Pseudorandom Obfuscation and Applications

Pedro Branco
Bocconi

Nico Döttling
CISPA

Abhishek Jain
NTT Research and JHU

Giulio Malavolta
Bocconi

Surya Mathialagan
MIT

Spencer Peters
Cornell

Vinod Vaikuntanathan
MIT

Abstract

We introduce the notion of *pseudorandom obfuscation* (PRO), a way to obfuscate (keyed) pseudorandom functions f_K in an average-case sense. We introduce several variants of pseudorandom obfuscation and show constructions and applications. For some of our applications that can be achieved using full-fledged indistinguishability obfuscation (iO), we show constructions using lattice-based assumptions alone; the other applications we enable using PRO are simply not known even assuming iO. We briefly summarize our contributions below.

1. **Constructions of PRO:** We show how to construct the strongest version of PRO, assuming the sub-exponential hardness of the learning with errors (LWE) problem, and of the evasive LWE problem (Wee, EUROCRYPT 2022; Tsabary, CRYPTO 2022).
2. **Applications outside the iO World:** We show how to construct a succinct witness encryption scheme from PRO, where the size of the ciphertext is independent of the witness size. Such a witness encryption scheme is not known to exist even assuming iO.
3. **Applications in the iO World:** Our weakest variant of pseudorandom obfuscation, named obfuscation for identical pseudorandom functions (iPRO), is weaker than iO: rather than obfuscating arbitrary circuits as in iO, iPRO only obfuscates circuits computing pseudorandom functions. We show that iPRO already enables several applications of iO, such as unleveled fully homomorphic encryption (without assuming circular security) and succinct randomized encodings.
4. **From iPRO to iO:** Despite being a seemingly weaker notion than iO, we show two pathways to constructing full-fledged iO from iPRO. Our first construction builds iO from iPRO and (standard assumptions on) cryptographic bilinear maps. Combined with our construction of iPRO, this gives us a construction of iO from a new combination of assumptions, namely LWE, evasive LWE and bilinear maps. Our second construction builds iO (and even ideal obfuscation) from iPRO in the pseudorandom oracle model (Jain, Lin, Luo and Wichs, CRYPTO 2023). To our knowledge, this is the first purely lattice-based, and hence plausibly post-quantum secure, construction of iO with a proof of security from LWE and evasive LWE.

Finally, we highlight some barriers in achieving the strongest version of pseudorandom obfuscation.

Contents

1 Introduction

The goal of program obfuscation is to make computer programs unintelligible, while preserving their functionality. Because of its intrinsic ability to hide secrets in plain software, obfuscation is often considered to be the quintessential cryptographic primitive. The possibility and utility of program obfuscation was already foreshadowed in Diffie and Hellman's landmark paper [DH76]; however, it took more than two decades for program obfuscation to be placed on solid definitional foundations in [Had00, BGI+01], where it is shown that strong (but natural) notions for obfuscation, namely ideal obfuscation and virtual black-box obfuscation, are impossible to achieve. To remedy this, the authors of [BGI+01] suggested the notion of *indistinguishability obfuscation* (iO), which guarantees that the obfuscations of any pair of functionally-equivalent circuits (or programs) are computationally indistinguishable [BGI+01, GR07].

The study of iO has had a lasting impact on cryptography and has generated a remarkable body of research devoted to constructing general-purpose obfuscators [GGH+13, BGK+14, BR14, PST14, BMSZ16, MSZ16, Lin16, LV16, AS17, Lin17, LT17, CVW18, GJ18, JLMS19, Agr19, BIJ+20, AP20, BDGM20, JLS21, GP21, WW21, BDGM22, JLS22] as well as enabling fascinating new applications in cryptography and complexity theory [GGH+13, SW14, BZ14, BPR15, BGL+15, Zha19, CLLZ21, BGK+23, ILW23]. A milestone in this line of research is the work of Jain, Lin and Sahai [JLS21, JLS22] who constructed an iO scheme from the hardness of three well-founded assumptions: the hardness of Diffie-Hellman like problems on bilinear maps; the hardness of learning parity with noise (LPN); and the existence of pseudorandom generators in NC^0. Further recent work [RVV24] replaced the last of these assumptions with sparse LPN over the binary field. However, owing to their dependence on bilinear maps, none of these constructions are post-quantum secure.

Thus, despite the resounding success of iO, the question of constructing useful program obfuscators from the hardness of (post-quantum secure) lattice problems has remained frustratingly open, despite several attempts [CVW18, BIJ+20, GP21, BDGM20]. The objective of this work is to make progress on this problem: by proposing new notions of obfuscation different from iO, both weaker and stronger; constructing them from plausibly post-quantum secure assumptions; and studying their implications for cryptography.

1.1 Our Results

In this work, we propose the notion of *pseudorandom obfuscation* (PRO) and investigate general-purpose constructions. We consider two variants of PRO: the stronger one yields cryptographic primitives that are currently unknown even assuming iO, whereas the weaker variant suffices for many applications of iO. We show that the weaker variant also provides an alternative pathway to constructing full-fledged iO. We summarize our contributions in more detail below.

Pseudorandom Obfuscation: A New Notion. A pseudorandom obfuscation scheme $\mathsf{PRO.Obf}$ allows one to obfuscate any (keyed) function $f_K : \mathcal{X} \to \mathcal{Y}$; that is, $\mathsf{PRO.Obf}$ takes as input a key K and outputs a circuit computing f_K. Security, in an informal sense, says that if the function table of f_K, for a uniformly random K, is pseudorandom (a condition that we will call truth-table pseudorandom), then the obfuscation of f_K hides all information about K. We formalize this in several different ways (see Table 1 for a summary of definitions and applications):

- **The Stronger Variants, PRO and dPRO:** If the function table of f_K is computationally indis-

tinguishable from uniform in the presence of an auxiliary input aux_K, that is,

$$\{f_K(x)\}_{x\in\mathcal{X}}, \mathsf{aux}_K \approx_c \{u_x : u_x \leftarrow \mathcal{Y}\}_{x\in\mathcal{X}}, \mathsf{aux}_K \tag{1.1}$$

over the random choice of K, then the PRO definition requires that

$$\mathsf{PRO.Obf}(f_K), \mathsf{aux}_K \approx_c \mathsf{PRO.Obf}(f_{K'}), \mathsf{aux}_K \tag{1.2}$$

where K, K' are i.i.d. random keys and the computational indistinguishability holds over the random coins used in PRO.Obf. That is, if the function family $\{f_K\}_{K\in\mathcal{K}}$ is truth-table pseudorandom in the presence of the auxiliary input aux_K, then the obfuscation of f_K is (computationally) independent of the auxiliary input.

Crucially, we require the entire function table of f_K to be pseudorandom to adversaries who can run in time polynomial in the size of the function table, a stronger notion of pseudorandomness than the standard one where a polynomial-time adversary can make queries to the function table of f_K.

We also consider a stronger version, named doubly pseudorandom obfuscation or dPRO, which replaces condition 1.2 with the following stronger condition (the pre-condition 1.1 remains the same):

$$\mathsf{dPRO.Obf}(f_K), \mathsf{aux}_K \approx_c U_\ell, \mathsf{aux}_K \tag{1.3}$$

where U_ℓ is a uniformly sampled bit-string of length $\ell = |\mathsf{PRO.Obf}(f_K)|$. That is, we require that the *obfuscated program itself* is also pseudorandom.

While both these notions in general suffer from a contrived impossibility (more discussion on this later), they imply cryptographic primitives that we do not yet know how to construct, even from iO.

- **The Weaker Variant, iPRO:** If the function table of f_K is truth-table pseudorandom in the presence of an auxiliary input aux_K (that is, assuming precondition 1.1), for any two K, K' such that $f_K \equiv f_{K'}$, the iPRO definition guarantees condition (1.2), namely that

$$\mathsf{iPRO.Obf}(f_K), \mathsf{aux}_K \approx_c \mathsf{iPRO.Obf}(f_{K'}), \mathsf{aux}_K$$

Here, indistinguishability holds only for function pairs $f_K, f_{K'}$ which have the same function table. Indeed, this definition can be seen both as a weakening of PRO (and dPRO) but also a weakening of the standard notion of iO, which has no requirement on the pseudorandomness of the function table of f_K. In particular, an immediate observation is that if iO exists, so does iPRO.

For a more detailed discussion of these notions, along with the comparison with existing ones, we refer the reader to Section 1.4.

Guarantee \ Pre-condition	$f_K \equiv f_{K'}$	$f_K \equiv f_{K'}$, and pseudorandom (1.1)	f_K and $f_{K'}$ are pseudorandom (1.1)
$\mathsf{Obf}(f_K) \approx_c \mathsf{Obf}(f_{K'})$ (1.2)	iO	iPRO **Applications:** Null-iO, Witness Encryption [VWW22] dvSNARGs for UP [MPV24] (+LWE) FHE (Section 7.1) Succinct RE (Section 7.2) (+Bilinear maps) iO (Section 8.3) (+PROM [JLLW23]) iO (Section 8.4)	PRO **Applications:** PRO for TMs (Section 6.2) Succinct WE (Section 7.3)
$\mathsf{Obf}(f_K) \approx_c U_\ell$ (1.3)	*impossible*	*same as* dPRO	dPRO

Table 1: Notions of obfuscation in this paper, and their applications. The column represents the assumption on the pre-condition, and the row represents the guarantee of the obfuscation. Notation: dvSNARG represents a designated verifier succinct non-interactive argument; FHE is fully homomorphic encryption; WE is witness encryption; RE is randomized encoding; and PROM is the pseudorandom oracle model from [JLLW23]. The combination of notions in the bottom left cell (colored blue) is unachievable: indeed, an obfuscation of f_K cannot be pseudorandom if f_K is an arbitrary function.

Pseudorandom Obfuscation: Constructions. The main technical contribution of this work is the construction of a dPRO scheme for all truth-table-pseudorandom functions (namely, ones whose truth table is indistinguishable from random) assuming the sub-exponential hardness of the learning with errors (LWE) problem [Reg05] and of the evasive LWE problem [Wee22, Tsa22, VWW22].

Theorem 1.1 (Informal)**.** *If LWE and evasive LWE are sub-exponentially hard, then there exists a doubly pseudorandom obfuscation (dPRO) scheme for all polynomial-time computable truth-table-pseudorandom families.*

We defer a discussion of these assumptions to Section 1.5, and focus here on the most significant ideas in our construction. At a high-level, we follow the outline of Brakerski, Döttling, Garg and Malavolta [BDGM20] to construct iO, but with several new ideas and techniques. In a nutshell, our construction proceeds in three steps.

- *Step 1: Exponentially-efficient dPRO.* Our starting point is to construct an exponentially-efficient doubly pseudorandom obfuscation scheme (xdPRO) where the only efficiency constraint is that the size of the obfuscated circuit should be *sublinear* in the domain size of f_K (Section 4). This requirement is akin to how one weakens iO to XiO [LPST16]. Our construction is inspired by heuristic lattice-based obfuscation schemes [GP21, WW21, BDGM22], except that here, we can prove its security based on LWE and evasive LWE.

- *Step 2: Delegating Computation.* In order to bootstrap xdPRO to dPRO (in the next step), we will need to reduce the computation time of the xdPRO scheme by *delegating* its computation to the decoder. To do this, we construct (Section 5) a laconic function evaluation (LFE) scheme [QWW18] that satisfies a notion of *blindness*. That is, provided that the output of the

computation is pseudorandom, the ciphertext in the blind laconic FE scheme is computationally indistinguishable from uniformly random bits. Our scheme is secure assuming only the hardness of the LWE problem.

- *Step 3: Bootstrapping.* To obtain a full-fledged (d)PRO scheme, we finally apply a bootstrapping theorem (Section 6), akin to the well-known transformations for iO [AJ15, BV15]. Crucially, for the bootstrapping step to work (for constructing either dPRO or even the weaker PRO), we need the underlying xPRO scheme to be *doubly* pseudorandom, i.e. it should be an xdPRO scheme.

In addition, as a contribution of independent interest, we show that one can easily turn a (d)PRO scheme for circuits into a (d)PRO scheme for Turing machines, using standard cryptographic techniques.

Pseudorandom Obfuscation: Applications. Our first observation is that many of the applications of iO already obfuscate pseudorandom functions. Thus, we can use known ideas from the literature to build interesting cryptographic primitives from the weakest version of pseudorandom obfuscation, namely iPRO (Section 7). For instance, we show how to construct fully-homomorphic encryption [Gen09] (without any circular security assumption) and succinct randomized encodings [BGL+15] from iPRO. Although these primitives are also known from iO, using iPRO yields constructions from a weaker notion of obfuscation, and gives us constructions that are fully lattice-based (and therefore plausibly post-quantum secure) [SW14]. (If one relies on the stronger notion of PRO, the constructions have a simple security analysis without any puncturing argument.)

We pause to mention that our framework gives a construction of witness encryption and null-IO from iPRO, generalizing the work of [VWW22, Tsa22]; and an adaptive designated verifier SNARG for the complexity class UP, generalizing the work of [MPV24]. In both cases [VWW22, Tsa22, MPV24], the authors based their constructions on LWE and evasive LWE, which we recover via iPRO.

In addition, we show that the stronger notion of PRO implies the existence of a *succinct* witness encryption scheme [GGSW13], i.e., a scheme where the size of the ciphertext is independent of the witness size. To the best of our knowledge, it is currently unknown how to construct such a scheme from iO, because of the dependency of the obfuscated circuit on the size of the input. PRO allows us to bypass this input-size barrier, and we view this as evidence that the security guarantees offered by PRO may be stronger than those of iO. We provide more details on how we achieve these applications in Section 1.2.

A New Pathway Towards iO. As a contribution of independent interest, we show how iPRO can be used to construct general-purpose iO (Section 8), thus obtaining new constructions of iO from assumptions that were not known previously known to imply obfuscation. First, we show how to combine iPRO with bilinear pairings in order to obtain iO.

Theorem 1.2 (Informal)**.** *If the LWE, the evasive LWE, the SXDH, and the DLIN assumptions hold against sub-exponential distinguishers, then there exists indistinguishability obfuscation for all polynomial-time computable function families.*

Compared with known construction of iO from standard assumptions [JLS21, JLS22], the above construction does not rely on the hardness of coding-related problems (such as the learning parity

with noise problem), nor it requires the existence of low-complexity pseudorandom generators (implementable by constant-depth circuits). The flip side is that we require the hardness of the LWE and the evasive LWE problem, which was not needed by prior approaches.

Finally, we present an entirely *lattice-based* construction of iO, with security in the pseudorandom oracle model (PROM) [JLLW23]. The PROM is a recently introduced idealized model that allows one to reason formally about the interplay between random oracles and obfuscation. This yields the first construction of iO that is entirely based on lattices with a proof of security, albeit in an idealized model and with strong cryptographic assumptions.

Theorem 1.3 (Informal)**.** *If the LWE and the evasive LWE assumptions hold against sub-exponential distinguishers, then there exists indistinguishability obfuscation for all polynomial-time computable function families in the PROM.*

Besides providing new candidate constructions of iO, our transformations elucidate the relation between iO and iPRO, and the kind of cryptographic structure that is needed to bridge between these two primitives.

Impossibility of PRO. We observe that there is a (contrived) truth-table pseudorandom function family $\mathcal{F} = \{f_K\}_{K \in \mathcal{K}}$ and an auxiliary input function $\mathsf{aux} : \mathcal{K} \to \{0,1\}^*$ such that $\mathcal{F}$ cannot be PRO-obfuscated. The idea, drawing from [GK05, BCC+14, VWW22], is to let the auxiliary input function be a witness encryption for the NP statement x which is (a large part of) the truth table of f_K whose witness w is a small program that computes f_K. The pre-condition in the definition of PRO is true via its pseudorandomness and the semantic security of the witness encryption scheme. However, one can easily break the PRO guarantee using the obfuscated program as a witness to decrypt the auxiliary input. This is a counterexample not only to PRO, but also xPRO, the slightly compressing version. With a little more work, we can extend this to the setting where there is no auxiliary input: By including the auxiliary input in the truth table itself, and by ensuring that the witness encryption ciphertext (in the *no* case) is pseudorandom. For a more in-depth discussion on the implications of this result, we refer the reader to Section 1.4, and for the details of the impossibility result, see Section 9.

Overview of the rest of this section. In the rest of this section, we will first give an overview of the applications in Section 1.2 and Section 1.3. We then discuss our notions of pseudorandom obfuscation in the context of existing notions of obfuscations in Section 1.4. Finally, we discuss the evasive LWE assumption in Section 1.5 and outline our construction of (d)PRO in Section 1.6.

1.2 Technical Overview I: Applications

We outline some of the applications of our PRO scheme; for details, we refer the reader to Section 7. In fact, for all but one of our applications, the weaker notion of iPRO will suffice.

Fully homomorphic encryption. As a first application, we consider the problem of turning a leveled homomorphic encryption scheme into a fully homomorphic one. This precise problem was already considered in [CLTV15] and in fact the same idea works here: Simply obfuscate a circuit that on input an index $i \in \{1, \ldots, d\}$, returns a key that allows one to evaluate up to depth d. Setting $d = 2^{\omega(\log(\lambda))}$ allows one to evaluate circuits of any polynomial depth, without affecting

the efficiency of the construction. The only subtlety here is that we need to select a base leveled homomorphic encryption scheme with *pseudorandom keys and ciphertexts* (such as [BV11, GSW13]) to be able to appeal to the security of iPRO.

Succinct randomized encodings. We can similarly construct succinct randomized encodings [BGL+15] by obfuscating the program that generates a garbled circuit locally. That is, given the description of a gate i, the program produces the garbled version of the gate. To make this template go through with PRO, we need two additional ingredients: (a) ensure that the output of the garbled circuit is pseudorandom, which we ensure by garbling a circuit that computes not $C(x)$ but rather $C(x) \oplus r$ for a random r, and publishing the garbled circuit together with r; and (b) ensure that the garbled circuit itself is random, which we ensure by appealing to the notion of blind garbling [BLSV18].

Making this template go through with the weaker notion of iPRO requires a puncturing argument and follows closely the machinery of [BGL+15].[1]

Succinct witness encryption. Next, we discuss how to construct a *succinct* witness encryption scheme [GGSW13]. This is the only application we discuss that uses the full power of PRO. In particular, this application is a case-study in how to overcome the informal "input-size barrier" faced by iO-based constructions; for more on this barrier, see the work of Jain and Jin [JJ22].

As a first step, assume we have a pseudorandom witness encryption WE scheme with long ciphertexts of length polynomial in the statement and witness length. In such a witness encryption scheme, the ciphertext for any $x \notin L$ and any message m is pseudorandom. To turn this into a succinct WE scheme, think of the witness encryption ciphertext as the truth-table of a function (with the NP statement x and the message bit b hard-coded) which, on input i, outputs the i-th bit of the ciphertext. Now, apply PRO for TMs (which exists assuming PRO for circuits) and let the PRO obfuscation be the succinct WE ciphertext. The size of the ciphertext only depends on the size of the Turing machine description of the underlying program, which is independent of the witness size (although it does depend on the statement size).

A pseudorandom witness encryption can be constructed in a few different ways:

- The scheme of [VWW22] constructs a pseudorandom witness encryption scheme directly from evasive LWE.
- The works of [WZ17, GKW17] construct compute-and-compare obfuscation from LWE, which allows one to generically upgrade a witness encryption scheme into one with pseudorandom ciphertexts. Therefore, using either iO or even iPRO along with LWE, one can construct a pseudorandom witness encryption scheme.

1.3 Technical Overview II: From iPRO to iO

Finally, we show how iPRO can be used generically to achieve the regular notion of iO, if one is willing to rely on additional computational assumptions.

To get an intuitive understanding of the idea behind our transformation, it is useful to pretend that we already have a general-purpose iO and see how to combine it with a iPRO in order to get an XiO. We then rely on known bootstrapping theorems [LPST16] to obtain another general-purpose

[1] The version we present only works for small-space computations, although we suspect this restriction can be removed using similar techniques to those in [KLW15, AL18, GS18].

iO. This is of course a vacuous statement but, later in the overview, we will discuss how to relax the initial assumptions without affecting the conclusion.

To obfuscate a circuit C, we sample n PRF keys K_i and partition the truth table of C into n blocks. For the i-th block, we compute an iPRO-obfuscation $\tilde{C}_i$ of the circuit $C_i^*(x) = C(x) \oplus \mathsf{PRF}(K_i, x)$, restricted to the x's in the i-th input block. Additionally, we compute an iO-obfuscation $\tilde{F}$ of the function $F(i, x) = \mathsf{PRF}(K_i, x)$. To evaluate the obfuscated circuit on some input x belonging to the i-th block, one can compute:

$$\tilde{C}_i(x) \oplus \tilde{F}(i, x) = C(x) \oplus \mathsf{PRF}(K_i, x) \oplus \mathsf{PRF}(K_i, x) = C(x).$$

To show why the scheme is secure, we begin with an obfuscation of a circuit C, which we gradually change to an obfuscation of a functionally-equivalent C', in a series of hybrid distributions. In the i-th hybrid, we program the obfuscated circuit $\tilde{F}$ to hardwire the values $\{\mathsf{PRF}(K_i, x)\}_x$ for all x in the i-th block. Clearly, the functionality of the circuit is preserved, so this modification is computationally indistinguishable, by the security of iO. Let us now make an important observation: The value of $\mathsf{PRF}(K_i, x)$ can alternatively be computed as

$$\tilde{C}_i(x) \oplus C(x) = \mathsf{PRF}(K_i, x)$$

where $\tilde{C}_i(x)$ is publicly available to the distinguisher. This effectively transfers the dependency on the circuit C from $\tilde{C}_i(x)$ to $\tilde{F}$. In other words, the function table $\{\tilde{C}_i(x)\}_x$ of the i-th input block is now pseudorandom, since any other item in the view of the distinguisher can be computed from $\{\tilde{C}_i(x)\}_x$ and the truth-table $\{C_i(x)\}_x$ (which is public information), and the output of the PRF masks the real output of the circuit.

This means that we can now appeal to the security of iPRO, in order to switch the obfuscated function from $C(x) \oplus \mathsf{PRF}(K_i, x)$ to $C'(x) \oplus \mathsf{PRF}(K_i, x)$, which are also functionally equivalent. Undoing the previous change and continuing this way for all input blocks, yields the desired implication. As a note for the advanced reader, we remark that this puncturing strategy requires us to set the size of $\tilde{F}$ to be proportional to the size of an input block, but this will be amortized by taking n to be large enough, thus allowing us to achieve XiO.

To tie the loose ends, we have to show how to instantiate the iO-obfuscation for $\tilde{F}$ without resorting to general purpose iO. Our observation is that it suffices to (i) build an obfuscation for a *single* (arbitrary) PRF and that (ii) even an iO with sublinear compactness (in the truth-table) suffices, since we are constructing XiO. We propose two different instantiations.

- First, note that a *random oracle* is already an obfuscation of a pseudorandom function! Therefore, if we are willing to rely on idealized models of security, iPRO alone already suffices to construct iO. To prove this, we rely on the pseudorandom oracle model [JLLW23], which allows one to formally prove theorems about obfuscations of idealized functions.
- Our second instantiation is in the standard model, and it relies on the pseudorandomness of the (small domain) PRF

  $$\mathsf{PRF}(K, (i, j)) = g^{x_i y_j}$$

  where the PRF key K is the collection of all x_i and y_j. By the decisional Diffie-Hellman assumption,[2] we have that the two distributions

  $$\{g^{x_i y_j}\} \approx_c \{g^{z_{i,j}} : z_{i,j} \leftarrow \mathbb{Z}_p\}$$

[2]Technically we will rely on the hardness of the Diffie-Hellman problem over asymmetric bilinear groups, which is known as the SXDH assumption.

are computationally indistinguishable. The reason why we use such a PRF, is that existing constructions of functional encryption schemes for quadratic functions [Wee20] can be seen (up to a rearranging of the algorithms) as XiO-obfuscation for this particular function. Since the scheme from [Wee20] can be proven secure assuming the hardness of the DLIN assumption on bilinear groups, we obtain the final implication that iPRO plus the two computational assumptions mentioned above (namely, SXDH and DLIN on bilinear groups) suffice to construct iO.

A New Perspective and a Challenge for the Future. We view our transformation from iPRO to iO above as a generalization of the FHE with hints approach of [BDGM20]. The BDGM construction of iO involves two steps: the first step is to produce encrypted outputs of C; in [BDGM20], this is achieved by releasing the FHE encryption of C. The second step is to produce decryption hints. This is the hard part since we need to hide both the secret key and the randomness.

Our iPRO to iO transformation cleanly separates out these two tasks:

- The iPRO is responsible for generating encrypted outputs of C. Importantly, we do not need to use FHE for encryption; rather, a specific PRF-based encryption scheme is sufficient.
- The iO for (small-domain) PRFs is responsible for decryption hints.

The use of bilinear maps is only in the second step, a clearly defined goal. This sets up a clear challenge for future, namely, building such scheme without using functional encryption for quadratic functions, which itself seems hard to build from LWE.

1.4 Discussion on Pseudorandom Obfuscation

We discuss in more detail the different variants of PRO and how the notion compares with existing notions of obfuscation.

On the Meaning of the Impossibility Result for PRO. In this work, we consider three variants of the notion of pseudorandom obfuscation — dPRO, PRO and iPRO. Our main theorem shows that the former variant exists for all pseudorandom functions, under the evasive LWE assumption. However, in Section 9 we also show that there are contrived truth-table-pseudorandom function family for which PRO does not exist. In light of this, we advise maximum caution in the use of (d)PRO to build new cryptography.

Backtracking, we see that the fact that we have just proven the existence of a cryptographic primitive that is also impossible, is due to the fact that we started from a computational assumption that cannot be true. It was already known that evasive LWE cannot hold in general, although the variant that we use in this work is even *weaker* than that used in prior work [Tsa22, VWW22, MPV24]. So, a pessimistic interpretation of our work is the discovery of a new counterexample for a weaker variant of evasive LWE. We refer the reader to Section 1.5 for a more thorough discussion on evasive LWE, whereas in what follows we focus on notion of PRO itself.

Our interpretation of our collection of results is much more optimistic, for the following reasons:

First, we remark that the *weaker* definition of iPRO, where security is guaranteed for pseudorandom functions with identical truth tables, does not suffer from this impossibility and in fact it is

weaker than the standard notion of iO. Furthermore, barring a single exception (succinct witness encryption), iPRO suffices to obtain all applications presented in this work.

Secondly, while PRO for general functionalities is impossible to achieve, it is entirely plausible that restricting the attention to specific classes of functions can yield a feasible notion. For instance, obfuscation of *null functions* exists assuming LWE, as shown in [WZ17, GKW17], and we conjecture here that PRO for the class of *puncturable* pseudorandom functions is also not subject to our impossibility result.

Thirdly, from PRO, we construct succinct witness encryption which is currently not known to exist under falsifiable computational assumptions.[3] Nor do existing impossibility results rule out this notion. Loosely speaking, known impossibility results rely on *incompressibility arguments*, where one ultimately reaches the conclusion that an obfuscation can compress random strings, a contradiction. However, in order to set this up, one needs the ability to include in the output of the program some pseudorandom bitstrings. The interface of witness encryption offers no such option, making this approach not viable. Overall, we have no evidence against the existence of succinct witness encryption as a primitive. Thus, we view our work as a heuristic first step towards understanding the feasibility of succinct witness encryption which is currently outside of the reach of our techniques. This is not unlike proofs in the random oracle or in the generic group model, two well-known heuristic models in cryptography, that have proven very robust and successful in advancing our understanding.

