Universal Computational Extractors from Lattice Assumptions

Yilei Chen [*] Xinyu Mao [†]

February 13, 2024

Abstract

Universal computational extractors (UCEs), introduced by Bellare, Hoang, and Keelveedhi [BHK13], can securely replace random oracles in various applications, including KDM-secure encryption, deterministic encryption, RSA-OAEP, etc. Despite its usefulness, constructing UCE in the standard model is challenging. The only known positive result is given by Brzuska and Mittelbach [BM14], who construct UCE with strongly computationally unpredictable one-query source assuming indistinguishability obfuscation (iO) and the existence of point obfuscators with auxiliary input (AIPO); they also construct UCE with q-query sources assuming iO and composable AIPO. On the other hand, Brzuska, Farshim, and Mittelbach [BFM14] show that the most potent version of UCE does not exist, assuming the existence of iO.

In this paper, we construct UCE with strongly computationally unpredictable one-query sources from lattice assumptions based on the GGH15 encodings [GGH15], without using iO. Security is proven under the following assumptions: (1) LWE with subexponential hardness; (2) evasive LWE, which is a new assumption proposed by Wee [Wee22]; (3) the existence of AIPO in NC^1. Our UCE directly implies a universal hardcore function that outputs a polynomial number of bits, giving the first lattice-based universal hardcore function without using iO. We also put forth a new primitive called *obliviously programmable function* as an intermediate abstraction; it makes our analysis more modularized and could be of independent interest.

1 Introduction

The Random Oracle Methodology refers to the popular paradigm of designing cryptographic schemes that comprises two steps: One first designs a scheme whose security can be proved in the random oracle (RO) model; then, the random oracle is instantiated by a good 'cryptographic hash function' (e.g., SHA-3), hoping the resulting scheme is still safe. Well-known applications of the RO methodology include the Fiat-Shamir transform [FS87] and the Fujisaki-Oakamoto transform [FO99]. However, this is only a rule of thumb and has been proven to be theoretically unsound: In a seminal work, Canetti et al. [CGH04] designed a scheme that is secure in the random oracle model but insecure when the random oracle is replaced by *any* function.

Even with these negative results, the random oracle methodology remains popular as people deem the known counterexamples as artificially contrived. The hope is that in natural and practical

[*]Tsinghua University and Shanghai Qi Zhi Institute. Email: `chenyilei.ra@gmail.com`

[†]University of Southern California. Email: `xinyumao@usc.edu`

cases, the random oracle can be safely instantiated. A natural remedy is to identify 'RO-like' properties that are sufficient for important applications and then construct hash functions with such properties under well-formed assumptions. Along this line, a number of security notions have been proposed in existing literature, such as correlation intractability [CGH04], correlated-input security [GOR11], and universal computational extractors [BHK13]. In this paper, we focus on the construction of universal computational extractors (UCEs).

UCE security. ***Universal computational extractors***, introduced by Bellare, Hoang, and Keelveedhi (BHK, [BHK13]), enable instantiations of random oracles in various applications, e.g., universal hardcore function, deterministic encryption, RSA-OAEP, etc. For a keyed function family H, the UCE security is defined by a two-stage game as follows. An adversary in this game is a pair of algorithms (S, D), where S is called the *source* and D the *distinguisher*. S has access to an oracle $\textsc{Hash}$ that is either $\mathsf{H.Eval}(\mathsf{hk}, \cdot)$ or a random oracle; D receives a message L from S and is given hk; D has to guess to which oracle S has access to. The adversary wins if D guesses correctly.

Without any restrictions, we can easily design a winning adversary: S queries $\textsc{Hash}$ on a random point x and send $L = (x, y)$ to D, where y is the oracle answer; then D simply check whether $\mathsf{H.Eval}(\mathsf{hk}, x) = y$. Therefore, we restrict the source to be in some set $\mathcal{S}$, and allow D to be any PPT algorithm. H is said to be $\mathcal{S}$-UCE-secure if for every $S \in \mathcal{S}$ and PPT D, the winning probability of (S, D) in the UCE game is negligible.

BHK first considers the set of all computationally unpredictable sources, denoted by $\mathcal{S}^{\mathsf{cup}}$. Roughly speaking, $S \in \mathcal{S}^{\mathsf{cup}}$ if no PPT predictor, given L, can predict the queries made by S to $\textsc{Hash}$. [BFM14] shows that UCE security for computationally unpredictable sources $\mathcal{S}^{\mathsf{cup}}$ cannot be achieved in the standard model, assuming indistinguishability obfuscation (iO) [BGI+12, GGH+16] exists.

Later, [BM14] strengthened the restriction to be *strongly* computationally unpredictable, meaning that the oracle answers to the source are also given to the predictor. Assuming the existence of iO and point obfuscation with auxiliary input (AIPO) [Can97, Wee05, GK05, GKPV10], they constructed a UCE-secure function for sources that are (1) strongly computationally unpredictable and (2) only make one query to $\textsc{Hash}$. We denote the set of such sources as $\mathcal{S}_1^{\mathsf{scup}}$.

Though the sources are restricted to making only one query, a $\mathcal{S}_1^{\mathsf{scup}}$-UCE-secure function is still powerful — it implies hardcore function for any one-way function and the hardcore function outputs a polynomial number of bits. As iO appears to be a much stronger primitive than UCE, a natural question arises:

Can we construct UCE-secure functions without using iO (in the standard model)?

1.1 Our Results

In this paper, we give a simple construction of $\mathcal{S}_1^{\mathsf{scup}}$-UCE-secure function from lattice assumptions based on the GGH15 encoding [GGH15]. Our main result is

Theorem 1.1. *Under the subexponential LWE assumption and the evasive LWE assumption, provided that there exists an AIPO in* NC^1*, then there exists a* $\mathcal{S}_1^{\mathsf{scup}}$*-UCE-secure function.*

Here, we need the existence of AIPO where the evaluation algorithm of the obfuscated program can be computed by a circuit in NC^1. Such AIPO exists assuming some non-standard yet plausible variants of LWE or DDH, which we will explain later.

According to [BHK13], an $\mathcal{S}_1^{\mathsf{scup}}$-UCE-secure function is a universal hardcore function; that is, it is a hardcore for any one-way function.

Corollary 1.2. *Under the subexponential LWE assumption and the evasive LWE assumption, provided that there exists an AIPO in* NC^1*, then there exists a universal hardcore function that outputs a polynomial number of bits.*

Among previous constructions of universal hardcore function, only the constructions due to Zhandry are free of iO: One [Zha16] uses extractable witness PRF, and the other [Zha19] uses extremely loss functions. Extractable witness PRF contains strong knowledge assumptions and the only instantiation of extremely lossy functions based on the exponential hardness of the decisional Diffie-Hellman problem. Our construction is the first lattice-based one without using iO.

Since subexponential LWE is rather a standard assumption, we discuss the other two assumptions in what follows. Evasive LWE is a new assumption proposed by Wee [Wee22]. Suppose that we have some joint distributions over matrices $\mathbf{P}, \mathbf{S}$ and auxiliary information aux. The evasive LWE assumption postulates that, for a uniformly random (and secret) matrix $\mathbf{B}$,

$$\begin{aligned} &\textbf{if} \quad \left(\mathbf{SB}+\mathbf{E}, \mathbf{SP}+\mathbf{E}',\ \mathsf{aux}\right) \approx_c \left(\mathbf{U}, \mathbf{U}',\ \mathsf{aux}\right) \\ &\textbf{then} \quad \left(\mathbf{SB}+\mathbf{E}, \mathbf{B}^{-1}(\mathbf{P}),\ \mathsf{aux}\right) \approx_c \left(\mathbf{U}, \mathbf{B}^{-1}(\mathbf{P}),\ \mathsf{aux}\right) \end{aligned}$$

where $\mathbf{U}, \mathbf{U}'$ are uniformly random matrices, and $\mathbf{E}, \mathbf{E}'$ are chosen from the LWE error distribution with appropriate parameters. Essentially the above says that given $\mathbf{SB}+\mathbf{E}$, getting the additional component $\mathbf{B}^{-1}(\mathbf{P})$ is no more useful than just getting the product $(\mathbf{SB}+\mathbf{E}) \cdot \mathbf{B}^{-1}(\mathbf{P}) \approx \mathbf{SP}+\mathbf{E}'$. Evasive LWE has proven to be useful in constructing advanced primitives such as witness encryption [Tsa22, VWW22] and attribute-based encryption [HLL23]; (heuristic) attacks are only known for highly contrived distribution of aux [VWW22].

On the AIPO assumption. We consider a variant of point obfuscators. Loosely speaking, we require the obfuscation of any point function to be indistinguishable from the obfuscation of the all-zero function. Construction of AIPO in NC^1 with statistically unpredictable auxiliary inputs is known from standard LWE [GKPV10]. We conjecture that the same construction is an AIPO for computationally unpredictable auxiliary inputs (with our definition).

It is worth noting that the AIPO is only used in the security proof; the construction of UCE itself only involves lattice computations. That is, we only need the existence of an AIPO in NC^1. Therefore, regardless of the status of the AIPO assumption, our construction is still a good candidate for UCE in light of its simplicity. It is possible that the same construction can be proven secure without making use of AIPO, and even more ambitiously, without evasive LWE.

1.2 Technical Overview

Our starting point is the construction of UCE based on iO and AIPO due to Brzuska and Mittelbach [BM14]. Their construction of UCE uses the iO of a puncturable pseudorandom function (PRF); that is, the hash key $\mathsf{hk} \leftarrow \mathsf{iO}(\mathsf{F}(msk, \cdot))$ is an obfuscated program, where F is a puncturable PRF and msk is the master key of F. The security proof in [BM14] essentially uses iO to privately switch the real PRF key to a punctured key that involves the query information from the source. This hints that privately constrained PRFs (PCPRFs) [BLW17] may be good candidates for UCEs, where the hash key hk is simply the constrained key for an empty circuit. Given that there exist a few constructions of PCPRFs from lattice assumptions without using iO (see e.g. [BKM17, CC17, BTVW17, PS18]), we might be able to construct a UCE without iO using ideas from those PCPRFs.

Let us now briefly explain how [BM14] shows that iO of puncturable PRF is a UCE. Here we assume readers have some familiarity of the iO and puncturable PRF methodology (for readers who are not familiar with the iO and puncturable PRF methodology, [SW14, BM14] are good references). Suppose that the source S makes only one query, denoted by x^*, and receives the oracle answer $y^* = \mathsf{F}(msk, x^*)$. The proof in [BM14] comprises two major steps.

1. First, switch y^* to a uniformly random value, and meanwhile, hk is changed to $\mathsf{iO}(P_{x^*,y^*,k_{x^*}})$ where k_{x^*} is the punctured key at point x^* and $P_{x^*,y^*,k_{x^*}}(\cdot)$ is the following program:
    - On input x, if $x = x^*$, output y^*; otherwise, output $\mathsf{F}(k_{x^*}, x)$.
2. Next, hk is switched back to $\mathsf{iO}(\mathsf{F}(msk, \cdot))$, but y^* is still chosen uniformly at random. This step is where AIPO comes into play: Instead of hardcoding x^* into $P_{x^*,y^*,k_{x^*}}$, it suffices to hardcode a point obfuscation of x^* denoted by $p_{x^*} \leftarrow \mathsf{AIPO}(x^*)$, because we only need to check whether $x = x^*$ or not. Moreover, we can use msk instead of k_{x^*}, since $\mathsf{F}(k_{x^*}, x) = \mathsf{F}(msk, x)$ for all $x \neq x^*$. More precisely, we replace $P_{x^*,y^*,k_{x^*}}$ by
    - $P_{p_{x^*},y^*}$: On input x, if $p_{x^*}(x) = 1$, output y^*; otherwise, output $\mathsf{F}(msk, x)$.

    By the strong unpredictability of S and the security of AIPO, it is hard to find x^* given p_{x^*}, y^*. Note that x^* is the only input that $\mathsf{F}(msk, \cdot)$ and $P_{p_{x^*},y^*}(\cdot)$ differs. Then we can finish the proof by a technique from [BCP14]: It is shown that iO for all circuits in $\mathsf{P/poly}$ is also a differing-inputs obfuscator for circuits that differ on at most polynomially many inputs.

Nevertheless, if we wish to use a PCPRF G instead, we will get stuck even in the first step: In a wishful thinking, we want to first switch the key to the punctured key k_{x^*} while using $\mathsf{G}(msk, x^*)$ as the oracle answer; then, we can replace the oracle answer $\mathsf{G}(msk, x^*)$ by a random value due to pseudorandomness. However, to switch from msk (i.e., constrained key for an empty circuit) to k_{x^*} relying on the constraint-hiding property, the adversary (for G) has no access to $\mathsf{G}(msk, x^*)$, and thus it cannot properly reply $\mathsf{G}(msk, x^*)$ to the query by the source. To fix this issue, we conceive the following strategy:

- Imagine that the punctured key is generated by first sampling y^* (uniformly at random) and then producing a key k_{x^*} such that $\mathsf{G}(k_{x^*}, x^*) = y^*$. This way, in the reduction from the constraint-hiding property, the reduction can compute $y^* = \mathsf{G}(k_{x^*}, x^*)$ and reply y^* to the source.

That is, we want to program the value at x^* to be y^*, a random value sampled before the generation of the puncture key k_{x^*}.

Even if we manage to do such programming, the AIPO also causes a problem: In the second step, we want to use $p_{x^*} \leftarrow \mathsf{AIPO}(x^*)$ instead of x^* to generate the key. Hence, we have to program the value at x^* to be y^* without explicitly knowing x^* — we are only given p_{x^*}, a circuit that computes the point function $\mathbf{1}_{x^*}$. Therefore, we roughly require the following functionality:

- Given a circuit C that computes the point function $\mathbf{1}_{x^*}$ and a value y^*, one can generate a 'programmed key' k_C such that $\mathsf{G}(k_C, x^*) = y^*$.

This functionality is supported by the PCPRF constructed from iO [BLW17], but is not supported by the existing lattice-based PCPRFs [BKM17, CC17, BTVW17, PS18]. This also motivates our definition of a new primitive called *obliviously programmable function*. We mean by oblivious that the programmed key is generated only given a circuit C that computes $\mathbf{1}_{x^*}$, without explicitly knowing x^*.

Obliviously programmable function (OPF). Let $\mathcal{C}$ be a circuit class. An OPF $\mathsf{OPF}(\cdot,\cdot)$ for $\mathcal{C}$, like PRFs, is a keyed function where the first input is viewed as the key. It provides an algorithm $\mathsf{OPF.Program}(C, y)$ that takes as input a circuit $C \in \mathcal{C}$ and a value y, and outputs a programmed key k_C. The following properties are required.

