmed key $k\{x^*, y^*\}$.*
- *The evaluation algorithm in the programmed mode* $\mathsf{Program.Eval}(k\{x^*, y^*\}, x)$ *takes a programmed key $k\{x^*, y^*\}$, an input x, outputs $F_{k\{x^*,y^*\}}(x)$.*

$\mathcal{F}$ is required to satisfy the following properties.

Functionality. *For an input $x \in D_\lambda$, with probability $1 - \mathsf{negl}(\lambda)$ over the randomness in algorithms* Gen *and* Program*,*

$$\mathsf{Program.Eval}(k\{x^*, y^*\}, x) = \begin{cases} \mathsf{Eval}(\mathsf{MSK}, x) & \text{if } x \neq x^* \\ y^* & \text{if } x = x^* \end{cases}.$$

Pseudorandomness on the programmed points. *Consider the following experiment between a challenger and an adversary. The adversary can ask 3 types of oracle queries: program oracle, evaluation oracle, and challenge oracle. For $b \in \{0,1\}$, the challenger responds to each oracle query in the following manner:*

- *Program oracle. Given an input-output pair x^*, y^*, the challenger outputs a programmed key $k\{x^*, y^*\} \leftarrow \mathsf{Program}(1^\lambda, \mathsf{MSK}, x^*, y^*)$.*
- *Evaluation oracle. Given an input $x \in D_\lambda$, the challenger outputs $y \leftarrow \mathsf{Eval}(\mathsf{MSK}, x)$.*
- *Challenge oracle. Given an input $x_c \in D_\lambda$, the challenger outputs $y \leftarrow \mathsf{Eval}(\mathsf{MSK}, x_c)$ if $b = 1$; outputs $y \leftarrow U(R_\lambda)$ if $b = 0$.*

The challenge query x_c must be equal to x^*, and x_c is not sent among evaluation queries. At the end of the experiment, the adversary chooses b' and wins if $b' = b$. The scheme satisfies the pseudorandomness property if the winning probability of any p.p.t. adversary is bounded by $1/2 + \mathsf{negl}(\lambda)$.

Indistinguishability-based point-hiding. *Consider the following experiment between a challenger and an adversary. The adversary can ask 2 types of oracle queries: program oracle or evaluation oracle. For $b \in \{0,1\}$, the challenger responds to each oracle query in the following manner:*

- *Program oracle. Given two inputs x_0^* and x_1^*, the challenger samples a random $y^* \leftarrow U(R_\lambda)$, outputs a programmed key: $k\{x_b^*, y^*\} \leftarrow \mathsf{Program}(1^\lambda, \mathsf{MSK}, x_b^*, y^*)$.*
- *Evaluation oracle. Given an input $x \in D_\lambda$, the challenger outputs $\mathsf{Eval}(\mathsf{MSK}, x)$.*

x_0^* and x_1^* are not allowed to be asked to the evaluation oracle. At the end of the experiment, the adversary chooses b' and wins if $b' = b$. The scheme is called point-hiding if the winning probability of any p.p.t. adversary is bounded by $1/2 + \mathsf{negl}(\lambda)$.

7.2 Rerandomizable puncturability implies programmability

A constraining algorithm is rerandomizable if the output on an input x such that $C(x) = 0$ is rerandomizable.

Definition 7.2 (Rerandomizability in the constraining algorithm). *A constraining algorithm is rerandomizable if for all but negligibly many master secret key* MSK*, for a constraint C, for all x s.t. $C(x) = 0$, the following holds:*

$$\mathsf{Constrain.Eval}(\mathsf{Constrain}(1^\lambda, \mathsf{MSK}, C; r), x) \approx_s U(R_\lambda),$$

where the constraining randomness r is sampled uniformly random.

Achieving rerandomizable puncturability. We show how to support rerandomization in the puncturing algorithm of Construction 5.11. Observe that the evaluation of the constrained key on an input x s.t. $C(x) = 0$ outputs $\left\lfloor \prod_{i=1}^{z} s_i^{x_{\iota(i)}} \mathbf{A}_{z+1}^{(2)} \right\rceil_p$ (cf. Eqn. (6)). By our parameter setting, each $s_i^{x_{\iota(i)}}$ is invertible in R with all but negligible probability. So the simplest way to rerandomize the punctured output is to pick a uniformly random $\mathbf{A}_{z+1}^{(2)}$ using the randomness in the puncturing algorithm.

We note that the constraint-hiding puncturable PRFs in [BKM17, BTVW17, CVW18] can also be modified to provide rerandomizable puncturing algorithms.

The transformation. Next we construct a private programmable PRF from a rerandomizable constraint-hiding puncturable PRF. For simplicity we only construct a private programmable PRF with 1-bit output. The one with m-bit output follows immediately by sampling m independent keys from the 1-bit solution.