Finally, in addition to the quest for new cryptographic primitives, we argue that PRO may be useful also in the study of obfuscation on its own. In fact, our construction of PRO is based entirely on *lattice-based* techniques, and has a proof of security against evasive LWE and the LWE assumptions. On the other hand, known iO constructions from well-founded assumptions [JLS21, JLS22] rely on different assumptions, that are *insecure* against quantum algorithms. Thus, our PRO construction is, at present, the only candidate *post-quantum* construction of obfuscation for a large class of functions with a proof of security against a popular (although heuristic) computational assumption. (The one exception is compute-and-compare obfuscation [WZ17, GKW17] which obfuscates a much more restricted class of functions, similar in spirit to null functions; see below for a more detailed discussion.) Given the similarity of our construction with existing lattice-based iO candidates [GP21, WW21, BDGM22], we optimistically view our PRO scheme as evidence of the existence of post-quantum iO.

PRO vs iO. The notion of PRO is both stronger and weaker than iO (and hence incomparable). First of all, it is clear that the security of iO applies to a wider class of functions (in fact, all polynomially-computable ones) and it is a *worst case* guarantee, that is, it applies to any pair of functionally equivalent circuits. On the other hand, PRO only offers security for functions that are computationally indistinguishable from uniform, and it is an *average case* guarantee, since security only holds over the random choice of the key K. However, when the security offered by PRO applies, it appears to be slightly stronger than that of iO. In particular, it gives a security notion that is akin to ideal obfuscation.

In fact, the security guarantee provided by iO, namely indistinguishable obfuscations for functionally equivalent programs, turned out to be a bit cumbersome, and it took considerable effort to harness the full potential of iO (thirteen years from [BGI+01] to [SW14]). To get some evidence

[3] Constructions are known via differing-inputs (aka extractability) obfuscation or extractable witness encryption [BCP14, ABG+13]

for this claim, recall that the typical iO proof proceeds by *puncturing* out bad inputs one by one, each time making sure that the functionality of the program is unchanged, in order to appeal to the indistinguishability guarantee offered by iO. On the other hand, no such arguments are needed with PRO: if one can establish that the function table of the obfuscated program is pseudorandom, then security allows one to *replace* the obfuscated circuit in one shot. Note that this step does *not* preserve the functionality of the obfuscated program which, if we use a dPRO scheme, is in fact substituted by something maximally non-functional, i.e. truly random bits. The adversary cannot tell that functionality has been removed, and this is only possible since PRO applies only to pseudorandom function families. This security argument is typically simpler and it allows for better parameters, which in turn enables applications not known from iO.

PRO vs. Probabilistic iO. The notion of probabilistic iO (pIO) [CLTV15] gives us a way to obfuscate a probabilistic circuit $C(x;r)$ and requires that the obfuscations of two probabilistic circuits $C(x;r)$ and $D(x;r)$ that generate indistinguishable distributions for all x are themselves indistinguishable. [CLTV15] show a construction of (one of their variants of) pIO from IO. Their construction obfuscates the (deterministic) circuit $C'_K(x) = C(K; \mathsf{prf}_K(x))$, i.e. it generates the random coins for each input x by applying a pseudorandom function prf to x. Now, the two circuits C'_K and D'_K have indistinguishable, *but not necessarily pseudorandom*, truth tables. Our construction of PRO, on the other hand, crucially requires the truth table to be pseudorandom; thus, we view PRO as a weakening of pIO.

PRO vs. Compute-and-Compare Obfuscation. Yet another average-case notion of obfuscation where the function truth-table is required to have entropy is the notion of compute-and-compare obfuscation (also called lockable obfuscation) [WZ17, GKW17]. In lockable obfuscation corresponding to a function $f : \{0,1\}^n \to \{0,1\}^m$, one obfuscates a program $\Pi_{f,y}(x) = 1$ if $f(x) = y$ and 0 otherwise. Importantly, the string $y \in \{0,1\}^m$ is required to have sufficient pseudo-entropy. [WZ17, GKW17] construct compute-and-compare (CC) obfuscation from the hardness of LWE.

In comparison to PRO, the truth-table in CC obfuscation is definitely not pseudorandom, but it does have structured entropy that allows constructions from LWE.

1.5 The Computational Assumptions

The LWE assumption [Reg05] is, at this point, *the* standard assumption in lattice-based cryptography, so we do not discuss it any further here and just refer to Section 2.3 for details. In the following, we assume typical LWE parameters, i.e., we work over a ring $\mathbb{Z}_q$ and assume for convenience and simplicity that q is a power of 2, whereas error terms are drawn from discrete Gaussian distributions with suitable parameters. The *evasive* LWE problem is a recent but popular assumption introduced in the context of broadcast encryption [Wee22] and witness encryption [Tsa22, VWW22]; we use a variant from the recent work of Brzuska, Unal and Woo [BUW24]. Let $\mathbf{P} \in \mathbb{Z}_q^{m' \times n}$ be a uniformly random matrix. Fix some efficiently samplable distribution $(\mathbf{S}, \mathsf{aux})$, possibly depending on $\mathbf{P}$, where $\mathbf{S} \in \mathbb{Z}_q^{n \times \ell}$ is a matrix and aux is a bit-string. The evasive LWE assumption states that if

$$(\mathbf{B}, \mathbf{P}, \mathbf{BS} + \mathbf{E}, \mathbf{PS} + \mathbf{E}', \mathsf{aux}) \approx_c (\mathbf{B}, \mathbf{P}, \mathbf{U}, \mathbf{U}', \mathsf{aux})$$

where $(\mathbf{B}, \mathbf{U}, \mathbf{U}')$ are uniformly sampled matrices with $\mathbf{B} \in \mathbb{Z}_q^{m \times n}$ and $(\mathbf{E}, \mathbf{E}')$ are Gaussian error matrices, then

$$(\mathbf{B}, \mathbf{P}, \mathbf{BS} + \mathbf{E}, \mathbf{B}^{-1}(\mathbf{P}), \mathsf{aux}) \approx_c (\mathbf{B}, \mathbf{P}, \mathbf{U}, \mathbf{B}^{-1}(\mathbf{P}), \mathsf{aux})$$

where $\mathbf{B}^{-1}(\mathbf{P})$ is a matrix with small entries such that $\mathbf{B}^{-1}(\mathbf{P}) \cdot \mathbf{B} = \mathbf{P} \pmod q$. The design philosophy behind this assumption is that the pre-condition appears to rule out known attack strategies from the literature, such as the well-known *zeroizing* attacks. In other words, the assumption postulates that, given a matrix $\mathbf{D} := \mathbf{B}^{-1}(\mathbf{P})$ which is a "Gaussian pre-image" of a random matrix $\mathbf{P}$ under $\mathbf{B}$, the only thing that the distinguisher can do with LWE samples $\mathbf{BS} + \mathbf{E}$ is expand them into LWE "pseudo-samples" $\mathbf{D} \cdot (\mathbf{BS} + \mathbf{E}) = \mathbf{PS} + \mathbf{DE}$. Note that the auxiliary information aux may depend on $\mathbf{P}$, whereas $\mathbf{B}$ is chosen after aux[4]. We refer the reader to [Wee22, Tsa22, VWW22] for a more thorough discussion, and we mention here that the version that we define in Section 2.4 and use is seemingly *weaker* than the variant proposed in the context of witness encryption [Tsa22, VWW22], in that the matrix $\mathbf{P}$ is sampled uniformly and provided to the sampler as an input, whereas in their version the matrix $\mathbf{P}$ is sampled jointly with $\mathbf{S}$ and aux.

We nevertheless caution the reader that evasive LWE is a non-falsifiable assumption (borrowing terminology from [Nao03]) and that counterexamples are known [VWW22]. Somewhat interestingly though, known counterexamples from [VWW22] do not directly apply to the variant that we consider in this work, since the matrix $\mathbf{P}$ here is sampled without a trapdoor. Even more recent attacks [BUW24] do not apply to our variant since the matrices $(\mathbf{P}, \mathbf{B})$ are given to the distinguisher both in the pre- and post-condition. This roughly corresponds to the *binding* variant of evasive LWE, which is conjectured to hold in [BUW24]. With that being said, we do not suggest that our version of evasive LWE is sound (in fact we prove the exact opposite) and therefore the usual caveats of evasive LWE also apply to our work.

Overall, evasive LWE was motivated as a *bridge* between the standard LWE assumption and iO, and was used to prove the security of lattice-based constructions of broadcast encryption [Wee22], witness encryption [Tsa22, VWW22], multi-authority [WWW22] and unbounded-depth [HLL23] attribute-based encryption, all tools that we can construct from iO but not from LWE. In this work, we show that evasive LWE implies cryptography that seems out of reach even for iO, in our current understanding.

1.6 Technical Overview III: Construction of dPRO

As discussed above, in this work we pursue the *compactness approach* to obfuscation, along the lines of [AJ15, BV15, LPST16]. Roughly speaking, this approach consists of three main steps:

- Achieving non-trivial compactness: We construct a obfuscation scheme whose only effect is to slightly reduce the size of the obfuscated program, compared to the trivial obfuscation that just returns the function table.
- Delegation: We then add a mechanism to delegate computation, making the runtime of the obfuscator slightly sublinear, compared to the trivial obfuscation that evaluates the program on all inputs.
- Bootstrapping: Once we have achieved the desired efficiency, we can bootstrap our construction to a full-fledged obfuscation.

[4] However, note that the matrix $\mathbf{B}$ can be easily recovered from $\mathbf{D}$ and $\mathbf{P}$ via basic linear algebra for typical parameters, i.e. when $\mathbf{P}$ is wider than $\mathbf{B}$.

We describe these steps in more detail next. In the following, we will assume familiarity with basic lattice techniques such as trapdoor sampling, gadget matrices and binary decomposition. For details on these techniques, the reader is referred to Section 2.

Compactness from Homomorphic Encryption. Our starting point to achieve initial compactness is the candidate indistinguishability obfuscator of Brakerski et al. [BDGM20] (see also [GP21, WW21, BDGM22]). A circuit $C : \{1, \ldots, \kappa\} \to \{0,1\}^k$ is obfuscated by encrypting it under a suitable (leveled) FHE scheme (for concreteness, we think of the GSW scheme [GSW13]). Given an encryption $c = \mathsf{Enc}_{\mathsf{sk}}(C)$ of C, the evaluator can homomorphically expand it into a ciphertext vector encrypting the entire truth table of C, i.e., we obtain

$$(c_1, \ldots, c_\kappa) \quad \text{where} \quad c_i = \mathsf{Enc}_{\mathsf{sk}}(C(i)).$$

Note now that revealing the corresponding secret key sk to the evaluator is not an option, as it would allow him to also decrypt $\mathsf{Enc}(C)$. Furthermore, revealing the ciphertext randomnesses of $(c_1, \ldots, c_\kappa)$ is also not an option, as it is large in the sense that it scales with the size of the truth table of C and would furthermore also reveal information about C.

The critical idea of [BDGM20] is that we can *recrypt* the ciphertext vector $(c_1, \ldots, c_\kappa)$ into an encryption scheme which has a succinct randomness. This recryption is facilitated by the fact that approximate decryption for GSW is a linear function (of the secret key). Hence, by providing an encryption of sk under a suitable linearly homomorphic encryption scheme, we can recrypt $(c_1, \ldots, c_\kappa)$ into a batch-ciphertext $\mathbf{A}$ which can be "opened" by revealing a succinct random string. In [BDGM20], the choice for this linearly homomorphic encryption scheme was the Damgård-Jurik [DJ01] cryptosystem, whereas in [BDGM22, GP21, WW21] these were schemes following the *dual-Regev* framework [GPV08]. In the latter, the secret key is a matrix $\mathbf{R} \in \mathbb{Z}_q^{n \times km}$, whereas an encryption of a message $\mathbf{M} \in \mathbb{Z}_q^{k \times kn}$ consists of a uniformly random matrix $\mathbf{P} \in \mathbb{Z}_q^{k \times n}$ and a matrix $\mathbf{A} = \mathbf{PR} + \bar{\mathbf{E}} + \mathbf{M} \otimes \mathbf{g}^\top \in \mathbb{Z}_q^{k \times km}$ (where $\mathbf{g}^\top = (1, 2, 4, \ldots, 2^{\log(q)})$ is a *gadget* vector). Here, $m = n \log q$. For a linear function given by a matrix $\mathbf{F} \in \mathbb{Z}_q^{kn \times \kappa}$, we can obtain an encryption of $\mathbf{M} \cdot \mathbf{F}$ by computing

$$\mathbf{A} \cdot \mathbf{G}^{-1}(\mathbf{F}) = \mathbf{P} \cdot \mathbf{R} \cdot \mathbf{G}^{-1}(\mathbf{F}) + \bar{\mathbf{E}}\mathbf{G}^{-1}(\mathbf{F}) + \mathbf{MF} \approx \mathbf{P} \cdot \mathbf{R} \cdot \mathbf{G}^{-1}(\mathbf{F}) + \mathbf{MF}.$$

The important observation is that, for a suitable choice of parameters, the size of the randomness $\mathbf{RG}^{-1}(\mathbf{F}) \in \mathbb{Z}_q^{n \times \kappa m}$ is substantially smaller than the size of the encrypted message $\mathbf{MF}$. Hence, by revealing $\mathbf{RG}^{-1}(\mathbf{F})$ we can allow an evaluator to learn $\mathbf{MF}$.

In our setting, the encrypted message $\mathbf{M}$ is of the form $\mathbf{I} \otimes \mathsf{sk}^\top$, where sk is the secret key of the GSW scheme, whereas the linear function $\mathbf{F}$ is the *vectorization* of the (approximate) decryption function of GSW with the ciphertexts $(c_1, \ldots, c_\kappa)$ hardwired, i.e.

$$(\mathbf{I} \otimes \mathsf{sk}^\top) \cdot \mathbf{F} \approx (C(1), \ldots, C(\kappa)) \in \mathbb{Z}_q^{k \times \kappa}.$$

First observe that the asymptotics work in our favor now: For $\kappa = k^2$, the complete function table $(C(1), \ldots, C(\kappa))$ is of size $\sim k \times \kappa = k^3$, whereas both $\mathbf{P}$ and $\mathbf{RG}^{-1}(\mathbf{F})$ are of size $\lesssim k^2$, so we do achieve compression, as we can discount the size of $c = \mathsf{Enc}(C)$.

However this presentation is too simplistic, and in fact a closer look at the above scheme reveals critical security issues: The term $\mathbf{RG}^{-1}(\mathbf{F})$ together with the matrix $\mathbf{F}$ reveal $\mathbf{R}$ and therefore sk

entirely as we can recover $\mathbf{R}$ by solving a linear system of equations. To address this issue, all above-mentioned work [BDGM20, BDGM22, GP21, WW21] resorted to some mechanism to generate a *magic encryption of 0*, that is a ciphertext $\mathbf{C}'$ of the form $\mathbf{C}' = \mathbf{PS} + \mathbf{E}' \in \mathbb{Z}_q^{k \times \kappa}$. Given such a $\mathbf{C}'$, the obfuscator can provide $\mathbf{H} = \mathbf{RG}^{-1}(\mathbf{F}) + \mathbf{S}$ to the evaluator, who can recover $(C(1), \ldots, C(\kappa))$ by computing

$$\mathbf{A} \cdot \mathbf{G}^{-1}(\mathbf{F}) + \mathbf{C}' - \mathbf{PH} \approx (C(1), \ldots, C(\kappa)).$$

The issue with this approach though is that the ciphertext $\mathbf{C}'$ is of size $\sim k \times \kappa$, and thus no compression takes place if $\mathbf{C}'$ was included in the obfuscation.

The aforementioned works [BDGM20, BDGM22, GP21, WW21] resorted to new ad-hoc "circularity" assumptions in order to generate such ciphertexts $\mathbf{C}'$ that are simultaneously compressible and capable of performing the above rerandomization task. While none of these schemes were broken, some of the underlying assumptions turned out to be invalid [HJL21, JLLS23].

Anchoring Compactness in Evasive LWE. In this work, we adopt a different strategy for the case of circuits C that compute pseudorandom functions, by relying on the evasive LWE assumption.

We will assume that C is a pseudorandom function with range $\mathbb{Z}_q^k$, and we will not concern ourselves with noise that is added on the low-order bits of the outputs of C. This is a valid simplification, as we can always encode the PRF output we are interested in into the most-significant bit of C, while computing all lower order bits using a different PRF with a fresh key. Furthermore, note that in the setting of pseudorandom obfuscation we do not try to hide the circuit C, but just a key K[5]. Hence, in the following we will write $\mathsf{Enc}(K)$ instead of $\mathsf{Enc}(C)$.

In a nutshell, we compress the $\mathbf{C}'$ ciphertext via two additional components: A *short* matrix $\mathbf{D} \in \mathbb{Z}^{k \times m}$, and a matrix $\mathbf{C} = \mathbf{BS} + \mathbf{E} \in \mathbb{Z}_q^{m \times \kappa m}$, where $\mathbf{B} \leftarrow \mathbb{Z}_q^{m \times n}$ is chosen uniformly random and $\mathbf{D}$ is such that $\mathbf{D} \cdot \mathbf{B} = \mathbf{P}$. First note that $\mathbf{D}$ is of size $\sim k$ and $\mathbf{C}$ of size $\sim \kappa$, hence together their size is sublinear in $k \cdot \kappa$. Now note that $\mathbf{D}$ and $\mathbf{C}$ together allow us to achieve a similar functionality as required, since we can compute

$$\mathbf{D} \cdot \mathbf{C} = \mathbf{DBS} + \mathbf{DE} = \mathbf{PS} + \mathbf{DE},$$

where the term $\mathbf{DE}$ is short as both $\mathbf{D}$ and $\mathbf{E}$ are short. The obvious question at this point is, why should providing $\mathbf{C}'$ via $\mathbf{D}$ and $\mathbf{C}$ yield a secure scheme, which in our case is a pseudorandom obfuscator. In other words, we want to argue that the tuple $(\mathsf{Enc}(K), \mathbf{P}, \mathbf{A}, \mathbf{D}, \mathbf{C})$ is pseudorandom.

A first issue here is that the matrix $\mathbf{D}$ follows a (component-wise) discrete Gaussian distribution, hence it is easily distinguishable from uniform. To address this issue, we will not provide the matrix $\mathbf{D}$ itself, but only the random coins used to sample it. However, if $\mathbf{D}$ was sampled using a conventional Gaussian sampler, we would be in trouble if we had to provide random coins that *explain* a matrix $\mathbf{D}$, which we did not sample ourselves. We overcome this issue by relying on a primitive we call *invertible* Gaussian sampler: An invertible sampler comes with an efficient procedure which, given a sample t of the distribution, generates coins r such that running the sampler using coins r outputs t. For *narrow* discrete Gaussians, i.e., having polynomial variance, we can immediately obtain an invertible sampler via classical *inverse transform sampling* (see e.g. [Ros98]). For wide discrete Gaussians, i.e., having super-polynomial variance, [AWY20] shows an efficient inverter for the sampler in [GPV08].

[5] In case C is a universal circuit, the key K actually encodes a circuit, hence in general encrypting keys and circuits is equivalent.

We can now argue that $(\mathsf{Enc}(K), \mathbf{P}, \mathbf{A}, \mathbf{H}, \mathbf{D}, \mathbf{C})$ is pseudorandom. In order to appeal to the evasive LWE assumption, we will include $\mathsf{Enc}(K), \mathbf{P}, \mathbf{A}, \mathbf{H}$ in auxiliary information aux. To keep this outline concise we will not consider additional auxiliary information relating to the key K, as it does not introduce additional challenges. Assume for now that the precondition of evasive LWE holds, i.e., that

$$(\mathbf{B}, \mathbf{P}, \mathbf{BS} + \mathbf{E}, \mathbf{PS} + \mathbf{E}', \mathsf{aux}) \approx_c (\mathbf{B}, \mathbf{P}, \mathbf{U}, \mathbf{U}', \mathsf{aux}).$$

Evasive LWE then postulates that

$$(\mathbf{B}, \mathbf{P}, \mathbf{BS} + \mathbf{E}, \mathbf{D}, \mathsf{aux}) \approx_c (\mathbf{B}, \mathbf{P}, \mathbf{U}, \mathbf{D}, \mathsf{aux}),$$

in other words

$$(\mathbf{B}, \mathbf{P}, \mathsf{Enc}(K), \mathbf{A}, \mathbf{H}, \mathbf{D}, \mathbf{C}) \approx_c (\mathbf{B}, \mathbf{P}, \mathsf{Enc}(K), \mathbf{A}, \mathbf{H}, \mathbf{D}, \mathbf{U}),$$

i.e., we can replace $\mathbf{C}$ with a uniformly random matrix $\mathbf{U}$. We will get back to the task of establishing the evasive LWE precondition later, and for now just assume that it holds.

At this point, instead of computing $\mathbf{H}$ via $\mathbf{H} = \mathbf{S} + \mathbf{RG}^{-1}(\mathbf{F})$ we can just choose $\mathbf{H}$ uniformly at random, as $\mathbf{S}$ is chosen uniformly random and does not appear anywhere else anymore. In the next hybrid step, we want to replace $\mathbf{A} = \mathbf{PR} + \bar{\mathbf{E}} + \mathbf{I} \otimes \mathsf{sk}^\top \otimes \mathbf{g}^\top$ (which encrypts the GSW secret key sk) with a uniformly random matrix. We cannot immediately use the LWE assumption as the matrix $\mathbf{D}$ is correlated with $\mathbf{P}$ via $\mathbf{DB} = \mathbf{P}$. However, instead of first choosing $\mathbf{D}$ and $\mathbf{B}$ and setting $\mathbf{P} = \mathbf{DB}$, we can choose $\mathbf{P}$ uniformly at random and generate a (statistically close to) uniform matrix $\mathbf{B}$ together with a lattice trapdoor td [GPV08, MP12]. The lattice trapdoor td enables us to sample $\mathbf{D}$ from a discrete gaussian under the constraint that $\mathbf{D} \cdot \mathbf{B} = \mathbf{P}$.

Before we proceed, recall that we need to represent $\mathbf{D}$ via the random coins used to sample it. However, now $\mathbf{D}$ is sampled via an entirely different process! Thus we would be indeed be stuck if $\mathbf{D}$ was sampled by a plain Gaussian sampler. However, as $\mathbf{D}$ was sampled via an invertible Gaussian sampler, we can use the inverse sampler to simulate random coins r which *explain* the matrix $\mathbf{D}$ as an output of this sampler. Hence, we can now replace $\mathbf{A} = \mathbf{PR} + \bar{\mathbf{E}} + \mathbf{I} \otimes \mathsf{sk}^\top \otimes \mathbf{g}^\top$ with a uniformly random matrix via a routine reduction to LWE. Finally, now that we have removed the encryption of the GSW secret key sk, we can appeal to the ciphertext and key pseudorandomness of GSW to replace $\mathsf{Enc}(K)$ with a uniformly random string.

Establishing the Evasive-LWE Precondition. The more involved step of this proof is to establish the evasive LWE precondition. That is, we have to show that

$$(\mathbf{B}, \mathbf{P}, \mathbf{BS} + \mathbf{E}, \mathbf{PS} + \mathbf{E}', \mathsf{aux}) \approx_c (\mathbf{B}, \mathbf{P}, \mathbf{U}, \mathbf{U}', \mathsf{aux}),$$

where $\mathsf{aux} = (\mathsf{Enc}(K), \mathbf{P}, \mathbf{A}, \mathbf{H})$ with $\mathbf{A} = \mathbf{PR} + \bar{\mathbf{E}} + \mathbf{I} \otimes \mathsf{sk}^\top \otimes \mathbf{g}^\top$ and $\mathbf{H} = \mathbf{S} + \mathbf{RG}^{-1}(\mathbf{F})$. Let us define $\mathbf{C} = \mathbf{BS} + \mathbf{E}$ and $\mathbf{C}' = \mathbf{PS} + \mathbf{E}'$. The first step is just a syntactic one, we choose $\mathbf{H}$ uniformly at random and set $\mathbf{S} = \mathbf{H} - \mathbf{RG}^{-1}(\mathbf{F})$. Hence, we compute $\mathbf{C}$ and $\mathbf{C}'$ via

$$\mathbf{C} = \mathbf{BH} - \mathbf{BRG}^{-1}(\mathbf{F}) + \mathbf{E} \quad \text{and} \quad \mathbf{C}' = \mathbf{PH} - \mathbf{PRG}^{-1}(\mathbf{F}) + \mathbf{E}'.$$

Furthermore, we introduce a new noise term $\mathbf{E}^*$ and set

$$\mathbf{C} = \mathbf{BH} - (\mathbf{BR} + \mathbf{E}^*)\mathbf{G}^{-1}(\mathbf{F}) + \mathbf{E}^*\mathbf{G}^{-1}(\mathbf{F}) + \mathbf{E}.$$

By introducing another variable $\mathbf{U}^* = \mathbf{BR} + \mathbf{E}^*$ we can write $\mathbf{C}$ as

$$\mathbf{C} = \mathbf{BH} - \mathbf{U}^*\mathbf{G}^{-1}(\mathbf{F}) + \mathbf{E}^*\mathbf{G}^{-1}(\mathbf{F}) + \mathbf{E}.$$

Similarly, we rewrite $\mathbf{C}'$ as

$$\mathbf{C}' = \mathbf{PH} - \mathbf{AG}^{-1}(\mathbf{F}) - (C(K,1),\ldots,C(K,\kappa)) + \bar{\mathbf{E}}\mathbf{G}^{-1}(\mathbf{F}) - \tilde{\mathbf{E}} + \mathbf{E}',$$

where we have used that $\mathbf{PR} = \mathbf{A} - \bar{\mathbf{E}} - \mathbf{I} \otimes \mathsf{sk}^\top \otimes \mathbf{g}^\top$ and $(\mathbf{I} \otimes \mathsf{sk}^\top \otimes \mathbf{g}^\top) \cdot \mathbf{G}^{-1}(\mathbf{F}) = (C(K,1),\ldots,C(K,\kappa)) + \tilde{\mathbf{E}}$ (by the approximate linear decryption property of GSW). By choosing the gaussian parameter of $\mathbf{E}$ and $\mathbf{E}'$ sufficiently large, we can appeal to a standard drowning argument to argue that $\mathbf{E}$ drowns $\mathbf{E}^*\mathbf{G}^{-1}(\mathbf{F})$ and $\mathbf{E}'$ drowns $\bar{\mathbf{E}}\mathbf{G}^{-1}(\mathbf{F}) - \tilde{\mathbf{E}}$. This hybrid replacement allow us to compute $\mathbf{C}$ and $\mathbf{C}'$ via

$$\mathbf{C} = \mathbf{BH} - \mathbf{U}^*\mathbf{G}^{-1}(\mathbf{F}) + \mathbf{E} \quad \text{and}$$
$$\mathbf{C}' = \mathbf{PH} - \mathbf{AG}^{-1}(\mathbf{F}) - (C(K,1),\ldots,C(K,\kappa)) + \mathbf{E}'.$$

That is, now we just need $\mathbf{U}^*$ to compute $\mathbf{C}$, but not $\mathbf{R}$ and $\mathbf{E}^*$. Likewise, we can compute $\mathbf{C}'$ from $\mathbf{A}$ (as well as K and additional public information), but without knowledge of $\mathbf{R}$ and $\bar{\mathbf{E}}$.

Realizing that only $\mathbf{U}^* = \mathbf{BR} + \mathbf{E}^*$ and $\mathbf{A} = \mathbf{PR} + \bar{\mathbf{E}} + \mathbf{I} \otimes \mathsf{sk}^\top \otimes \mathbf{g}^\top$ depend on $\mathbf{R}$ as well as $\mathbf{E}^*$ and $\bar{\mathbf{E}}$ respectively, we are set to make both $\mathbf{U}^*$ and $\mathbf{A}$ uniform via a routine reduction to LWE. Subsequently, since $\mathbf{A}$ does not depend on the GSW secret key sk anymore, we can replace $c = \mathsf{Enc}(K)$ with $c = \mathsf{Enc}(0)$ using the security of GSW.

Notice that we have not made use of fact the C computes a pseudorandom function so far. In the next step we will make use of this property, and replace the matrix $(C(K,1),\ldots,C(K,\kappa))$ by uniformly random values, hence we can just choose the matrix $\mathbf{C}'$ uniformly at random (as no other term depends on K anymore).