- Correctness. If C computes the point function $\mathbf{1}_x$, then $\mathsf{OPF}(k_C, x) = y$.
- Privacy. k_C computationally hides C if (1) C computes a point function or the all-zero function and (2) the programmed value y is chosen uniformly at random.
- Value-Hiding for the all-zero function. If C computes the all zero function, k_C computationally hides the value y.

The value-hiding property intuitively says that if C computes the all-zero function, the value y has no effect. In section 3, we shall prove that $\mathsf{H}(\mathsf{hk}, x) \overset{\text{def}}{=} \mathsf{OPF}(\mathsf{hk}, x)$ is a UCE, with the following key generation algorithm:

- The key generation algorithm $\mathsf{H.Gen}$ chooses a value y uniformly at random, and outputs $\mathsf{hk} \leftarrow \mathsf{OPF.Program}(C_\emptyset, y)$, where $C_\emptyset \in \mathcal{C}$ is any (fixed) circuit that computes the all-zero function.[1]

Theorem 1.3. *(Theorem 3.3, informally) Assume that there exists an AIPO that always outputs a circuit in the circuit class $\mathcal{C}$ (e.g., NC^1). If OPF is an OPF for $\mathcal{C}$, then H defined above is $\mathcal{S}_1^{\mathsf{scup}}$-UCE-secure.*

Realizing OPF via GGH15 encodings. We start with a brief introduction of GGH15 encodings. In this framework, circuits are encoded as matrix branching programs (MBPs). A read-once MBP Γ is specified by $\mathbf{v} \in \{0,1\}^w$ and $\left\{\mathbf{M}_{i,b} \in \{0,1\}^{w\times w}\right\}_{i\in[h],b\in\{0,1\}}$. On inputs $\mathbf{x} \in \{0,1\}^h$, the output of the MBP is 1 if $\mathbf{v}^\top \mathbf{M}_\mathbf{x} = \mathbf{0}$ and otherwise 0, where $\mathbf{M}_\mathbf{x} \overset{\text{def}}{=} \prod_{i\in[h]} \mathbf{M}_{i,x_i}$. To encode such a MBP, we first construct $\left\{\widehat{\mathbf{S}}_{i,b}\right\}_{i\in[h],b\in\{0,1\}}$:

$$\widehat{\mathbf{S}}_{1,b} = \left(\mathbf{I}_n \mid \mathbf{v}^\top \mathbf{M}_{1,b} \otimes \mathbf{S}_{1,b}\right), \widehat{\mathbf{S}}_{i,b} = \begin{pmatrix} \mathbf{I}_n & \\ & \mathbf{M}_{i,b} \otimes \mathbf{S}_{i,b} \end{pmatrix} \text{ for } i = 2, \dots, h,$$

where $\mathbf{S}_{j,b} \leftarrow \mathcal{D}_\sigma^{n\times n}$. Then GGH15 encodings of such an MBP is given by

$$\mathsf{GGH.Encode}(\{\widehat{\mathbf{S}}_{i,b}\}) = \left\{\underset{\sim}{\widehat{\mathbf{S}}_{1,b}\mathbf{A}_1}, \mathbf{A}_1^{-1}(\underset{\sim}{\widehat{\mathbf{S}}_{2,b}\mathbf{A}_2}), \dots, \mathbf{A}_{h-1}^{-1}(\underset{\sim}{\widehat{\mathbf{S}}_{h,b}\mathbf{A}_h})\right\}_{b\in\{0,1\}},$$

where $\mathbf{A}_i \leftarrow \mathbb{Z}_q^{(n+nw)\times m}$ for some $m = \Theta(nw \log q)$ and wavy underline is put in place of noise terms. Given the encoding and $\mathbf{x} \in \{0,1\}^h$, we can approximate $\widehat{\mathbf{S}}_\mathbf{x}\mathbf{A}_h$, where $\widehat{\mathbf{S}}_\mathbf{x} \overset{\text{def}}{=} \prod_{i\in[h]} \widehat{\mathbf{S}}_{i,x_i}$.

[1]Actually, by the value-hiding property, we can set y to be any fixed value.

Intuitively, to generate a programmed key for Γ, we let the encoding be the programmed key k_Γ and let $\mathsf{OPF.Eval}(k_\Gamma, \mathbf{x})$ be the approximation of $\widehat{\mathbf{S}}_\mathbf{x}\mathbf{A}_h$ given by the encoding. But how to program the value at $\mathbf{x}^*$ (assuming Γ computes the point function $\mathbf{1}_{\mathbf{x}^*}$)?

Note that we add a seemingly useless $\mathbf{I}_n$-track in the $\widehat{\mathbf{S}}$-matrices. Remarkably, it is this modification that allows us to program on the target point. To see this, observe that

$$\begin{aligned}\mathsf{OPF.Eval}(k_\Gamma, \mathbf{x}) \approx \widehat{\mathbf{S}}_\mathbf{x}\mathbf{A}_h &= \left(\mathbf{I}_n \mid \mathbf{v}^\top \mathbf{M}_\mathbf{x} \otimes \mathbf{S}_\mathbf{x}\right) \cdot \mathbf{A}_h \\ &= \overline{\mathbf{A}_h} + (\mathbf{v}^\top \mathbf{M}_\mathbf{x} \otimes \mathbf{S}_\mathbf{x}) \cdot \underline{\mathbf{A}}_h,\end{aligned} \tag{1}$$

where $\overline{\mathbf{A}}_h$ denotes the top n rows of $\mathbf{A}_h$ and $\underline{\mathbf{A}}_h$ is the rest part. If Γ computes the point function $\mathbf{1}_{\mathbf{x}^*}$, i.e., $\mathbf{v}^\top \mathbf{M}_{\mathbf{x}^*} = \mathbf{0}$, then $\mathsf{OPF.Eval}(k_\Gamma, \mathbf{x}) \approx \widehat{\mathbf{S}}_{\mathbf{x}^*}\mathbf{A}_h = \overline{\mathbf{A}}_h$. Hence, we can program the value at $\mathbf{x}^*$ by controlling $\overline{\mathbf{A}}_h$.

For privacy, we need to show the encoding computationally hides Γ *provided that* Γ *computes a point function or the all-zero function*. This is reminiscent of witness encryption or null-iO, where security is required only when the underlying circuit computes the all-zero function. In our case, we need to show security also for circuits that compute point functions. To this end, we observe that though [VWW22] aims at constructing witness encryption and null-iO, they in fact give a general reduction assuming evasive LWE: In order to prove the encodings are pseudorandom (thus hiding Γ), it suffices to show that the evaluated products $\left\{\widehat{\mathbf{S}}_\mathbf{x}\mathbf{A}_h + \mathbf{E}_\mathbf{x}\right\}_{\mathbf{x}\in\{0,1\}^h}$ are pseudorandom, where $\mathbf{E}_\mathbf{x}$ are independent noises. Continuing eq. (1), we have

$$\begin{aligned}\widehat{\mathbf{S}}_\mathbf{x}\mathbf{A}_h + \mathbf{E}_\mathbf{x} &= \overline{\mathbf{A}_h} + (\mathbf{v}^\top \mathbf{M}_\mathbf{x} \otimes \mathbf{I}) \cdot (\mathbf{I} \otimes \mathbf{S}_\mathbf{x}) \cdot \underline{\mathbf{A}}_h + \mathbf{E}_\mathbf{x} \\ &\approx \overline{\mathbf{A}_h} + (\mathbf{v}^\top \mathbf{M}_\mathbf{x} \otimes \mathbf{I}) \cdot \overbrace{(\mathbf{I} \otimes \mathbf{S}_\mathbf{x}) \cdot \underline{\mathbf{A}}_h}^{\text{pseudorandom}}.\end{aligned}$$

Since there is at most one point $\mathbf{x}^*$ with $\mathbf{v}^\top \mathbf{M}_{\mathbf{x}^*} = \mathbf{0}$, we have that

- the value at $\mathbf{x}^*$ approximately equals to $\overline{\mathbf{A}_h}$ so it is uniformly random as long as $\overline{\mathbf{A}_h}$ is;
- the value at $\mathbf{x} \neq \mathbf{x}^*$ is pseudorandom since $(\mathbf{I} \otimes \mathbf{S}_\mathbf{x}) \cdot \underline{\mathbf{A}}_h$ is pseudorandom and $(\mathbf{v}^\top \mathbf{M}_\mathbf{x} \otimes \mathbf{I}) \neq \mathbf{0}$.

The value-hiding property follows from a similar argument. The difference is that we want to show the programmed value $\overline{\mathbf{A}}_h$ is computationally hidden, provided that Γ computes the all-zero function. This is true because if Γ computes the all-zero function, then for all $\mathbf{x} \in \{0,1\}^h$, $\widehat{\mathbf{S}}_\mathbf{x}\mathbf{A}_h + \mathbf{E}_\mathbf{x}$ is randomized by $(\mathbf{I} \otimes \mathbf{S}_\mathbf{x}) \cdot \underline{\mathbf{A}}_h$, hence the evaluated products are still pseudorandom even if $\overline{\mathbf{A}}_h$ is chosen and fixed by the distinguisher. Consequently, the encoding is pseudorandom by the aforementioned reduction.

We can generalize the above proof strategy to work with a larger class called read-c MBPs for any polynomial c. Any NC^1 circuit can be translated into such MBPs; therefore, we get an OPF for NC^1:

Theorem 1.4. *Assuming subexponential LWE and evasive LWE, there exists an OPF for* NC^1 *(based on GGH15 encodings).*

Now theorem 1.1 follows by combining theorem 1.4 and theorem 1.3.

1.3 Discussion

OPF compared with PCPRF. OPF is different from PCPRF in both syntax and security.

- Syntax. OPF has no master key and it is more like a plain PRF. The programmed circuit is chosen before key generation.
- Security. OPF does not emphasize pseudorandomness; it is more about programmability. It allows us to program an input-value pair (x^*, y^*) into the function where x^* is not explicitly known — only a circuit computing the point function $\mathbf{1}_{x^*}$ is given. Interestingly, though in our construction, the OPF is indeed a PRF when programmed on the all-zero function, but it seems difficult to prove 'every OPF programmed on the all-zero function is a PRF' from the definitions.

Our OPF construction is somewhat similar to the construction of PCPRFs by Chen, Vaikuntanathan, and Wee (CVW, [CVW18]). The CVW construction also uses a two-track structure in the $\widehat{\mathbf{S}}$-matrices. While we put all identity matrices in the top track to support programming, the CVW construction puts $\mathbf{S}_{i,b}$ in the top track of $\widehat{\mathbf{S}}_{i,b}$ so that the value computed by the top track (i.e., $\mathbf{S}_{\mathbf{x}}\overline{\mathbf{A}_h}$) is the evaluation of the constrained PRF with the master key.

We would like to emphasize that the OPF we construct from lattice assumptions is not a constrained PRF, in the sense that we do not support a general constraining algorithm. Constructing an obliviously programmable constrained PRF without using iO remains an open problem.

Programming on more points? Our result suggests that for a keyed function H, if we can program one point in H obliviously, then assuming AIPO, we show that H behaves like an RO with one query. It is conceivable that if we manage to program q points obliviously, then assuming obfuscation for functions that take the value 1 on at most q values, we can prove H behaves like an RO with q-queries, e.g., proving H is a UCE that supports q-queries. Indeed, this is realized in [BM14] using iO plus *composable virtual grey box point obfuscators*.

Another direction is to construct OPF for larger circuit classes, e.g., $\mathsf{P}/\mathtt{poly}$, without using iO. If we can do so then there is no need to restrict the AIPO from being in NC^1. One potential approach is to start from the PCPRFs in [BTVW17, PS18], which support $\mathsf{P}/\mathtt{poly}$ constraints.

Remove the evasive LWE assumption? Another interesting open problem is proving our construction of UCE is secure without using the evasive LWE assumption. Evasive LWE is a plausible but yet strong and unfalsifiable assumption. Let us briefly explain why we currently need to assume evasive LWE by looking back to the previous applications of the GGH15 encoding. For the constructions of PCPRFs and lockable obfuscations [CC17, GKW17, WZ17, CVW18] from the GGH15 encoding, their security properties can be converted *locally* into each level of the GGH15 encoding, so they can be proven from standard LWE. For the constructions of witness encryption [Tsa22, VWW22] from GGH15, the security property (i.e., there is no witness) is *global*, and it is not clear how to convert it to local properties in GGH15, so they use evasive LWE instead. In our proof, "the AIPO has only one input evaluated to 1" is a global property, and we don't know how to convert it to each level of GGH15 since we don't know which input evaluates to 1 in the AIPO, so we can only argue security using evasive LWE. However, there might be a chance of finding a smart proof technique that allows us to base UCE (and witness encrytion) from GGH15 encoding on standard LWE, and we leave it as an open problem.

2 Preliminaries

Notations. We use lowercase bold symbols for vectors (e.g., $\mathbf{v}$) and uppercase for matrices (e.g., $\mathbf{A}$); denoted by $\mathbf{I}_n$ the identity matrix of dimension n. For a set S, we use $\mathcal{U}(S)$ to denote the uniform distribution over S. We use $\leftarrow$ to denote sampling from a distribution or choosing an element from a set uniformly at random. For two distributions χ_1 and χ_2, we write $\chi_1 \approx_s \chi_2$ if χ_1 and χ_2 are statistically close; and $\chi_1 \approx_c \chi_2$ means that they are computationally indistinguishable. In experiments or games, $[\![E]\!]$ equals 1 if the event E happens and otherwise 0; Expr $\Rightarrow 1$ means the experiment Expr outputs 1. For $x \in \mathbb{Z}_q$, let $\lfloor x \rceil_p \overset{\text{def}}{=} \left\lfloor x \cdot \frac{p}{q} \right\rceil$ denote the rounding of x to $\mathbb{Z}_p$; and $\lfloor \mathbf{A} \rceil_p$ is the matrix resulting from rounding every entry of $\mathbf{A}$ to $\mathbb{Z}_p$. For a function $\nu : \mathbb{N} \to [0, 1]$, we write $\nu = \mathtt{negl}(\lambda)$ if for every $c \in \mathbb{N}$, $\nu(\lambda) \leq 1/(c\lambda^c)$ for sufficiently large λ.

2.1 Lattice Background

Gassian. For $\sigma > 0$, we use $\mathcal{D}_\sigma$ to denote the Guassain over $\mathbb{Z}$ with standard deviation σ, namely

$$\forall x \in \mathbb{Z}, \mathcal{D}_\sigma(x) \propto e^{-\pi x/\sigma^2}.$$

Learning with Error. We recall the learning with errors problem.