Construction 7.3. *Given a rerandomizable constraint-hiding puncturable PRF $\mathcal{F} : D_\lambda \to \{0,1\}$, construct a private programmable PRF $\mathcal{F}' : D_\lambda \to \{0,1\}$ as follows.*

Key Gen. $\mathcal{F}'.\mathsf{Gen}(1^\lambda)$ *runs* $\mathsf{MSK} \leftarrow \mathcal{F}.\mathsf{Gen}(1^\lambda)$*, set* $\mathsf{MSK}' = \mathsf{MSK}$.

Eval. $\mathcal{F}'.\mathsf{Eval}(\mathsf{MSK}', x) = \mathcal{F}.\mathsf{Eval}(\mathsf{MSK}, x)$.

Program. $\mathcal{F}'.\mathsf{Program}(\mathsf{MSK}', x^*, y^*)$ *keeps running* $k\{x^*\} \leftarrow \mathcal{F}.\mathsf{Puncture}(\mathsf{MSK}, x^*; r)$ *with fresh randomness r, until* $\mathcal{F}.\mathsf{Puncture.Eval}(k\{x^*\}, x^*) = y^*$*. Set $k\{x^*\}$ as the programmed key $k'\{x^*, y^*\}$.*

Program Eval. $\mathcal{F}'.\mathsf{Program.Eval}(k'\{x^*, y^*\}, x) = \mathcal{F}.\mathsf{Puncture.Eval}(k\{x^*\}, x)$.

Theorem 7.4. *Construction 7.3 satisfies Definition 7.1.*

Proof. The functionality follows immediately from that of $\mathcal{F}$. So does the indistinguishability-based point-hiding property of $\mathcal{F}'$, since in the game the programmed output is chosen from random by the challenger, which makes the game identical to the indistinguishability constraint-hiding game for CHCPRF.

It remains to prove the pseudorandomness of the original evaluation on the programmed input. Suppose by contradiction that there is an adversary A' that predicts the "real" or "random" in the pseudorandomness game of $\mathcal{F}'$ with probability $1/2 + \nu(\lambda)$ where ν is non-negligible, we show an adversary A that breaks the pseudorandomness game of $\mathcal{F}$. Once A' makes a program query on $x^*, y^* \in D_\lambda \times \{0,1\}$, A makes a puncture key query on x^*, gets back a punctured key $k\{x^*\}$. If $\mathcal{F}.\mathsf{Puncture.Eval}(k\{x^*\}, x^*) = y^*$, A continues by sending $k\{x^*\}$ as the programmed key to A'; in the end choose "real" or "random" according to the choice of A'. If $\mathcal{F}.\mathsf{Puncture.Eval}(k\{x^*\}, x^*) \neq y^*$, A aborts the game. Then the success probability of A is $1/2 + \nu(\lambda)/2$, breaking the pseudorandomness game of $\mathcal{F}$. □

Acknowledgments

We thank Leonid Reyzin for the helpful discussions and the formalization of LWE-hardness.

We thank Shai Halevi, Vinod Vaikuntanathan, Hoeteck Wee, Sam Kim, David Wu, Hart Montgomery, Shweta Agrawal, Ryo Nishimaki, and Takashi Yamakawa, for the valuable suggestions from them after the initial publication of the paper.

References

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Advances in Cryptology-CRYPTO 2009*, pages 595–618. Springer, 2009.

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *CRYPTO (2)*, volume 8043 of *LNCS*, pages 500–518. Springer, 2013.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *CRYPTO (1)*, volume 9215 of *LNCS*, pages 308–326. Springer, 2015.

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11-15, 1999, Proceedings*, volume 1644 of *LNCS*, pages 1–9. Springer, 1999.

[AMN$^+$18] Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Constrained prfs for nc1 in traditional groups. To appear at Crypto 2018. Cryptology ePrint Archive, Report 2018/154, 2018.

[AP11] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3):535–553, 2011.

[AR17] Shweta Agrawal and Alon Rosen. Functional encryption for bounded collusions, revisited. In *TCC (1)*, volume 10677 of *Lecture Notes in Computer Science*, pages 173–205. Springer, 2017.

[Bar86] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. In Juris Hartmanis, editor, *STOC*, pages 1–5. ACM, 1986.

[BFP$^+$15] Abhishek Banerjee, Georg Fuchsbauer, Chris Peikert, Krzysztof Pietrzak, and Sophie Stevens. Key-homomorphic constrained pseudorandom functions. In *TCC (2)*, volume 9015 of *Lecture Notes in Computer Science*, pages 31–60. Springer, 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 *Advances in Cryptology–EUROCRYPT 2014*, pages 533–556. Springer, 2014.