To establish the evasive LWE precondition we still have to argue that the matrix $\mathbf{C}$ is pseudorandom, and this turns out to be somewhat tricky. The matrix $\mathbf{U}^*$ is distributed uniformly random and appears nowhere else. However, a simple dimensional analysis reveals that $\mathbf{U}^* \in \mathbb{Z}_q^{m \times km}$ has insufficient entropy to fully randomize $\mathbf{C} \in \mathbb{Z}_q^{m \times \kappa}$, as $\kappa \approx k^2 \gg km$. Hence we have to rely on pseudorandomness for this step, and it is tempting to once again rely on the LWE assumption to argue that $-\mathbf{U}^*\mathbf{G}^{-1}(\mathbf{F}) + \mathbf{E}$ is pseudorandom. However, this is not readily possible as the matrix $\mathbf{F}$ is not distributed uniformly random, and in fact it is the result of homomorphically expanding the ciphertext c. Fortunately, a small modification of our scheme will help us make this argument go through: We will just add n uniformly random rows to the matrix $\mathbf{F}$. This will make enough room to embed an LWE challenge into $\mathbf{F}$ and $\mathbf{C}$. Specifically, let

$$\mathbf{F}' = \begin{pmatrix} \mathbf{F} \\ \mathbf{V} \end{pmatrix} \quad \text{for a uniformly random matrix } \mathbf{V}.$$

To preserve correctness of our scheme we also need to modify how the matrix $\mathbf{A}$ is computed in the scheme, which we set to

$$\mathbf{A} = \mathbf{PR} + \bar{\mathbf{E}} + (\mathbf{I}, \mathbf{0}) \otimes \mathsf{sk}^\top \otimes \mathbf{g}^\top,$$

where the additional column $\mathbf{0}$ in $(\mathbf{I}, \mathbf{0})$ ensures that any terms resulting from $\mathbf{V}$ are cancelled out. Furthermore, we include the matrix $\mathbf{V} \in \mathbb{Z}_q^{n \times \kappa}$ into the obfuscation, which does not violate compactness as it is of size $\kappa \cdot \mathsf{poly}(\lambda) = k^2 \cdot \mathsf{poly}(\lambda) \ll k^3$.

With this modification, we can now argue that

$$\mathbf{C} = \mathbf{BH} - \mathbf{U}^*\mathbf{G}^{-1}(\mathbf{F}') + \mathbf{E}$$

is pseudorandom under the LWE assumption. First, since $\mathbf{U}^*$ is distributed uniformly random we can equivalently compute it as $\mathbf{U}^* = \hat{\mathbf{U}} - (\mathbf{0}, \mathbf{T} \cdot \mathbf{G})$ for uniformly random matrices $\hat{\mathbf{U}}$ and $\mathbf{T}$ and an all-zero matrix $\mathbf{0}$ of appropriate size. Hence it holds that

$$\begin{aligned}\mathbf{C} &= \mathbf{BH} - \hat{\mathbf{U}}\mathbf{G}^{-1}(\mathbf{F}') + (\mathbf{0}, \mathbf{T} \cdot \mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{F}') + \mathbf{E} \\ &= \mathbf{BH} - \hat{\mathbf{U}}\mathbf{G}^{-1}(\mathbf{F}') + \mathbf{0} \cdot \mathbf{G}^{-1}(\mathbf{F}) + \mathbf{T} \cdot \mathbf{V} + \mathbf{E} \\ &= \mathbf{BH} - \hat{\mathbf{U}}\mathbf{G}^{-1}(\mathbf{F}') + \mathbf{TV} + \mathbf{E}.\end{aligned}$$

Hence, again via a standard reduction to LWE we can replace $\mathbf{TV} + \mathbf{E}$ and therefore $\mathbf{C}$ by a uniformly random matrix.

All that remains now to finish the proof for the evasive LWE precondition is to make c, $\mathbf{A}$ and $\mathbf{H}$ well-formed again, i.e., we replace $c = \mathsf{Enc}(0)$ by $c = \mathsf{Enc}(K)$, we set $\mathbf{A} = \mathbf{PR} + \bar{\mathbf{E}} + \mathbf{I} \otimes \mathsf{sk}^\top \otimes \mathbf{g}^\top$, and finally we set $\mathbf{H} = \mathbf{S} + \mathbf{RG}^{-1}(\mathbf{F})$. Hence we have established that

$$(\mathbf{B}, \mathbf{P}, \mathbf{BS} + \mathbf{E}, \mathbf{PS} + \mathbf{E}', \mathsf{aux}) \approx_c (\mathbf{B}, \mathbf{P}, \mathbf{C}, \mathbf{C}', \mathsf{aux}),$$

where $\mathbf{C}$ and $\mathbf{C}'$ are chosen uniformly random and $\mathsf{aux} = (\mathsf{Enc}(K), \mathbf{P}, \mathbf{A}, \mathbf{H})$ with $\mathbf{A} = \mathbf{PR} + \bar{\mathbf{E}} + \mathbf{I} \otimes \mathsf{sk}^\top \otimes \mathbf{g}^\top$ and $\mathbf{H} = \mathbf{S} + \mathbf{RG}^{-1}(\mathbf{F})$. This concludes the proof sketch for our exponentially efficient pseudorandom obfuscator.

Output-Compressing Laconic Function Evaluation. Thus, we are now equipped with an xPRO scheme which enables us to compress *pseudorandom objects*. The next step is to delegate the bulk of the computation to the evaluator, in order to achieve a *bootstrappable* construction [LPST16] (see also [BDGM22]). A sensible primitive for this task is an *output-compressing* laconic function evaluation (LFE) scheme [QWW18]. In a nutshell, this primitive allows an encrypter, who has an input x and a hash value h of a circuit C, to compute an encryption of $c = \mathsf{Enc}(h, x)$ from which a decrypter, who has the circuit C, can compute $C(x)$ but, critically, learns nothing else about x.

In terms of efficiency, the crucial feature of LFE is that both the encryption time and the ciphertext size are sublinear (and ideally independent) of the *size* of the circuit C. In the LWE-based construction of [QWW18] both the encryption time and ciphertext size depend on the depth d of the circuit C, as well as its input and output size. However, this is where we can rely on xPRO to reduce the dependency on the output size: Instead of outputting the LFE ciphertext, we return an xPRO obfuscation of the circuit that computes the ciphertext. This way we combine the delegation properties of LFE, with the compression properties of xPRO. Unfortunately, we cannot yet prove the security of this combined scheme. So far, we have no guarantee that the LFE ciphertext $c = \mathsf{Enc}(h, K)$ looks pseudorandom *given the circuit* C, and therefore we cannot appeal to the pseudorandomness of PRO.

Blind Laconic Function Evaluation. To address this issue we introduce an LFE scheme with a stronger security notion, inspired by the notion of blind garbled circuits of [BLSV18]. We say that an LFE scheme is *blind*, if for any pseudorandom $C(K)$ (for a randomly sampled K) the ciphertext $c = \mathsf{Enc}(h, K)$ is also pseudorandom even given the circuit C, the hash h, and potentially some

additional auxiliary information. In other words, blindness means that the LFE scheme preserves pseudorandomness, which is precisely the property that we need for our transformation above.

To keep things simple, in this overview we will actually focus on the weaker notion of attribute-based LFE (AB-LFE). AB-LFE has a similar syntax to LFE, but with a few small differences:

- The circuit C outputs just a single bit.
- The encrypter provides an input $x \in \{0,1\}^n$ as well as a message μ.
- The decrypter gets the ciphertext $c = \mathsf{Enc}(h, x, \mu)$ *as well as* x. If $C(x) = 0$ the decrypter will learn μ, otherwise the decrypter learns nothing about μ.

Let us now make more concrete the desiderata for a blind AB-LFE:

- If $C(x) = 0$, then c is computationally indistinguishable from uniform among all strings z satisfying $\mathsf{Dec}(C, x, z) = \mu$, given C, x, crs, h and μ
- If $C(x) = 1$, then c is computationally indistinguishable from uniform, given C, x, crs, h and μ.

Our blind AB-LFE scheme is identical to that of [QWW18], except for a single but crucial modification, that we describe next.

- The common reference string crs consists of uniformly random LWE matrices $\mathbf{A}_1, \ldots, \mathbf{A}_n$.
- The hashing algorithm takes the common reference string crs and a boolean circuit C and *deterministically* computes an LWE matrix $\mathbf{A}_C$ via homomorphic key evaluation [BGG+14, BTVW17].
- The encryption algorithm takes as input the common reference string crs, a hash value $\mathbf{A}_C$, input bits $x_1, \ldots, x_n \in \{0,1\}$ and a message $\mu \in \{0,1\}$. It computes ciphertexts

$$\boldsymbol{b}_i^\top = \boldsymbol{s}^\top \cdot (\mathbf{A}_i + x_i \cdot \mathbf{G}) + \boldsymbol{e}_i^\top \quad \text{for } i \in \{1, \ldots, n\},$$

  chooses a uniformly random $\boldsymbol{a}^* \in \mathbb{Z}_q^n$ and computes

$$\beta = \boldsymbol{s}^\top \mathbf{A}_c \mathbf{G}^{-1}(\boldsymbol{a}^*) + e' + \mu \cdot q/2$$

  where e' is an error term chosen uniformly from $[-q/4, q/4)$. The distribution of e' differs from the one used in [QWW18], and it will be crucial for our later proof. The ciphertext c consists of all of the above elements.
- The decryption algorithm takes as input the common reference string crs, the circuit C and a ciphertext c. Using $(\boldsymbol{b}_1^\top, \ldots, \boldsymbol{b}_n^\top, \mathbf{A}_1, \ldots, \mathbf{A}_n)$ as well as $x = (x_1, \ldots, x_n)$ it computes

$$\boldsymbol{b}_C^\top = \boldsymbol{s}^\top (\mathbf{A}_c + C(x) \cdot \mathbf{G}) + \boldsymbol{e}_C^\top$$

  via homomorphic ciphertext evaluation and outputs $\mathsf{MSB}(\beta - \boldsymbol{b}_C^\top \mathbf{G}^{-1}(\boldsymbol{a}^*))$.

To establish correctness, it suffices to observe that

$$\begin{aligned}
&\beta - \boldsymbol{b}_C^\top \mathbf{G}^{-1}(\boldsymbol{a}^*) \\
&= \boldsymbol{s}^\top \mathbf{A}_C \mathbf{G}^{-1}(\boldsymbol{a}^*) + e' + \mu \cdot q/2 - (\boldsymbol{s}^\top(\mathbf{A}_c + C(x) \cdot \mathbf{G}) + \boldsymbol{e}_C^\top)\mathbf{G}^{-1}(\boldsymbol{a}^*) \\
&= C(x) \cdot \boldsymbol{s}^\top \mathbf{G} \cdot \mathbf{G}^{-1}(\boldsymbol{a}^*) + \boldsymbol{e}_C^\top \mathbf{G}^{-1}(\boldsymbol{a}^*) + e' + \mu \cdot q/2 \\
&= C(x) \cdot \boldsymbol{s}^\top \boldsymbol{a}^* + \boldsymbol{e}_C^\top \mathbf{G}^{-1}(\boldsymbol{a}^*) + e' + \mu \cdot q/2.
\end{aligned}$$

For $C(x) = 0$, it holds that $\boldsymbol{e}_C^\top \mathbf{G}^{-1}(\boldsymbol{a}^*) + e' \in [-q/4, q/4)$ with high probability and therefore $\mathsf{MSB}(\beta - \boldsymbol{b}_C^\top \mathbf{G}^{-1}(\boldsymbol{a}^*)) = \mu$. To prove blindness, note that

$$\boldsymbol{s}^\top \mathbf{A}_C = \boldsymbol{b}_C^\top - C(x) \cdot \boldsymbol{s}^\top \cdot \mathbf{G} - \boldsymbol{e}_C^\top$$

since $\boldsymbol{b}_C^\top = \boldsymbol{s}^\top(\mathbf{A}_C + C(x) \cdot \mathbf{G}) + \boldsymbol{e}_C^\top$. Hence it holds that

$$\begin{aligned}
\beta &= (\boldsymbol{b}_C^\top - C(x) \cdot \boldsymbol{s}^\top \cdot \mathbf{G} - \boldsymbol{e}_C^\top) \cdot \mathbf{G}^{-1}(\boldsymbol{a}^*) + e' + \mu \cdot q/2 \\
&= \boldsymbol{b}_C^\top \cdot \mathbf{G}^{-1}(\boldsymbol{a}^*) - C(x) \cdot (\boldsymbol{s}^\top \cdot \boldsymbol{a}^* + \tilde{e}) - C(x) \cdot \tilde{e} - \boldsymbol{e}_C^\top \cdot \mathbf{G}^{-1}(\boldsymbol{a}^*) + e' + \mu \cdot q/2 \\
&=_s \boldsymbol{b}_C^\top \cdot \mathbf{G}^{-1}(\boldsymbol{a}^*) - C(x) \cdot \tilde{b} + e' + \mu \cdot q/2
\end{aligned}$$

where we set $\tilde{b} = \boldsymbol{s}^\top \boldsymbol{a}^* + \tilde{e}$. The third line follows from the fact that e' drowns the short terms $-C(x) \cdot \tilde{e} - \boldsymbol{e}_C^\top \cdot \mathbf{G}^{-1}(\boldsymbol{a}^*)$, by a standard statistical argument. We can now establish blindness as follows:

- For $C(x) = 0$, it holds that $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ are uniform, $\boldsymbol{a}^*$ is uniform and furthermore e' is uniform in $[-q/4, q/4)$. As $\boldsymbol{b}_C$ can be deterministically computed from $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$, $\mathbf{A}_1, \ldots, \mathbf{A}_n$, C and x, it holds that β is uniform across all values which satisfy $\mathsf{MSB}(\beta - b^* \mathbf{G}^{-1}(\boldsymbol{a}^*)) = \mu$. Hence the ciphertext c is uniform among all strings that decrypt to μ.
- For $C(x) = 1$, it holds that $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ are uniform, $\boldsymbol{a}^*$ is uniform. As $\tilde{b}$ is also uniform (and independent of the aforementioned terms), β is also uniform. Hence c is uniform.

Hence, we have obtained a blind AB-LFE scheme. To obtain a fully-fledged blind LFE scheme, we can rely on the same transformation as [GKP+13, QWW18], except using blind randomized encodings [BLSV18] and an FHE scheme with pseudorandom keys and ciphertexts[6]. We omit details here, and we refer the interested reader to Section 5 for details.

Bootstrapping Pseudorandom Obfuscation. Equipped with a blind and output-compressing LFE we now outline the final bootstrapping step in our construction of pseudorandom obfuscation. The main idea of the transformation comes from [BV15, AJ15] and it is based on the *input-extension* method. First, observe that it is easy to obfuscate a circuit $C(K, \cdot)$ (computing a pseudorandom function) where the input domain is a single bit, by simply computing

$$c_0 = \mathsf{Enc}(h, (K, 0)) \quad \text{and} \quad c_1 = \mathsf{Enc}(h, (K, 1))$$

where h is the hash of $C(\cdot, \cdot)$. One can recover $C(K, x)$ by running the LFE decryption, and security immediately follows from the blindness of the LFE, i.e. both c_0 and c_1 are pseudorandom. Next,

[6] For completeness, we added a sketch of this transformation in Appendix A.

we extend the input domain of this obfuscation by hashing increasingly longer functions. Assuming that we are inductively given an obfuscation for length i inputs, we can use it to construct an obfuscation for length $i+1$ inputs as follows: Obfuscate the circuit C_i that on input $x \in \{0,1\}^i$ return

$$c_{i+1,0} = \mathsf{Enc}(h_{i+1}, (K, x\|0)) \quad \text{and} \quad c_{i+1,1} = \mathsf{Enc}(h_{i+1}, (K, x\|1))$$

where h_{i+1} is the LFE-hash of the $i+1$-st circuit in this sequence. This way, we can construct an obfuscation for an arbitrary input domain, at the cost of one iteration per input bit. The most delicate aspect of this recursion is to ensure that the resulting obfuscation runs in polynomial time, regardless of the input length. It turns out that output-compressing LFE suffices to instantiate this transformation, thanks to the compactness and delegation properties. For more details, we refer the reader to Section 6.

1.7 Organization of the Paper

We now describe the organization of the rest of the paper. In Section 2, we introduce the relevant tools and definitions used in the paper. In Section 3, we introduce our various notions of pseudorandom obfuscation. In Section 4, we construct exponentially inefficient pseudorandom obfuscation. In Section 5, we construct a laconic function evaluation scheme satisfying blindness. In Section 6, we show how to bootstrap our xPRO construction to the setting to construct (d)PRO. Additionally, we also show how to construct PRO for Turing machines. In Section 7, we show how to use iPRO/PRO to construct fully homomorphic encryption (without circular security assumptions), succinct randomized encodings and succinct witness encryption. In Section 8, we show how to construct iO from PRO either using bilinear maps, or in the pseudorandom oracle model. Finally, in Section 9, we show counterexamples to our notions of PRO.

2 Preliminaries

Throughout this work, we write λ to denote the security parameter. We say a function f is negligible in the security parameter λ if $f(\lambda) = \lambda^{-\omega(1)}$. We say an algorithm is efficient if it runs in probabilistic polynomial time (PPT) in the length of its input. We say that two distributions are *computationally indistinguishable* (denoted by $\approx_c$) if no PPT distinguisher can tell them apart with probability negligibly better than $1/2$.

The statistical distance $\Delta(X; X')$ of two random variables X and X' supported on a set S is defined as $\Delta(X; X') = \frac{1}{2}\sum_{x \in S} |\Pr[X = x] - \Pr[X' = x]|$. If X and X' have statistical distance ε, we write $X \approx_\varepsilon X'$. If X and X' have negligible statistical distance, we write $X =_s X'$. Unless stated otherwise, we will generally assume such a negligible statistical distance is of order $\tilde{O}(2^{-\lambda}) = \mathsf{poly}(\lambda) \cdot 2^{-\lambda}$.

2.1 Linear Algebra

We denote vectors $\mathbf{v}$ in bold lowercase and matrices $\mathbf{M}$ in bold uppercase. We typically assume that vectors are column vectors and denote row vectors as transposes of column vectors, i.e. we write $\boldsymbol{v}^\top$ to denote a row vector corresponding to a column vector $\boldsymbol{v}$. The tensor (Kronecker) product

for matrices $\mathbf{A}$ and $\mathbf{B}$ is defined as

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{1,1}\mathbf{B}, & \dots, & a_{1,m}\mathbf{B} \\ \dots, & \dots, & \dots, \\ a_{n,1}\mathbf{B}, & \dots, & a_{n,m}\mathbf{B} \end{pmatrix}.$$

2.2 Lattices and Gaussians

For a vector $\boldsymbol{x} \in \mathbb{R}^n$ we will use $\|\boldsymbol{x}\| := \sqrt{\sum_i x_i^2}$ to denote the L_2 norm of $\boldsymbol{x}$ and $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$ to denote the L_∞ norm of $\boldsymbol{x}$.

We recall a few basic facts about lattices. An integer lattice $\Lambda \subseteq \mathbb{Z}^n$ is defined via

$$\Lambda = \Lambda(\mathbf{B}) = \{\mathbf{B} \cdot \boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{Z}^k\},$$

where $\mathbf{B} \in \mathbb{Z}^{n \times k}$ is a full-rank integer matrix. We call $k \leq n$ the rank of Λ. The dual lattice $\Lambda^* = \Lambda^*(\Lambda)$ of an integer lattice Λ is defined by

$$\Lambda^*(\Lambda) = \{\boldsymbol{x} \in \mathbb{R}^n \mid \forall \boldsymbol{y} \in \Lambda : \boldsymbol{x}^\top \boldsymbol{y} \in \mathbb{Z}\}.$$

For any $\sigma > 0$ the Gaussian function $\rho_\sigma : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is defined by

$$\rho_\sigma(\boldsymbol{x}) = e^{-\pi \cdot \|\boldsymbol{x}\|^2/\sigma^2}.$$

For any lattice $\Lambda \subseteq \mathbb{R}^n$ the quantity $\rho_\sigma(\Lambda) = \sum_{\boldsymbol{x} \in \Lambda} \rho_\sigma(\boldsymbol{x})$ is finite, hence we can define the discrete Gaussian distribution $D_{\Lambda,\sigma}$ via the probability mass function

$$\Pr[\boldsymbol{e} = \boldsymbol{x}] = \begin{cases} \rho_\sigma(\boldsymbol{x})/\rho_\sigma(\Lambda) \text{ for } \boldsymbol{x} \in \Lambda \\ 0 \text{ otherwise} \end{cases}$$

for an $\boldsymbol{e} \sim D_{\Lambda,\sigma}$.

The following tail bound for discrete Gaussians follows immediately from the fact that discrete gaussians are sub-gaussian and a simple tail bound for sub-gaussians, see e.g. [MP12].

Lemma 2.1. *Let $\sigma > 0$ and let $X \sim D_{\mathbb{Z},\sigma}$. Then it holds that*

$$\Pr[|X| \geq t\sigma] \leq 2 \cdot e^{-\pi \cdot t^2}.$$

Consequently, it holds that

$$\Pr[|X| \geq \sqrt{\lambda}\sigma] \leq 2 \cdot e^{-\pi\lambda} \leq \tilde{O}(2^{-\lambda}).$$

Recall the definition of the smoothing parameter of a lattice [MR04].

Definition 2.2. For a lattice Λ the smoothing parameter $\eta_\varepsilon(\Lambda)$ is defined to be the smallest σ for which $\rho_{1/\sigma}(\Lambda^*) \leq 1 + \varepsilon$.

We will use the following upper bound for the smoothing parameter of a lattice due to Micciancio and Regev [MR04].

Lemma 2.3. *Let Λ be an n-dimensional lattice and let $\varepsilon > 0$. Then it holds that*

$$\eta_\varepsilon(\Lambda) \leq \sqrt{\frac{\ln(2n(1+1/\varepsilon))}{\pi}} \cdot \lambda_n(\Lambda).$$

Hence it holds that

$$\eta_{2^{-\lambda}}(\Lambda) \leq O(\sqrt{\lambda}) \cdot \lambda_n(\Lambda).$$

For the special case of $\Lambda = \mathbb{Z}^n$ it holds that $\eta_{2^{-\lambda}}(\mathbb{Z}^n) = O(\sqrt{\lambda})$.

If $\sigma \geq \eta_\varepsilon(\Lambda)$, it holds by the Poisson summation formula (see e.g. [Reg05]) that

$$\rho_\sigma(\Lambda) = \sigma^n \det(\Lambda^*) \cdot \rho_{1/\sigma}(\Lambda^*),$$

and hence we get $\sigma^n \leq \rho_\sigma(\mathbb{Z}^n) \leq \sigma^n(1+\varepsilon)$.

The following is a general version of many *smudging* or *drowning* lemmata which have been used in many prior works, e.g. [GKPV10, AIK11, AJL+12].

Lemma 2.4. *Let χ be a symmetric and monotonously decreasing distribution supported on either $\mathbb{Z}$ or $\mathbb{Z}+\frac{1}{2}$, that is for $e \sim \chi$ we have for all $x' \geq x \geq 0$ in the support that*

$$\begin{aligned}\Pr[e = -x] &= \Pr[e = x] \\ \Pr[e = x'] &\leq \Pr[e = x].\end{aligned}$$

Then it holds for any $t \in \mathbb{Z}$ that

$$\Delta(e+t; e) = \Pr[e \in [-t/2, t/2)],$$

where $e \sim \chi$. Concretely:

- *If e is the uniform distribution on an interval $[-B_0/2, B_0/2]$, then $\Delta(e+t;e) = t/B_0$.*
- *If $e \sim D_{\mathbb{Z},\sigma}$, then $\Delta(e+t;e) \leq t/\sigma$.*

Proof. It holds that

$$\begin{aligned}\Delta(e+t;e) &= \Delta(e+t/2; e-t/2) \\ &= \frac{1}{2} \cdot \sum_x |\Pr[e = x - t/2] - \Pr[e = x + t/2]| \\ &= \sum_{x \geq 0} (\Pr[e = x - t/2] - \Pr[e = x + t/2]) \\ &= \Pr[e \geq -t/2] - \Pr[e \geq t/2] \\ &= \Pr[e \in [-t/2, t/2)],\end{aligned}$$

where the third equality holds due to symmetry and monotony of χ.

If e is uniform on an interval of size B_0, it holds that $\Pr[e \in [-t/2, t/2)] = \frac{t}{B_0}$. If e follows $D_{\mathbb{Z},\sigma}$, it holds that

$$\Pr[e \in [-t/2, t/2)] \leq t \cdot \Pr[e = 0] = t \cdot \frac{\rho_\sigma(0)}{\rho_\sigma(\mathbb{Z})} \leq \frac{t}{\sigma}.$$

□

Gentry, Peikert and Vaikuntanathan [GPV08] provide a generic method to sample a discrete gaussian distribution from a rank k lattice in $\mathbb{Z}^n$ given a sufficiently good basis.

Theorem 2.5. *Let $\mathbf{B} \in \mathbb{Z}^{n\times k}$, $\boldsymbol{c} \in \mathbb{Z}^n$ and let $\sigma \geq O(\sqrt{\lambda}) \cdot \lambda_n(\mathbf{B})$, where $\lambda_n(\mathbf{B})$ is the L_2-norm of the longest column of $\mathbf{B}$. There exists an efficient sampler which produces an output distribution statistically close to $D_{\Lambda(\mathbf{B}),\sigma,\boldsymbol{c}}$. It follows that for any $\boldsymbol{t} \in \mathbb{Z}^n$ we can efficiently sample statistically close to $D_{\Lambda(\mathbf{B})+\boldsymbol{t},\sigma}$, as $D_{\Lambda(\mathbf{B})+\boldsymbol{t},\sigma} \equiv \boldsymbol{t} + D_{\Lambda(\mathbf{B}),\sigma,-\boldsymbol{t}}$.*

Lattice Trapdoors. We recall the notion of lattice trapdoors from [GPV08, MP12].

Theorem 2.6. *There exists a pair of algorithms* $(\mathsf{TrapGen}, \mathsf{SampPre})$ *with the following syntax.*

- $\mathsf{TrapGen}(1^\lambda, q, n)$*: Generates a matrix $\mathbf{B} \in \mathbb{Z}_q^{m\times n}$ with $m = O(n\log(q))$ and trapdoor information* td*. We will assume wlog that* td *includes the matrix $\mathbf{B}$.*
- $\mathsf{SampPre}(\mathsf{td}, \boldsymbol{t}, \tau)$*: Takes a trapdoor* td*, a target $\boldsymbol{t} \in \mathbb{Z}_q^n$ and a parameter $\tau \geq \Omega(\sqrt{\lambda})$ and outputs a sample $\boldsymbol{x} \in \mathbb{Z}^m$.*

The following two properties hold.

- *(Uniformity) The matrices $\mathbf{B}$ generated by $\mathsf{TrapGen}(1^\lambda, q, n)$ are statistically close to uniform in $\mathbb{Z}_q^{m\times n}$*
- *(Trapdoor Sampling) For a matrix $\mathbf{B} \in \mathbb{Z}_q^{m\times n}$ and a vector $\boldsymbol{t} \in \mathbb{Z}_q^n$ define the coset $L(\mathbf{B}, \boldsymbol{t}) = \{\boldsymbol{x} \in \mathbb{Z}^m | \boldsymbol{x}\cdot\mathbf{B} = \boldsymbol{t}\}$. Then it holds that*

$$(\boldsymbol{x}\cdot\mathbf{B}, \boldsymbol{x}, \mathsf{td}) =_s (\boldsymbol{t}, \mathsf{SampPre}(\mathsf{td}, \boldsymbol{t}, \tau), \mathsf{td})$$

where $(\mathbf{B}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^\lambda, q, n)$, $\boldsymbol{x} \leftarrow D_{L(\mathbf{B},\boldsymbol{t}),\tau}$ and $\boldsymbol{t} \leftarrow \mathbb{Z}_q^n$.