Definition 2.1 (Decisional learning with errors (LWE) [Reg09]). For $n, m \in \mathbb{N}$ and modulus $q \geq 2$, distributions θ, π, χ over $\mathbb{Z}_q$ for secret vectors, public matrices, and error vectors respectively. The $\mathsf{LWE}_{n,m,q,\theta,\pi,\chi}$ assumption states that

$$(\mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top \mod q, \mathbf{A}) \approx_c (\mathbf{u}^\top, \mathbf{A}),$$

where

$$\mathbf{s} \leftarrow \theta^n, \mathbf{A} \leftarrow \pi^{n \times m}, \mathbf{e} \leftarrow \chi^m, \mathbf{u} \leftarrow \mathcal{U}(\mathbb{Z}_q^m).$$

We rely on the ***subexponential LWE*** assumption, which states that for some $\delta > 0$, the above indistinguishability holds against adversaries running in 2^{n^δ} with advantage at most 2^{-n^δ} with the following parameters:

$$m = \mathtt{poly}(n), \theta = \pi = \mathcal{U}(\mathbb{Z}_q), \chi = \mathcal{D}_\sigma, q \leq 2^{n^\delta} \cdot \sigma.$$

Lemma 2.2 ([BLMR13]). *Subexponential LWE assumption with $\pi = \mathcal{D}_{2\sqrt{n}}$ (and other parameters the same) are implied by the subexponential LWE assumption (with $\pi = \mathcal{U}(\mathbb{Z}_q)$).*

Trapdoor and preimage sampling. We recall the background of lattice trapdoor and the capability of using the trapdoor to sample a short preimage of the Ajtai function.

Lemma 2.3 ([Ajt99, AP09, MP12]). *There is a PPT algorithm $\mathsf{TrapSam}(1^n, 1^m, q)$ that, given the modulus $q \geq 2$, dimensions n, m such that $m \geq 2n \log q$, outputs $\mathbf{A} \approx_s \mathcal{U}(\mathbb{Z}_q^{n \times m})$ with a trapdoor τ.*

Given any $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{y} \in \mathbb{Z}_q^n$, $\sigma > 0$, we use $\mathbf{A}^{-1}(\mathbf{y}, \sigma)$ to denote the distribution of a vector $\mathbf{d}$ sampled from $\mathcal{D}_\sigma^m$ conditioned on $\mathbf{A}\mathbf{d} = \mathbf{y} \pmod q$. We sometimes suppress σ when the context is clear.

Lemma 2.4 ([GPV08]). *There is a PPT algorithm that for $\sigma \geq 2\sqrt{n \log q}$, given $(\mathbf{A}, \tau) \leftarrow \mathsf{TrapSam}(1^n, 1^m, q)$, $\mathbf{y} \in \mathbb{Z}_q^n$, outputs a sample from $\mathbf{A}^{-1}(\mathbf{y}, \sigma)$.*

2.2 Matrix Branching Programs

Below we introduce the terminologies for matrix branching programs.

Definition 2.5 (Matrix branching program, MBP)**.** A matrix branching program Γ with width w, length h and input length ℓ consists of the following data:

$$\Gamma = \left(\iota : [h] \to [\ell], \mathbf{v} \in \{0,1\}^w, \{\mathbf{M}_{i,b} \in \{0,1\}^{w\times w}\}_{i\in[h],b\in\{0,1\}}\right).$$

Here, ι is called the index-to-input map, and it naturally defines an input-to-index map $\varpi : \{0,1\}^\ell \to \{0,1\}^h$ by letting $\varpi(\mathbf{x})_i = x_{\iota(i)}, \forall i \in [h], \mathbf{x} \in \{0,1\}^\ell$.

This branching program is computing the function $f_\Gamma : \{0,1\}^\ell \to \{0,1\}$, defined as

$$f_\Gamma(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{v}^\top \mathbf{M}_{\varpi(\mathbf{x})} = \mathbf{v}^\top \prod_{i\in[h]} \mathbf{M}_{i,x_{\iota(i)}} \neq \mathbf{0}; \\ 1 & \text{if } \mathbf{v}^\top \mathbf{M}_{\varpi(\mathbf{x})} = \mathbf{v}^\top \prod_{i\in[h]} \mathbf{M}_{i,x_{\iota(i)}} = \mathbf{0}. \end{cases}$$

For simplicity, we write $\Gamma(\mathbf{x})$ instead of $f_\Gamma(\mathbf{x})$ henthforth.

Write $c \overset{\text{def}}{=} h/\ell$. Γ is called a read-c matrix branching program if $\varpi : \{0,1\}^\ell \to \{0,1\}^h$ outputs $c \overset{\text{def}}{=} h/\ell$ copies of x, namely, $\varpi(\mathbf{x}) = \underbrace{\mathbf{x}|\mathbf{x}|\cdots|\mathbf{x}}_{c\text{ times}}$. In this paper, we shall focus on read-once branching programs, i.e., $c = 1$ with ι, ϖ being the identity function, where we simply write $\mathbf{M}_\mathbf{x} \overset{\text{def}}{=} \prod_{i\in[h]} \mathbf{M}_{i,x_i}$.

Definition 2.6. For $\ell, w, c \in \mathbb{N}$, let $\mathsf{MBP}^c_{\ell,w}$ denotes the set of read-c matrix branching programs with input length ℓ and width w.

Note that read-c MBPs are closely related to read-once MBPs: Given a read-c MBP Γ with input length ℓ, we can repeat the input c times then feed into a read-once MBP Γ', where Γ' is the same as Γ except that its index-to-input map is the identity function (so its input length is h). However, the correctness is only about Γ' on valid inputs, namely, inputs that are c-copies of an ℓ-bit string. To generalize our result to read-c-branching programs, we need to ensure that $\Gamma'(\mathbf{x}') = 0$ (i.e., $\mathbf{v}^\top \mathbf{M}_\mathbf{x} \neq 0$) for all invalid input $\mathbf{x}'$. This is done by the following lemma, proven in appendix A.

Lemma 2.7. *Let Γ be a read-c MBP with width w, length h, and input length $\ell = h/c$. Let $\mathsf{repeat}(\mathbf{x}) = \underbrace{\mathbf{x}|\mathbf{x}|\cdots|\mathbf{x}}_{c\text{ times}}$. Then there exists a read-once MBP Γ' with the following properties.*

1. *Γ' has width $h + w$ and length h.*
2. *For all $\mathbf{x} \in \{0,1\}^\ell, \Gamma(\mathbf{x}) = \Gamma'(\mathsf{repeat}(\mathbf{x}))$.*
3. *For all invalid $\mathbf{x}' \in \{0,1\}^h$, i.e., $\mathbf{x}' \neq \mathsf{repeat}(\mathbf{x})$ for any $\mathbf{x} \in \{0,1\}^\ell$, it holds that $\Gamma'(\mathbf{x}') = 0$.*

In particular, if Γ computes a point function or all-zero function, then so is Γ'.

2.3 Point Obfuscation

For a point $x \in \{0,1\}^*$, define the point function $\mathbf{1}_x : \{0,1\}^{|x|} \to \{0,1\}$ as

$$\mathbf{1}_x(u) \overset{\mathsf{def}}{=} \begin{cases} 1 & \text{if } u = x \\ 0 & \text{otherwise} \end{cases}.$$

We consider a variant of point function obfuscators. Loosely speaking, we require the obfuscation of any point function to be indistinguishable from the obfuscation of the all-zero function. Note that for many distributional VBB obfuscators for point functions or even evasive functions (e.g. [WZ17, GKW17]) appeared in the literature, their simulators are essentially simulating an obfuscated null circuit that is indistinguishable from the real obfuscated circuit, so those constructions immediately satisfy our definition.

Formally, let us first define unpredictable distributions which are used in the definition of obfuscators for point functions.

Definition 2.8 (Computationally unpredictable distribution)**.** A distribution ensemble on $D = \{D_\lambda = (Z_\lambda, X_\lambda)\}_{\lambda\in\mathbb{N}}$ is computationally unpredictable if for every poly-size circuit family $\{C_\lambda\}_{\lambda\in\mathbb{N}}$ and for all sufficiently large λ,

$$\Pr_{(z,x)\leftarrow D_\lambda}[C_\lambda(z) = x] = \mathtt{negl}(\lambda).$$

Definition 2.9 (Auxiliary input point obfuscation for computationally unpredictable distributions (AIPO))**.** A PPT algorithm AIPO is a ***point obfuscator for computationally unpredictable distributions*** if it satisfies the following properties.

- Correctness. On input x, it outputs a polynomial-size circuit that computes that point function $\mathbf{1}_x$; on input $(1^\lambda, \mathsf{null})$, it outputs a polynomial-size circuit that computes the all-zero function of input length λ, where null is a special symbol.

- Indistinguishability from a null program. For every (efficiently samplable) unpredictable distribution $\mathcal{B}_1$ over $\{0,1\}^* \times \{0,1\}^\lambda$ and every PPT algorithm $\mathcal{B}_2$,

$$\left|\mathbf{Pr}\left[\mathrm{AIPO}_{\mathsf{AIPO},(\mathcal{B}_1,\mathcal{B}_2)}(1^\lambda) \Rightarrow 1\right] - \frac{1}{2}\right| = \mathtt{negl}(\lambda),$$

  where the experiment $\mathrm{AIPO}_{\mathsf{AIPO},(\mathcal{B}_1,\mathcal{B}_2)}(1^\lambda)$ proceeds as follows.

  1. $(x,z) \leftarrow \mathcal{B}_1(1^\lambda)$;
  2. $b \leftarrow \{0,1\}$;
  3. $C_0 \leftarrow \mathsf{AIPO}(x)$, $C_1 \leftarrow \mathsf{AIPO}(1^\lambda, \mathsf{null})$;
  4. $b' \leftarrow \mathcal{B}_2(z, C_b)$; the experiment outputs 1 iff $b = b'$.

The circuit complexity of AIPO. Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a circuit class, where each $C \in \mathcal{C}_\lambda$ has input length $\ell_{\mathsf{in}}(\lambda)$. We say AIPO is supported in $\mathcal{C}$ if for all $\lambda \in \mathbb{N}, x \in \{0,1\}^{\ell_{\mathsf{in}}(\lambda)}$, $\mathsf{AIPO}(x)$ always output a circuit in $\mathcal{C}_\lambda$.

Candidate AIPO in NC^1 from a variant of the LWE assumption. Although we only need the *existence* of AIPO in NC^1, let us mention a *concrete* candidate construction of AIPO in NC^1 from a variant of the LWE assumption.

The construction is very simple: To obfuscate a point $\mathbf{s} \in \{0,1\}^n$, we treat $\mathbf{s}$ as an LWE secret, and outputs $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{y} = \mathbf{A}^T\mathbf{s} + \mathbf{e} \bmod q$ as the point obfuscation for $\mathbf{s}$, where q is a prime, $\mathbf{e} \in \mathbb{Z}^m$ satisfies $\|\mathbf{e}\|_\infty < B$ for some bound $B \in \mathtt{poly}(n)$ such that $B^m < q^{m-n}$. The evaluation algorithm takes a point $\mathbf{x} \in \{0,1\}^n$, outputs 1 iff $\|\mathbf{A}^T\mathbf{x} - \mathbf{y} \bmod q\|_\infty < B$. For security, q is chosen to be super-polynomial in n. Suppose the min-entropy of $\mathbf{s}$ is k, then $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{y} = \mathbf{A}^T\mathbf{s} + \mathbf{e} \bmod q$ are indistinguishable from random (based on the standard LWE assumption where the secret is sampled uniformly from $\mathbb{Z}_q^\ell$ where $\ell = \frac{k-\omega(\log n)}{\log q}$) [GKPV10]. Furthermore, we conjecture the security holds when $(\mathbf{s}, z)$, where z is the auxiliary input, is sampled from a computational unpredictable distribution (in which case no reduction from the standard LWE is known). Other parameter settings can be found in, e.g., [BD20].

3 Constructing UCE from Oblivious Programmable Function and AIPO

In this section, we formally define UCE and present a construction from (1) a new primitive called obliviously programmable function (defined in section 3.2) plus (2) AIPO.

3.1 UCE

UCE is a security notion defined for a keyed function family. We first recall its syntax.

A keyed function family $\mathsf{H} = \left\{\mathsf{H}_\lambda : \mathcal{K}_\lambda \times \{0,1\}^{\mathsf{H}.\ell_{\mathsf{in}}(\lambda)} \to \mathcal{R}_\lambda\right\}_{\lambda\in\mathbb{N}}$ consists of a pair of algorithms $(\mathsf{H}.\mathsf{Gen}, \mathsf{H}.\mathsf{Eval})$ with the following syntax.

- $\mathsf{Gen}(1^\lambda) \mapsto \mathsf{hk} \in \mathcal{K}_\lambda$. The key generation algorithm Gen outputs a key hk on input security parameter 1^λ.
- $\mathsf{Eval}(\mathsf{hk}, x) \mapsto y \in \mathcal{R}_\lambda$. When $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda), x \in \{0,1\}^{\mathsf{H}.\ell_{\mathsf{in}}(\lambda)}$, the evaluation algorithm outputs a value $y \in \mathcal{R}_\lambda$.

The UCE security is defined by a two-stage game. The first player has access to an oracle Hash that is either $\mathsf{H}.\mathsf{Eval}(\mathsf{hk}, \cdot)$ or a random oracle. The second player receives a message from the first player and is given hk; it has to guess to which oracle the first player has access.

Definition 3.1. Let (S, D) be an adversary, where S is called the source, and D is called the distinguisher. We associate them with the game in fig. 1. Define the advantage as

$$\mathbf{Adv}^{\mathsf{uce}}_{\mathsf{H},(S,D)}(\lambda) \overset{\mathsf{def}}{=} \left|\mathbf{Pr}\left[\mathrm{UCE}^{\mathsf{H}}_{S,D}(\lambda) \Rightarrow 1\right] - \frac{1}{2}\right|.$$

We say H is $\mathcal{S}$-UCE-secure, denoted by $\mathsf{H} \in \mathrm{UCE}[\mathcal{S}]$, if for every source $S \in \mathcal{S}$ and PPT D, $\mathbf{Adv}^{\mathsf{uce}}_{\mathsf{H},(S,D)}(\lambda) = \mathtt{negl}(\lambda)$.

```
UCE$^{\mathsf{H}}_{S,D}(\lambda)$
1 :  $\beta \leftarrow \{0,1\}$, $\mathsf{hk} \leftarrow \mathsf{H.Gen}(1^\lambda)$
2 :  $L \leftarrow S^{\textsc{Hash}}$
3 :  $\beta' \leftarrow D(L, \mathsf{hk})$
4 :  return $[\![\beta = \beta']\!]$

Hash($x$)
1 :  if $x \notin Q$ then
2 :      $Q \leftarrow Q \cup \{x\}$
3 :      if $\beta = 0$ then $T[x] \leftarrow \mathcal{R}_\lambda$
4 :      else $T[x] \leftarrow \mathsf{H}(\mathsf{hk}, x)$
5 :      fi
6 :  fi
7 :  return $T[x]$
```

Figure 1: Games for defining UCE security.

```
Pred$^{S}_{P}(\lambda)$
1 :  $\mathsf{hk} \leftarrow \mathsf{H.Gen}(1^\lambda)$
2 :  $L \leftarrow S^{\textsc{Hash}}$
3 :  $x \leftarrow P(L)$
4 :  return $[\![x \in Q]\!]$

SPred$^{S}_{P}(\lambda)$
1 :  $\mathsf{hk} \leftarrow \mathsf{H.Gen}(1^\lambda)$
2 :  $L \leftarrow S^{\textsc{Hash}}$
3 :  $x \leftarrow P(L, T.\mathsf{values})$
4 :  return $[\![x \in Q]\!]$

Hash($x$)
1 :  if $x \notin Q$ then
2 :      $Q \leftarrow Q \cup \{x\}$
3 :      $T[x] \leftarrow \mathcal{R}_\lambda$
4 :  fi
5 :  return $T[x]$
```

Figure 2: Games for defining unpredictable sources. Here, $T.\mathsf{values}$ denote the set of all oracle answers (not query-answer pairs).