[BGI$^+$12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *Public Key Cryptography*, volume 8383 of *LNCS*, pages 501–519. Springer, 2014.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325. ACM, 2012.

[BKM17] Dan Boneh, Sam Kim, and Hart William Montgomery. Private puncturable prfs from standard lattice assumptions. In *EUROCRYPT (1)*, volume 10210 of *Lecture Notes in Computer Science*, pages 415–445, 2017.

[BKW17] Dan Boneh, Sam Kim, and David J. Wu. Constrained keys for invertible pseudorandom functions. In *TCC (1)*, volume 10677 of *Lecture Notes in Computer Science*, pages 237–263. Springer, 2017.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 410–428, 2013.

[BLP$^+$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 575–584. ACM, 2013.

[BLW17] Dan Boneh, Kevin Lewi, and David J. Wu. Constraining pseudorandom functions privately. To appear at PKC 2017, Cryptology ePrint Archive, Report 2015/1167, 2017.

[BNPW16] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. In *TCC (B2)*, volume 9986 of *LNCS*, pages 391–418, 2016.

[BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 353–370, 2014.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *LNCS*, pages 719–737. Springer, 2012.

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324(1):71–90, 2003.

[BS16] Jean-François Biasse and Fang Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In *SODA*, pages 893–902. SIAM, 2016.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference*, pages 253–273. Springer, 2011.

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained prfs (and more) from LWE. In *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, pages 264–302, 2017.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 97–106. IEEE, 2011.

[BV15a] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *FOCS*, pages 171–190. IEEE Computer Society, 2015.

[BV15b] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic prfs from standard lattice assumptions. In *Theory of Cryptography*, pages 1–30. Springer, 2015.

[BVWW16] Zvika Brakerski, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Obfuscating conjunctions under entropic ring LWE. In *ITCS*, pages 147–156. ACM, 2016.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology-ASIACRYPT 2013*, pages 280–300. Springer, 2013.

[CDPR16] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In *EUROCRYPT (2)*, volume 9666 of *LNCS*, pages 559–585. Springer, 2016.

[CDW17] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short stickelberger class relations and application to ideal-svp. To appear at Eurocrypt 2017. Cryptology ePrint Archive, Report 2016/885, 2017.

[CGH17] Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of candidate branching program obfuscators. To appear at Eurocrypt 2017. Cryptology ePrint Archive, Report 2016/998, 2017.

[CHL$^+$15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *EUROCRYPT (1)*, volume 9056 of *LNCS*, pages 3–12. Springer, 2015.

[CLLT16] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Cryptanalysis of GGH15 multilinear maps. In *CRYPTO (2)*, volume 9815 of *LNCS*, pages 607–628. Springer, 2016.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *CRYPTO (1)*, pages 476–493, 2013.

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. Ggh15 beyond permutation branching programs: Proofs, attacks, and candidates. To appear at Crypto 2018, Cryptology ePrint Archive, Report 2018/360, 2018.

[DD12] Léo Ducas and Alain Durmus. Ring-lwe in polynomial rings. In *International Workshop on Public Key Cryptography*, pages 34–51. Springer, 2012.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.

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

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

[GKP$^+$13] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 555–564. ACM, 2013.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, 2013.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, pages 545–554. ACM, 2013.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, 2015.

[Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 443–457. Springer, 2000.

[Hal15] Shai Halevi. Graded encoding, variations on a scheme. Cryptology ePrint Archive, Report 2015/866, 2015.

[HJ16] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. In *EUROCRYPT (1)*, volume 9665 of *LNCS*, pages 537–565. Springer, 2016.

[Kle00] Philip N. Klein. Finding the closest lattice vector when it's unusually close. In *SODA*, pages 937–941. ACM/SIAM, 2000.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *ACM Conference on Computer and Communications Security*, pages 669–684. ACM, 2013.

[KW17] Sam Kim and David J. Wu. Watermarking cryptographic functionalities from standard lattice assumptions. In *CRYPTO (1)*, volume 10401 of *Lecture Notes in Computer Science*, pages 503–536. Springer, 2017.

[LPR13a] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43, 2013.

[LPR13b] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 35–54, 2013.

[LPST16] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In *Public Key Cryptography (2)*, volume 9615 of *Lecture Notes in Computer Science*, pages 447–462. Springer, 2016.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptography*, 75(3):565–599, 2015.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology–EUROCRYPT 2012*, pages 700–718. Springer, 2012.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measure. *SIAM Journal on Computing*, 37(1):267–302, 2007.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *STOC*, pages 333–342, 2009.

[PR06] Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *Theory of Cryptography*, pages 145–166. Springer, 2006.

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

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.