To simplify notation, we will overload $\mathsf{SampPre}$ for matrices. For a matrix $\mathbf{T} = (\boldsymbol{t}_1, \ldots, \boldsymbol{t}_k) \in \mathbb{Z}_q^{n\times k}$ we will denote

$$\mathsf{SampPre}(\mathsf{td}, \mathbf{T}, \tau) = (\mathsf{SampPre}(\mathsf{td}, \boldsymbol{t}_1, \tau), \ldots, \mathsf{SampPre}(\mathsf{td}, \boldsymbol{t}_k, \tau)) \in \mathbb{Z}^{m\times k}.$$

Reverse Sampling Gaussians Samples from discrete Gaussian distributions have a discernible structure and are thus far from uniform. For one of our constructions, we need to represent samples from a discrete gaussian distribution in a way that looks pseudorandom. If such a sample x was generated by a sampling algorithm and we *happen to know* the random coins r that were used to generate x, we can simply represent x by the uniformly random coins r. For general sampling algorithms, recovering random coins that lead to a specific sample is infeasible, as the sampling algorithm may be a one-way function. For the case of Gaussians, however [AWY20] showed that the discrete Gaussian sampler of [GPV08] allows for *efficient reverse sampling*. That is, there exists an efficient algorithm $\mathcal{S}^{-1}$ which, given a sample $\boldsymbol{x}$ from a discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$ outputs random coins $r' \in \{0,1\}^{\mathsf{poly}(\lambda)}$ such that

$$(\mathcal{S}_\sigma(r), r) \approx_s (\boldsymbol{x}, \mathcal{S}_\sigma^{-1}(\boldsymbol{x})),$$

where $r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$ are random coins for the Gaussian sampler $\mathcal{S}$.

The idea for this reverse sampler is conceptually simple: The gaussian sampler of [GPV08] produces samples statistically close to $D_{\mathbb{Z},\sigma}$ by efficiently rejecting samples of a uniform distribution on the interval $[-\lambda\sigma, \lambda\sigma]$. As $D_{\mathbb{Z},\sigma}$ is supported on this interval, except with negligible probability, the output distribution of this sampler is statistically close to $D_{\mathbb{Z},\sigma}$. Reverse sampling this rejection sampler is now fairly straightforward.

Lemma 2.7 (Gaussian Reverse Sampling [GPV08, AWY20])**.** *There exists an efficient algorithm $\mathcal{S}^{-1}$ which, given a gaussian parameter σ and a sample $x \sim D_{\mathbb{Z},\sigma}$ outputs random coins $r' \in \{0,1\}^{\mathsf{poly}(\lambda)}$, such that*

$$(\mathcal{S}_\sigma(r), r) \approx_s (\boldsymbol{x}, \mathcal{S}_\sigma^{-1}(x)).$$

Note that this sampler immediately gives rise to an reversible sampler for $D_{\mathbb{Z}^{n\times m},\sigma}$, as this distribution can be sampled by sampling its components independently from $D_{\mathbb{Z},\sigma}$

Gadget Matrix. Let $\mathbf{g} = (1, 2, \ldots, 2^{\log(q)-1})$. The gadget matrix $\mathbf{G}$ is defined as

$$\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g} \in \mathbb{Z}_q^{n\times n\log(q)}.$$

We are going to use the fact that there exists an efficiently computable function $\mathbf{G}^{-1} : \mathbb{Z}_q^n \to \{0,1\}^{n\log(q)}$ such that for all $\mathbf{u} \in \mathbb{Z}_q^n$ we have $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{u})$.

For vectors $\mathbf{f} \in \mathbb{Z}_q^{k\cdot n}$ and $\boldsymbol{r} \in \{0,1\}^m$ we also define define a *randomized* gadget matrix inverse [MP12, BdMW16] as

$$\mathbf{G}_{\boldsymbol{r}}^{-1}(\boldsymbol{f}) = \begin{pmatrix} \tilde{\boldsymbol{f}} \\ \boldsymbol{r} \end{pmatrix},$$

where $\tilde{\boldsymbol{f}} \in \{0,1\}^{k\cdot n\cdot\log(q)}$ is the unique vector with $(\mathbf{I}_k \otimes \mathbf{G}) \cdot \tilde{\boldsymbol{f}} = \boldsymbol{f}$.

Homomorphic Evaluation. For an integer η, we consider the *lattice encodings*

$$\mathbf{b}_i = \mathbf{s}^T(\mathbf{A}_i - x_i\mathbf{G}) + \mathbf{e}_i^T \quad \text{such that} \quad \|\mathbf{e}_i\|_\infty \le B$$

where $\mathbf{s} \in \mathbb{Z}_q^n$, $\mathbf{A}_i \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{x} \in \{0,1\}^\eta$. And we are going to make use of the following deterministic homomorphic evaluation algorithm for matrix-valued circuits $C : \{0,1\}^\eta \to \mathbb{Z}_q^{n\times m}$ as described in [BGG+14, BTVW17].

- $\mathrm{EvalPK}(C, (\mathbf{A}_1, \ldots, \mathbf{A}_\eta))$: Takes as input the circuit C and n matrices $(\mathbf{A}_1, \ldots, \mathbf{A}_\eta)$ and returns a matrix $\mathbf{A}_C$.
- $\mathrm{EvalCT}(C, (\mathbf{A}_1, \ldots, \mathbf{A}_\eta), (\mathbf{b}_1, \ldots, \mathbf{b}_\eta), \mathbf{x})$: Takes as input the circuit C, matrices $(\mathbf{A}_1, \ldots, \mathbf{A}_\eta)$, encodings $(\mathbf{b}_1, \ldots, \mathbf{b}_\eta)$, and the input $\mathbf{x}$ and returns an encoding $\mathbf{b}_C$.

The key equation of homomorphic evaluation is:

$$\mathbf{b}_C = \mathbf{s}^T(\mathbf{A}_C - C(\mathbf{x})) + \mathbf{e}_C^T \quad \text{where } \|\mathbf{e}_C\|_\infty \le (m+2)^d \cdot B \cdot \log(q) \cdot m \cdot \eta$$

where $d = \mathsf{depth}(C)$.

2.3 The Learning with Errors Problem

We recall the definition of the decisional Learning with Errors (LWE) problem, introduced by Regev [Reg05].

Definition 2.8 (Decisional Learning with Errors)**.** Let $n = n(\lambda)$, $m = m(\lambda)$, $q = q(\lambda)$, and $\rho = \rho(\lambda)$ be integer parameters. The LWE problem is to distinguish between the distributions:

$$\left\{ (\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) : \begin{array}{l} \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m} \\ \mathbf{s} \leftarrow \mathbb{Z}_q^n \\ \mathbf{e} \leftarrow D_{\mathbb{Z},\rho}^m \end{array} \right\} \approx_c \left\{ (\mathbf{A}, \mathbf{u}) : \begin{array}{l} \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m} \\ \mathbf{u} \leftarrow \mathbb{Z}_q^m \end{array} \right\}.$$

The $\mathsf{LWE}(n, q, \rho)$ assumption is that no polynomial time algorithm can solve this problem with non-negligible advantage, for any polynomial number of samples m.

Throughout this work, we will always assume that the modulus q is of the form $q = 2^p$ for some $p \in \mathbb{Z}$. It is known [PRS17] that LWE is as hard as worst-case lattice problems for any choice of modulus.

2.4 Evasive LWE

The evasive LWE assumption [Wee22, Tsa22, VWW22] is a family of assumptions that postulates the hardness of a post-condition, if a certain pre-condition holds. In this work, we use a version called *private-coin binding evasive LWE*, a terminology introduced in [BUW24]. In this version, both the distinguisher for the pre-condition and the distinguisher for the post-condition get the matrices $\mathbf{B}$ and $\mathbf{P}$ as explicit inputs. Compared to the version used in prior works, our assumption appears to be slightly weaker than the one used in [VWW22]: in their definition, the matrix $\mathbf{S}$, the matrix $\mathbf{P}$ and the auxiliary information aux are jointly sampled. In our version, the sampling algorithm takes the matrix $\mathbf{P}$ as an explicit input.

There is another subtle definitional issue concerning the way evasive LWE is commonly phrased. Specifically, we typically grant the distinguisher $\mathcal{D}'$ for the post-condition more runtime than the pre-condition distinguisher $\mathcal{D}$. Furthermore, we also allow for a polynomial loss in the distinguishing advantage of $\mathcal{D}'$. In e.g. [MPV24] this is handled by requiring that for every sampler Samp there exists a polynomial Q such that when compared to $\mathcal{D}$ the runtime of $\mathcal{D}'$ grows at most by a factor of $Q(\lambda)$ and the distinguishing advantage of $\mathcal{D}$ drops at most by a factor of $1/Q(\lambda)$.

This is however problematic if the evasive LWE assumption is used a super-constant number of times in a hybrid argument, e.g., as done in [MPV24] and in our work. Specifically, in such a sequence of hybrids we construct a sequence of samplers $\mathrm{Samp}_1, \ldots, \mathrm{Samp}_k$. Now the polynomial Q_i, which is only existentially quantified by the sampler Samp_i could, e.g., have degree 2^{2^i}. This is a constant as long as k is a constant and thus i is a constant. However, this quickly becomes a problem once k grows with λ.

To deal with this definitional issue, we additionally require that the degree of this polynomial Q is bounded by $c \cdot d$ for some universal constant $c > 0$, where d is such that $|\mathrm{Samp}| \leq \lambda^d$ and $|\mathrm{Samp}|$ is the circuit-size of the sampler Samp.

We state our final definition in the following which, for notational convenience, we simply refer to as evasive LWE.

Definition 2.9 (Evasive LWE, adapted from [MPV24] and [BUW24])**.** Let $m, n, k, \kappa > 0$ be integers and let q be a modulus. Let $\tau, \sigma, \sigma' > 0$. Let Samp be an algorithm which takes 1^λ and a matrix $\mathbf{P} \in \mathbb{Z}_q^{k \times n}$ and outputs a matrix $\mathbf{S} \in \mathbb{Z}_q^{n \times \kappa}$ and auxiliary information aux. Let

$$\begin{aligned}
&\mathbf{D} \leftarrow D_{\mathbb{Z},\tau}^{k \times m} \\
&\mathbf{B} \leftarrow \mathbb{Z}_q^{m \times n} \\
&\mathbf{P} = \mathbf{D} \cdot \mathbf{B} \\
&(\mathbf{S}, \mathsf{aux}) \leftarrow \mathrm{Samp}(1^\lambda, \mathbf{P}) \\
&\mathbf{E} \leftarrow D_{\mathbb{Z},\sigma}^{m \times \kappa}, \mathbf{E}' \leftarrow D_{\mathbb{Z},\sigma'}^{k \times \kappa} \\
&\mathbf{C} \leftarrow \mathbb{Z}_q^{m \times \kappa}, \mathbf{C}' \leftarrow \mathbb{Z}_q^{k \times \kappa}
\end{aligned}$$

For PPT distinguishers $\mathcal{D}'$ and $\mathcal{D}$ define the following functions:

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{D}'}^{pre}(\lambda) &= |\Pr[\mathcal{D}'(\mathbf{B}, \mathbf{P}, \mathbf{BS} + \mathbf{E}, \mathbf{PS} + \mathbf{E}', \mathsf{aux}) = 1] - \Pr[\mathcal{D}'(\mathbf{B}, \mathbf{P}, \mathbf{C}, \mathbf{C}', \mathsf{aux}) = 1]| \\
\mathsf{Adv}_{\mathcal{D}}^{post}(\lambda) &= |\Pr[\mathcal{D}'(\mathbf{B}, \mathbf{P}, \mathbf{BS} + \mathbf{E}, \mathbf{D}, \mathsf{aux}) = 1] - \Pr[\mathcal{D}'(\mathbf{B}, \mathbf{P}, \mathbf{C}, \mathbf{D}, \mathsf{aux}) = 1]|
\end{aligned}$$

We say that the evasive LWE assumption $\mathsf{evLWE}(q, m, n, k, \kappa, \mathrm{Samp}, \tau, \sigma, \sigma')$ holds, if there exists polynomial Q such that $\deg_\lambda(Q) \leq c \cdot \deg_\lambda(|\mathrm{Samp}|)$ (for some universal constant $c > 0$), such that for every PPT distinguisher $\mathcal{D}$ there exists a PPT distinguisher $\mathcal{D}'$ such that

$$\mathsf{Adv}_{\mathcal{D}'}^{pre}(\lambda) \geq \mathsf{Adv}_{\mathcal{D}}^{post}(\lambda)/Q(\lambda) - \mathsf{negl}(\lambda)$$

and $\mathsf{time}(\mathcal{D}') \leq \mathsf{time}(\mathcal{D}) \cdot Q(\lambda)$.

Note that this is a non-falsifiable assumption (using the terminology of [Nao03]), as it does not provide an efficient *test* which can determine whether a given distinguisher $\mathcal{D}$ breaks the assumption.

2.5 The Gentry-Sahai-Waters Scheme

We recall the syntax and some properties of interest of the (levelled) homomorphic encryption from [GSW13]. The scheme consists of the following algorithms.

- $\mathsf{KeyGen}(1^\lambda, 1^d)$: On input the security parameter 1^λ and the depth parameter 1^d, the key generation algorithm returns a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk}, m)$: On input a public key pk and a message m, the encryption algorithm returns a ciphertext c.
- $\mathsf{Eval}(\mathsf{pk}, C, (c_1, \ldots, c_\eta))$: On input the public key pk, an η-inputs circuit C, and a vector of ciphertexts $(c_1, \ldots, c_\eta)$, the evaluation algorithm returns an evaluated ciphertext c.
- $\mathsf{Dec}(\mathsf{sk}, c)$: On input the secret key sk and a ciphertext c, the decryption algorithm returns a message m.

We recall some useful properties of the GSW homomorphic encryption scheme, which we are going to use extensively throughout this work.

- (Evaluation Correctness) Let $c_i \in \mathsf{Enc}(\mathsf{pk}, m_i)$ and let

$$\mathbf{C} \leftarrow \mathsf{Eval}(\mathsf{pk}, C, (c_1, \ldots, c_\eta)) \in \mathbb{Z}_q^{(n+1)\times(n+1)\log(q)}$$

and let $\mathsf{sk} = (\mathbf{s}, -1)$ where $\mathbf{s} \in \mathbb{Z}_q^n$. Then it holds that

$$(\mathbf{s}^T, -1)\mathbf{C} = m \cdot (\mathbf{s}^T, -1)\mathbf{G} + \mathbf{e}^T \quad \text{where} \quad \|\mathbf{e}\|_\infty \leq \tilde{B}$$

where $\tilde{B}$ is some integer. Note that, even this bound is independent from the depth of the computation and can be achieved without loss of generality by adding one key-switching in the very end of the computation. Furthermore, observe that we can isolate the message by computing

$$(\mathbf{s}^T, -1)\mathbf{C}\mathbf{v} = (\mathbf{0}, -q/2 \cdot m) + \mathbf{e}^T\mathbf{v}$$

where $\mathbf{v} = \mathbf{G}^{-1}(\mathbf{0}, q/2)$. In a slight abuse of notation, we sometimes refer to the decryption process as a linear function in $\mathbf{s}$, whose coefficient can be computed from the ciphertext, followed by a rounding.

- (Pseudorandomness) We require that for all messages m and any polynomial $d = d(\lambda)$, the following distributions are indistinguishable:

$$\begin{aligned}\left\{\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m) : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^d)\right\} &\approx_c \left\{\mathbf{u}, \mathsf{Enc}(\mathsf{pk}, m) : \mathbf{u} \leftarrow \{0,1\}^{|\mathsf{pk}|}\right\} \\ &=_s \left\{\mathbf{u} : \mathbf{u} \leftarrow \{0,1\}^{|\mathsf{pk}|+|c|}\right\}.\end{aligned}$$

In other words, public keys and ciphertexts are computationally indistinguishable from random bits. This is shown assuming the hardness of the LWE problem and an application of the leftover-hash Lemma [HILL99, Reg05].

2.6 Pseudorandom Functions and Generators

We briefly recall the standard notion of a pseudorandom function (PRF), which is a function

$$\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\eta \to \{0,1\}^\ell$$

whose output is computationally indistinguishable from uniform, for a randomly sampled key k, for an algorithm having oracle access to $\mathsf{PRF}(K, \cdot)$. In this work, we are going to use a slightly different security notion, where the distinguisher is directly given the entire truth-table of the function. In other words, we require that the following distributions are computationally indistinguishable:

$$\{\mathsf{PRF}(K, x)\}_{x\in\{0,1\}^\eta} \approx_c \left\{\mathbf{u}_x : \mathbf{u}_x \leftarrow \{0,1\}^\lambda\right\}_{x\in\{0,1\}^\eta}$$

for a uniformly sampled $K \leftarrow \{0,1\}^\lambda$. Note that this notion is equivalent to the standard security for a PRF whenever the distinguisher is allowed to run in time 2^η, which is the setting that we consider in this work, since it can query the function on all points.

We also consider the standard notion of a pseudorandom generator $\mathrm{PRG} : \{0,1\}^\lambda \to \{0,1\}^*$, whose output is computationally indistinguishable from uniform, for a randomly sampled seed. It is well known that both PRFs and PRGs can be constructed from any one-way function [GGM84, HILL99].

3 Pseudorandom Obfuscation

In the following we present formal definitions for the notion of pseudorandom obfuscation (PRO).

Definition 3.1 (Pseudorandom Obfuscation)**.** A pseudorandom obfuscation (PRO) scheme for a family of keyed circuits $C : \{0,1\}^{\mathfrak{h}} \times \{0,1\}^{\eta} \to \{0,1\}^{\ell}$ consists of two PPT algorithms $(\mathsf{PRO.Obf}, \mathsf{PRO.Eval})$ with the following syntax.

- $\mathsf{PRO.Obf}(1^{\lambda}, C, K)$: On input the security parameter 1^{λ}, a circuit C, and a key $K \in \{0,1\}^{\mathfrak{h}}$, the probabilistic obfuscation algorithm returns an obfuscated circuit $\tilde{C}$.
- $\mathsf{PRO.Eval}(C, \tilde{C}, x)$: On input the circuit C, an obfuscated circuit $\tilde{C}$, and an input x, the deterministic evaluation algorithm returns an output y.

We require the following correctness property to hold.

- (ε-Correctness) For all $\lambda \in \mathbb{N}$, all $K \in \{0,1\}^{\mathfrak{h}}$, and all circuits C it holds that

$$\Pr_{\tilde{C} \leftarrow \mathsf{PRO.Obf}(1^{\lambda}, C, K)} \Big[\forall\, x \in \{0,1\}^{\eta} : C(K,x) = \mathsf{PRO.Eval}(C, \tilde{C}, x) \Big] \geq 1 - \varepsilon$$

  where the probability is taken over the random coins of $\mathsf{PRO.Obf}$.

We define three variants of security for pseudorandom obfuscation.

- Indistinguishable Pseudorandomness (iPRO): Let $\mathsf{KeySamp}$ be a sampling algorithm. If

$$\{C(K,x)\}_{x \in \{0,1\}^{\eta}}, \mathsf{aux}_K \approx_c \left\{ u_x : u_x \leftarrow \{0,1\}^{\ell} \right\}_{x \in \{0,1\}^{\eta}}, \mathsf{aux}_K$$

  for a randomly sampled $(K, \mathsf{aux}_K) \leftarrow \mathsf{KeySamp}(1^{\lambda})$, then we have that

$$\mathsf{PRO.Obf}(1^{\lambda}, C, K), \mathsf{aux}_K \approx_c \mathsf{PRO.Obf}(1^{\lambda}, C, K'), \mathsf{aux}_K$$

  where $(K, \mathsf{aux}_K) \leftarrow \mathsf{KeySamp}(1^{\lambda})$ and K' is such that $C(K, \cdot)$ and $C(K', \cdot)$ are functionally equivalent.

- Pseudorandomness (PRO): Let $\mathsf{KeySamp}$ be a sampling algorithm. If

$$\{C(K,x)\}_{x \in \{0,1\}^{\eta}}, \mathsf{aux}_K \approx_c \left\{ u_x : u_x \leftarrow \{0,1\}^{\ell} \right\}_{x \in \{0,1\}^{\eta}}, \mathsf{aux}_K$$

  for a randomly sampled $(K, \mathsf{aux}_K) \leftarrow \mathsf{KeySamp}(1^{\lambda})$, then we have that

$$\mathsf{PRO.Obf}(1^{\lambda}, C, K), \mathsf{aux}_K \approx_c \mathsf{PRO.Obf}(1^{\lambda}, C, K'), \mathsf{aux}_K$$

  where both $(K, \mathsf{aux}_K) \leftarrow \mathsf{KeySamp}(1^{\lambda})$ and $(K', \mathsf{aux}_{K'}) \leftarrow \mathsf{KeySamp}(1^{\lambda})$ are randomly sampled.

- Double Pseudorandomness (dPRO): Let $\mathsf{KeySamp}$ be a sampling algorithm. If

$$\{C(K,x)\}_{x \in \{0,1\}^{\eta}}, \mathsf{aux}_K \approx_c \left\{ u_x : u_x \leftarrow \{0,1\}^{\ell} \right\}_{x \in \{0,1\}^{\eta}}, \mathsf{aux}_K$$

  for a randomly sampled $(K, \mathsf{aux}_K) \leftarrow \mathsf{KeySamp}(1^{\lambda})$, then we have that

$$\mathsf{PRO.Obf}(1^{\lambda}, C, K), \mathsf{aux}_K \approx_c \tilde{u}, \mathsf{aux}_K$$

  where both $(K, \mathsf{aux}_K) \leftarrow \mathsf{KeySamp}(1^{\lambda})$ and $\tilde{u} \leftarrow \{0,1\}^{|\tilde{C}|}$ are randomly sampled.

In this work, we also consider (as an intermediate primitive) the notion of *exponentially efficient* PRO, denoted by xPRO, directly inspired by the analogous notion in indistinguishability obfuscation [LPST16]. An xPRO scheme $(\mathsf{xPRO.Obf}, \mathsf{xPRO.Eval})$ is defined identically as above, except that we allow the obfuscation algorithm to run in time polynomial in the size of the truth table (2^η) of C, and we only impose the following requirements:

- (Non-Trivial Efficiency) We require that $|\tilde{C}| \leq 2^{\eta(1-\delta)}$ for some constant $\delta > 0$.
- (Shallowness) We require that $\mathsf{depth}(\mathsf{xPRO.Obf}) = \mathsf{poly}(\lambda, |C|)$.

4 Exponentially (In)efficient Pseudorandom Obfuscation

In this section we will provide a construction of xPRO for *any* pseudorandom function that satisfies the strong notion of *double pseudorandomness*. Specifically, let $k, \kappa, \mathfrak{h} \in \mathbb{N}$ be integers. We present a construction of xPRO for *any* circuit

$$C : \{0,1\}^{\mathfrak{h}} \times \{0,1\}^{\log(\kappa)} \to \{0,1\}^k$$

with pseudorandom outputs. In other words, we require that the following distributions are computationally indistinguishable

$$\{C(K,i)\}_{i\in\{1,\ldots,\kappa\}}, \mathsf{aux}_K \approx_c \left\{\mathbf{u}_i : \mathbf{u}_i \leftarrow \{0,1\}^k\right\}_{i\in\{1,\ldots,\kappa\}}, \mathsf{aux}_K$$

over the random choice of $(K, \mathsf{aux}_K) \leftarrow \mathsf{KeySamp}(1^\lambda)$. In other words, we regard $C(K,\cdot)$ as the succinct representation of $\mathtt{TT}(C(K,\cdot))$, which is a pseudorandom binary string of length $k \cdot \kappa$.

Parameters, Ingredients, and Notation. We will first discuss the cryptographic building blocks and problem parameters needed for our construction.

- Let $q = 2^t$ be modulus, which we assume to be a power of 2.
- Let $\tau, \rho, \sigma, \sigma' > \eta_{2^{-\lambda}}(\mathbb{Z})$ be parameters for discrete Gaussians.
- Let $n \in \mathbb{N}$ be an LWE dimension and $m = \Omega(n \cdot \log(q))$.
- We define the *most-significant bit* operator $\mathsf{MSB} : \mathbb{Z}_q \to \{0,1\}$ as

$$\mathsf{MSB}(x) = \begin{cases} 0 & \text{if } x \in [-q/4, q/4) \\ 1 & \text{otherwise} \end{cases}$$

  and in a slight abuse of notation, we also use the same operator on vectors to perform the same operation component-wise.
- Let $\hat{B} > 0$ and let PRF be a pseudorandom function which outputs values in $[-q/4 + \hat{B}, q/4 - \hat{B})^k$.

- For a keyed circuit $C : \{0,1\}^{\mathfrak{h}} \times \{0,1\}^{\log(\kappa)} \to \{0,1\}^k$ we will define a circuit $\bar{C}$ as follows:

$$\bar{C}((K, K'), i) = \frac{q}{2} \cdot C(K, i) + \mathsf{PRF}(K', i) \in \mathbb{Z}_q^k.$$

  Note that it holds for all $i \in \{1, \ldots, \kappa\}$ and every $\hat{\boldsymbol{e}}_i \in \mathbb{Z}^k$ with $\|\hat{\boldsymbol{e}}_i\|_\infty \leq \hat{B}$ that

$$\mathsf{MSB}(\bar{C}((K, K'), i) + \hat{\boldsymbol{e}}_i) = C(K, i).$$

- Let $\mathsf{GSW} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be the d-leveled GSW encryption scheme, where we assume that KeyGen takes the modulus q and the parameter d as explicit inputs. Furthermore, we assume that GSW has a decryption error bounded by a parameter $\tilde{B}$ (which we assume to be independent of q and d), i.e. it holds for all ciphertexts c (obtained via encryption or homomorphic evaluation) encrypting a value m that the noisy decryption $\mathsf{GSW.Dec}$ of c returns

$$\mathsf{GSW.Dec}(\mathsf{sk}, c) = m + \tilde{e},$$

  where $|\tilde{e}| \leq \tilde{B}$. Further recall that for a given ciphertext c noisy decryption is a linear function of the secret key $\mathsf{sk} \in \mathbb{Z}_q^n$, i.e. for every c in the ciphertext space there exists a vector $\boldsymbol{f}$ in $\mathbb{Z}_q^n$ such that

$$\mathsf{GSW.Dec}(\mathsf{sk}, c) = \mathsf{sk}^\top \cdot \boldsymbol{f}.$$

  We call $\boldsymbol{f}$ the *vectorization* of $\mathsf{GSW.Dec}(\cdot, c)$.

- Let $\ell = \mathsf{poly}(\lambda)$ and let $(\mathcal{S}_{k\times m,\tau}, \mathcal{S}^{-1}_{k\times m,\tau})$ be a reversible sampler such that $\mathcal{S}_{k\times m,\tau}(r)$ takes uniformly random coins $r \in \{0,1\}^{\ell \cdot k \cdot m}$ and produces samples (statistically close to) $(D_{\mathbb{Z},\tau})^{k\times m}$

Constraints. Before providing our construction, we briefly state the constraints that the parameters we introduced above will need to satisfy. These constraints will arise due to efficiency (Lemma 4.1 below), correctness (Lemma 4.2 below) and pseudorandomness (Theorem 4.3 below).

1. We require that $\log(q), n, m$ depend at most poly-logarithmically on k and κ (Lemma 4.1).
2. We need to choose $m = O(n \log(q))$ to enable lattice trapdoor sampling (Theorem 2.6)
3. The modulus q needs to satisfy $q > 4\kappa k \hat{B}$, where $\hat{B} := \tilde{B} + \lambda m(k+1)\rho + \lambda m \tau \sigma$ (Lemma 4.2).
4. The Gaussian parameter σ needs to be large enough such that $D_{\mathbb{Z},\sigma}$ *drowns* a value of absolute value $\tilde{B} + \lambda(k+1)m\rho$ (Theorem 4.3). Via Lemma 2.4 this can be achieved by choosing $\sigma \geq 2^\lambda(\tilde{B} + \lambda(k+1)m\rho)$.
5. We require $\bar{C}$ to produce pseudorandom outputs over $\mathbb{Z}_q$. This is the case if the uniform-distribution on $[-q/4 + \hat{B}, q/4 - \hat{B})^k$ is statistically close to the uniform distribution on $[-q/4, q/4)^k$, which in turn is the case if $\hat{B}/q \leq 2^{-\lambda}$.