Definition 3.2. Consider the games in fig. 2. We say S is ***unpredictable*** if for every PPT predictor P, it holds that

$$\left|\mathbf{Pr}\left[\mathrm{Pred}^{S}_{P}(\lambda) \Rightarrow 1\right] - \frac{1}{2}\right| = \mathtt{negl}(\lambda).$$

Moreover, S is called ***strongly unpredictable*** if for every PPT predictor P,

$$\left|\mathbf{Pr}\left[\mathrm{SPred}^{S}_{P}(\lambda) \Rightarrow 1\right] - \frac{1}{2}\right| = \mathtt{negl}(\lambda).$$

Define $\mathcal{S}^{\mathsf{cup}}$ and $\mathcal{S}^{\mathsf{scup}}$ as the sets of unpredictable and strongly unpredictable sources, respectively. Furthermore, given a number $q \in \mathbb{N}$ and a class of sources $\mathcal{S}$, let

$$\mathcal{S}_q \overset{\mathsf{def}}{=} \{S \in \mathcal{S} : S \text{ makes at most } q \text{ queries to } \textsc{Hash}\}.$$

3.2 Obliviously Programmable Functions (OPF)

Syntax. Let $\ell_{\mathsf{in}} = \ell_{\mathsf{in}}(\lambda)$ be a length function. Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be a circuit class such that every circuit $C \in \mathcal{C}_\lambda$ has input length $\ell_{\mathsf{in}}(\lambda)$, namely, $C : \{0,1\}^{\ell_{\mathsf{in}}(\lambda)} \to \{0,1\}$. C is called a ***point circuit*** if $|C^{-1}(1)| \overset{\mathsf{def}}{=} \left|\left\{x \in \{0,1\}^{\ell_{\mathsf{in}}(\lambda)} : C(x) = 1\right\}\right| \leq 1$. We assume that there is a simple representation of point functions in $\mathcal{C}_\lambda$, i.e., for every $x \in \{0,1\}^{\ell_{\mathsf{in}}(\lambda)}$, there exists a simple and explicit $P_x \in \mathcal{C}_\lambda$ that computes $\mathbf{1}_x$.

An ***obliviously programmable function (OPF)*** for $\mathcal{C}$ consists of two algorithms $\Pi = (\mathsf{Program}, \mathsf{Eval})$ and specifies an codomain $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{N}}$.

1. $\mathsf{Program}(1^\lambda, C, y) \mapsto k_C$. The programming algorithm Program takes as input a point circuit $C \in \mathcal{C}_\lambda$ and $y \in \mathcal{R}_\lambda$, outputs a key k_C.

2. $\mathsf{Eval}(k, x) \mapsto y$. The evaluation algorithm Eval, on input a key k and a point $x \in \{0,1\}^{\ell_{\mathsf{in}}(\lambda)}$, outputs $y = \mathsf{Eval}(k, x) \in \mathcal{R}_\lambda$.

Correctness. There exists a negligible function ν such that for all $\lambda \in \mathbb{N}$, $C \in \mathcal{C}_\lambda$ and $y \in \mathcal{R}_\lambda$, if C computes the point function $\mathbf{1}_{x^*}$, then

$$\underset{k_C \leftarrow \Pi.\mathsf{Program}(1^\lambda, C, y)}{\mathbf{Pr}} [\mathsf{Eval}(k_C, x^*) = y] \geq 1 - \nu(\lambda).$$

The security of OPF consists of the following two features.

Privacy. We require $k_C \leftarrow \Pi.\mathsf{Program}(C, y)$ computationally hides C if (1) C is a point circuit and (2) y is chosen uniformly at random. Concretely, for every PPT adversary $\mathcal{A}$,

$$\left| \mathbf{Pr}\left[\mathrm{Priv}_{\Pi,\mathcal{A}}(1^\lambda) \Rightarrow 1 \right] - 1/2 \right| = \mathtt{negl}(\lambda),$$

where the experiment $\mathrm{Priv}_{\Pi,\mathcal{A}}(1^\lambda)$ is defined as follows:

1. On input 1^λ, $\mathcal{A}$ submit two point circuits $C_0, C_1 \in \mathcal{C}_\lambda$ to the challenger.

2. The challenger samples $b \leftarrow \{0,1\}$, $y \leftarrow \mathcal{R}_\lambda$, and send $k^* := \Pi.\mathsf{Program}(1^\lambda, C_b, y)$ back to $\mathcal{A}$.

3. On receiving k^*, $\mathcal{A}$ outputs b'; the experiment output 1 iff $b = b'$.

Value-Hiding (when programming on the all-zero function). We require that $k \leftarrow \Pi.\mathsf{Program}(C, y)$ computationally hides the value y whenever C computes the all-zero function. Formally, for every PPT adversary $\mathcal{A}$,

$$\left| \mathbf{Pr}\left[\mathrm{VH}_{\Pi,\mathcal{A}}(1^\lambda) \Rightarrow 1 \right] - 1/2 \right| = \mathtt{negl}(\lambda),$$

where the experiment $\mathrm{VH}_{\Pi,\mathcal{A}}(1^\lambda)$ is defined as follows:

1. On input 1^λ, $\mathcal{A}$ submits (C, y_0, y_1) to the challenger, where $y_0, y_1 \in \mathcal{R}_\lambda$ and C computes the all-zero function.

2. The challenger samples $b \leftarrow \{0,1\}$, and sends $k^* := \Pi.\mathsf{Program}(1^\lambda, C, y_b)$ back to $\mathcal{A}$.

3. On receiving k^*, $\mathcal{A}$ outputs b'; the experiment output 1 iff $b = b'$.

3.3 The Construction

Construction 1. Let $\Pi = (\Pi.\mathsf{Program}, \Pi.\mathsf{Eval})$ be an OPF for circuit class $\mathcal{C}$ with input length $\ell_{\mathsf{in}} = \ell_{\mathsf{in}}(\lambda)$ and codomain $\{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{N}}$. Consider the following construction of a keyed function family $\mathsf{H} = (\mathsf{H.Gen}, \mathsf{H.Eval})$:

- $\mathsf{H.Gen}(1^\lambda) \mapsto \mathsf{hk}$: $y \leftarrow \mathcal{R}_\lambda, k_\emptyset \leftarrow \Pi.\mathsf{Program}(C_\emptyset, y)$, where $C_\emptyset$ denotes the all-zero function; output $\mathsf{hk} := k_\emptyset$.
- $\mathsf{H.Eval}(\mathsf{hk}, x) \mapsto y$: output $y := \Pi.\mathsf{Eval}(\mathsf{hk}, x)$.

Theorem 3.3. *Let Π, H be as in construction 1. Assume that there exists an AIPO supported in $\mathcal{C}$, then* $\mathsf{H} \in \mathrm{UCE}[\mathcal{S}_1^{\mathsf{scup}}]$.

Proof. Let $S \in \mathcal{S}_1^{\mathsf{scup}}$ be a source and D be a PPT distinguisher. We start with Game_0, as shown in fig. 3, which is exactly the UCE game $\mathrm{UCE}^{\mathsf{H}}_{S,D}$ when the hidden bit β is fixed to 1 (i.e., HASH returns the real hash value). Since S only query HASH once, we use a variable x^* to record this query. Our goal is to replace $y^* := \mathsf{H.Eval}(\mathsf{hk}, x)$ by $y^* \leftarrow \mathcal{R}_\lambda$ (as in the last game Game_5) via a sequence of undetectable change.

```
Game_0, Game_5
1 : $x^* := \perp$
2 : $\mathsf{hk} \leftarrow$ H.Gen($1^\lambda$)
        1 : $y \leftarrow \mathcal{R}_\lambda$
        2 : $k_\emptyset \leftarrow \Pi.\mathsf{Program}(1^\lambda, C_\emptyset, y)$
        3 : $\mathsf{hk} := k_\emptyset$
3 : $L \leftarrow S^{\textsc{Hash}}$
4 : $\beta' \leftarrow D(\mathsf{hk}, L)$
5 : return $[\![1 = \beta']\!]$

Hash($x$)
1 : $x^* := x$
2 : $y^* := \mathsf{H.Eval}(\mathsf{hk}, x^*)$    // Game 0
3 : $y^* \leftarrow \mathcal{R}_\lambda$    // Game 5
4 : return $y^*$
```

Figure 3: Game_0 and Game_5

1. Game_0 in fig. 4 is the same as fig. 3, but we move the generation of $k_\emptyset$ into HASH, which is just a conceptual change.

2. Game_1. Game_1 is identical to Game_0 except that

   - $\mathsf{hk} := k_{x^*}$ where $k_{x^*} \leftarrow \Pi.\mathsf{Program}(1^\lambda, \mathsf{AIPO}(x^*), y)$.

   $\mathsf{Game}_0 \approx_c \mathsf{Game}_1$ readily follows from the privacy of Π (lemma 3.4).

3. Game_2. Game_2 is identical to Game_1 except that HASH directly returns the programmed value instead of computing $\Pi.\mathsf{Eval}(k_{x^*}, x^*)$. Also, we move the generation of k_{x^*} out of the oracle, so that the oracle behaves exactly as a random oracle. By the correctness of Π,

$$|\mathbf{Pr}[\mathsf{Game}_2 \Rightarrow 1] - \mathbf{Pr}[\mathsf{Game}_1 \Rightarrow 1]| = \mathtt{negl}(\lambda).$$

```
Game_0
1 : $x^* := \bot$
2 : $L \leftarrow S^{\textsc{Hash}}$
3 : $\mathsf{hk} := k_\emptyset$
4 : $\beta' \leftarrow D(\mathsf{hk}, L)$
5 : return $[\![1 = \beta']\!]$
```

```
Game_1
1 : $x^* := \bot$
2 : $L \leftarrow S^{\textsc{Hash}'}$
3 : $\mathsf{hk} := k_{x^*}$
4 : $\beta' \leftarrow D(\mathsf{hk}, L)$
5 : return $[\![1 = \beta']\!]$
```

```
Game_2
1 : $x^* := \bot$
2 : $L \leftarrow S^{\mathrm{RO}}$
3 : $k_{x^*} \leftarrow \Pi.\mathsf{Program}(1^\lambda, \mathsf{AIPO}(x^*), y^*))$
4 : $\mathsf{hk} := k_{x^*}$
5 : $\beta' \leftarrow D(\mathsf{hk}, L)$
6 : return $[\![1 = \beta']\!]$
```

```
Game_3
1 : $x^* := \bot$
2 : $msk \leftarrow \Pi.\mathsf{Gen}(1^\lambda)$
3 : $L \leftarrow S^{\mathrm{RO}}$
4 : $k_\emptyset \leftarrow \Pi.\mathsf{Program}(msk, \mathsf{AIPO}(\mathsf{null}), y^*)$
5 : $\mathsf{hk} := k_\emptyset$
6 : $\beta' \leftarrow D(\mathsf{hk}, L)$
7 : return $[\![1 = \beta']\!]$
```

```
Game_4
1 : $x^* := \bot$
2 : $msk \leftarrow \Pi.\mathsf{Gen}(1^\lambda)$
3 : $L \leftarrow S^{\mathrm{RO}}$
4 : $y \leftarrow \mathcal{R}_\lambda$
5 : $k_\emptyset \leftarrow \Pi.\mathsf{Program}(msk, \mathsf{AIPO}(\mathsf{null}), y)$
6 : $\mathsf{hk} := k_\emptyset$
7 : $\beta' \leftarrow D(\mathsf{hk}, L)$
8 : return $[\![1 = \beta']\!]$
```

```
Game_5
1 : $x^* := \bot$
2 : $msk \leftarrow \Pi.\mathsf{Gen}(1^\lambda)$
3 : $L \leftarrow S^{\mathrm{RO}}$
4 : $y \leftarrow \mathcal{R}_\lambda$
5 : $k_\emptyset \leftarrow \Pi.\mathsf{Program}(msk, C_\emptyset, y)$
6 : $\mathsf{hk} := k_\emptyset$
7 : $\beta' \leftarrow D(\mathsf{hk}, L)$
8 : return $[\![1 = \beta']\!]$
```

```
Hash(x)
1 : $x^* := x$
2 : $y \leftarrow \mathcal{R}_\lambda$
3 : $k_\emptyset \leftarrow \Pi.\mathsf{Program}(1^\lambda, C_\emptyset, y)$
4 : $y^* := \Pi.\mathsf{Eval}(k_u, x^*)$
5 : return $y^*$
```

```
Hash'(x)
1 : $x^* := x$
2 : $y \leftarrow \mathcal{R}_\lambda$
3 : $k_{x^*} \leftarrow \Pi.\mathsf{Program}(1^\lambda, \mathsf{AIPO}(x^*), y))$
4 : $y^* := \Pi.\mathsf{Eval}(k_{x^*}, x^*)$
5 : return $y^*$
```

```
RO(x)
1 : $x^* := x$
2 : $y^* \leftarrow \mathcal{R}_\lambda$
3 : return $y^*$
```

Figure 4: Games in the proof

$$\begin{tikzcd}
\mathsf{Game}_0 \arrow[r, bend left, "\text{Privacy}"] \arrow[d, phantom, "\equiv"] & \mathsf{Game}_1 \arrow[r, bend left, "\text{Correctness}"] & \mathsf{Game}_2 \arrow[r, bend left, "\text{AIPO}"] & \mathsf{Game}_3 \arrow[r, bend left, "\text{Value-Hiding}"] & \mathsf{Game}_4 \arrow[r, bend left, "\text{Privacy}"] & \mathsf{Game}_5 \arrow[d, phantom, "\equiv"] \\
\text{UCE game with real hash value } (\beta = 1) & & & & & \text{UCE game with random value } (\beta = 0)
\end{tikzcd}$$

Figure 5: Outline of the proof

4. Game_3. Game_3 is identical to Game_2 except that hk is replaced by

$$k_\emptyset \leftarrow \Pi.\mathsf{Program}(msk, \mathsf{AIPO}(\mathsf{null}), y^*).$$

   We shall prove $\mathsf{Game}_2 \approx_c \mathsf{Game}_3$ via the security of AIPO (lemma 3.5).