   Then a simple hybrid argument shows that if C is a pseudorandom function with respect to some auxiliary input aux_K, then so is $\bar{C}$, i.e.

$$\left\{\bar{C}((K, K'), i)\right\}_{i\in\{1,\ldots,\kappa\}}, \mathsf{aux}_K \approx_c \{u_i\}_{i\in\{1,\ldots,\kappa\}}, \mathsf{aux}_K,$$

   for $(K, \mathsf{aux}_K) \leftarrow \mathsf{KeySamp}(1^\lambda)$, $K' \leftarrow \{0,1\}^\lambda$ and $u_1, \ldots, u_\kappa$ uniform in $\mathbb{Z}_q^k$.

$$\sigma = 2^\lambda(\tilde{B} + \lambda(k+1)m\rho)$$
$$q = 2^\lambda \kappa k \hat{B} = 2^\lambda \kappa k(\tilde{B} + \lambda m(k+1)\rho + \lambda m\tau\sigma) = O(2^{2\lambda}\kappa k\lambda m\tau(\tilde{B} + \lambda(k+1)m\rho))$$
$$m = O(n\log(q)).$$

Finally, by choosing $n = \mathsf{poly}(\lambda)$ for a sufficiently large polynomial (e.g. $n = \lambda^3$), we will be able to base the security of our scheme on LWE and evasive LWE with sub-exponential modulus-to-noise ratio (Theorem 4.3).

4.1 Construction

We present our construction of xPRO in the following. Note that the evaluation xPRO.Eval algorithm does not input any point, and instead it directly reconstructs the entire function table. This is done for notational convenience, and it is anyway how we are going to use the algorithm later on.

- $\mathsf{xPRO.Obf}(1^\lambda, C, K)$:
  - Sample a key $K' \leftarrow \{0,1\}^\lambda$ uniformly at random
  - Set $\bar{K} = (K, K')$.
  - Sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, q, 1^d)$, where d is the circuit depth of C, and compute $c \leftarrow \mathsf{Enc}(\mathsf{pk}, \bar{K})$.
  - Choose $\mathbf{B} \leftarrow \mathbb{Z}_q^{m\times n}$ and $r \leftarrow \{0,1\}^{k\cdot m\cdot \ell}$ uniformly at random.
  - Sample $\mathbf{D} = \mathcal{S}_{k\times m,\tau}(r)$ using random coins r and set $\mathbf{P} = \mathbf{D}\cdot\mathbf{B} \in \mathbb{Z}_q^{k\times n}$.
  - Parse $\mathsf{sk}^\top = (s_1, \ldots, s_n) \in \mathbb{Z}_q^n$, sample $\mathbf{R} \leftarrow \mathbb{Z}_q^{n\times(k+1)\cdot m}$ and $\bar{\mathbf{E}} \leftarrow D_{\mathbb{Z},\rho}^{k\times(k+1)\cdot m}$ and compute
  $$\mathbf{A} = \mathbf{P}\cdot\mathbf{R} + \bar{\mathbf{E}} + (\mathbf{I}_k, \mathbf{0})\otimes(s_1,\ldots,s_n)\otimes \boldsymbol{g}^\top \in \mathbb{Z}_q^{k\times(k+1)m},$$
  where $\mathbf{0} \in \mathbb{Z}_q^k$ is the all-zero column vector.
  - For $i \in \{1,\ldots,\kappa\}$:
    - Compute $c_i \leftarrow \mathsf{Eval}(\mathsf{pk}, \bar{C}(\cdot, i), c)$
    - Let $\mathbf{f}_i \in \mathbb{Z}_q^{k\cdot n}$ be the vectorization of the linear function $\mathsf{Dec}(\cdot, c_i)$.
    - Choose $\boldsymbol{r}_i \leftarrow \{0,1\}^m$ uniformly at random.
  - Compute
  $$\mathbf{F} = (\mathbf{G}_{\boldsymbol{r}_1}^{-1}(\boldsymbol{f}_1), \ldots, \mathbf{G}_{\boldsymbol{r}_\kappa}^{-1}(\boldsymbol{f}_\kappa)) \in \{0,1\}^{(k+1)m\times\kappa}.$$
  - Choose $\mathbf{S} \leftarrow \mathbb{Z}_q^{n\times\kappa}$ uniformly at random and set $\mathbf{H} = \mathbf{S} + \mathbf{R}\cdot\mathbf{F} \in \mathbb{Z}_q^{n\times\kappa}$.
  - Choose $\mathbf{E} \leftarrow D_{\mathbb{Z},\sigma}^{m\times\kappa}$ and compute $\mathbf{C} = \mathbf{B}\cdot\mathbf{S} + \mathbf{E} \in \mathbb{Z}_q^{m\times\kappa}$.
  - Output $\tilde{K} = (\mathbf{C}, r, \mathbf{P}, \mathbf{A}, \mathsf{pk}, c, \{\boldsymbol{r}_i\}_{i\in\{1,\ldots,k\}}, \mathbf{H})$.
- $\mathsf{xPRO.Eval}(C, \tilde{K})$:
  - Parse $\tilde{K} = (\mathbf{C}, r, \mathbf{P}, \mathbf{A}, \mathsf{pk}, c, \{\boldsymbol{r}_i\}_{i\in\{1,\ldots,k\}}, \mathbf{H})$.

- Recompute $\mathbf{D} = \mathcal{S}_{k\times m}(r)$ using random coins r.
- For $i \in \{1, \ldots, \kappa\}$:
  - Compute $c_i \leftarrow \mathsf{Eval}(\mathsf{pk}, \bar{C}(\cdot, i), c)$
  - Let $\mathbf{f}_i \in \mathbb{Z}_q^{k\cdot n}$ be the vectorization of the linear function $\mathsf{Dec}(\cdot, c_i)$.
- Recompute
$$\mathbf{F} = (\mathbf{G}_{\boldsymbol{r}_1}^{-1}(\boldsymbol{f}_1), \ldots, \mathbf{G}_{\boldsymbol{r}_\kappa}^{-1}(\boldsymbol{f}_\kappa)) \in \{0,1\}^{(k+1)m\times\kappa}.$$
- Compute $\mathbf{Y} = \mathbf{A}\cdot\mathbf{F} + \mathbf{D}\cdot\mathbf{C} - \mathbf{P}\cdot\mathbf{H} \in \mathbb{Z}_q^{k\times\kappa}$ and parse $\mathbf{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\kappa)$ column-wise.
- Output $\mathrm{MSB}(\boldsymbol{y}_1), \ldots, \mathrm{MSB}(\boldsymbol{y}_\kappa)$.

4.2 Efficiency and Compactness

We discuss the efficiency of the scheme xPRO. Clearly, both xPRO.Obf and xPRO.Eval run in time polynomial in k, κ as well as $n, m, \log(q), d, \ell = \mathsf{poly}(\lambda)$. Note further that xPRO.Obf is shallow: Computing the c_i can be achieved in depth $\mathsf{depth}(\bar{C}\cdot\mathsf{poly}(\lambda)$, sampling of the matrices can be performed component-wise, and matrix-operations can be performed in poly-logarithmic depth.

In terms of compactness, we will now analyze the size of

$$\tilde{K} = (\mathbf{C}, r, \mathbf{P}, \mathbf{A}, \mathsf{pk}, c, \{\boldsymbol{r}_i\}_{i\in\{1,\ldots,k\}}, \mathbf{H})$$

and compare it to the bit-size of the function table of C. In that regard, note first that the size of $\mathtt{TT}(C(K,\cdot))$ is $\kappa\cdot k$ bits.

- The matrix $\mathbf{C} \in \mathbb{Z}_q^{m\times\kappa}$ has a bit-size of $\kappa\cdot m\cdot\log(q) = \kappa\cdot\mathsf{poly}(\lambda)$.
- The random coins $r \in \{0,1\}^{k\cdot m\cdot\ell}$ have a bit-size of $k\cdot m\cdot\ell = k\cdot\mathsf{poly}(\lambda)$.
- The matrix $\mathbf{P} \in \mathbb{Z}_q^{k\times n}$ has a bit-size of $k\cdot n\cdot\log(q) = k\cdot\mathsf{poly}(\lambda)$.
- The matrix $\mathbf{A} \in \mathbb{Z}_q^{k\times(k+1)\cdot m}$ has a bit-size of $k\cdot(k+1)\cdot m\log(q) = k^2\cdot\mathsf{poly}(\lambda)$.
- The public key pk has a bit-size $d\cdot\mathsf{poly}(\lambda) = \mathsf{poly}(\lambda)$.
- The ciphertext has a bit-size $\mathfrak{h}\cdot\mathsf{poly}(\lambda)$.[7]
- The list $\{\boldsymbol{r}_i\}_{i\in\{1,\ldots,k\}}$ has a bit-size of $k\cdot m = k\cdot\mathsf{poly}(\lambda)$.
- The matrix $\mathbf{H} \in \mathbb{Z}_q^{n\times\kappa}$ has a bit-size of $\kappa\cdot n\cdot\log(q) = \kappa\cdot\mathsf{poly}(\lambda)$.

Thus, taken together $\tilde{K}$ has a bit-size of $(k^2 + \kappa + |K|)\cdot\mathsf{poly}(\lambda)$. Setting $\kappa = O(k^2)$ and assuming that $|K| = k^2\cdot\mathsf{poly}(\lambda)$ we get

$$|\tilde{K}| = k^2\cdot\mathsf{poly}(\lambda)$$

versus a truth table size of

$$|\mathtt{TT}(C(K,\cdot))| = k^3.$$

Hence, it holds that

$$|\tilde{K}| = |\mathtt{TT}(C(K,\cdot))|^{2/3}\cdot\mathsf{poly}(\lambda),$$

i.e., we achieve non-trivial efficiency. We summarize this in the following Lemma.

[7] This can easily be improved to $\mathfrak{h} + \mathsf{poly}(\lambda)$ via hybrid encryption.

Lemma 4.1. *Given that $\log(q), n, m, \ell$ depend at most poly-logarithmically on k and κ, the scheme* xPRO *defined above satisfies the non-trivial efficiency requirement.*

4.3 Correctness

We now establish correctness of the scheme xPRO.

Lemma 4.2. *Assume that the GSW encryption scheme has decryption noise bounded by $\tilde{B}$ and that $q > 4\kappa k(\tilde{B} + \lambda(k+1)m\rho + \lambda m\tau\sigma)$. Then the scheme* xPRO *has correctness error at most $\tilde{O}(2^{-\lambda})$.*

Proof. For all $i \in \{1, \ldots, k\}$ it holds that

$$\begin{aligned}
&\mathbf{A} \cdot \mathbf{G}_{\boldsymbol{r}_i}^{-1}(\boldsymbol{f}_i) \\
&\quad = \mathbf{P} \cdot \mathbf{R} \cdot \mathbf{G}_{\boldsymbol{r}_i}^{-1}(\boldsymbol{f}_i) + \bar{\mathbf{E}} \cdot \mathbf{G}_{\boldsymbol{r}_i}^{-1}(\boldsymbol{f}_i) + ((\mathbf{I}_k, \mathbf{0}) \otimes (s_1, \ldots, s_n) \otimes \boldsymbol{g}^\top) \cdot \mathbf{G}_{\boldsymbol{r}_i}^{-1}(\boldsymbol{f}_i) \\
&\quad = \mathbf{P} \cdot \mathbf{R} \cdot \mathbf{G}_{\boldsymbol{r}_i}^{-1}(\boldsymbol{f}_i) + \bar{\mathbf{E}} \cdot \mathbf{G}_{\boldsymbol{r}_i}^{-1}(\boldsymbol{f}_i) + \mathsf{Dec}(\mathsf{sk}, c_i) \\
&\quad = \mathbf{P} \cdot \mathbf{R} \cdot \mathbf{G}_{\boldsymbol{r}_i}^{-1}(\boldsymbol{f}_i) + \bar{\mathbf{E}} \cdot \mathbf{G}_{\boldsymbol{r}_i}^{-1}(\boldsymbol{f}_i) + \mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, \bar{C}(\cdot, i), c)) \\
&\quad = \bar{C}(K, i) + \tilde{\boldsymbol{e}}_i + \mathbf{P} \cdot \mathbf{R} \cdot \mathbf{G}_{\boldsymbol{r}_i}^{-1}(\boldsymbol{f}_i) + \bar{\mathbf{E}} \cdot \mathbf{G}_{\boldsymbol{r}_i}^{-1}(\boldsymbol{f}_i),
\end{aligned}$$

where the second equality holds via the definition of $\boldsymbol{f}_i$, and the third equality holds by the the definition of the c_i, and the fourth equality holds by the correctness of FHE, where the $\tilde{\boldsymbol{e}}_i$ are decryption noise terms. Hence it holds that

$$\mathbf{A} \cdot \mathbf{F} = (\bar{C}(\bar{K}, 1), \ldots, \bar{C}(\bar{K}, \kappa)) + \tilde{\mathbf{E}} + \mathbf{P} \cdot \mathbf{R} \cdot \mathbf{F} + \bar{\mathbf{E}} \cdot \mathbf{F},$$

where $\tilde{\mathbf{E}} = (\tilde{\boldsymbol{e}}_1, \ldots, \tilde{\boldsymbol{e}}_\kappa)$. Further note that

$$\mathbf{D} \cdot \mathbf{C} = \mathbf{D} \cdot \mathbf{B} \cdot \mathbf{S} + \mathbf{D} \cdot \mathbf{E} = \mathbf{P} \cdot \mathbf{S} + \mathbf{D} \cdot \mathbf{E}$$

and by definition of $\mathbf{H}$:

$$\mathbf{PH} = \mathbf{P} \cdot \mathbf{S} + \mathbf{P} \cdot \mathbf{R} \cdot \mathbf{F}.$$

Hence it holds that

$$\begin{aligned}
\mathbf{Y} &= \mathbf{A} \cdot \mathbf{F} + \mathbf{D} \cdot \mathbf{C} - \mathbf{PH} \\
&= (\bar{C}(\bar{K}, 1), \ldots, \bar{C}(\bar{K}, \kappa)) + \tilde{\mathbf{E}} + \bar{\mathbf{E}}\mathbf{F} + \mathbf{DE}.
\end{aligned}$$

Let $\hat{\mathbf{E}} = (\hat{\boldsymbol{e}}_1, \ldots, \hat{\boldsymbol{e}}_\kappa) := \tilde{\mathbf{E}} + \bar{\mathbf{E}}\mathbf{F} + \mathbf{DE}$. Since $\bar{\mathbf{E}} \sim D_{\mathbb{Z},\rho}^{k \times (k+1)m}$, it holds for each column $\bar{\boldsymbol{e}}_i$ of $\bar{\mathbf{E}}$ that $\|\boldsymbol{e}_i\|_\infty \leq \sqrt{\lambda} \cdot \rho \leq \log(1/\varepsilon) \cdot \rho$, except with probability $k \cdot (k+1)m \cdot \tilde{O}(2^{-\lambda}) = \tilde{O}(2^{-\lambda})$. Hence it holds for all i that $\|\bar{\mathbf{E}} \cdot \boldsymbol{f}_i\|_\infty \leq \sqrt{\lambda} \cdot (k+1)m\rho$. Furthermore, it holds for every column $\boldsymbol{e}_i$ of $\mathbf{E}$ that $\|\boldsymbol{e}_i\|_\infty \leq \sqrt{\lambda} \cdot \sigma$ and for every row $\boldsymbol{d}_j$ of $\boldsymbol{D}$ that $\|\boldsymbol{d}_j\|_\infty \leq \sqrt{\lambda}\tau$, except with probability $\tilde{O}(2^{-\lambda})$. Hence it holds that $\|\mathbf{D} \cdot \boldsymbol{e}_i\|_\infty \leq \lambda \cdot m\tau\sigma$, except with probability $\tilde{O}(2^{-\lambda})$. We conclude that the norm of each column $\hat{\boldsymbol{e}}_i$ of the error matrix $\hat{\mathbf{E}}$ can be bounded via

$$\hat{B} := \tilde{B} + \lambda m(k+1)\rho + \lambda m\tau\sigma \geq \|\hat{\boldsymbol{e}}_i\|_\infty.$$

By the definition of $\bar{C}$ it holds for all i that $\bar{C}(\bar{K}, i) = q/2 \cdot \mathsf{PRF}(K, i) + \mathsf{PRF}(K', i)$. Recalling that $\mathbf{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\kappa)$, it holds for all $i \in \{1, \ldots, \kappa\}$ that

$$\boldsymbol{y}_i = q/2 \cdot C(K, i) + \mathsf{PRF}(K', i) + \hat{\boldsymbol{e}}_i.$$

Now observe that by definition of PRF it holds for every $i \in \{1, \ldots, \kappa\}$ that $\mathsf{PRF}(K', i) \in [-q/4 + \hat{B}, q/4 - \hat{B})^k$. Hence it holds that $\mathsf{PRF}(K', i) + \hat{\boldsymbol{e}}_i \in [-q/4, q/4)^k$ as $\hat{\boldsymbol{e}}_i \in [-\hat{B}, \hat{B})^k$. We can conclude that

$$\mathrm{MSB}(\boldsymbol{y}_i) = \mathrm{MSB}(q/2 \cdot C(K, i) + \mathsf{PRF}(K', i) + \hat{\boldsymbol{e}}_i) = C(K, i).$$

We can conclude that xPRO is correct, except with probability $\tilde{O}(2^{-\lambda})$. □

4.4 Pseudorandomness

We now show that xPRO satisfies the strongest pseudorandomness property.

Theorem 4.3. *Assume that $\bar{C}$ is a pseudorandom function, that* GSW *has pseudorandom keys and pseudorandom ciphertexts. Assume further that the assumptions $(\mathcal{S}_{k\times m}, \mathcal{S}^{-1}_{k\times m})$ is a reversible sampler for $D_{\mathbb{Z}^{k\times m},\tau}$ (as by Lemma 2.7). Also assume that $\sigma \geq 2^{\lambda} \cdot (\tilde{B} + \lambda(k+1)m\rho)$. Assume further that $\mathsf{LWE}(q, n, \rho)$ and $\mathsf{evLWE}(q, m, n, k, \kappa, \mathrm{Samp}, \tau, \sigma, \sigma')$ hold, where the sampler Samp is given in Algorithm 1. Then* xPRO *is a doubly pseudorandom obfuscator.*

Proof. Our proof will proceed in a sequence of hybrids. Consider the following hybrid distributions.

- Hybrid $\mathcal{H}_0$: This is identical to the real distribution.
- Hybrid $\mathcal{H}_1$: Identical to $\mathcal{H}_0$, except that we choose $\mathbf{C}$ uniformly at random. We will discuss this step in more detail below.
- Hybrid $\mathcal{H}_2$: Identical to $\mathcal{H}_1$, except that we choose the matrix $\mathbf{H} \leftarrow \mathbb{Z}_q^{n\times\kappa}$ uniformly at random. Note that in $\mathcal{H}_1$ the matrix $\mathbf{H}$ is computed via $\mathbf{H} = \mathbf{S} + \mathbf{R} \cdot \mathbf{F}$ for a uniformly random $\mathbf{S}$ (which is not used anywhere else). Hence $\mathcal{H}_1$ and $\mathcal{H}_2$ are identically distributed. Observe that after this modification the matrix $\mathbf{R}$ is only used for the computation of $\mathbf{A} = \mathbf{PR} + \bar{\mathbf{E}} + (\mathbf{I}_k, \mathbf{0}) \otimes (s_1, \ldots, s_n) \otimes \boldsymbol{g}^{\top}$.
- Hybrid $\mathcal{H}_3$: Identical to $\mathcal{H}_2$, except that we change the way $\mathbf{B}$ is computed. Instead of choosing $\mathbf{B}$ uniformly at random we compute $(\mathbf{B}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^{\lambda}, q, n)$. Statistical indistinguishability of hybrids $\mathcal{H}_2$ and $\mathcal{H}_3$ follows from the uniformity property of TrapGen (Theorem 2.6).
- Hybrid $\mathcal{H}_4$: Identical to hybrid $\mathcal{H}_3$, except that we change the way the coins r are generated. Instead of choosing r uniformly at random, we sample $\mathbf{D} \leftarrow D_{\mathbb{Z}^{k\times m},\tau}$ and set $r = \mathcal{S}^{-1}_{k\times m}(\mathbf{D})$. Statistical indistinguishability follows from the fact that $(\mathcal{S}_{k\times m}, \mathcal{S}^{-1}_{k\times m})$ is a reversible sampler for $D_{\mathbb{Z}^{k\times m},\tau}$.
- Hybrid $\mathcal{H}_5$: Identical to hybrid $\mathcal{H}_4$, except that we change the way $\mathbf{D}$ and $\mathbf{P}$ are computed. Instead of choosing $\mathbf{D} \leftarrow D_{\mathbb{Z}^{k\times m},\tau}$ and setting $\mathbf{P} = \mathbf{D} \cdot \mathbf{B}$, we choose $\mathbf{P} \leftarrow \mathbb{Z}_q^{k\times n}$ uniformly at random and set $\mathbf{D} = \mathsf{SampPre}(\mathsf{td}, \mathbf{P}, \tau)$. Statistical indistinguishability follows from the trapdoor sampling property of $(\mathsf{TrapGen}, \mathsf{SampPre})$ (Theorem 2.6).
- Hybrid $\mathcal{H}_6$: Identical to $\mathcal{H}_5$, except that instead of computing $\mathbf{A} = \mathbf{PR} + \bar{\mathbf{E}} + (\mathbf{I}_k, \mathbf{0}) \otimes (s_1, \ldots, s_n) \otimes \boldsymbol{g}^{\top}$, we choose the matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{k\times(k+1)\cdot m}$ uniformly at random. Computational indistinguishability follows routinely from $\mathsf{LWE}(q, n, \rho)$.

Algorithm 1 The Sampler Samp.

$\mathrm{Samp}(1^\lambda, \mathbf{P})$:

- Choose $(K, \mathsf{aux}_K) \leftarrow \mathsf{KeySamp}(1^\lambda)$.
- Choose $K' \leftarrow \{0,1\}^\lambda$ uniformly at random.
- Set $\bar{K} = (K, K')$.
- Sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^d)$.
- Compute $c \leftarrow \mathsf{Enc}(\mathsf{pk}, \bar{K})$.
- Parse $\mathsf{sk} = (s_1, \ldots, s_n) \in \mathbb{Z}_q^n$, sample $\mathbf{R} \leftarrow \mathbb{Z}_q^{n \times (k+1)\cdot m}$ and $\bar{\mathbf{E}} \leftarrow D_{\mathbb{Z},\rho}^{k \times (k+1)\cdot m}$ and compute

$$\mathbf{A} = \mathbf{P}\mathbf{R} + \bar{\mathbf{E}} + (\mathbf{I}_k, \mathbf{0}) \otimes (s_1, \ldots, s_n) \otimes \boldsymbol{g}^\top.$$

- For $i \in \{1, \ldots, \kappa\}$:
  - Compute $c_i \leftarrow \mathsf{Eval}(\mathsf{pk}, \bar{C}(\cdot, i), c)$.
  - Let $\mathbf{f}_i \in \mathbb{Z}_q^{k \cdot n}$ be the vectorization of the linear function $\mathsf{Dec}(\cdot, c_i)$.
  - Choose $\boldsymbol{r}_i \leftarrow \{0,1\}^m$ uniformly at random.
- Set $\mathbf{F} = (\mathbf{G}_{\boldsymbol{r}_1}^{-1}(\boldsymbol{f}_1), \ldots, \mathbf{G}_{\boldsymbol{r}_\kappa}^{-1}(\boldsymbol{f}_\kappa))$.
- Choose $\mathbf{S} \leftarrow \mathbb{Z}_q^{n \times \kappa}$ uniformly at random.
- Set $\mathbf{H} = \mathbf{S} + \mathbf{R} \cdot \mathbf{F}$.
- Set $\mathrm{aux} = (\mathbf{P}, \mathsf{pk}, c, \mathbf{A}, (\boldsymbol{r}_i)_{i \in \{1,\ldots,k\}}, \mathbf{H}, \mathsf{aux}_K)$.
- Output $\mathbf{S}, \mathrm{aux}$.

- Hybrid $\mathcal{H}_7$: Identical to $\mathcal{H}_6$, except that we undo the modification done in $\mathcal{H}_5$, i.e. we choose $\mathbf{D} \leftarrow D_{\mathbb{Z}^{k\times m},\tau}$ and set $\mathbf{P} = \mathbf{D} \cdot \mathbf{B}$. Statistical indistinguishability follows from the trapdoor sampling property of $(\mathsf{TrapGen}, \mathsf{SampPre})$ (Theorem 2.6).
- Hybrid $\mathcal{H}_8$: Identical to $\mathcal{H}_7$, except that we undo the modification of $\mathcal{H}_4$, i.e. we choose the coins r uniformly at random and set $\mathbf{D} = \mathcal{S}_{k\times m}(r)$. As before, statistical indistinguishability follows from the fact that $(\mathcal{S}_{k\times m}, \mathcal{S}_{k\times m}^{-1})$ is a reversible sampler for $D_{\mathbb{Z}^{k\times m},\tau}$.
- Hybrid $\mathcal{H}_9$: Identical to hybrid $\mathcal{H}_8$, except that we undo the change of $\mathcal{H}_3$, i.e. instead of computing $\mathbf{B}$ via $(\mathbf{B}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^\lambda, q, n)$, we choose $\mathbf{B} \leftarrow \mathbb{Z}_q^{m\times n}$ uniformly at random. Statistical indistinguishability follows from the uniformity property of $\mathsf{TrapGen}$ (Theorem 2.6).
- Hybrid $\mathcal{H}_{10}$: Identical to hybrid $\mathcal{H}_9$, except that we choose the ciphertext c uniformly at random. Indistinguishability follows from the ciphertext pseudorandomness of GSW.

- Hybrid $\mathcal{H}_{11}$: Identical to hybrid $\mathcal{H}_{10}$, except that we choose the public key pk uniformly at random. Indistinguishability follows from the public key pseudorandomness of GSW.

Note that in hybrid $\mathcal{H}_{11}$ the obfuscation $\tilde{K}$ is chosen uniformly random, i.e., $\mathcal{H}_{11}$ is the ideal distribution.

The heavy lifting of this proof occurs in the transition from $\mathcal{H}_0$ to $\mathcal{H}_1$. We will now focus on this step, which relies on the evasive LWE assumption $\mathsf{evLWE}(q, m, n, k, \kappa, \mathrm{Samp}, \tau, \sigma, \sigma')$. Recall the sampler Samp given in Algorithm 1.

The sampler Samp generates an LWE secret $\mathbf{S}$, an LWE matrix $\mathbf{P}$, as well as auxiliary information $\mathrm{aux} = (\mathbf{P}, \mathsf{pk}, c, \mathbf{A}, (\boldsymbol{r}_i)_{i\in\{1,\dots,k\}}, \mathbf{H}, \mathsf{aux}_K)$.

Now assume towards contradiction that there is a PPT distinguisher $\mathcal{D}$ which distinguishes $\mathcal{H}_0$ and $\mathcal{H}_1$ with non-negligible advantage ε. We will use $\mathcal{D}$ to construct a distinguisher $\mathcal{D}'$ which distinguishes $(\mathbf{B}, \mathbf{P}, \mathbf{BS} + \mathbf{E}, \mathbf{D}, \mathrm{aux})$ and $(\mathbf{B}, \mathbf{P}, \mathbf{U}, \mathbf{D}, \mathrm{aux})$ with non-negligible advantage ε. The distinguisher $\mathcal{D}'(\mathbf{B}, \mathbf{P}, \mathbf{C}, \mathbf{D}, \mathrm{aux})$ is given as follows.