5. Game_4. Game_4 is identical to Game_3 except that

$$k_\emptyset \leftarrow \Pi.\mathsf{Program}(1^\lambda, \mathsf{AIPO}(\mathsf{null}), y),$$

   where $y \leftarrow \mathcal{R}_\lambda$ is a fresh random value (like in Game_0). Note that when programming on the all-zero function, the value y is hidden by the programmed key. Therefore, $\mathsf{Game}_4 \approx_c \mathsf{Game}_3$ readily follows from the value-hiding property of Π (lemma 3.6).

6. Game_5. Finally, Game_5 is identical to Game_4 except that $\mathsf{hk} = k_\emptyset$ is generated in the original way, removing AIPO:

$$k_\emptyset \leftarrow \Pi.\mathsf{Program}(msk, C_\emptyset, y).$$

   The two programmed keys are indistinguishable according to the privacy of Π, which is similar to is similar to $\mathsf{Game}_1 \approx_c \mathsf{Game}_0$. Hence,

$$|\mathbf{Pr}[\mathsf{Game}_5 \Rightarrow 1] - \mathbf{Pr}[\mathsf{Game}_4 \Rightarrow 1]| = \mathtt{negl}(\lambda)$$

   .

We are happy to see that Game_5 is exactly the UCE game with $\beta = 0$. Hence,

$$\begin{aligned}
&\left|\mathbf{Pr}\left[\mathrm{UCE}^{\mathsf{H}}_{S,D}(\lambda) \Rightarrow 1 \;\middle|\; \beta = 1\right] - \mathbf{Pr}\left[\mathrm{UCE}^{\mathsf{H}}_{S,D}(\lambda) \Rightarrow 1 \;\middle|\; \beta = 0\right]\right| \\
&= |\mathbf{Pr}[\mathsf{Game}_0 \Rightarrow 1] - \mathbf{Pr}[\mathsf{Game}_5 \Rightarrow 1]| \\
&= \mathtt{negl}(\lambda),
\end{aligned}$$

which implies that $\mathbf{Adv}^{\mathsf{uce}}_{\mathsf{H},(S,D)}(\lambda)$ is negligible.

It remains to prove the following lemmas on game-hopping. See fig. 5 for an outline.

Lemma 3.4. *By the privacy of* Π, $|\mathbf{Pr}[\mathsf{Game}_1 \Rightarrow 1] - \mathbf{Pr}[\mathsf{Game}_0 \Rightarrow 1]| = \mathtt{negl}(\lambda)$.

Proof. Consider the following adversary $\mathcal{A}$ that aims to break the privacy of Π.

1. Simulate $S^{(\cdot)}$ until S issues the oracle query x^*.

2. Submit the challenge $(C_0 = C_\emptyset, C_1 = \mathsf{AIPO}(x^*))$ and receive k^* from the challenger where k^* is generated in the following way: $b \leftarrow \{0,1\}, y \leftarrow \mathcal{R}_\lambda, k^* \leftarrow \Pi.\mathsf{Program}(1^\lambda, C_b, y)$.

3. Forward $y := \Pi.\mathsf{Eval}(k^*, x^*)$ to $S^{\textsc{Hash}}$ and get L.

4. Simulate $D(\mathsf{hk} = k^*, L)$, and let β' denote the output of D.

5. Output $b' := \beta'$.

When $b = 0$, $\mathcal{A}$ simulates Game_0; when $b = 1$, $\mathcal{A}$ simulates Game_1. Therefore,

$$|\mathbf{Pr}[\mathsf{Game}_1 \Rightarrow 1] - \mathbf{Pr}[\mathsf{Game}_0 \Rightarrow 1]| = 2\left|\mathrm{Priv}_{\Pi,\mathcal{A}}(1^\lambda) - 1/2\right| = \mathtt{negl}(\lambda).$$

□

Lemma 3.5. *By the security of AIPO,*

$$|\mathbf{Pr}[\mathsf{Game}_3 \Rightarrow 1] - \mathbf{Pr}[\mathsf{Game}_2 \Rightarrow 1]| = \mathtt{negl}(\lambda).$$

Proof. Consider the adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that aims to break the security of AIPO, as shown in fig. 6. The output of $\mathcal{B}_1$ is an unpredictable ensemble since S is an unpredictable source. Recall that the AIPO experiment $\mathrm{AIPO}_{\mathsf{AIPO},\mathcal{B}}(1^\lambda)$ proceeds as follows:

1. $(x, z) \leftarrow \mathcal{B}_1(1^\lambda)$;
2. $b \leftarrow \{0, 1\}$;
3. $C_0 \leftarrow \mathsf{AIPO}(x)$, $C_1 \leftarrow \mathsf{AIPO}(\mathsf{null})$
4. $b' \leftarrow \mathcal{B}_2(z, C_b)$; the experiment outputs 1 iff $b = b'$.

By the assumption that S is strongly unpredictable, $\mathcal{B}_1$ is computationally unpredictable, and hence the security of the AIPO guarantees that

$$\left|\mathbf{Pr}\left[\mathrm{AIPO}_{\mathsf{AIPO},\mathcal{B}}(1^\lambda) \Rightarrow 1\right] - 1/2\right| = \mathtt{negl}(\lambda).$$

Note that conditioned on $b = 0$, the AIPO game is exactly Game_2, and conditioned on $b = 1$, it is exactly Game_3. Therefore,

$$\begin{aligned}|\mathbf{Pr}[\mathsf{Game}_3 \Rightarrow 1] - \mathbf{Pr}[\mathsf{Game}_2 \Rightarrow 1]| &\le 2\left|\mathbf{Pr}\left[\mathrm{AIPO}_{\mathsf{AIPO},\mathcal{B}}(1^\lambda) \Rightarrow 1\right] - 1/2\right| \\ &= \mathtt{negl}(\lambda).\end{aligned}$$

```
$\mathcal{B}_1(1^\lambda)$
1 : $y^* \leftarrow \mathcal{R}_\lambda$
2 : Simulate $S^{\text{HASH}}$ with $y^*$ as oracle answer
3 : let $x^*$ and $L$ be the query and output of $S$ respectively.
4 : return $(x^*, z = (L, y^*))$
$\mathcal{B}_2(z, C)$
1 : Parse $z = (L, y^*)$
2 : $k_C \leftarrow \Pi.\mathsf{Program}(1^\lambda, C, y^*)$
3 : return $D(\mathsf{hk} = k_C, L)$
```

Figure 6: Adversary $\mathcal{B}$ for AIPO

□

Lemma 3.6. *By the value-hiding property of* Π,

$$|\mathbf{Pr}[\mathsf{Game}_4 \Rightarrow 1] - \mathbf{Pr}[\mathsf{Game}_3 \Rightarrow 1]| = \mathtt{negl}(\lambda).$$

Proof. Consider the following adversary $\mathcal{A}$ that aims to break the value-hiding property of Π.

1. Simulate $S^{(\cdot)}$ until S issue the oracle query x^* and reply with $y_0 \leftarrow \mathcal{R}_\lambda$. Let L be the output of S.

2. Samples $y_1 \leftarrow \mathcal{R}_\lambda$ and $C \leftarrow \mathsf{AIPO}(\mathsf{null})$. Submit the challenge (C, y_0, y_1) and receive k^* from the challenger where k^* is generated as follows: $b \leftarrow \{0,1\}, k^* \leftarrow \Pi.\mathsf{Program}(1^\lambda, C, y_b)$.

3. Simulate $D(\mathsf{hk} = k^*, L)$, and let β' denote the output of D.

4. Output $b' := \beta'$.

When $b = 0$, $\mathcal{A}$ simulates Game_3; when $b = 1$, $\mathcal{A}$ simulate Game_4. Therefore,

$$|\mathbf{Pr}[\mathsf{Game}_3 \Rightarrow 1] - \mathbf{Pr}[\mathsf{Game}_4 \Rightarrow 1]| = 2\left|\mathrm{VH}_{\Pi,\mathcal{A}}(1^\lambda) - 1/2\right| = \mathtt{negl}(\lambda).$$

□

□

4 Obliviously Programmable Function from Lattice Assumptions

In this section, we present a construction of OPF based on GGH15 encodings [GGH15]. We draw on LWE with subexponential hardness and evasive LWE assumption to prove the security of our construction.

Theorem 4.1. *Let* λ *be the security parameter and* $\ell_{\mathsf{in}}(\lambda), w(\lambda) = \lambda^{O(1)}$. *Assume LWE with subexponential hardness and evasive LWE. Then there exists an OPF with input length* ℓ_{in} *for* $\mathsf{MBP}^1_{\ell_{\mathsf{in}},w}$, *where* $\mathsf{MBP}^1_{\ell_{\mathsf{in}},w} = \left\{\mathsf{MBP}^1_{\ell_{\mathsf{in}}(\lambda),w(\lambda)}\right\}_{\lambda\in\mathbb{N}}$. *(Recall that* $\mathsf{MBP}^1_{\ell,w}$ *denotes the set of read-once matrix branching programs with input length* ℓ *and width* w.*)*

Evasive LWE is a new assumption proposed by Wee [Wee22]; we formally state it in section 4.2 and use it to prove the security property of GGH 15 encodings we need.

Our construction can be generalized to any read-c (matrix) branching programs (see appendix A for more detail):

Theorem 4.2. *Let* λ *be the security parameter and* $\ell_{\mathsf{in}}(\lambda), c(\lambda), w(\lambda) = \lambda^{O(1)}$. *Under the subexponential LWE assumption and the evasive LWE assumption, there exists an OPF with input length* ℓ_{in} *for* $\mathsf{MBP}^c_{\ell_{\mathsf{in}},w}$.

Together with theorem 3.3, we conclude that

Theorem 4.3. *Let* λ *be the security parameter and* $\ell_{\mathsf{in}}(\lambda), c(\lambda), w(\lambda) = \lambda^{O(1)}$. *Under the subexponential LWE assumption and the evasive LWE assumption, if there exists an AIPO supported in* $\mathsf{MBP}^c_{\ell_{\mathsf{in}},w}$, *then there exists a* $\mathcal{S}_1^{\mathsf{scup}}$*-UCE-secure function with input length* ℓ_{in}.

Such branching programs can represent the NC^1 circuit class. Meanwhile, a $\mathcal{S}_1^{\mathsf{scup}}$-UCE-secure function is a universal hardcore function, proven in [BHK13, BM14]. Hence we conclude that

Corollary 4.4. *Under the subexponential LWE assumption and the evasive LWE assumption, if there exists an AIPO that always outputs an* NC^1 *circuit, then there exists a* $\mathcal{S}_1^{\mathsf{scup}}$*-UCE-secure function. Consequently, there exists a universal hard-core function that outputs a polynomial number of bits.*

4.1 OPF Construction Based on GGH15 Encodings

GGH15 encodings. We first describe generalized GGH15 encodings, following [GGH15, CVW18, VWW22].

Construction 2 (GGH15 encodings)**.** The randomized algorithm $\mathsf{GGH.Encode}$ takes the following inputs

- parameters $1^\lambda, h, m, q, \widehat{n}_0, \widehat{n} \in \mathbb{N}$ and Gaussian parameters $\sigma_1, \sigma_2, \sigma_3, \sigma_4$,
- matrices $\left\{\widehat{\mathbf{S}}_{1,b} \in \mathbb{Z}_q^{\widehat{n}_0 \times \widehat{n}}, \widehat{\mathbf{S}}_{2,b}, \dots, \widehat{\mathbf{S}}_{h,b} \in \mathbb{Z}_q^{\widehat{n} \times \widehat{n}}\right\}_{b \in \{0,1\}}, \mathbf{A}_h \in \mathbb{Z}_q^{\widehat{n} \times m}$,

and proceeds as follows.

1. Sample $(\mathbf{A}_i, \tau_i) \leftarrow \mathsf{TrapGen}(1^\lambda, q)$ for $i \in [h-1]$.
2. Sample $\mathbf{E}_{1,b} \leftarrow \mathcal{D}_{\sigma_1}^{\widehat{n}_0 \times \widehat{n}}$ and $\mathbf{E}_{2,b}, \dots, \mathbf{E}_{h,b} \leftarrow \mathcal{D}_{\sigma_4}^{\widehat{n} \times \widehat{n}}$ for $i = 2, \dots h, b \in \{0,1\}$.
3. Compute
$$\mathbf{C}_b := \widehat{\mathbf{S}}_{1,b}\mathbf{A}_1 + \mathbf{E}_{1,b} \text{ and } \mathbf{D}_{i,b} := \mathbf{A}_{i-1}^{-1}(\widehat{\mathbf{S}}_{i,b}\mathbf{A}_i + \mathbf{E}_{i,b}, \sigma_3)$$
for $i = 2, \dots, h, b \in \{0,1\}$, where $\mathbf{A}_{i-1}^{-1}(\cdot, \sigma_3)$ is computed using trapdoor τ_{i-1}.
4. Outputs $\mathbf{C}_0, \mathbf{C}_1, \{\mathbf{D}_{i,b}\}_{i=2,\dots,h,b\in\{0,1\}}$.

The functionality of GGH15 encodings allows us to approximate $\widehat{\mathbf{S}}_{\mathbf{x}}\mathbf{A}_h$ by $\mathbf{C}_{x_1} \cdot \prod_{i=2}^h \mathbf{D}_{1,x_i}$, which is stated formally in the next lemma.

Lemma 4.5 (Correctness of GGH15 encodings, see, e.g., [CVW18], Lemma 5.3)**.** *Let $\mathbf{A}_h \in \mathbb{Z}_q^{\widehat{n} \times m}$. For all $\mathbf{x} \in \{0,1\}^h$, with high probability over*

$$\mathbf{C}_0, \mathbf{C}_1, \{\mathbf{D}_{i,b}\}_{i=2,\dots,h,b\in\{0,1\}} \leftarrow \mathsf{GGH.Encode}(\left\{\widehat{\mathbf{S}}_{i,b}\right\}_{i\in[h],b\in\{0,1\}}, \mathbf{A}_h),$$

it holds that

$$\left\|\mathbf{C}_{x_1} \cdot \prod_{i=2}^{h} \mathbf{D}_{1,x_i} - \prod_{i=1}^{h} \widehat{\mathbf{S}}_{i,x_i} \cdot \mathbf{A}_h\right\|_\infty \leq h \cdot \sigma_1 \cdot \left((\sigma_3 + \sigma_4)m \cdot \max_{i\in[h],b\in\{0,1\}} \left\|\widehat{\mathbf{S}}_{i,b}\right\|_\infty\right)^h,$$

where $\left\|\widehat{\mathbf{S}}_{i,b}\right\|_\infty$ is the largest absolute value among all entries of $\widehat{\mathbf{S}}_{i,b}$.

The OPF construction. For simplicity, we present the construction for read-once MBPs. The construction can be generalized to read-c MBPs, which is presented in appendix A.