- Parse $\mathrm{aux} = (\mathbf{P}, \mathsf{pk}, c, \mathbf{A}, (\boldsymbol{r}_i)_{i\in\{1,\dots,k\}}, \mathbf{H}, \mathsf{aux}_K)$
- Run $\mathcal{D}$ on input $\tilde{K} = (\mathbf{C}, r, \mathbf{P}, \mathbf{A}, \mathsf{pk}, c, (\boldsymbol{r}_i)_{i\in\{1,\dots,k\}}), \mathbf{H})$ and aux_K and output whatever $\mathcal{D}$ outputs.

We will now analyze the advantage of $\mathcal{D}'$. Let $\mathbf{P}$ be distributed uniformly random and let $(\mathbf{S}, \mathrm{aux}) \leftarrow \mathrm{Samp}(1^\lambda, \mathbf{P})$. Further let $\mathbf{B} \leftarrow \mathbb{Z}_q^{m\times n}$ be chosen uniformly random and let $\mathbf{D} \leftarrow \mathbf{B}^{-1}(\mathbf{P})$.

First assume that $\mathbf{C} = \mathbf{BS} + \mathbf{E}$, where , $\mathbf{E} \leftarrow D_{\mathbb{Z},\sigma}^{m\times km}$. It follows that the $\tilde{K}$ generated by $\mathcal{D}'$ is identically distributed to $\mathcal{H}_0$. On the other hand, if $\mathbf{C}$ is chosen uniformly random, then the $\tilde{K}$ generated by $\mathcal{D}'$ is identically distributed to $\mathcal{H}_1$. We conclude that

$$\begin{aligned}&|\Pr[\mathcal{D}'(\mathbf{B}, \mathbf{P}, \mathbf{BS} + \mathbf{E}, \mathbf{D}, \mathrm{aux}) = 1] - \Pr[\mathcal{D}'(\mathbf{B}, \mathbf{P}, \mathbf{U}, \mathbf{D}, \mathrm{aux}) = 1]|\\ &\quad = |\Pr[\mathcal{D}(\mathcal{H}_0) = 1] - \Pr[\mathcal{D}(\mathcal{H}_1) = 1]| \geq \varepsilon.\end{aligned}$$

In other words $\mathcal{D}'$ breaks the post-condition of evasive LWE with advantage ε. It remains to show that the pre-condition of evasive LWE holds. This will immediately lead to the desired contradiction. Consider the following hybrids.

- Hybrid $\mathcal{P}_0$:
  - Choose $\mathbf{P} \leftarrow \mathbb{Z}_q^{k\times n}$ uniformly at random.
  - Choose $(K, \mathsf{aux}_K) \leftarrow \mathsf{KeySamp}(1^\lambda)$.
  - Choose $K' \leftarrow \{0,1\}^\lambda$ uniformly at random.
  - Set $\bar{K} = (K, K')$.
  - Sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^d)$.
  - Compute $c \leftarrow \mathsf{Enc}(\mathsf{pk}, \bar{K})$.
  - Parse $\mathsf{sk} = (s_1, \dots, s_n) \in \mathbb{Z}_q^n$, sample $\mathbf{R} \leftarrow \mathbb{Z}_q^{n\times(k+1)\cdot m}$ and $\bar{\mathbf{E}} \leftarrow D_{\mathbb{Z},\rho}^{k\times(k+1)\cdot m}$ and compute $\mathbf{A} = \mathbf{PR} + \bar{\mathbf{E}} + (\mathbf{I}_k, \mathbf{0}) \otimes (s_1, \dots, s_n) \otimes \boldsymbol{g}^\top$.
  - For $i \in \{1, \dots, \kappa\}$:

* Compute $c_i \leftarrow \mathsf{Eval}(\mathsf{pk}, \bar{C}(\cdot, i), c)$.
* Let $\mathbf{f}_i \in \mathbb{Z}_q^{k \cdot n}$ be the vectorization of the linear function $\mathsf{Dec}(\cdot, c_i)$.
* Choose $\boldsymbol{r}_i \leftarrow \{0,1\}^m$ uniformly at random.

– Set $\mathbf{F} = (\mathbf{G}^{-1}_{\boldsymbol{r}_1}(\boldsymbol{f}_1), \ldots, \mathbf{G}^{-1}_{\boldsymbol{r}_\kappa}(\boldsymbol{f}_\kappa))$.
– Choose $\mathbf{S} \leftarrow \mathbb{Z}_q^{n \times \kappa}$ uniformly at random.
– Set $\mathbf{H} = \mathbf{S} + \mathbf{R} \cdot \mathbf{F}$.
– Set $\mathrm{aux} = (\mathbf{P}, \mathsf{pk}, c, \mathbf{A}, (\boldsymbol{r}_i)_{i \in \{1,\ldots,k\}}, \mathbf{H}, \mathsf{aux}_K)$.
– Choose $\mathbf{B} \leftarrow \mathbb{Z}_q^{m \times n}$ uniformly at random.
– Compute $\mathbf{C} = \mathbf{BS} + \mathbf{E}$ where $\mathbf{E} \leftarrow D^{m \times \kappa}_{\mathbb{Z},\sigma}$.
– Compute $\mathbf{C}' = \mathbf{PS} + \mathbf{E}'$ where $\mathbf{E}' \leftarrow D^{k \times \kappa}_{\mathbb{Z},\sigma'}$.
– Output $(\mathbf{B}, \mathbf{P}, \mathbf{C}, \mathbf{C}', \mathrm{aux})$.

This is the real distribution of the pre-condition with the algorithm $\mathrm{Samp}(1^\lambda, \mathbf{P})$ unrolled.

- Hybrid $\mathcal{P}_1$: This is identical to $\mathcal{P}_0$, except for the way we compute $\mathbf{H}$, $\mathbf{C}$ and $\mathbf{C}'$:

  – Choose $\mathbf{H}$ uniformly at random.
  – Compute $\mathbf{U} = \mathbf{BR} + \mathbf{E}^*$, where $\mathbf{E}^* \leftarrow D^{m \times (k+1)m}_{\mathbb{Z},\rho}$.
  – Compute $\mathbf{C} = \mathbf{BH} - \mathbf{UF} + \mathbf{E}^*\mathbf{F} + \mathbf{E}$ where $\mathbf{E} \leftarrow D^{m \times \kappa}_{\mathbb{Z},\sigma}$.
  – Compute $\mathbf{C}' = \mathbf{PH} - \mathbf{A} \cdot \mathbf{F} + (C(\bar{K}, 1), \ldots, C(\bar{K}, \kappa)) + \bar{\mathbf{E}}\mathbf{F} + \tilde{\mathbf{E}} + \mathbf{E}'$ where $\mathbf{E}' \leftarrow D^{k \times \kappa}_{\mathbb{Z},\sigma'}$ and $\tilde{\mathbf{E}}$ is such that $((\mathbf{I}_k, \mathbf{0}) \otimes \mathsf{sk}^\top \times \boldsymbol{g}^\top)\mathbf{F} = (\bar{C}(1), \ldots, \bar{C}(\kappa)) + \tilde{\mathbf{E}}$.

  I.e., instead of choosing $\mathbf{S}$ uniformly at random and setting $\mathbf{H} = \mathbf{S} + \mathbf{RF}$, we choose $\mathbf{H}$ uniformly at random and substitute $\mathbf{S}$ with $\mathbf{H} - \mathbf{RF}$. Furthermore, note that the terms involving the matrix $\mathbf{E}^*$ in $\mathbf{C}$ just cancel out. Furthermore, we have used the correctness property of GSW to substitute the term $\mathbf{PS}$ in the definition of $\mathbf{C}'$ with

$$\begin{aligned}\mathbf{PS} &= \mathbf{PH} - \mathbf{PRF} \\ &= \mathbf{PH} - (\mathbf{A} - \bar{\mathbf{E}} - (\mathbf{I}_k, \mathbf{0}) \otimes \mathsf{sk}^\top \times \boldsymbol{g}^\top)\mathbf{F} \\ &= \mathbf{PH} - \mathbf{AF} - \bar{\mathbf{E}}\mathbf{F} - (\bar{C}(1), \ldots, \bar{C}(\kappa)) - \tilde{\mathbf{E}}.\end{aligned}$$

  Hence hybrids $\mathcal{P}_0$ and $\mathcal{P}_1$ are identically distributed.

- Hybrid $\mathcal{P}_2$: This is identical to $\mathcal{P}_1$, except for the way we compute $\mathbf{C}$ and $\mathbf{C}'$:

  – Compute $\mathbf{C} = \mathbf{BH} - \mathbf{UF} + \mathbf{E}$ where $\mathbf{E} \leftarrow D^{m \times \kappa}_{\mathbb{Z},\sigma}$.
  – Compute $\mathbf{C}' = \mathbf{PH} - \mathbf{A} \cdot \mathbf{F} - (\bar{C}(\bar{K}, 1), \ldots, \bar{C}(\bar{K}, \kappa)) + \mathbf{E}'$ where $\mathbf{E}' \leftarrow D^{k \times \kappa}_{\mathbb{Z},\sigma'}$.

  Hybrids $\mathcal{P}_1$ and $\mathcal{P}_2$ are statistically close, as since $\|\tilde{\mathbf{E}}\|_\infty \leq \tilde{B}$ (by our assumption on GSW) and $\|\mathbf{E}^*\mathbf{F}\|_\infty \leq \lambda(k+1)m\rho)$ (except with probability $\tilde{O}(2^{-\lambda})$ as in the proof of Lemma 4.2), and further since by assumption $\sigma \geq 2^\lambda \cdot (\tilde{B} + \lambda(k+1)m\rho)$ we have by Lemma 2.4 that

$$\mathbf{E}^*\mathbf{F} + \mathbf{E} =_s \mathbf{E}$$

and

$$\bar{\mathbf{E}}\mathbf{F} - \tilde{\mathbf{E}} + \mathbf{E}' =_s \mathbf{E}'.$$

- Hybrid $\mathcal{P}_3$: This is identical to $\mathcal{P}_2$, except for the way we compute $\mathbf{A}$ and $\mathbf{U}$:
  - Choose $\mathbf{A}$ uniformly at random.
  - Choose $\mathbf{U}$ uniformly at random.

  Computational indistinguishability of $\mathcal{P}_2$ and $\mathcal{P}_3$ follows routinely from the $\mathsf{LWE}(q, n, \rho)$ assumption.

- Hybrid $\mathcal{P}_4$: This is identical to hybrid $\mathcal{P}_3$, except that we compute c via:
  - Compute $c \leftarrow \mathsf{Enc}(\mathsf{pk}, 0)$.

  Computational indistinguishability between $\mathcal{P}_3$ and $\mathcal{P}_4$ follows from the IND-CPA security of the FHE scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$.

  For the sake of presentational clarity, the full $\mathcal{P}_4$ is given as follows.
  - Choose $\mathbf{P} \leftarrow \mathbb{Z}_q^{k \times n}$ uniformly at random.
  - Choose $(K, \mathsf{aux}_K) \leftarrow \mathsf{KeySamp}(1^\lambda)$.
  - Choose $K' \leftarrow \{0,1\}^\lambda$ uniformly at random.
  - Set $\bar{K} = (K, K')$.
  - Sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^d)$.
  - Compute $c \leftarrow \mathsf{Enc}(\mathsf{pk}, 0)$.
  - Choose $\mathbf{A} \leftarrow \mathbb{Z}_q^{k \times (k+1)m}$ uniformly at random.
  - For $i \in \{1, \ldots, \kappa\}$:
    - Compute $c_i \leftarrow \mathsf{Eval}(\mathsf{pk}, \bar{C}(\cdot, i), c)$.
    - Let $\mathbf{f}_i \in \mathbb{Z}_q^{k \cdot n}$ be the vectorization of the linear function $\mathsf{Dec}(\cdot, c_i)$.
    - Choose $\boldsymbol{r}_i \leftarrow \{0,1\}^m$ uniformly at random.
  - Set $\mathbf{F} = (\mathbf{G}_{\boldsymbol{r}_1}^{-1}(\boldsymbol{f}_1), \ldots, \mathbf{G}_{\boldsymbol{r}_\kappa}^{-1}(\boldsymbol{f}_\kappa))$.
  - Choose $\mathbf{H}$ uniformly at random.
  - Set $\mathrm{aux} = (\mathbf{P}, \mathsf{pk}, c, \mathbf{A}, (\boldsymbol{r}_i)_{i \in \{1,\ldots,k\}}, \mathbf{H}, \mathsf{aux}_K)$.
  - Choose $\mathbf{B} \leftarrow \mathbb{Z}_q^{m \times n}$ uniformly at random.
  - Choose $\mathbf{U} \leftarrow \mathbb{Z}_q^{m \times (k+1)m}$ uniformly at random.
  - Compute $\mathbf{C} = \mathbf{BH} - \mathbf{UF} + \mathbf{E}$ where $\mathbf{E} \leftarrow D_{\mathbb{Z},\sigma}^{m \times \kappa}$.
  - Compute $\mathbf{C}' = \mathbf{PH} - \mathbf{A} \cdot \mathbf{F} - (\bar{C}(\bar{K}, 1), \ldots, \bar{C}(\bar{K}, \kappa)) + \mathbf{E}'$ where $\mathbf{E}' \leftarrow D_{\mathbb{Z},\sigma'}^{k \times \kappa}$.
  - Output $(\mathbf{B}, \mathbf{P}, \mathbf{C}, \mathbf{C}', \mathrm{aux})$.

- Hybrid $\mathcal{P}_5$: This is the same as $\mathcal{P}_4$, except that we compute $\mathbf{C}'$ via
  - Choose $\mathbf{C}'$ uniformly at random.

Note that both the matrix $\mathbf{A}$ and the matrix $\mathbf{F}$ are public, hence we cannot use LWE in this step to argue that $\mathbf{C}'$ is uniform. However, by the pseudorandomness of $\bar{C}$ it holds that

$$(\bar{C}((K,K'),1),\ldots,\bar{C}((K,K'),\kappa)),\mathsf{aux}_K \approx_c (\boldsymbol{c}_1,\ldots,\boldsymbol{c}_\kappa),\mathsf{aux}_K$$

for uniformly random $\boldsymbol{c}_1,\ldots,\boldsymbol{c}_\kappa \in \mathbb{Z}_q^k$. It follows that hybrids $\mathcal{P}_4$ and $\mathcal{P}_5$ are computationally indistinguishable.

- Hybrid $\mathcal{P}_6$: This is the same as $\mathcal{P}_5$, except that we choose $\mathbf{C}$ via:
  - Choose $\mathbf{C}$ uniformly at random.

  Computational indistinguishability follows from the $\mathsf{LWE}(q,n,D_{\mathbb{Z},\sigma})$ assumption.

  Specifically we will show that an distinguisher $\mathcal{D}$ for $\mathcal{P}_5$ and $\mathcal{P}_6$ gives rise to an LWE distinguisher $\mathcal{D}'$ with the same advantage. The distinguisher $\mathcal{D}'$ gets as input a matrices $\mathbf{V}$ and $\mathbf{Z}$, and needs to distinguish whether $\mathbf{Z}$ follows $\mathbf{T}\cdot\mathbf{V}+\mathbf{E}$ or is chosen uniformly random. $\mathcal{D}'$ simulates $\mathcal{P}_5$, except with the following modifications concerning how the $\boldsymbol{r}_i$ and $\mathbf{C}$ are computed.

  - Choose a matrix $\mathbf{U}'$ uniformly at random
  - For $i\in\{1,\ldots,\kappa\}$ set $\boldsymbol{r}_i = \mathbf{G}^{-1}(\boldsymbol{v}_i)$
  - Compute the matrix $\mathbf{C}$ via $\mathbf{C} = \mathbf{BH}+\mathbf{Z}+\mathbf{U}'\cdot\mathbf{F}$

  First observe that the $\boldsymbol{r}_i$ are i.i.d. uniformly random, as the matrix $\mathbf{V}$ is distributed uniformly random.

  Assume that $\mathbf{Z}$ is distributed according to $\mathbf{Z}=\mathbf{T}\cdot\mathbf{V}+\mathbf{E}$. Then it holds that

$$\begin{aligned}\mathbf{Z}+\mathbf{U}'\cdot\mathbf{F} &= \mathbf{TV}+\mathbf{E}+\mathbf{U}'\cdot\mathbf{F}\\ &= (\mathbf{0},\mathbf{T}\cdot\mathbf{G})\cdot\mathbf{F}+\mathbf{U}'\cdot\mathbf{F}+\mathbf{E}\\ &= ((\mathbf{0},\mathbf{T}\cdot\mathbf{G})+\mathbf{U}')\cdot\mathbf{F}+\mathbf{E}\\ &= \mathbf{U}\cdot\mathbf{F}+\mathbf{E},\end{aligned}$$

  where $\mathbf{U}=(\mathbf{0},\mathbf{T}\cdot\mathbf{G})+\mathbf{U}'$ is distributed uniformly random. Hence, in this case it holds that

$$\mathbf{C}=\mathbf{BH}+\mathbf{Z}+\mathbf{U}'\cdot\mathbf{F}=\mathbf{BH}+\mathbf{UF}+\mathbf{E}$$

  and therefore the distribution of $\tilde{K}$ produced by $\mathcal{D}'$ is identical to $\mathcal{P}_5$.

  On the other hand, if $\mathbf{Z}$ is distributed uniformly random, then the $\mathbf{C}=\mathbf{BH}+\mathbf{Z}+\mathbf{U}'\cdot\mathbf{F}$ is also distributed uniformly random, and therefore the distribution of $\tilde{K}$ produced by $\mathcal{D}'$ is identical to $\mathcal{P}_6$. Hence, $\mathcal{D}'$ has the same advantage as $\mathcal{D}$, and it follows by the hardness of LWE that $\mathcal{P}_5$ and $\mathcal{P}_6$ are indistinguishable.

In the following hybrids we will undo the modifications of $\mathcal{P}_1,\ldots,\mathcal{P}_4$

- Hybrid $\mathcal{P}_7$: In hybrid $\mathcal{P}_7$ we undo the modification done by hybrid $\mathcal{P}_4$, specifically we compute c via

– Compute $c \leftarrow \mathsf{Enc}(\mathsf{pk}, K)$.

Computational indistinguishability follows from the IND-CPA security of the FHE scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$, analogous to the hybrid step between $\mathcal{P}_3$ and $\mathcal{P}_4$.

- Hybrid $\mathcal{P}_8$: In this hybrid we undo a change done in hybrid $\mathcal{P}_3$, specifically we compute $\mathbf{A}$ via

  – Parse $\mathsf{sk} = (s_1, \ldots, s_n) \in \mathbb{Z}_q^n$, sample $\mathbf{R} \leftarrow \mathbb{Z}_q^{n \times (k+1) \cdot m}$ and $\bar{\mathbf{E}} \leftarrow D_{\mathbb{Z},\rho}^{k \times (k+1) \cdot m}$ and compute $\mathbf{A} = \mathbf{P}\mathbf{R} + \bar{\mathbf{E}} + (\mathbf{I}_k, \mathbf{0}) \otimes (s_1, \ldots, s_n) \otimes \boldsymbol{g}^\top$.

  Computational indistinguishability follows from the LWE assumption, analogous to the hybrid step between $\mathcal{P}_2$ and $\mathcal{P}_3$.

- Hybrid $\mathcal{P}_9$: In $\mathcal{P}_9$ we undo a change done in hybrid $\mathcal{P}_1$, specifically we compute $\mathbf{H}$ via

  – Choose $\mathbf{S} \leftarrow \mathbb{Z}_q^{n \times \kappa}$ uniformly at random.
  – Set $\mathbf{H} = \mathbf{S} + \mathbf{R} \cdot \mathbf{F}$.

  Hybrid $\mathcal{P}_9$ is identically distributed to hybrid $\mathcal{P}_8$, this is analogous to the hybrid step between $\mathcal{P}_0$ and $\mathcal{P}_1$.

  We provide $\mathcal{P}_9$ in full below and note that hybrid $\mathcal{P}_9$ is the ideal distribution of the evasive LWE precondition, with the the algorithm $\mathrm{Samp}(1^\lambda, \mathbf{B})$ unrolled.

  – Choose $\mathbf{P} \leftarrow \mathbb{Z}_q^{k \times n}$ uniformly at random.
  – Choose $(K, \mathsf{aux}_K) \leftarrow \mathsf{KeySamp}(1^\lambda)$.
  – Choose $K' \leftarrow \{0,1\}^\lambda$ uniformly at random.
  – Set $\bar{K} = (K, K')$.
  – Sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^d)$.
  – Compute $c \leftarrow \mathsf{Enc}(\mathsf{pk}, \bar{K})$.
  – Parse $\mathsf{sk} = (s_1, \ldots, s_n) \in \mathbb{Z}_q^n$, sample $\mathbf{R} \leftarrow \mathbb{Z}_q^{n \times (k+1) \cdot m}$ and $\bar{\mathbf{E}} \leftarrow D_{\mathbb{Z},\rho}^{k \times (k+1) \cdot m}$ and compute $\mathbf{A} = \mathbf{P}\mathbf{R} + \bar{\mathbf{E}} + (\mathbf{I}_k, \mathbf{0}) \otimes (s_1, \ldots, s_n) \otimes \boldsymbol{g}^\top$.
  – For $i \in \{1, \ldots, \kappa\}$:
    * Compute $c_i \leftarrow \mathsf{Eval}(\mathsf{pk}, \bar{C}(\cdot, i), c)$.
    * Let $\mathbf{f}_i \in \mathbb{Z}_q^{k \cdot n}$ be the vectorization of the linear function $\mathsf{Dec}(\cdot, c_i)$.
    * Choose $\boldsymbol{r}_i \leftarrow \{0,1\}^m$ uniformly at random.
  – Set $\mathbf{F} = (\mathbf{G}_{\boldsymbol{r}_1}^{-1}(\boldsymbol{f}_1), \ldots, \mathbf{G}_{\boldsymbol{r}_\kappa}^{-1}(\boldsymbol{f}_\kappa))$.
  – Choose $\mathbf{S} \leftarrow \mathbb{Z}_q^{n \times \kappa}$ uniformly at random.
  – Set $\mathbf{H} = \mathbf{S} + \mathbf{R} \cdot \mathbf{F}$.
  – Set $\mathrm{aux} = (\mathbf{P}, \mathsf{pk}, c, \mathbf{A}, (\boldsymbol{r}_i)_{i \in \{1,\ldots,k\}}, \mathbf{H})$.
  – Choose $\mathbf{C}$ uniformly at random.
  – Choose $\mathbf{C}'$ uniformly at random.

– Output $(\mathbf{B}, \mathbf{P}, \mathbf{C}, \mathbf{C}', \mathrm{aux})$.

Hence we have established that the evasive LWE precondition holds. As detailed above this leads to the desired contradiction via the evasive LWE assumption. This concludes the proof.

□

5 Blind Laconic Function Evaluation

5.1 Definitions

We recall the definition of laconic function evaluation (LFE) from [QWW18], while additionally defining the new blindness property that we consider in this work.

Definition 5.1 (Laconic Function Evaluation)**.** An LFE scheme for a class of circuits $C : \{0,1\}^\eta \to \{0,1\}^\ell$ consists of four PPT algorithms

$$(\mathsf{LFE.Setup}, \mathsf{LFE.Hash}, \mathsf{LFE.Enc}, \mathsf{LFE.Dec})$$

with the following syntax.

- $\mathsf{LFE.Setup}(1^\lambda)$: On input the security parameter 1^λ, the setup algorithm returns the common reference string crs.
- $\mathsf{LFE.Hash}(\mathsf{crs}, C)$: On input the common reference string crs and a circuit C, the hash algorithm returns deterministically a digest hash_C.
- $\mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{hash}_C, \mathbf{x})$: On input the common reference string crs, a digest hash_C, and an input $\mathbf{x}$, the encryption algorithm returns a ciphertext ct.
- $\mathsf{LFE.Dec}(\mathsf{crs}, C, \mathsf{ct})$: On input the common reference string crs, the circuit C, and the ciphertext ct, the decryption algorithm returns a message $\mathbf{y}$.

We require the following properties to hold.

- (ε-Correctness) For all $\lambda \in \mathbb{N}$, all $\mathsf{crs} \in \mathsf{LFE.Setup}(1^\lambda)$, all circuits C, and all inputs $\mathbf{x}$, it holds that
$$\Pr\left[\mathsf{LFE.Dec}(\mathsf{crs}, C, \mathsf{ct}) = C(\mathbf{x}) : \begin{array}{l} \mathsf{hash}_C \leftarrow \mathsf{LFE.Hash}(\mathsf{crs}, C) \\ \mathsf{ct} \leftarrow \mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{hash}_C, \mathbf{x}) \end{array}\right] \geq 1 - \varepsilon$$
where the probability is taken over the random coins of $\mathsf{LFE.Enc}$.
- (Efficiency) The runtime of the $\mathsf{LFE.Enc}$ algorithm, and consequently the size of the ciphertexts, must be sublinear in the size of the circuit C, that is
$$|\mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{hash}_C, \mathbf{x})| = \mathsf{poly}(\lambda) \cdot o(|C|).$$
- (Blindness) Let $\mathrm{InputSamp}$ be a sampling algorithm which produces pairs $(\mathbf{x}, \mathsf{aux}_\mathbf{x})$ and let C be a circuit. We require that if
$$\{C(\mathbf{x}), \mathsf{aux}_\mathbf{x}\} \approx_c \left\{\mathbf{u}, \mathsf{aux}_\mathbf{x} : \mathbf{u} \leftarrow \{0,1\}^\ell\right\}$$
then
$$\{\mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{hash}_C, \mathbf{x}), \mathsf{crs}, \mathsf{hash}_C, \mathsf{aux}_\mathbf{x}\} \approx_c \left\{\mathbf{u}, \mathsf{crs}, \mathsf{hash}_C, \mathsf{aux}_\mathbf{x} : \mathbf{u} \leftarrow \{0,1\}^{|\mathsf{ct}|}\right\}$$
where $\mathsf{crs} \leftarrow \mathsf{LFE.Setup}(1^\lambda)$, $\mathsf{hash}_C \leftarrow \mathsf{LFE.Hash}(\mathsf{crs}, C)$, and $(\mathbf{x}, \mathsf{aux}_\mathbf{x}) \leftarrow \mathrm{InputSamp}(1^\lambda)$.

5.2 Base Construction

We describe an LFE scheme that satisfies the blindness property. Our construction largely follows the scheme of [QWW18], with some small but crucial modifications needed to make the output of the LFE uniform. For notational convenience, we describe a construction for the class of circuits $C : \{0,1\}^\eta \to \{0,1\}$, whereas the more general construction for ℓ-bit output circuits is obtained by hashing circuits computing individual output bits, and encrypting the input ℓ times with respect to each hash.

- $\mathsf{LFE.Setup}(1^\lambda)$: Sample uniform

$$\mathsf{crs} = (\mathbf{A}_1, \ldots, \mathbf{A}_L) \in \mathbb{Z}_q^{n \times m}$$

where L equals η times the size of an FHE ciphertext and $m = (n+1)\log(q)$.

- $\mathsf{LFE.Hash}(\mathsf{crs}, C)$: Compute and output

$$\mathsf{hash}_C = \mathbf{A}_{\tilde{C}} \leftarrow \mathrm{EvalPK}(\tilde{C}, (\mathbf{A}_1, \ldots, \mathbf{A}_L))$$

where $\tilde{C}$ is the circuit that computes the homomorphic evaluation of C and removes the last row of the ciphertext.