Construction 3. $\Pi = (\Pi.\mathsf{Program}, \Pi.\mathsf{Eval})$. Input length $\ell_{\mathsf{in}} = h$ and codomain $\mathcal{R}_\lambda = \mathbb{Z}_2^{n\times m}$. Let $\widehat{n}_0 \stackrel{\mathsf{def}}{=} n, \widehat{n} \stackrel{\mathsf{def}}{=} n + nw$. For a matrix $\mathbf{U} \in \mathbb{Z}_q^{\widehat{n}\times t}$, we use $\overline{\mathbf{U}} \in \mathbb{Z}_q^{n\times t}$ to denote the top n rows of $\mathbf{U}$ and $\underline{\mathbf{U}} \in \mathbb{Z}_q^{nw\times t}$ the bottom nw rows.

- $\mathsf{Program}(1^\lambda, C, \mathbf{Y} \in \mathbb{Z}_2^{n\times m}) \mapsto k_C$.

1. Parse C as a read-once MBP $\Gamma = (\mathbf{v}, \{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}})$.
2. Sample $\mathbf{S}_{i,b} \leftarrow \mathcal{D}^{n\times n}_{2\sqrt{n}}$ for $i \in [h], b \in \{0,1\}$ and set
$$\widehat{\mathbf{S}}_{1,b} = \left(\mathbf{I}_n \mid \mathbf{v}^\top \mathbf{M}_{1,b} \otimes \mathbf{S}_{1,b}\right), \widehat{\mathbf{S}}_{i,b} = \begin{pmatrix} \mathbf{I}_n & \\ & \mathbf{M}_{i,b} \otimes \mathbf{S}_{i,b} \end{pmatrix},$$
for $i = 2, \ldots, h, b \in \{0,1\}$.
3. Sample $\mathbf{A}_h \leftarrow \mathbb{Z}_q^{\widehat{n}\times m}$ conditioned on $\lfloor \overline{\mathbf{A}_h} \rceil_2 = \mathbf{Y}$ and output
$$k_C := \mathbf{C}_0, \mathbf{C}_1, \{\mathbf{D}_{i,b}\}_{i=2,\ldots,h,b\in\{0,1\}} \leftarrow \mathsf{GGH.Encode}(\left\{\widehat{\mathbf{S}}_{i,b}\right\}_{i\in[h],b\in\{0,1\}}, \mathbf{A}_h).$$

- $\mathsf{Eval}(k_C, \mathbf{x} \in \{0,1\}^h) \mapsto \mathbf{Y} \in \mathbb{Z}_2^{n\times m}$. Parse $k_C = \mathbf{C}_0, \mathbf{C}_1, \{\mathbf{D}_{i,b}\}_{i=2,\ldots,h,b\in\{0,1\}}$ and output
$$\mathbf{Y} := \left\lfloor \mathbf{C}_{x_1} \prod_{i=2}^{h} \mathbf{D}_{i,x_i} \right\rceil_2.$$

Parameter setting. Our parameter setting is similar to that of [VWW22]. The adversary runs in time $\texttt{poly}(\lambda)$. We rely on 2^{n^δ}-hardness for LWE (i.e., indistinguishability against adversaries 2^{n^δ} and a modulus-to-noise ratio of 2^{n^δ}). We have the following requirements for our parameters:

$$\begin{aligned}
2^{n^\delta} &> \max\left\{2^{h\lambda^2}, q/\sigma_4\right\} && \text{LWE hardness}\\
\sigma_2 &= \lambda^h \cdot \sigma_4 \cdot \lambda^{\omega(1)} && \text{noise flooding}\\
\sigma_1 &= \sigma_2 \cdot \lambda^{\omega(1)} && \text{evasive LWE \& noise flooding}\\
q &\geq 4h \cdot \sigma_1 \cdot ((\sigma_3 + \sigma_4)m \cdot \lambda\sqrt{n})^h && \text{correctness}\\
\sigma_3 &= 2\sqrt{n(w+1)\log q}, m = 2n(w+1)\log q && \text{trapdoor sampling}
\end{aligned}$$

A possible setting satisfying the constraints above is:
$$n = \left(h^2\lambda\right)^{1/\delta}, \quad q = 2^{n^\delta} = 2^{h^2\lambda}, \quad \sigma_4 = \Theta(n), \quad m = 2n(w+1)\log q.$$

Correctness. Correctness of construction 3 readily follows from the correctness of GGH15 encoding and the parameter setting above. Suppose that
$$k_C = \mathbf{C}_0, \mathbf{C}_1, \{\mathbf{D}_{i,b}\}_{i=2,\ldots,h,b\in\{0,1\}} \leftarrow \Pi.\mathsf{Program}(1^\lambda, C, \mathbf{Y}).$$

With overwhelming probability, we have $\max_{i\in[h],b\in\{0,1\}} \left\|\widehat{\mathbf{S}}_{i,b}\right\| \leq \lambda\sqrt{n}$, Hence, by lemma 4.5, for all $\mathbf{x} \in \{0,1\}^h$, it holds (with high probability) that
$$\mathbf{C}_{1,x_1} \prod_{i=2}^{h} \mathbf{D}_{i,x_i} \approx \widehat{\mathbf{S}}_{\mathbf{x}} \mathbf{A}_h = \left(\mathbf{I}_n \mid \mathbf{v}^\top \mathbf{M}_{\mathbf{x}} \otimes \mathbf{S}_{\mathbf{x}}\right) \cdot \mathbf{A}_h = \overline{\mathbf{A}_h} + (\mathbf{v}^\top \mathbf{M}_{\mathbf{x}} \otimes \mathbf{S}_{\mathbf{x}}) \cdot \underline{\mathbf{A}_h},$$

where the $\approx$ is up to an additive factor of $B = h \cdot \sigma_1 \cdot ((\sigma_3 + \sigma_4)m \cdot \lambda\sqrt{n})^h$.

If $C(x) = 1$, meaning that $\mathbf{v}^\top \mathbf{M}_\mathbf{x} = \mathbf{0}$, we will have

$$\mathsf{Eval}(k_C, \mathbf{x}) = \left\lfloor \mathbf{C}_{x_1} \prod_{i=2}^{h} \mathbf{D}_{i,x_i} \right\rceil_2 = \left\lfloor \overline{\mathbf{A}_h} + (\mathbf{v}^\top \mathbf{M}_\mathbf{x} \otimes \mathbf{S}_\mathbf{x}) \cdot \underline{\mathbf{A}_h} \right\rceil_2 = \left\lfloor \overline{\mathbf{A}_h} \right\rceil_2,$$

where the second equality holds since we set $q \geq 4B$. In the running of Program, it is guaranteed that $\left\lfloor \overline{\mathbf{A}_h} \right\rceil_2 = \mathbf{Y}$, and hence we have $\mathsf{Eval}(k_C, \mathbf{x}) = \mathbf{Y}$ as required.

4.2 Security of GGH15 Encodings from Evasive LWE

We now prove the pseudorandomness of GGH15 encoding with respect to the distributions of $\left\{\widehat{\mathbf{S}}_{i,b}\right\}_{i\in[h], b\ in\{0,1\}}, \mathbf{A}_h$ involved in our construction: lemma 4.6 and lemma 4.7.

Lemma 4.6. *Let* $\Gamma = \left(\mathbf{v} \in \{0,1\}^w, \{\mathbf{M}_{i,b} \in \{0,1\}^{w\times w}\}_{i\in[h], b\in\{0,1\}}\right)$ *be a read-once MBP such that* $|\Gamma^{-1}(1)| \leq 1$. *Let* $\mathbf{C}_0, \mathbf{C}_1, \{\mathbf{D}_{i,b}\}_{i=2,\ldots,h, b\in\{0,1\}}$ *be generated as follows.*

1. *Sample* $\mathbf{S}_{i,b} \leftarrow \mathcal{D}^{n\times n}_{2\sqrt{n}}$ *for* $i \in [h], b \in \{0,1\}$, $\mathbf{A}_h \leftarrow \mathbb{Z}_q^{\widehat{m}\times m}$ *and set*

$$\widehat{\mathbf{S}}_{1,b} := \left(\mathbf{I}_n \mid \mathbf{v}^\top \mathbf{M}_{1,b} \otimes \mathbf{S}_{1,b}\right), \widehat{\mathbf{S}}_{i,b} := \begin{pmatrix} \mathbf{I}_n & \\ & \mathbf{M}_{i,b} \otimes \mathbf{S}_{i,b} \end{pmatrix},$$

2. *Output* $\mathbf{C}_0, \mathbf{C}_1, \{\mathbf{D}_{i,b}\}_{i=2,\ldots,h, b\in\{0,1\}} \leftarrow \mathsf{GGH.Encode}(\left\{\widehat{\mathbf{S}}_{i,b}\right\}_{i\in[h], b\in\{0,1\}}, \mathbf{A}_h)$.

Then, by LWE and the evasive LWE assumption,

$$\{\mathbf{C}_b\}_{b\in\{0,1\}}, \{\mathbf{D}_{i,b}\}_{i=2,\ldots,h, b\in\{0,1\}} \approx_c \{\mathcal{U}(\mathbb{Z}_q^{n\times m})\}_{b\in\{0,1\}}, \{\mathcal{D}^{m\times m}_{\sigma_3}\}_{i=2,\ldots,h, b\in\{0,1\}}.$$

Lemma 4.7. *Fix* $\overline{\mathbf{A}}_h \in \mathbb{Z}_q^{n\times m}$. *Let* $\Gamma = \left(\mathbf{v} \in \{0,1\}^w, \{\mathbf{M}_{i,b} \in \{0,1\}^{w\times w}\}_{i\in[h], b\in\{0,1\}}\right)$ *be a read-once MBP that computes the all-zero function. Let* $\mathbf{C}_0, \mathbf{C}_1, \{\mathbf{D}_{i,b}\}_{i=2,\ldots,h, b\in\{0,1\}}$ *be generated as follows.*

1. *Sample* $\mathbf{S}_{i,b} \leftarrow \mathcal{D}^{n\times n}_{2\sqrt{n}}$ *for* $i \in [h], b \in \{0,1\}$, $\underline{\mathbf{A}_h} \leftarrow \mathbb{Z}^{nw\times m}$, *and set*

$$\widehat{\mathbf{S}}_{1,b} := \left(\mathbf{I}_n \mid \mathbf{v}^\top \mathbf{M}_{1,b} \otimes \mathbf{S}_{1,b}\right), \widehat{\mathbf{S}}_{i,b} := \begin{pmatrix} \mathbf{I}_n & \\ & \mathbf{M}_{i,b} \otimes \mathbf{S}_{i,b} \end{pmatrix}, \mathbf{A}_h := \begin{pmatrix} \overline{\mathbf{A}_h} \\ \underline{\mathbf{A}_h} \end{pmatrix}.$$

2. *Output* $\mathbf{C}_0, \mathbf{C}_1, \{\mathbf{D}_{i,b}\}_{i=2,\ldots,h, b\in\{0,1\}} \leftarrow (\left\{\widehat{\mathbf{S}}_{i,b}\right\}_{i\in[h], b\in\{0,1\}}, \mathbf{A}_h)$.

Then, by LWE and the evasive LWE assumption,

$$\{\mathbf{C}_b\}_{b\in\{0,1\}}, \{\mathbf{D}_{i,b}\}_{i=2,\ldots,h, b\in\{0,1\}}, \overline{\mathbf{A}_h} \approx_c \{\mathcal{U}(\mathbb{Z}_q^{n\times m})\}_{b\in\{0,1\}}, \{\mathcal{D}^{m\times m}_{\sigma_3}\}_{i=2,\ldots,h, b\in\{0,1\}}, \overline{\mathbf{A}_h}.$$

The proofs of the two lemmas above follow the proof structure of [VWW22].

1. First, relying on evasive LWE, proving the pseudorandomness of GGH15 encodings

$$\mathbf{C}_0, \mathbf{C}_1, \{\mathbf{D}_{i,b}\}_{i=2,\ldots,h,b\in\{0,1\}} \leftarrow \mathsf{GGH.Encode}(\left\{\widehat{\mathbf{S}}_{i,b}\right\}_{i\in[h],b\in\{0,1\}}, \mathbf{A}_h)$$

   is reduced to proving that all the evaluation products $\left\{\widehat{\mathbf{S}}_{\mathbf{x}'}\mathbf{A}_j + \mathbf{E}_{\mathbf{x}'}\right\}_{j\in[h],\mathbf{x}'\in\{0,1\}^j}$ are pseudorandom. Here, $\mathbf{E}_{\mathbf{x}'}$ are independent errors and $\mathbf{A}_j \leftarrow \mathbb{Z}_q^{n\times m}$ for $j \in [h-1]$, but the distribution of $\mathbf{A}_h$ could be tailored. This is done by lemma 4.8, which is directly from [VWW22].

2. It remains to show that all the evaluation products are indeed pseudorandom; we call this a 'precondition'. The only difference between lemma 4.6 and lemma 4.7 is that we have different distributions for $\mathbf{A}_h$. Hence, we verify the preconditions respectively in claim 4.10 and claim 4.11.

Below, we formally state the evasive LWE assumption.

Evasive LWE. Let Samp be a PPT algorithm that on input 1^λ, outputs

$$\mathbf{B} \in \mathbb{Z}_q^{n'\times n}, \mathbf{P} \in \mathbb{Z}_q^{n\times t}, \mathsf{aux} \in \{0,1\}^* .$$

Define the following two advantage functions (for adversaries $\mathcal{A}_0, \mathcal{A}_1$):

$$\mathbf{Adv}^{\mathsf{post}}_{\mathcal{A}_0}(\lambda) \overset{\mathsf{def}}{=} \left|\mathbf{Pr}\left[\mathcal{A}_0(\boxed{\mathbf{SB}+\mathbf{E}}, \boxed{\mathbf{SP}+\mathbf{E}'}, \mathsf{aux}) = 1\right] - \mathbf{Pr}\left[\mathcal{A}_0(\mathbf{C}, \mathbf{C}', \mathsf{aux}) = 1\right]\right|,$$
$$\mathbf{Adv}^{\mathsf{pre}}_{\mathcal{A}_1}(\lambda) \overset{\mathsf{def}}{=} \left|\mathbf{Pr}\left[\mathcal{A}_1(\boxed{\mathbf{SB}+\mathbf{E}}, \mathbf{D}, \mathsf{aux}) = 1\right] - \mathbf{Pr}\left[\mathcal{A}_1(\mathbf{C}, \mathbf{D}, \mathsf{aux}) = 1\right]\right|,$$

where

$$\begin{aligned}
&(\mathbf{S}, \mathbf{P}, \mathsf{aux}) \leftarrow \mathsf{Samp}\left(1^\lambda\right),\\
&\mathbf{B} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{E} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_1}^{n'\times m}, \mathbf{E}' \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_2}^{n'\times t},\\
&\mathbf{C} \leftarrow \mathbb{Z}_q^{n'\times m}, \mathbf{C}' \leftarrow \mathbb{Z}_q^{n'\times t},\\
&\mathbf{D} \leftarrow \mathbf{B}^{-1}(\mathbf{P}, \sigma_1).
\end{aligned}$$

We say that the evasive LWE assumption holds if for every PPT Samp there exists some polynomial $p(\cdot)$ such that for every PPT $\mathcal{A}_1$, there exists another PPT $\mathcal{A}_0$ satisfying

$$\mathbf{Adv}^{\mathsf{post}}_{\mathcal{A}_0}(\lambda) \geq \mathbf{Adv}^{\mathsf{pre}}_{\mathcal{A}_1}(\lambda)/p(\lambda) \text{ and } \mathsf{time}(\mathcal{A}_0) \leq \mathsf{time}(\mathcal{A}_1)\cdot p(\lambda).$$

We consider parameter settings for which $\sigma_2 \ll \sigma_1$ so that the precondition is stronger, which in turn makes evasive LWE weaker.