- $\mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{hash}_C, \mathbf{x})$: Sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and parse $\mathbf{s}$ as the secret key. For all $i \in [\eta]$ compute $\mathbf{C}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, x_i)$ and let $\mathbf{c} = \mathbf{G}^{-1}(\mathbf{C}_1, \ldots, \mathbf{C}_\eta) \in \{0,1\}^L$ be the vectorized binary decomposition of $(\mathbf{C}_1, \ldots, \mathbf{C}_\eta)$. Then for all $i \in [L]$ compute

$$\mathbf{b}_i = \mathbf{s}^T(\mathbf{A}_i - c_i\bar{\mathbf{G}}) + \mathbf{e}_i^T$$

where $\mathbf{e}_i \leftarrow D_{\mathbb{Z},\rho}^m$ is a B-bounded noise term and $\bar{\mathbf{G}}$ denotes the gadget matrix $\mathbf{G} = \mathbf{I}_{n+1} \otimes \mathbf{g}$ without the last row. Sample $e \leftarrow [-q/4, q/4)$ and $\mathbf{t} \leftarrow \{0,1\}^m$ such that

$$\mathrm{MSB}((\mathbf{s}^T, -1)\mathbf{G}\mathbf{t} + e) = 1 \tag{5.1}$$

where MSB is the operator that returns the most significant bit. Finally, set

$$\beta = \mathbf{s}^T\mathbf{A}_{\tilde{C}}\mathbf{t} + e.$$

Return $\mathsf{ct} = (\beta, \mathbf{t}, \mathsf{pk}, \mathbf{b}_1, \ldots, \mathbf{b}_L, \mathbf{c})$.

- $\mathsf{LFE.Dec}(\mathsf{crs}, C, \mathsf{ct})$: Compute

$$\mathbf{C} \leftarrow \mathsf{Eval}(\mathsf{pk}, C, (\mathbf{C}_1, \ldots, \mathbf{C}_\eta))$$

and let $\mathbf{c}_{n+1}$ be the last row of $\mathbf{C}$. Then compute

$$\mathbf{b}_{\tilde{C}} \leftarrow \mathrm{EvalCT}(\tilde{C}, (\mathbf{A}_1, \ldots, \mathbf{A}_L), (\mathbf{b}_1, \ldots, \mathbf{b}_L), \mathbf{c}).$$

Return

$$\mathrm{MSB}\left(\beta - (\mathbf{b}_{\tilde{C}}^T + \mathbf{c}_{n+1})\mathbf{t}\right).$$

Correctness. Correctness follows by observing that

$$\begin{aligned}\mathrm{MSB}\left(\beta-(\mathbf{b}_{\tilde{C}}^T+\mathbf{c}_{n+1})\mathbf{t}\right) &= \mathrm{MSB}\left(\beta-(\mathbf{s}^T(\mathbf{A}_{\tilde{C}}-\bar{\mathbf{C}})+\mathbf{e}_{\tilde{C}}+\mathbf{c}_{n+1})\mathbf{t}\right)\\ &= \mathrm{MSB}\left(e+((\mathbf{s}^T,-1)\mathbf{C}+\mathbf{e}_{\tilde{C}})\mathbf{t}\right)\\ &= \mathrm{MSB}\left(e+(C(\mathbf{x})(\mathbf{s}^T,-1)\mathbf{G}+\mathbf{e}_{\tilde{C}}+\mathbf{e})\mathbf{t}\right)\\ &= \mathrm{MSB}\left(e+C(\mathbf{x})(\mathbf{s}^T,-1)\mathbf{G}\mathbf{t}+(\mathbf{e}_{\tilde{C}}+\mathbf{e})\mathbf{t}\right)\end{aligned}$$

where $\bar{\mathbf{C}}$ denotes all but the last row of $\mathbf{C}$, and the equalities hold by the correctness of the lattice homomorphic evaluation. Note that

$$\left|(\mathbf{e}_{\tilde{C}}+\mathbf{e})\mathbf{t}\right| \leq m\cdot\left\|\mathbf{e}_{\tilde{C}}+\mathbf{e}\right\|_\infty \leq (m+2)^d\cdot B\cdot\log(q)\cdot m^2\cdot L+\tilde{B}\cdot m$$

where d is the depth of C. For $q\geq 2^{\omega(1)}\cdot((m+2)^d\cdot B\cdot\log(q)\cdot m^2\cdot L+\tilde{B}\cdot m)$, we have that

$$e =_s e+(\mathbf{e}_{\tilde{C}}+\mathbf{e})\mathbf{t}$$

by Lemma 2.4, therefore with overwhelming probability we have that

$$\mathrm{MSB}\left(e+C(\mathbf{x})(\mathbf{s}^T,-1)\mathbf{G}\mathbf{t}+(\mathbf{e}_{\tilde{C}}+\mathbf{e})\mathbf{t}\right)=\mathrm{MSB}\left(e+C(\mathbf{x})(\mathbf{s}^T,-1)\mathbf{G}\mathbf{t}\right).$$

Observe that

$$\mathrm{MSB}\left(e+C(\mathbf{x})(\mathbf{s}^T,-1)\mathbf{G}\mathbf{t}\right)=\begin{cases}\mathrm{MSB}\left(e\right)=0 & \text{if } C(\mathbf{x})=0\\ \mathrm{MSB}\left(e+(\mathbf{s}^T,-1)\mathbf{G}\mathbf{t}\right)=1 & \text{if } C(\mathbf{x})=1\end{cases}$$

where the first case follows by the fact that $e\in[-q/4,q/4)$ and the second case follows by Eq. (5.1).

Blindness. Before presenting our main theorem, we prove a useful fact that we are going to use repeatedly in our proof.

Lemma 5.2. *For any fixed e^* such that $|e^*|\leq B$ and any fixed u^*, the following distributions are all statistically close:*

$$\begin{aligned}&\{e\mid e\leftarrow[-q/4,q/4)\ \&\ \mathrm{MSB}\left(e+u^*\right)=1\}\\ =_s\ &\{e+e^*\mid e\leftarrow[-q/4,q/4)\ \&\ \mathrm{MSB}\left(e+e^*+u^*\right)=1\}\\ =_s\ &\{e+e^*\mid e\leftarrow[-q/4,q/4)\ \&\ \mathrm{MSB}\left(e+u^*\right)=1\}.\end{aligned}$$

Proof. The first equation is a standard application of Lemma 2.4. The second equation is also a smudging argument, and it follows by observing that the distribution of the top random variable can be alternatively represented by a uniform distribution whose support is the vector of size $q/2$ defined as

$$(\underbrace{e_1,\ldots,e_{|S|}}_{e_i\in S},\perp,\ldots,\perp)$$

where S is the set of contiguous e such that $\mathrm{MSB}\left(e+u^*\right)=1$. Then adding any B-bounded element e_B to e has the effect of shifting the entries by e_B:

$$(\underbrace{e_1+e_B,\ldots,e_{|S|}+e_B}_{e_i\in S},\perp,\ldots,\perp)$$

and therefore the two vectors differ in at most $2B$ entries, by rearranging them appropriately. Thus we can conclude that the statistical distance between the top and the bottom distributions is bounded by the probability of sampling one such element, which is $4B/q$. The final claim follows by a triangle inequality. □

We are now in the position of showing that the scheme satisfies blindness.

Theorem 5.3. *If the* $\mathsf{LWE}(q, n, \rho)$ *problem is hard, then the LFE scheme as described above satisfies blindness.*

Proof. The crux of the proof is to describe a simulator $\mathsf{LFE.Sim}$ that simulates the ciphertext distribution, given as input the computation output $C(\mathbf{x})$, and in particular without knowing the input $\mathbf{x}$. We gradually modify the view induced by our scheme in a series of hybrid experiments.

- Hybrid $\mathcal{H}_0$: This is identical to the real distribution.
- Hybrid $\mathcal{H}_1$: In this hybrid, we modify the sampling procedure of $\mathbf{u} = (\bar{\mathbf{u}}, u) = \mathbf{G}\mathbf{t}$ and e as follows:
  - $\mathbf{u}$ is uniformly sampled over $\mathbb{Z}_q^{n+1}$.
  - e is sampled uniformly from $[-q/4, q/4)$ together with a B-bounded noise term e_B, and abort if $\mathrm{MSB}\left(e + e_B + (\mathbf{s}^T, -1)\mathbf{u}\right) \neq 1$. Return $e + e_B$.

  Note that $\mathbf{G}$ is injective and the distribution post-selected on not aborting is the same as conditional sampling, thus this distribution is statistically close to the original one by an application of Lemma 5.2.
- Hybrid $\mathcal{H}_2$: In this hybrid, we sample the β component of the ciphertext as follows:

$$\beta = e + C(\mathbf{x})(\underbrace{e_B + \mathbf{s}^T\bar{\mathbf{u}}}_{=v} - u) + (\mathbf{b}_{\tilde{C}}^T + \mathbf{c}_{n+1})\mathbf{t}.$$

  To analyze this distribution, we consider two cases separately.
  - $C(\mathbf{x}) = 0$: In this case we have that

$$\begin{aligned}\beta &= e + (\mathbf{b}_{\tilde{C}}^T + \mathbf{c}_{n+1})\mathbf{t} \\ &=_s e + e_B + (\mathbf{b}_{\tilde{C}}^T + \mathbf{c}_{n+1})\mathbf{t} \\ &=_s e + e_B + (\mathbf{b}_{\tilde{C}}^T + \mathbf{c}_{n+1})\mathbf{t} + (\mathbf{e}_{\tilde{C}} + \mathbf{e})\mathbf{t}\end{aligned}$$

    by Lemma 5.2.
  - $C(\mathbf{x}) = 1$: In this case we have that

$$\begin{aligned}\beta &= e + v - u + (\mathbf{b}_{\tilde{C}}^T + \mathbf{c}_{n+1})\mathbf{t} \\ &= e + e_B + (\mathbf{s}^T, -1)\mathbf{u} + (\mathbf{b}_{\tilde{C}}^T + \mathbf{c}_{n+1})\mathbf{t} \\ &=_s e + e_B + (\mathbf{s}^T, -1)\mathbf{u} + (\mathbf{b}_{\tilde{C}}^T + \mathbf{c}_{n+1})\mathbf{t} + (\mathbf{e}_{\tilde{C}} + \mathbf{e})\mathbf{t}\end{aligned}$$

    by Lemma 5.2.

In both cases, the latter string is precisely what the distinguisher is expecting in the previous hybrid.

- Hybrid $\mathcal{H}_3$: In this hybrid we sample $(\mathbf{b}_1, \ldots, \mathbf{b}_L, \mathsf{pk}, v)$ uniformly at random.

  Indistinguishability follow by a reduction against the LWE assumption, since all terms in the real distributions are LWE samples with respect to the secret $\mathbf{s}$. Let $\mathbf{A}_i^*$ be the LWE matrix corresponding to $\mathbf{b}_i$, the reduction can simulate the view of the distinguisher by computing $\mathbf{A}_i = \mathbf{A}_i^* + c_i\bar{\mathbf{G}}$, β as specified above, sampling u uniformly in $\mathbb{Z}_q$ and e uniformly from $[-q/4, q/4)$, aborting if $\mathrm{MSB}(e + v - u) \neq 1$.

  It is clear that the reduction perfectly simulates the view of the distinguisher of $\mathcal{H}_2$ (in the LWE case) or of $\mathcal{H}_3$ (in the uniform case). Thus, all is left to be shown is that the reduction does not abort too often. It is easy to see that in the uniform case the reduction aborts with probability exactly $1/2$, since v is uniform. Therefore the same (up to a negligible term) must hold for the LWE case, as otherwise this test could be used to distinguish the two distributions.

- Hybrid $\mathcal{H}_4$: In this hybrid, we sample $(\mathbf{C}_1, \ldots, \mathbf{C}_\eta)$ uniformly.

  Indistinguishability is statistical, and follows by the security of the GSW encryption scheme.

- Hybrid $\mathcal{H}_5$: In this hybrid, we compute β as follows:

$$\beta = q/2 \cdot C(\mathbf{x}) + r + (\mathbf{b}_{\tilde{C}}^T + \mathbf{c}_{n+1})\mathbf{t}.$$

  where $r \leftarrow [-q/4, q/4)$ is randomly sampled.

  We claim that this distribution is identical to the previous one. Recall that in the previous hybrid β was computed under the constraint that

$$\mathrm{MSB}\left(\beta - (\mathbf{b}_{\tilde{C}}^T + \mathbf{c}_{n+1})\mathbf{t}\right) = \mathrm{MSB}\left(e + C(\mathbf{x})(v-u)\right) = C(\mathbf{x}).$$

  For $C(\mathbf{x}) = 0$, e is uniform subject to $\mathrm{MSB}(e) = 0$ because it is conditioned upon a random v that is not present in the view of the distinguisher and therefore its marginal is uniform in $[-q/4, q/4)$. For $C(\mathbf{x}) = 1$, the term $e+v-u$ is uniform subject to $\mathrm{MSB}(e+v-u) = 1$ because v is uniformly sampled.

We then define our simulator $\mathsf{LFE.Sim}$ to be identical to the last hybrid, where all bits of the ciphertexts are uniformly sampled, except for the most significant bit of $\beta - (\mathbf{b}_{\tilde{C}}^T + \mathbf{c}_{n+1})\mathbf{t}$, which is set to be identical to $C(\mathbf{x})$. For multi-bit outputs, the simulator acts identically, using independent randomness for each ciphertext, and in particular for each β_i. At this point, we can appeal to the fact that

$$\{C(\mathbf{x}), \mathsf{aux}_{\mathbf{x}}\} \approx_c \left\{\mathbf{u}, \mathsf{aux}_{\mathbf{x}} : \mathbf{u} \leftarrow \{0,1\}^\ell\right\}$$

to conclude that the output of the simulator is computationally indistinguishable from uniform.

□

Efficiency. Overall, we presented a blind LFE scheme for a circuit $C : \{0,1\}^\eta \to \{0,1\}^\ell$ with depth $d = \mathsf{depth}(C)$, with the following asymptotic sizes:

$$|\mathsf{crs}| = \mathsf{poly}(\lambda, d) \cdot \eta \quad \text{and} \quad |\mathsf{hash}_C| = \mathsf{poly}(\lambda, d) \cdot \ell \quad \text{and} \quad |\mathsf{ct}| = \mathsf{poly}(\lambda, d) \cdot \eta \cdot \ell.$$

Furthermore, we have the following asymptotic runtimes of the algorithms:

$$|\mathsf{LFE.Setup}| = \mathsf{poly}(\lambda, d) \cdot \eta \quad \text{and} \quad |\mathsf{LFE.Hash}| = |\mathsf{LFE.Dec}| = \mathsf{poly}(\lambda, |C|)$$

whereas

$$|\mathsf{LFE.Enc}| = \mathsf{poly}(\lambda, d) \cdot \eta \cdot \ell \quad \text{and} \quad \mathsf{depth}(\mathsf{LFE.Enc}) = \mathsf{poly}(\lambda, d).$$

5.3 Decreasing the Depth of the Encryption Algorithm

In the following we show a generic transformation to decrease the depth of the encryption algorithm of our blind LFE scheme, without changing the complexity of the other algorithms.

Blind Randomized Encodings. We recall the notion of blind randomized encodings (RE) from [BLSV18].

Definition 5.4 (Blind Randomized Encodings)**.** A blind RE consists of two PPT algorithm $(\mathrm{RE.Enc}, \mathrm{RE.Enc})$ with the following syntax.

- $\mathrm{RE.Enc}(C, \mathbf{x})$: On input a circuit C and an input $\mathbf{x}$, the encoding algorithm returns an encoding $\tilde{E}$.
- $\mathrm{RE.Dec}(\tilde{E})$: On input an encoding $\tilde{E}$, the decoding algorithm returns an output $\mathbf{y}$.

We require the following properties to hold.

- (Correctness) For all circuits C and inputs $\mathbf{x}$ it holds that
$$\mathrm{RE.Dec}(\mathrm{RE.Enc}(C, \mathbf{x})) = C(\mathbf{x}).$$
- (Depth-Independence) The circuit depth of the encoding algorithm is independent of C, in other words
$$\mathsf{depth}(\mathrm{RE.Enc}) = \mathsf{poly}(\lambda).$$
- (Blindness) Let $\mathrm{InputSamp}$ be a sampling algorithm which produces pairs $(\mathbf{x}, \mathsf{aux}_\mathbf{x})$ and let C be a circuit. We require that if
$$\{C(\mathbf{x}), \mathsf{aux}_\mathbf{x}\} \approx_c \left\{\mathbf{u}, \mathsf{aux}_\mathbf{x} : \mathbf{u} \leftarrow \{0,1\}^\ell\right\}$$
then
$$\{\mathrm{RE.Enc}(C, \mathbf{x}), \mathsf{aux}_\mathbf{x}\} \approx_c \left\{\mathbf{u}, \mathsf{aux}_\mathbf{x} : \mathbf{u} \leftarrow \{0,1\}^{|\tilde{E}|}\right\}.$$
where $(\mathbf{x}, \mathsf{aux}_\mathbf{x}) \leftarrow \mathrm{InputSamp}(1^\lambda)$.

It is shown that blind randomized encodings can be constructed from any PRF, and we recall here a sketch of the construction for completeness. For a detailed analysis, we refer the reader to [BLSV18].

Theorem 5.5 ([BLSV18])**.** *If there exists a PRF, then there exists a blind randomized encoding scheme for all polynomial-size circuits.*

Proof Sketch. For each wire w in the circuit sample a permutation bit $\alpha_w \leftarrow \{0,1\}$ and two PRF keys $s_{w,b} \leftarrow \{0,1\}^\lambda$. For any logical gate $g : \{0,1\}^2 \to \{0,1\}$, with input wires w_1, w_2 and output wire w_3, and for all $\beta_1, \beta_2 \in \{0,1\}$ compute the table entry

$$T_{g,\beta_1,\beta_2} = (s_{w_3,\alpha_3\oplus\beta_3}, \beta_3) \oplus \mathsf{PRF}(s_{w_1,\alpha_1\oplus\beta_1}, (g,\beta_1,\beta_2)) \oplus \mathsf{PRF}(s_{w_2,\alpha_2\oplus\beta_2}, (g,\beta_1,\beta_2))$$

where $\beta_3 = g(\alpha_1 \oplus \beta_1, \alpha_2 \oplus \beta_2) \oplus \beta_3$.

The output of the randomized encoding contains all the table entries $\{T_{g,\beta_1,\beta_2}\}$, together with the the permutation bits $\{\alpha_j\}$ for all output wires j, and the pairs $\{s_{i,x_i}, x_i \oplus \alpha_i\}$ for all input wires i. The evaluate the encoding, follow the gates in topological order, and unmask the PRF output with the provided information. Follow the pointers in the last bit to decide which entry of the table to unmask. □

Transformation. We now compile our blind LFE scheme into one with shallow encryption algorithm. The LFE.Setup and LFE.Hash algorithms are unchanged, so we only describe the modification of the remaining algorithms.

- $\mathsf{LFE.Enc}^*(\mathsf{crs}, \mathsf{hash}_C, \mathbf{x})$: Sample $\mathbf{r} \leftarrow \{0,1\}^r$ where r is the amount of random coins needed by the LFE.Enc algorithm. Compute

  $$\tilde{E} \leftarrow \mathrm{RE.Enc}(\mathsf{LFE.Enc}, (\mathsf{crs}, \mathsf{hash}_C, \mathbf{x}, \mathbf{r}))$$

  and return $\mathsf{ct} = \tilde{E}$.

- $\mathsf{LFE.Dec}^*(\mathsf{crs}, C, \mathsf{ct})$: Return

  $$\mathsf{LFE.Dec}(\mathsf{crs}, C, \mathrm{RE.Dec}(\tilde{E})).$$

Since the blind RE scheme is perfectly correct, the correctness of the compiled scheme follows immediately from that of the underlying blind LFE. Next we show that the scheme is still blind.

Theorem 5.6. *If the RE scheme and the LFE scheme are blind, then the LFE scheme as described above satisfies blindness.*

Proof. Follows by observing that the output of $\mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{hash}_C, \mathbf{x})$ is uniformly distributed if so is $C(\mathbf{x})$, and consequently so is the output of

$$\mathrm{RE.Enc}(\mathsf{LFE.Enc}, (\mathsf{crs}, \mathsf{hash}_C, \mathbf{x}); \mathbf{r})$$

Thus we can conclude the proof by appealing to the blindness of the RE scheme. □

Finally, we argue that the complexity of the scheme is not increased by this transformation. In particular, the new size of the ciphertext is bounded by the complexity of the old LFE.Enc algorithm, times a fixed polynomial in the security parameter, which is $|\mathsf{ct}| = \mathsf{poly}(\lambda, d) \cdot \eta \cdot \ell$, same as before. On the other hand, the depth of the new encryption algorithm is

$$\mathsf{depth}(\mathsf{LFE.Enc}^*) = \mathsf{poly}(\lambda)$$

regardless of the depth of C.

5.4 Decreasing the Correctness Error

In the following we show how to arbitrarily decrease the correctness error of our blind LFE scheme. Note that the blind randomized encoding scheme of [BLSV18] has perfect correctness, and therefore the only error happens when the base LFE scheme causes an error. Upon closer inspection, we note that the only event that triggers an error is whenever

$$\mathrm{MSB}\left(e + C(\mathbf{x})(\mathbf{s}^T, -1)\mathbf{G}\mathbf{t} + (\mathbf{e}_{\tilde{C}} + \mathbf{e})\mathbf{t}\right) \neq \mathrm{MSB}\left(e + C(\mathbf{x})(\mathbf{s}^T, -1)\mathbf{G}\mathbf{t}\right).$$

Note that this can only happen if

$$e + C(\mathbf{x})(\mathbf{s}^T, -1)\mathbf{G}\mathbf{t} \in [-B_e, B_e] \cup [q/2 - B_e, q/2 + B_e] \tag{5.2}$$

where $B_e = (m+2)^d \cdot B \cdot \log(q) \cdot m^2 \cdot L + \tilde{B} \cdot m$ is the bound on the magnitude of the noise term. Thus, we can drive down the correctness error by letting the encrypter run N parallel instances of the encryption algorithm, and each time check if Eq. (5.2) does *not* hold for both $C(\mathbf{x}) = 0$ and $C(\mathbf{x}) = 1$. The probability that none of the instances passes this check decreases exponentially in N. The runtime of the encryption algorithm is increased by a factor of N and its depth is increased by a factor $\log(N) < \lambda$, and thus it is asymptotically unchanged. Therefore we henceforth assume that the LFE has an arbitrary ε correctness error and the runtime of the encryption algorithm is changed to

$$|\mathsf{LFE.Enc}| = \mathsf{poly}(\lambda, d, \log(1/\varepsilon)) \cdot \eta \cdot \ell$$

with all other asymptotics unchanged.

5.5 Decreasing the Size of the Ciphertext

Finally, we further reduce the ciphertext of our LFE scheme by another generic transformation, which uses a standard pseudorandom generator PRG and a blind obfuscation $(\mathsf{xPRO.Obf}, \mathsf{xPRO.Eval})$. For a circuit C, let us define the circuit C^* as

$$C^*(\mathbf{x}, K) = \mathsf{xPRO.Obf}(1^\lambda, C, \mathbf{x}; \mathsf{PRG}(K)).$$

We define the new LFE scheme to hash C^* instead, but except for this the algorithms LFE.Setup and LFE.Hash are unchanged. Below we describe the modifications to the encryption and decryption algorithms.

- $\mathsf{LFE.Enc}^*(\mathsf{crs}, \mathsf{hash}_{C^*}, \mathbf{x})$: Sample $K \leftarrow \{0,1\}^\lambda$ and return

$$\mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{hash}_{C^*}, (\mathbf{x}, K))$$

- $\mathsf{LFE.Dec}^*(\mathsf{crs}, C^*, \mathsf{ct})$: Define $\tilde{K} = \mathsf{LFE.Dec}(\mathsf{crs}, C^*, \mathsf{ct})$ and return

$$\mathsf{xPRO.Eval}(C, \tilde{K})$$

  where $\mathsf{xPRO.Eval}(C, \tilde{K})$ evaluates the obfuscated circuit on all inputs.

Correctness of the scheme follows immediately by a union bound on the correctness of the LFE and the xPRO schemes. The next theorem establishes the blindness of the construction.

Theorem 5.7. *If the PRG is pseudorandom and the xPRO scheme and the LFE scheme are blind, then the LFE scheme as described above satisfies blindness.*

Proof. Assuming that $C(\mathbf{x})$ is computationally indistinguishable from uniform, we have that

$$\begin{aligned}\left\{\mathsf{xPRO.Obf}(1^\lambda, C, \mathbf{x}; \mathsf{PRG}(K)), \mathsf{aux}_{\mathbf{x}}\right\} &\approx_c \left\{\mathsf{xPRO.Obf}(1^\lambda, C, \mathbf{x}), \mathsf{aux}_{\mathbf{x}}\right\} \\ &\approx_c \{\mathbf{u}, \mathsf{aux}_{\mathbf{x}}\}\end{aligned}$$

where the first implication follows by the pseudorandomness of the PRG and the second implication follows by the security of xPRO. Appealing to the blindness of the LFE, we can conclude that $\mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{hash}_{C^*}, (\mathbf{x}, K))$ is computationally close to uniform. □

Note that the asymptotic complexity of the algorithms is unchanged, with the exception of the size of the ciphertext, which is now bounded by $|\mathsf{ct}| = \mathsf{poly}(\lambda, d) \cdot \eta \cdot \ell^{1-\delta}$, for some constant $\delta > 0$. Since we can always increase ℓ artificially by, e.g., padding the output with pseudorandom bits, we can assume without loss of generality that ℓ dominates the above expression and therefore that

$$|\mathsf{ct}| \leq \frac{\ell}{2} \tag{5.3}$$

which is going to be a more convenient bound to work with.

6 Bootstrapping Pseudorandom Obfuscation

We provide here our bootstrapping theorems for PRO. First we show how to construct a PRO for circuits starting from a blind LFE with appropriate efficiency properties. Then we show how to bootstrap any PRO for circuits into a PRO for Turing machines.

6.1 Pseudorandom Obfuscation for Circuits

We provide here our bootstrapping theorem for pseudorandom obfuscation, which is similar in spirit to the transformation shown in [BV15, AJ15]. We are going to obfuscate an arbitrary pseudorandom function

$$\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\eta \to \{0,1\}^\ell$$

provided that it is *truth-table pseudorandom*, namely that it is pseudorandom against any distinguisher running in time polynomial in 2^η that gets to see the entire truth table of $\mathsf{PRF}(K, \cdot)$ for a random K. To achieve this, we are going to use the following ingredients:

- The blind LFE scheme $(\mathsf{LFE.Setup}, \mathsf{LFE.Hash}, \mathsf{LFE.Enc}, \mathsf{LFE.Dec})$ presented in Section 5.2, after applying the transformations described in Section 5.3, Section 5.4, and Section 5.5.
- A pseudorandom function $\mathsf{PRF}^* : \{0,1\}^\lambda \times \{0,1\}^\eta \to \{0,1\}^r$ where $r = r(\lambda)$ is the number of random coins needed by the $\mathsf{LFE.Enc}$ algorithm.

The security of both primitives is governed by the security parameter λ, which we set in such a way that security holds also for *sub-exponential* adversaries running in time polynomial in 2^η.

Construction. Our construction proceeds as follows.

- $\mathsf{PRO.Obf}(1^\lambda, \mathsf{PRF}, K)$: Sample keys $(K_2^*, \ldots, K_\eta^*) \leftarrow \{0,1\}^\lambda$. Then sample $\mathsf{crs}_\eta \leftarrow \mathsf{LFE.Setup}(1^\lambda)$ and compute $\mathsf{hash}_\eta \leftarrow \mathsf{LFE.Hash}(\mathsf{crs}, C_\eta)$ where the circuit C_η is defined as:

$$C_\eta(K, x) = \mathsf{PRF}(K, x).$$

  Then, for all $i = \{\eta - 1, \ldots, 1\}$ compute $\mathsf{crs}_i \leftarrow \mathsf{LFE.Setup}(1^\lambda)$ and $\mathsf{hash}_i \leftarrow \mathsf{LFE.Hash}(\mathsf{crs}_i, C_i)$ where the circuit C_i is defined as:

$$C_i(K, (\mathsf{crs}_j, \mathsf{hash}_j, K_j^*)_{j=i+1,\ldots,\eta}, x) = \\ \left(\begin{array}{l} \mathsf{LFE.Enc}(\mathsf{crs}_{i+1}, \mathsf{hash}_{i+1}, (K, (\mathsf{crs}_j, \mathsf{hash}_j, K_j^*)_{j=i+1,\ldots,\eta}, x, 0); \mathsf{PRF}^*(K_{i+1}^*, (x, 0))), \\ \mathsf{LFE.Enc}(\mathsf{crs}_{i+1}, \mathsf{hash}_{i+1}, (K, (\mathsf{crs}_j, \mathsf{hash}_j, K_j^*)_{j=i+1,\ldots,\eta}, x, 1); \mathsf{PRF}^*(K_{i+1}^*, (x, 1))) \end{array} \right).$$

  Note that representations of the C_i as boolean circuits can be publicly and efficiently computed given only the security parameter 1^λ.