The reduction step is by the following lemma.

Lemma 4.8 (Lemma 5.1 in [VWW22]). *Fix some distributions for $\left\{\widehat{\mathbf{S}}_{i,b}\right\}_{i\in[h],b\in\{0,1\}}$ and let $\mathcal{E}$ be an efficiently samplable (and publicly known) distribution over $\mathbb{Z}_q^{\widehat{n}\times m}$. Suppose that for all $j \in [h]$, we have*

$$\left\{\boxed{\widehat{\mathbf{S}}_{\mathbf{x}'}\mathbf{A}_j + \mathbf{E}_{\mathbf{x}'}}\right\}_{\mathbf{x}'\in\{0,1\}^j}, \left\{\widehat{\mathbf{S}}_{i,b}\right\}_{i\in[h],b\in\{0,1\}} \approx_c \left\{\mathcal{U}(\mathbb{Z}_q^{\widehat{n}_0\times m})\right\}_{\mathbf{x}'\in\{0,1\}^j}, \left\{\widehat{\mathbf{S}}_{i,b}\right\}_{i\in[h],b\in\{0,1\}} \quad (2)$$

where $\mathbf{E}_{\mathbf{x}'} \leftarrow \mathcal{D}_{\sigma_1}^{\widehat{n}_0 \times m}$, $\mathbf{A}_j \leftarrow \mathbb{Z}_q^{\widehat{n} \times m}$ *for* $j \in [h-1]$, *and* $\mathbf{A}_h \leftarrow \mathcal{E}$. *Then, by the evasive LWE assumption, we have*

$$\{\mathbf{C}_b\}_{b\in\{0,1\}}, \{\mathbf{D}_{i,b}\}_{i=2,\ldots,h,b\in\{0,1\}}, \approx_c \left\{\mathcal{U}(\mathbb{Z}_q^{n\times m})\right\}_{b\in\{0,1\}}, \left\{\mathcal{D}_{\sigma_3}^{m\times m}\right\}_{i=2,\ldots,h,b\in\{0,1\}},$$

where

$$\{\mathbf{C}_b\}_{b\in\{0,1\}}, \{\mathbf{D}_{i,b}\}_{i=2,\ldots,h,b\in\{0,1\}} \leftarrow \mathsf{GGH.Encode}(\left\{\widehat{\mathbf{S}}_{i,b}\right\}_{i\in[h],b\in\{0,1\}}, \mathbf{A}_h), \mathbf{A}_h \leftarrow \mathcal{E}.$$

Remark 4.9. Two comments are in order regarding lemma 4.8.

1. It is required that the precondition eq. (2) holds with hardness $2^{h^2\lambda}$, namely, any adversary running within $2^{h^2\lambda}$ times has advantage at most $2^{-h^2\lambda}$.
2. For parameters, in the proof of this lemma, it is also required that $\sigma_1 = \sigma_2 \cdot \lambda^{\omega(1)}$ for invoking evasive LWE and $\sigma_2 = \lambda^h \cdot \sigma_4 \cdot \lambda^{\omega(1)}$ for noise flooding.

The following two claims verify the preconditions for two different distributions of $\mathbf{A}_h$.

Claim 4.10 (Precondition 1). *Fix a MBP*

$$\Gamma = \left(\mathbf{v} \in \{0,1\}^w, \{\mathbf{M}_{i,b} \in \{0,1\}^{w\times w}\}_{i\in[h],b\in\{0,1\}}\right)$$

such that $|\Gamma^{-1}(1)| \leq 1$. *Let*

$$\widehat{\mathbf{S}}_{1,b} := \left(\mathbf{I}_n \mid \mathbf{v}^\top \mathbf{M}_{1,b} \otimes \mathbf{S}_{1,b}\right), \widehat{\mathbf{S}}_{i,b} := \begin{pmatrix} \mathbf{I}_n & \\ & \mathbf{M}_{i,b} \otimes \mathbf{S}_{i,b} \end{pmatrix}.$$

where $\mathbf{S}_{i,b} \leftarrow \mathcal{D}_{2\sqrt{n}}^{n\times n}$ *for* $i \in [h], b \in \{0,1\}$. *Then, by the LWE assumption, for every* $j \in [h]$ *we have*

$$\left\{\boxed{\widehat{\mathbf{S}}_{\mathbf{x}'}\mathbf{A}_j + \mathbf{E}_{\mathbf{x}'}}\right\}_{\mathbf{x}'\in\{0,1\}^j}, \{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}} \approx_c \left\{\mathcal{U}(\mathbb{Z}_q^{n\times m})\right\}_{\mathbf{x}'\in\{0,1\}^j}, \{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \tag{3}$$

where $\mathbf{E}_{\mathbf{x}'} \leftarrow \mathcal{D}_{\sigma_1}^{n\times m}$, $\mathbf{A}_j \leftarrow \mathbb{Z}_q^{\widehat{n}\times m}$ *for* $j \in [h]$.

Proof. Fix an arbitrary $j \in [h]$. For all $\mathbf{x}' \in \{0,1\}^j$, we have

$$\begin{aligned}
F(\mathbf{x}') &\overset{\text{def}}{=} \widehat{\mathbf{S}}_{\mathbf{x}'}\mathbf{A}_j + \mathbf{E}_{\mathbf{x}'} \\
&= \left(\mathbf{I}_n \mid \mathbf{v}^\top\mathbf{M}_{\mathbf{x}'} \otimes \mathbf{S}_{\mathbf{x}'}\right) \cdot \mathbf{A}_h + \mathbf{E}_{\mathbf{x}'} \\
&= \overline{\mathbf{A}_h} + (\mathbf{v}^\top\mathbf{M}_{\mathbf{x}'} \otimes \mathbf{S}_{\mathbf{x}}) \cdot \underline{\mathbf{A}_h} + \mathbf{E}_{\mathbf{x}'} \\
&= \overline{\mathbf{A}_h} + (\mathbf{v}^\top\mathbf{M}_{\mathbf{x}'} \otimes \mathbf{I}_n) \cdot (\mathbf{I}_w \otimes \mathbf{S}_{\mathbf{x}'}) \cdot \underline{\mathbf{A}_h} + \mathbf{E}_{\mathbf{x}'} \\
&\approx_s \overline{\mathbf{A}_h} + (\mathbf{v}^\top\mathbf{M}_{\mathbf{x}'} \otimes \mathbf{I}_n) \cdot \underbrace{\left((\mathbf{I}_w \otimes \mathbf{S}_{\mathbf{x}'}) \cdot \underline{\mathbf{A}_h} + \mathcal{D}_{\sigma_2}^{nw\times m}\right)}_{\overset{\text{def}}{=} G(\mathbf{x}')} + \mathbf{E}_{\mathbf{x}'},
\end{aligned}$$

where the last step is by noise flooding ($\sigma_1 = \sigma_2 \cdot \lambda^{\omega(1)}$). Next, by the security of the BLMR PRF [BLMR13], $G(\mathbf{x}')$ is pseudorandom, i.e.,

$$\left\{\boxed{(\mathbf{I}_w \otimes \mathbf{S}_{\mathbf{x}'}) \cdot \underline{\mathbf{A}_h} + \mathcal{D}_{\sigma_2}^{nw\times m}}\right\}_{\mathbf{x}'\in\{0,1\}^j}, \{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}} \approx_c \left\{\mathcal{U}(\mathbb{Z}_q^{nw\times m})\right\}_{\mathbf{x}'\in\{0,1\}^j}, \{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}.$$

This relies on $((\mathbf{I}_w \otimes \mathbf{S})\mathbf{A} + \mathbf{E}, \mathbf{S}) \approx_c \left(\mathcal{U}(\mathbb{Z}_q^{nw\times m}), \mathbf{S}\right)$, which follows from LWE via a simple reduction to the case of lemma 2.2.

- $\underline{j \in [h-1]}$. Since $|\Gamma^{-1}(1)| \leq 1$, we have $\mathbf{v}^\top \mathbf{M}_{\mathbf{x}'} \neq \mathbf{0}$ for all $\mathbf{x}' \in \{0,1\}^j$. Therefore, $(\mathbf{v}^\top \mathbf{M}_{\mathbf{x}} \otimes \mathbf{I}_n) \cdot G(\mathbf{x}') \approx_c \mathcal{U}(\mathbb{Z}_q^{nw \times m})$ and hence $\{F(\mathbf{x}')\}_{\mathbf{x}' \in \{0,1\}^j} \approx_c \left\{\mathcal{U}(\mathbb{Z}_q^{nw \times m})\right\}_{\mathbf{x}' \in \{0,1\}^j}$.
- $\underline{j = h}$. If Γ computes the all-zero function, the argument in the first item still goes for $j = h$. Now assume that $\Gamma^{-1}(1) = \{\mathbf{x}^*\}$. Then

$$F(\mathbf{x}) = \begin{cases} \overline{\mathbf{A}_h} + \mathbf{E}_{\mathbf{x}} & \text{if } \mathbf{x} = \mathbf{x}^* \\ \overline{\mathbf{A}_h} + (\mathbf{v}^\top \mathbf{M}_{\mathbf{x}} \otimes \mathbf{I}_n) \cdot G(\mathbf{x}) + \mathbf{E}_{\mathbf{x}} & \text{if } \mathbf{x} \neq \mathbf{x}^* \end{cases}$$

Since $\overline{\mathbf{A}_h}$ is uniformly distributed and $(\mathbf{v}^\top \mathbf{M}_{\mathbf{x}} \otimes \mathbf{I}_n) \cdot G(\mathbf{x}) \approx_c \mathcal{U}(\mathbb{Z}_q^{nw \times m})$ for all $\mathbf{x} \neq \mathbf{x}^*$, we have $\{F(\mathbf{x})\}_{\mathbf{x} \in \{0,1\}^h} \approx_c \left\{\mathcal{U}(\mathbb{Z}_q^{nw \times m})\right\}_{\mathbf{x} \in \{0,1\}^h}$, proving eq. (3) for $j = h$.

$\square$

Claim 4.11. *Fix* $\overline{\mathbf{A}}_h \in \mathbb{Z}_q^{n \times m}$. *Let*

$$\Gamma = \left(\mathbf{v} \in \{0,1\}^w, \{\mathbf{M}_{i,b} \in \{0,1\}^{w \times w}\}_{i \in [h], b \in \{0,1\}}\right)$$

be an MBP that computes the all-zero function and let

$$\widehat{\mathbf{S}}_{1,b} := \left(\mathbf{I}_n \mid \mathbf{v}^\top \mathbf{M}_{1,b} \otimes \mathbf{S}_{1,b}\right), \widehat{\mathbf{S}}_{i,b} := \begin{pmatrix} \mathbf{I}_n & \\ & \mathbf{M}_{i,b} \otimes \mathbf{S}_{i,b} \end{pmatrix}, \mathbf{A}_h := \begin{pmatrix} \overline{\mathbf{A}_h} \\ \underline{\mathbf{A}_h} \end{pmatrix},$$

where $\mathbf{S}_{i,b} \leftarrow \mathcal{D}_{2\sqrt{n}}^{n \times n}$ *for* $i \in [h], b \in \{0,1\}$, $\underline{\mathbf{A}_h} \leftarrow \mathbb{Z}_q^{\widehat{n} \times m}$. *Then, by the LWE assumption, for every* $j \in [h]$ *we have*

$$\left\{\boxed{\widehat{\mathbf{S}}_{\mathbf{x}'} \mathbf{A}_j + \mathbf{E}_{\mathbf{x}'}}\right\}_{\mathbf{x}' \in \{0,1\}^j}, \{\mathbf{S}_{i,b}\}_{i \in [h], b \in \{0,1\}}, \overline{\mathbf{A}}_h \approx_c \{\mathcal{U}(\mathbb{Z}_q^{\widehat{n}_0 \times m})\}_{\mathbf{x}' \in \{0,1\}^j}, \{\mathbf{S}_{i,b}\}_{i \in [h], b \in \{0,1\}}, \overline{\mathbf{A}}_h, \quad (4)$$

where $\mathbf{E}_{\mathbf{x}'} \leftarrow \mathcal{D}_{\sigma_1}^{n \times m}$, $\mathbf{A}_j \leftarrow \mathbb{Z}_q^{\widehat{n} \times m}$ *for* $j \in [h-1]$.

Proof. For $j \leq h-1$, the proof is the same as that of the previous claim (Precondtion 1). For $j = h$, note that when $\mathbf{v}^\top \mathbf{M}_{\mathbf{x}} \neq \mathbf{0}$ for all $\mathbf{x} \in \{0,1\}^h$, so $F(\mathbf{x})$ is completely randomized by $G(\mathbf{x})$, and hence knowing $\overline{\mathbf{A}_h}$ is not helpful. $\square$

Proving the pseudorandomness of GGH15 encodings: lemma 4.6 and lemma 4.7.

Proof of lemma 4.6. Since $|\Gamma^{-1}(1)| \leq 1$, by claim 4.10, the precondition of lemma 4.8 holds with $\mathcal{E}$ being the uniform distribution over $\mathbb{Z}_q^{\widehat{n} \times m}$. Then the lemma follows from lemma 4.8. $\square$

Proof of lemma 4.7. Claim 4.11 shows that the precondition of lemma 4.8 holds for the assumed distribution of $\left\{\widehat{\mathbf{S}}_{i,b}\right\}_{i \in [h], b \in \{0,1\}}$ and $\mathbf{A}_h$. Note that $\mathcal{E}$ is publicly known and hence $\overline{\mathbf{A}_h}$ can be given to the distinguisher. Then the lemma follows from lemma 4.8. $\square$

4.3 Security Proof of Our OPF Construction

Finally, we show that construction 3 is indeed an OPF, proving theorem 4.1.

Theorem 4.12 (Security of construction 3)**.** *Under subexponential LWE assumption and evasive LWE assumption, Π in construction 3 is an obliviously programmable function for* $\mathsf{MBP}^1_{\ell_{\mathsf{in}},w}$.

Proof. Correctness is proven in section 4.1. Here we prove that the two security properties of OPF are satisfied.

Privacy. Note that when $\mathbf{Y} \leftarrow \mathbb{Z}_2^{n\times m}$, $\mathbf{A}_h$ sampled in $\Pi.\mathsf{Program}(1^\lambda, \Gamma, \mathbf{Y})$ is also uniformly distributed, and thus satisfies the condition of lemma 4.6. By lemma 4.6, for any $\Gamma \in \mathsf{ROBP}_w$, if $|\Gamma^{-1}(1)| \le 1$, then $k_\Gamma \leftarrow \Pi.\mathsf{Program}(1^\lambda, \Gamma, \mathbf{Y})$ is pseudorandom, where $\mathbf{Y} \leftarrow \mathbb{Z}_2^{n\times m}$. Therefore, for arbitrary (Γ_0, Γ_1) with $|\Gamma_0^{-1}(1)| \le 1, |\Gamma_1^{-1}(1)| \le 1$, no PPT adversary can distinguish $k_{\Gamma_0} \leftarrow \Pi.\mathsf{Program}(1^\lambda, \Gamma_0, \mathbf{Y})$ from $k_{\Gamma_1} \leftarrow \Pi.\mathsf{Program}(1^\lambda, \Gamma_1, \mathbf{Y})$ for $\mathbf{Y} \leftarrow \mathbb{Z}_2^{n\times m}$.

Value-Hiding when programming on the all-zero function. Let $\Gamma_\emptyset$ be a MBP that computes the all-zero function. Note that $\mathbf{Y} = \lfloor \overline{\mathbf{A}_h} \rceil_2$ is a deterministic function of $\overline{\mathbf{A}_h}$. By lemma 4.7, for any fixed $\mathbf{Y}$, $k_\Gamma \leftarrow \Pi.\mathsf{Program}(1^\lambda, \Gamma, \mathbf{Y})$ is pseudorandom even the adversary knows $\mathbf{Y}$. Therefore, for arbitrary $(\mathbf{Y}_0, \mathbf{Y}_1)$, no PPT adversary can distinguish $k_0 \leftarrow \Pi.\mathsf{Program}(1^\lambda, \Gamma_\emptyset, \mathbf{Y}_0)$ from $k_1 \leftarrow \Pi.\mathsf{Program}(1^\lambda, \Gamma_\emptyset, \mathbf{Y}_1)$. $\square$

Acknowledgments and Disclosure of Funding

Yilei Chen is supported by Tsinghua University startup funding. Xinyu Mao is supported by NSF CAREER award 2141536.

References

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In *ICALP*, volume 1644 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 1999. 8

[AP09] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In *STACS*, volume 3 of *LIPIcs*, pages 75–86. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2009. 8

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *Theory of cryptography conference*, pages 52–73. Springer, 2014. 4

[BD20] Zvika Brakerski and Nico Döttling. Hardness of LWE on general entropic distributions. In *EUROCRYPT (2)*, volume 12106 of *Lecture Notes in Computer Science*, pages 551–575. Springer, 2020. 11

[BFM14] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Indistinguishability obfuscation and uces: The case of computationally unpredictable sources. In *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34*, pages 188–205. Springer, 2014. 1, 2

[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):6:1–6:48, 2012. 2

[BHK13] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via uces. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 398–415. Springer, 2013. 1, 2, 3, 18

[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. 4, 5

[BLMR13] Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *Annual Cryptology Conference*, pages 410–428. Springer, 2013. 8, 23

[BLW17] Dan Boneh, Kevin Lewi, and David J. Wu. Constraining pseudorandom functions privately. In *Public Key Cryptography (2)*, volume 10175 of *Lecture Notes in Computer Science*, pages 494–524. Springer, 2017. 3, 5

[BM14] Christina Brzuska and Arno Mittelbach. Using indistinguishability obfuscation via uces. In *ASIACRYPT (2)*, volume 8874 of *Lecture Notes in Computer Science*, pages 122–141. Springer, 2014. 1, 2, 3, 4, 7, 18

[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. 4, 5, 7

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 1997. 2

[CC17] Ran Canetti and Yilei Chen. Constraint-hiding constrained PRFs for NC^1 from LWE. In *EUROCRYPT 2017, Part I*, pages 446–476, 2017. 4, 5, 7

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004. 1, 2

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In *CRYPTO (2)*, volume 10992 of *Lecture Notes in Computer Science*, pages 577–607. Springer, 2018. 7, 19

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings*, pages 537–554. Springer, 1999. 1

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Advances in cryptology—CRYPTO '86 (Santa Barbara, Calif., 1986)*, volume 263 of *Lecture Notes in Comput. Sci.*, pages 186–194. Springer, Berlin, 1987. 1

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *TCC 2015, Part II*, pages 498–527, 2015. 1, 2, 18, 19

[GGH$^+$16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.*, 45(3):882–929, 2016. 2

[GK05] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *FOCS*, pages 553–562. IEEE Computer Society, 2005. 2

[GKPV10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In *ICS*, pages 230–240. Tsinghua University Press, 2010. 2, 3, 11

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *FOCS*, pages 612–621, 2017. 7, 10

[GOR11] Vipul Goyal, Adam O'Neill, and Vanishree Rao. Correlated-input secure hash functions. In *Theory of cryptography*, volume 6597 of *Lecture Notes in Comput. Sci.*, pages 182–200. Springer, Heidelberg, 2011. 2

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008. 8

[HLL23] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 415–434. IEEE, 2023. 3

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, 2012. 8

[PS18] Chris Peikert and Sina Shiehian. Privately constraining and programming PRFs, the LWE way. In *PKC*, 2018. 4, 5, 7

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009. 8

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484. ACM, 2014. 4

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In *CRYPTO (1)*, volume 13507 of *Lecture Notes in Computer Science*, pages 535–559. Springer, 2022. 3, 7

[VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-io from evasive lwe. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 195–221. Springer, 2022. 3, 6, 7, 19, 20, 21, 22

[Wee05] Hoeteck Wee. On obfuscating point functions. In *STOC*, pages 523–532. ACM, 2005. 2

[Wee22] Hoeteck Wee. Optimal broadcast encryption and cp-abe from evasive lattice assumptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 217–241. Springer, 2022. 1, 3, 18

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In *FOCS*, pages 600–611, 2017. 7, 10

[Zha16] Mark Zhandry. How to avoid obfuscation using witness PRFs. In *Theory of cryptography. Part II*, volume 9563 of *Lecture Notes in Comput. Sci.*, pages 421–448. Springer, Berlin, 2016. 3

[Zha19] Mark Zhandry. The magic of elfs. *Journal of Cryptology*, 32:825–866, 2019. 3

A Generalization to Read-c MBPs

We first prove the lemma that allows us to represent a read-c MBP by a read-once MBP and ensures all invalid inputs are evaluated to zero.

Lemma A.1 (lemma 2.7 restated)**.** *Let Γ be a read-c MBP with width w, length h, and input length $\ell = h/c$. Let $\mathsf{repeat}(\mathbf{x}) = \underbrace{\mathbf{x}|\mathbf{x}|\cdots|\mathbf{x}}_{c\ times}$. Then there exists a read-once MBP Γ' with the following properties.*

1. *Γ' has width $h + w$ and length h.*
2. *For all $\mathbf{x} \in \{0,1\}^\ell$, $\Gamma(\mathbf{x}) = \Gamma'(\mathsf{repeat}(\mathbf{x}))$.*
3. *For all invalid $\mathbf{x}' \in \{0,1\}^h$, i.e., $\mathbf{x}' \neq \mathsf{repeat}(\mathbf{x})$ for any $\mathbf{x} \in \{0,1\}^\ell$, it holds that $\Gamma'(\mathbf{x}') = 0$.*

In particular, if Γ computes a point function or all-zero function, then so is Γ'.

Proof. Let $\Gamma = \left\{\mathbf{v} \in \{0,1\}^w, \{\mathbf{M}_{i,b} \in \{0,1\}^{w\times w}\}_{i\in[h],b\in\{0,1\}}\right\}$ be a read-c MBP (and thus we omit ι). Let

$$\mathbf{V}^{(0)} \overset{\mathsf{def}}{=} \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}, \mathbf{V}^{(1)} \overset{\mathsf{def}}{=} \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \in \{0,1\}^{c\times c}.$$

Claim A.2. *For all $z \in \{0,1\}^c$, $\prod_{i\in[c]} \mathbf{V}^{(z_i)} = \mathbf{0}$ if and only if z is all-zero or all-one.*

For $i \in [h], j \in [\ell], b \in \{0,1\}$, define $\mathbf{U}_{i,b,j} \in \{0,1\}^{c\times c}$ as follows.

$$\mathbf{U}_{i,b,j} = \begin{cases} \mathbf{V}^b, & \text{if } i \equiv j \pmod{\ell}; \\ \mathbf{I}_c, & \text{otherwise.} \end{cases},$$

For $i \in [h], b \in \{0,1\}$, set

$$\mathbf{M}'_{i,b} = \mathrm{diag}(\mathbf{U}_{i,b,1}, \mathbf{U}_{i,b,2}, \ldots, \mathbf{U}_{i,b,\ell}, \mathbf{M}_{i,b}) \in \{0,1\}^{(h+w)\times(h+w)}.$$

Let $\mathbf{v}' = (1\ldots1|\mathbf{v}) \in \{0,1\}^{h+w}$. We claim that the read-once MBP

$$\Gamma' = \left\{\mathbf{v}', \left\{\mathbf{M}'_{i,b} \in \{0,1\}^{(h+w)\times(h+w)}\right\}_{i\in[h],b\in\{0,1\}}\right\}$$

satisfies the said properties.

Note that $(\mathbf{v}')^\top \mathbf{M}'_{\mathbf{x}'} = (\mathbf{1}^\top \mathbf{U}^{(1)}|\cdots|\mathbf{1}^\top \mathbf{U}^{(\ell)}|\mathbf{v}^\top \mathbf{M}_{\mathbf{x}'})$, where $\mathbf{U}^{(j)} \overset{\mathsf{def}}{=} \prod_{i\in[h]} \mathbf{U}_{i,x'_i,j}$. And for all $j \in [\ell]$

$$\mathbf{U}^{(j)} = \prod_{i\in[h]} \mathbf{U}_{i,x'_i,j} = \prod_{i\in[h]:i\equiv j \pmod{\ell}} \mathbf{U}_{i,x'_i,j} = \prod_{k=1}^{c} \mathbf{V}^{(x'_{(k-1)c+j})}.$$

Therefore, by the claim above, $\mathbf{1}^\top \mathbf{U}^{(j)} = \mathbf{0}$ if and only if

$$x'_j = x'_{j+\ell} = x'_{j+2\ell} = \cdots = x'_{j+(c-1)\ell}.$$

Since this holds for all $j \in [\ell]$, we have the desired properties. $\square$

Construction 4. $\Pi = (\Pi.\mathsf{Program}, \Pi.\mathsf{Eval})$. Input length ℓ_{in} and codomain $\mathcal{R}_\lambda = \mathbb{Z}_2^{n\times m}$. Let $h \overset{\mathsf{def}}{=} c \cdot \ell_{\mathsf{in}}, w' \overset{\mathsf{def}}{=} w + h, \widehat{n}_0 \overset{\mathsf{def}}{=} n, \widehat{n} \overset{\mathsf{def}}{=} nw' + n$.

- $\mathsf{Program}(1^\lambda, C, \mathbf{Y} \in \mathbb{Z}_p^{n\times m}) \mapsto k_C$
  1. Parse C as a read-c MBP Γ. Let $\Gamma' = (\mathbf{v}', \left\{\mathbf{M}'_{i,b}\right\}_{i\in[h],b\in\{0,1\}})$ be the read-once MBP representation of Γ given by lemma 2.7.
  2. Sample $\mathbf{S}_{i,b} \leftarrow \mathcal{D}_{\sigma_1}^{n\times n}$ for $i \in [h], b \in \{0,1\}$ and set
  $$\widehat{\mathbf{S}}_{1,b} = \left(\mathbf{I}_n \mid (\mathbf{v}')^\top \mathbf{M'}_{1,b} \otimes \mathbf{S}_{1,b}\right), \widehat{\mathbf{S}}_{i,b} = \begin{pmatrix} \mathbf{I}_n & \\ & \mathbf{M'}_{i,b} \otimes \mathbf{S}_{i,b} \end{pmatrix},$$
  for $i = 2, \ldots, h, b \in \{0,1\}$.
  3. Sample $\mathbf{A}_h \leftarrow \mathbb{Z}_q^{\widehat{n}\times m}$ conditioned on $\lfloor \mathbf{A}_h \rceil_2 = \mathbf{Y}$ and output
  $$k_c := \mathbf{C}_0, \mathbf{C}_1, \{\mathbf{D}_{i,b}\}_{i=2,\ldots,h,b\in\{0,1\}} \leftarrow \mathsf{GGH.Encode}(\left\{\widehat{\mathbf{S}}_{i,b}\right\}_{i\in[h],b\in\{0,1\}}, \mathbf{A}_h).$$
- $\mathsf{Eval}(k_C, \mathbf{x} \in \{0,1\}^h) \mapsto \mathbf{Y} \in \mathbb{Z}_2^{n\times m}$. Parse $k_C = \mathbf{C}_0, \mathbf{C}_1, \{\mathbf{D}_{i,b}\}_{i=2,\ldots,h,b\in\{0,1\}}$ and let $\mathbf{x}' = \mathsf{repeat}(\mathbf{x}) \in \{0,1\}^h$. Output
$$\mathbf{Y} := \left\lfloor \mathbf{C}_{x'_1} \prod_{i=2}^{h} \mathbf{D}_{i,x'_i} \right\rceil_2 .$$

Analysis.

- Correctness. Since for all $\mathbf{x} \in \{0,1\}^{\ell_{\mathsf{in}}}$ and $\mathbf{x}' = \mathsf{repeat}(\mathbf{x}) \in \{0,1\}^h$, we have
$$\Gamma(\mathbf{x}) = 1 \iff \Gamma'(\mathbf{x}') = 1 \iff (\mathbf{v}')^\top \mathbf{M}'_{\mathbf{x}'} = \mathbf{0}.$$
And hence correctness follows from the same calculation in section 4.1.
- Security. Note that $\Gamma'^{-1}(1) = \Gamma^{-1}(1)$, and thus lemma 4.6, lemma 4.7 works perfectly for Γ'. Therefore, the argument in the proof of theorem 4.12 still goes, with Γ being replaced by Γ'.
- Efficiency. The only efficiency loss is that we need to augment the width from w to $w' = w + c \cdot \ell_{\mathsf{in}}$, which is still polynomial in λ. In the setting of parameters, we shall use w' in place of w.