  The obfuscated circuit $\tilde{K}$ consists of

$$\{\mathsf{crs}_i\}_{i \in \{1,\ldots,\eta\}}$$

  and

$$\left\{\mathsf{ct}_{1,x} \leftarrow \mathsf{LFE.Enc}(\mathsf{crs}_1, \mathsf{hash}_1, (K, (\mathsf{crs}_j, \mathsf{hash}_j, K_j^*)_{j=2,\ldots,\eta}, x))\right\}_{x \in \{0,1\}}.$$

- $\mathsf{PRO.Eval}(\mathsf{PRF}, \tilde{K}, x)$: To evaluate the obfuscated circuit at a point $x \in \{0,1\}^\eta$, proceed as follows. For all $i \in \{1, \ldots, \eta - 1\}$ compute recursively

$$(\mathsf{ct}_{i+1,0}, \mathsf{ct}_{i+1,1}) \leftarrow \mathsf{LFE.Dec}(\mathsf{crs}_i, C_i, \mathsf{ct}_{i,x_i})$$

  where $\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}$ are the LFE ciphertexts contained in $\tilde{K}$. Return

$$\mathsf{LFE.Dec}(\mathsf{crs}_\eta, C_\eta, \mathsf{ct}_{\eta,x_\eta})$$

Analysis. To see that the scheme is efficient, it suffices to argue that the size of each circuit C_i is bounded by some polynomial in the security parameter, which is also an upper bound on the runtime of the LFE algorithms. For C_η, this is trivial since it only computes the output of PRF, which is efficient. On the other hand, the complexity of other circuits C_i is dominated by two computations of PRF^*, together with two calls to the LFE.Enc ciphertext. Overall, we can bound this by

$$\begin{aligned} |C_i| &= \mathsf{poly}(\lambda, \mathsf{depth}(\mathsf{LFE.Enc})) \cdot (2\lambda + \eta) \cdot 2|\mathsf{ct}_{i+1}| \\ &= \mathsf{poly}(\lambda) \cdot 2|\mathsf{ct}_{i+1}| \end{aligned}$$

which holds because the depth of the LFE.Enc algorithm is bounded by a fixed polynomial in the security parameter and $\eta < \lambda$. Then, denoting by ℓ_i the size of the output of C_i and applying Eq. (5.3), we obtain that

$$|C_i| = \mathsf{poly}(\lambda) \cdot 2|\mathsf{ct}_{i+1}| \leq \mathsf{poly}(\lambda) \cdot 2\frac{\ell_{i+1}}{2} = \mathsf{poly}(\lambda) \cdot 2|\mathsf{ct}_{i+2}| \leq \cdots \leq \mathsf{poly}(\lambda) \cdot \ell_\eta.$$

Since $\ell_\eta = \ell$ is just the size of the output of PRF, then efficiency follows. In particular, the above recurcive bound establishes that the size of the obfuscated circuit is polynomial in the size of an LFE ciphertext evaluating PRF, which is in turn bounded by a factor $\mathsf{poly}(\lambda, d, \eta, \ell) = \mathsf{poly}(\lambda, d)$, where d is the circuit depth of PRF. Furthermore, the depth of the circuit computing the obfuscation is bounded by a fixed polynomial in λ, by the depth-independence of the LFE.

On the other hand, correctness follows by a union bound on the ε-correctness of the LFE: As argued above, each ciphertext is of size bounded by a polynomial in the security parameter, and we have at most 2^η of them. Setting $\varepsilon = 2^{-\omega(\eta, \log(\lambda))}$ we obtain an arbitrary small correctness error, where the only other algorithm that depends on this parameter is

$$|\mathsf{LFE.Enc}| = \mathsf{poly}(\lambda, d, \log(1/\varepsilon)) \cdot \eta \cdot \ell = \mathsf{poly}(\lambda, d) \cdot \eta \cdot \ell = \mathsf{poly}(\lambda)$$

since $\eta < \lambda$. As for security, the following theorem establishes our main result.

Theorem 6.1. *If PRF schemes are pseudorandom and the LFE scheme is blind, both against distinguishers running in time polynomial in 2^η, then the obfuscation scheme as described above satisfies pseudorandomness.*

Proof. By the pseudorandomness of PRF, we have that

$$\{\mathsf{PRF}(K, x)\}_{x \in \{0,1\}^\eta}, \mathsf{aux}_K \approx_c \{u_x : u_x \leftarrow \{0,1\}^\ell\}_{x \in \{0,1\}^\eta}, \mathsf{aux}_K$$

then, by appealing to the pseudorandomness of PRF^* and the blindness of the LFE scheme, we have that

$$\begin{aligned}
&\{\mathsf{LFE.Enc}(\mathsf{crs}_\eta, \mathsf{hash}_\eta, (K, x); \mathsf{PRF}^*(K^*_\eta, x))\}_{x \in \{0,1\}^\eta}, \mathsf{aux}_K \\
&\approx_c \{\mathsf{LFE.Enc}(\mathsf{crs}_\eta, \mathsf{hash}_\eta, (K, x); r_x) : r_x \leftarrow \{0,1\}^r\}_{x \in \{0,1\}^\eta}, \mathsf{aux}_K \\
&\approx_c \{z_x : z_x \leftarrow \{0,1\}^{|\mathsf{ct}_\eta|}\}_{x \in \{0,1\}^\eta}, \mathsf{aux}_K
\end{aligned}$$

Observing that the ciphertexts at level $i+1$ are the outputs of the computation at level i, we can apply the above argument recursively to conclude that

$$\begin{aligned}
\{\mathsf{LFE.Enc}(\mathsf{crs}_1, &\mathsf{hash}_1, (K, (\mathsf{crs}_j, \mathsf{hash}_j, K^*_j)_{j=2,\ldots,\eta}, x))\}_{x \in \{0,1\}}, \mathsf{aux}_K \\
&\approx_c \{z_x : z_x \leftarrow \{0,1\}^{|\mathsf{ct}_1|}\}_{x \in \{0,1\}}, \mathsf{aux}_K.
\end{aligned}$$

The proof is concluded by observing that the obfuscated circuit $\tilde{K}$ no longer depends on K. □

6.2 Pseudorandom Obfuscation for Turing Machines

Turing Machines. Formally, a Turing Machine is described by a tuple $M = (\Sigma, S, f)$, where Σ is a finite alphabet, S is a set of states containing the starting and the halting state, and f is a transition function. In this work, we always consider the alphabet and the set of states as fixed, and therefore a Turing Machine is completely described by its transition function, which we assume to be given in circuit form C_f. The Turing machine is given a worktape, that the machine can use to read/write information, and an output tape, that the machine can use to output some information. We assume without loss of generality that the input is loaded in the worktape and that the Turing machine is deterministic. Furthermore, we can also assume without loss of generality that the Turing machine

is *oblivious*, i.e., the movements of the head are deterministic and do not depend on the input of the computation, but only on its size [AB06, Imp11]. It is well-known that, fixing an input size s and a runtime T, an oblivious Turing machine can be converted into a circuit of size $O(sT)$, and we denote this size by $\mathrm{Circuit}(s, T)$.

We are going to define the *gate-synthesis* function G that takes as input the description of a Turing machine M and a gate index i, and returns the topology of the i-th gate of the corresponding circuit. The topology of a gate includes the logical operation performed by the gate, as well as the indices of the input and output wires. We are going to state a well-known fact from complexity theory.

Proposition 6.2. *Let M be a Turing machine, there exists a gate-synthesis function G such that*

$$|G| = \mathsf{poly}(|M|, \log(s), \log(T))$$

where $|G|$ denotes the circuit size of G.

Proof Sketch. Since the Turing machine is oblivious, this means that the position of the head on the tape at any given point in the computation is fixed, and in fact it can be efficiently computed without running the entire computation. For the simplest model of oblivious Turing machines, where the machine runs over the entire tape at each step, this can be done using simple modular arithmetic.

The circuit conversion process then defines a circuit that checks that each cell position is correct at every time step. This is done by checking that the neighboring cells in the current time step and the previous time step, along with the state of the Turing machine, are all valid. Since we need to verify a constant number of cells at each time, the overhead is constant. □

Obfuscating Turing Machines. We show how to use a PRO for circuits to obfuscate any pseudorandom function $\mathsf{PRF} : \{0,1\}^\eta \times \{0,1\}^\lambda \to \{0,1\}^\ell$, where the complexity of the obfuscation algorithm grows only with the size of the Turing machine description of PRF. We denote this scheme by

$$\mathsf{PRO}^{\mathrm{TM}} = (\mathsf{PRO.Obf}^{\mathrm{TM}}, \mathsf{PRO.Eval}^{\mathrm{TM}})$$

We are going to use as additional building blocks another pseudorandom function PRF^* and the blind randomized encoding scheme from Theorem 5.5. Specifically, on input a Turing machine M_{PRF}, an input size η, and a runtime T, we define the circuit $C_{M_{\mathsf{PRF}},\eta,T}(K, K^*, i, x)$ as follows:

- Compute $g_i \leftarrow G(M_{\mathsf{PRF}}, i)$ where $i \in \{1, \ldots, \mathrm{Circuit}(\eta, T)\}$.
- Let $W = \{w\}$ the set of wires associated with g_i, where we can assume without loss of generality that $|W| = O(1)$. Compute

$$\alpha_w \leftarrow \mathsf{PRF}^*(K^*, (\mathrm{perm}, x, w)) \quad \text{and} \quad \{s_{w,b} \leftarrow \mathsf{PRF}^*(K^*, (\mathrm{key}, x, w, b))\}_{b\in\{0,1\}}\,.$$

- If g_i is an input gate, then it consists of a single wire w, and we return $(s_{w,x_{w^*}}, \alpha_w \oplus x_{w^*})$ or $(s_{w,K_{w^*}}, \alpha_w \oplus K_{w^*})$, depending on whether it is a gate corresponding to x or K. Here w^* denotes the bit of the input corresponding to the w-th wire.
- If g_i is an input gate, then it consists of a single wire w, and we return α_w.
- Otherwise we compute and return the encoding table as specified in Theorem 5.5.

Our obfuscation consists of the output of $\tilde{K} \leftarrow \mathsf{PRO.Obf}(1^\lambda, C_{M_{\mathsf{PRF}},\eta,T}, (K, K^*))$, for a uniformly sampled $K^* \leftarrow \{0,1\}^\lambda$. To evaluate a circuit on input $x \in \{0,1\}^\eta$, we simply recompute

$$\tilde{E} \leftarrow (\mathsf{PRO.Eval}(C_{M_{\mathsf{PRF}},\eta,T}, \tilde{K}, (1, x)), \ldots, \mathsf{PRO.Eval}(C_{M_{\mathsf{PRF}},\eta,T}, \tilde{K}, (\mathrm{Circuit}(\eta, T), x)))$$

and output $\mathrm{RE.Dec}(\tilde{E})$. The next theorem establishes the pseudorandomness of the construction.

Theorem 6.3. *If the PRF schemes and the PRO scheme are pseudorandom and RE scheme is blind, all against distinguishers running in time polynomial in 2^η, then the obfuscation scheme as described above satisfies pseudorandomness.*

Proof. The proof follows routinely by invoking (in this order) the pseudorandomness of PRF and PRF^*, the blindness of the RE scheme, and the pseudorandomness of PRO. Since the truth-table of the obfuscated circuit is bounded by a polynomial in 2^η and the security parameter, this is also an upper bound in the runtime of the reductions. □

As for efficiency, note that the size of the obfuscated circuit $C_{M_{\mathsf{PRF}},\eta,T}$ only depends on $|M_{\mathsf{PRF}}|$ (in terms of the circuit-size of the transition function), η, and λ, since all operations performed by the circuit concern single gates of the computation, and are otherwise independent on the runtime T (except for a factor of $\log(T)$ in the input size which is anyway absorbed in the λ factor). Thus, we can bound the size of the obfuscated program, as well as the *circuit-size* of the obfuscator by

$$|\tilde{K}| = \left|\mathsf{PRO.Obf}^{\mathrm{TM}}\right| = \mathsf{poly}(\lambda, |M_{\mathsf{PRF}}|, \eta).$$

Plugging in the PRO for circuits constructed in Section 6.1, we can slightly improve the size of the obfuscated program to $|\tilde{K}| = \mathsf{poly}(\lambda, \mathsf{depth}(M_{\mathsf{PRF}}), \eta)$, where $\mathsf{depth}(M_{\mathsf{PRF}})$ is the depth of the circuit implementing the transition function of M_{PRF}.

7 Applications

We show how PRO allows us to construct a number of foundational cryptographic primitives, sometimes allowing us to bypass well-known limitations of indistinguishability obfuscation. Section 7.1 constructs an (unleveled) FHE scheme and Section 7.2 constructs a succinct randomized encoding scheme, both from iPRO. Section 7.3 constructs a succinct witness encryption scheme, this time assuming the stronger notion of PRO.

7.1 Fully Homomorphic Encryption

We show how PRO combined with the GSW (leveled) homomorphic encryption scheme allows us to construct fully-homomorphic encryption (FHE) without resorting to any circularity assumption, by instantiating the template from [CLTV15]. For simplicity, we will describe the secret key version of the protocol. One can then use the techniques of Rothblum [Rot11] to upgrade this scheme to a public-key FHE scheme.

Recall that in the GSW encryption, the public evaluation key for depth-d evaluation consists of d components

$$\mathbf{p} = (\mathbf{p}_1, \ldots, \mathbf{p}_d)$$

where the i-th component of the public key consists of an encryption of the $(i-1)$-th private key component $\mathbf{s}_i$ under the i-th private key component $\mathbf{s}_{i+1}$. In a slight abuse of notation, we denote by

$$\mathbf{p}_i \leftarrow \mathsf{Enc}_{\mathbf{s}_{i+1}}(\mathbf{s}_i)$$

the bit by bit encryption of $\mathbf{s}_i$ under key $\mathbf{s}_{i+1}$. We shall think of the index i as ranging over $[\ell]$ where $\ell = 2^{\log^2(\lambda)}$; that is, ranging over a slightly superpolynomial domain. Following the approach from [CLTV15], we will define a circuit that, intuitively, computes $\mathbf{p}_i$ given i. The FHE public key pk will be an obfuscation of this circuit. The only difference from [CLTV15] is that we use a iPRO to perform this obfuscation rather than an iO. We now describe the construction in more detail.

Construction. Let PRF be a subexponentially secure puncturable pseudorandom function. Let $\mathsf{KeyGen}(\mathsf{sk}, i)$ be a circuit that has sk hardcoded and takes as input $i \in [\ell(\lambda)]$ where $\ell(\lambda) = 2^{\log^2 \lambda}$, and performs the following operations.

- Compute $\mathbf{s}_i = \mathsf{GSW.Gen}(1^\lambda; \mathsf{PRF}(\mathsf{sk}, i))$ and $\mathbf{s}_{i+1} = \mathsf{GSW.Gen}(1^\lambda, \mathsf{PRF}(\mathsf{sk}, i+1))$ and $r_i = \mathsf{PRF}(\mathsf{sk}, (\mathrm{rand}, i))$, where rand is a label (i.e., an arbitrary fixed constant).
- Output $\mathsf{Enc}_{\mathbf{s}_{i+1}}(\mathbf{s}_i; r_i)$.

Now, we construct the new FHE scheme as follows.

- $\mathsf{FHE.Gen}(1^\lambda)$: Sample the secret key as $\mathsf{sk} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$, and the public key as $\mathsf{ek} \leftarrow \mathsf{iPRO.Obf}(1^\lambda, \mathsf{KeyGen}(\mathsf{sk}, \cdot))$.
- $\mathsf{FHE.Enc}(\mathsf{sk}, m)$: Compute $\mathbf{s}_1 = \mathsf{GSW.Gen}(1^\lambda; \mathsf{PRF}(\mathsf{sk}, 1))$. Output $\mathsf{GSW.Enc}_{\mathbf{s}_1}(m)$.
- $\mathsf{FHE.Eval}(\mathsf{ek}, C, (c_1, \ldots, c_\eta))$: Proceed just as in GSW homomorphic evaluation, but derive evaluation keys $\mathbf{p}_i$ via $\mathbf{p}_i \leftarrow \mathsf{iPRO.Eval}(C, \mathsf{ek}, i)$ when needed.
- $\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{ct}, i)$: Compute $\mathbf{s}_i = \mathsf{GSW.Gen}(1^\lambda; \mathsf{PRF}(\mathsf{sk}, i))$ and output $\mathsf{GSW.Dec}(\mathbf{s}_i, \mathsf{ct})$.

Note that, analogously to [CLTV15], we assume without loss of generality that the decryption circuit takes as input the depth i of the evaluation circuit. (One can always include this depth as an additional component of the ciphertext.)

Correctness and security. Correctness of the scheme follows directly from the correctness of GSW and the correctness of the PRO scheme.

To show semantic security, it suffices to show that the following two hybrids are indistinguishable.

- Hybrid $\mathcal{H}_\ell$: Sample $\mathsf{ek} \leftarrow \mathsf{iPRO.Obf}(1^\lambda, \mathsf{KeyGen}(\mathsf{sk}, \cdot))$ and set

  $$\mathbf{s}_1 := \mathsf{GSW.Gen}(1^\lambda, \mathsf{PRF}(\mathsf{sk}, 1))$$

  Output $(\mathsf{ek}, \mathbf{s}_1)$.
- Hybrid $\mathcal{H}_0$: Consider the program $f_{\mathsf{sk}'}$ which works for:

  $$f_{\mathsf{sk}'}(j) = \mathsf{PRF.Eval}(\mathsf{sk}', j)$$

  Set $\mathsf{ek}' \leftarrow \mathsf{iPRO.Obf}(1^\lambda, f_{\mathsf{sk}'})$ and $\mathbf{s}_1 := \mathsf{GSW.Gen}(1^\lambda, \mathsf{PRF}(\mathsf{sk}, 1))$. Output $(\mathsf{ek}', \mathbf{s}_1)$.

Since ek' in $\mathcal{H}_0$ is independent of $\mathbf{s}_1$, one can invoke GSW semantic security with respect $\mathbf{s}_1$. In particular, this must also mean that semantic security holds in $\mathcal{H}_\ell$, i.e. the construction.

To show $\mathcal{H}_\ell$ and $\mathcal{H}_0$ are indistinguishable, we follow the argument of [CLTV15] and puncture the truth-table one input at a time (the only difference is that we require that the truth-table to additionally be pseudorandom). We will also use the fact that for all $k \in [\ell]$,

$$\mathbf{s}_1, \mathsf{Enc}_{\mathbf{s}_2}(\mathbf{s}_1), \ldots, \mathsf{Enc}_{\mathbf{s}_k}(\mathbf{s}_{k-1}) \approx_C \mathbf{s}_1, u_2, \ldots, u_k$$

for uniformly sampled strings u_i of appropriate length.

we define the following intermediate hybrid $\mathcal{H}_i$ and show that $\mathcal{H}_{i+1}$ and $\mathcal{H}_i$ are indistinguishable.:

- Hybrid $\mathcal{H}_i$: Consider the program $f_{\mathsf{sk},\mathsf{sk}'}$ which works for:

$$f_{\mathsf{sk},\mathsf{sk}',i}(j) = \begin{cases} \mathsf{KeyGen}(\mathsf{sk}, j) & \text{for } j < i \\ \mathsf{PRF.Eval}(\mathsf{sk}', j) & \text{for } j \geq i. \end{cases}$$

  Set $\mathsf{ek} \leftarrow \mathsf{iPRO.Obf}(1^\lambda, f_{\mathsf{sk},\mathsf{sk}',i})$ and $\mathbf{s}_1 \leftarrow \mathsf{GSW.Gen}(1^\lambda, \mathsf{PRF}(\mathsf{sk}, 1))$.

We show how to go from $\mathcal{H}_{i+1}$ to $\mathcal{H}_i$ with more intermediate hybrids.

- Hybrid $\mathcal{H}_{i+1,1}$: Puncture sk on points $r_i = (\mathrm{rand}, i)$ and $i+1$ to get key $\mathsf{sk}\{i\}$ (note the abuse of notation here - sk is punctured on two points defined by the index i). Let $\alpha = \mathsf{GSW.Gen}(1^\lambda; \mathsf{PRF}(\mathsf{sk}, i))$ and $\beta = \mathsf{PRF}(\mathsf{sk}, (\mathrm{rand}, i))$. Let $\mathsf{ct} = \mathsf{Enc}_\alpha(\mathbf{s}_j, \beta)$.

  Now, let

$$f_{\mathsf{sk}\{i\},\mathsf{sk}',i,\mathsf{ct}} = \begin{cases} \mathsf{KeyGen}(\mathsf{sk}\{i\}, j) & \text{for } j < i \\ \mathsf{ct} & \text{for } j = i \\ \mathsf{PRF}(\mathsf{sk}', j) & \text{for } j \geq i+1. \end{cases}$$

  Set $\mathbf{e}_k \leftarrow \mathsf{iPRO}(1^\lambda, f_{\mathsf{sk}\{i\},\mathsf{sk}',i,\mathsf{ct}})$ and $\mathbf{s}_1 \leftarrow \mathsf{GSW.Gen}(1^\lambda, \mathsf{PRF}(\mathsf{sk}, 1))$.

  This hybrid follows from the functionality equivalence of the and the pseudorandomness of the truth table of the obfuscated program.

- Hybrid $\mathcal{H}_{i+1,2}$: Replace α and β with uniformly random strings. This follows from the punctured key security of $\mathsf{sk}\{i\}$.

- Hybrid $\mathcal{H}_{i+1,3}$: Replace ct with a uniformly random string. This hybrid follows from the fact that GSW ciphertexts are pseudorandom.

- Hybrid $\mathcal{H}_{i+1,4}$: Puncture sk' on i to get key $\mathsf{sk}'\{i\}$, and unpuncture sk. Obfuscate the program

$$f_{\mathsf{sk},\mathsf{sk}'\{i\},i,\mathsf{ct}} = \begin{cases} \mathsf{KeyGen}(\mathsf{sk}, j) & \text{for } j < i \\ \mathsf{ct} & \text{for } j = i \\ \mathsf{PRF}(\mathsf{sk}'\{i\}, j) & \text{for } j \geq i+1. \end{cases}$$

  This hybrid follows from the functionality equivalence of the and the pseudorandomness of the truth table of the obfuscated program.

- Hybrid $\mathcal{H}_{i+1,5}$: Replace ct with $\mathsf{PRF}(\mathsf{sk}', i)$. This follows from the punctured security of $\mathsf{sk}'\{i\}$.
- Hybrid $\mathcal{H}_{i+1,6}$: Unpuncture the key sk' and obfuscated the program:

$$f_{\mathsf{sk},\mathsf{sk}',i}(j) = \begin{cases} \mathsf{KeyGen}(\mathsf{sk}, j) & \text{for } j < i \\ \mathsf{PRF.Eval}(\mathsf{sk}', j) & \text{for } j \geq i. \end{cases}$$

  This hybrid follows from the functionality equivalence of the and the pseudorandomness of the truth table of the obfuscated program.

Therefore, via $O(\ell)$ hybrids, we have that $\mathcal{H}_\ell \approx_c \mathcal{H}_0$.

7.2 Succinct Randomized Encodings

A randomized encoding (RE) allows one to delegate the computation of a function $F : \{0,1\}^{|x|} \to \{0,1\}$ on an input x, while not revealing anything but $F(x)$. (For convenience, here we consider only functions with single-bit outputs but the more general case can be easily achieved by parallel repetition.) A randomized encoding is *succinct* [BGL+15] if, when F is represented as a Turing machine TM, the encoding of (F, x) has size polynomial in $|\mathrm{TM}|$, $|x|$, and λ, and the runtime of the decoding algorithm is polynomial in t and λ, where t is the runtime of TM on input x. In this section, we show two ways to use pseudorandom obfuscation to realize succinct RE. First, we show that succinct RE is an almost-immediate application of our PRO for Turing machines, using a one-time pad trick. Second, we show that even iPRO suffices for succinct RE, by instantiating the succinct garbling construction of Ananth and Lombardi [AL18] with the blind garbling scheme of Brakerski, Lombardi, Segev and Vaikuntanathan [BLSV18] and then using the same trick.

7.2.1 Succinct Randomized Encodings from PRO

Construction. Consider the keyed function $\tilde{F}$ which on key (x, r), outputs the single value

$$\tilde{F}(x, r) = F(x) \oplus r.$$

Our succinct randomized encoding consists of the obfuscated program $\tilde{K} \leftarrow \mathsf{PRO.Obf}^{\mathsf{TM}}(1^\lambda, \tilde{F}, (x, r))$, along with the uniformly sampled $r \leftarrow \{0,1\}$. To recover the output, simply compute

$$\mathsf{PRO.Eval}^{\mathsf{TM}}(\tilde{F}, \tilde{K}) \oplus r$$

noting that the obfuscated program takes no additional input.

Correctness is immediate. For efficiency, the runtime of the encoder (for single-bit functions) and the size of the encoding is bounded by a polynomial in the size of the input $|x|$, the size of the Turing machine computing F, and the security parameter λ.

Security. For security, it suffices to show that there exists a polynomial time simulator Sim such that:

$$(\tilde{K}, r), f(x) \approx_c \mathsf{Sim}(1^\lambda, f(x)), f(x).$$

We first note that by construction

$$F(x) \oplus r, F(x) \approx_c u, F(x)$$

where u is uniformly sampled. Therefore, by PRO security, we have that

$$\tilde{K}, F(x) \approx_c \tilde{u}, F(x)$$

for a uniformly sampled $\tilde{u}$. We now define the simulator Sim as follows: On input $F(x)$,

- Sample a uniformly random string $\tilde{u}$ to be the length of $\tilde{K}$.
- Compute $r' \leftarrow \mathsf{PRO.Eval}^{\mathsf{TM}}(\tilde{F}, \tilde{u}) \oplus F(x)$.
- Output $(\tilde{u}, r')$.

Then, we have that:

$$\begin{aligned}(\tilde{K}, r), F(x) &=_s (\tilde{K}, \mathsf{PRO}^{\mathsf{TM}}(\tilde{F}, \tilde{K}) \oplus F(x)), F(x) \\ &\approx_c (\tilde{u}, \mathsf{PRO}^{\mathsf{TM}}(\tilde{F}, \tilde{u}) \oplus F(x)), F(x) \\ &=_s \mathsf{Sim}(1^\lambda, F(x)), F(x)\end{aligned}$$

where the first statistical equality follows from the correctness of the PRO scheme, the second computational indistinguishability follows from the PRO scheme, and the third statistical equality follows from the definition of Sim.

7.2.2 Succinct Randomized Encodings from iPRO

In this section, we construct a succinct randomized encoding for small-space Turing machines (e.g., with maximum space usage polynomial in the security parameter), with runtime bounded by a polynomial that is known in advance.[8] Our encoding relies on the succinct garbling construction of [AL18], which upgrades a locally-simulatable garbling scheme to a succinct garbling scheme. In addition to iPRO, we will need a blind, locally simulatable garbling scheme, as guaranteed by the following. (We refer the reader to [BLSV18] and [AL18] for the definitions of blindness and local simulatability, respectively.) We assume subexponential security of all primitives.

Theorem 7.1. *Assume (subexponentially-secure) one-way functions exist. Then there exists a (subexponentially-secure), blind, (L_{sim}, L_{inp})-locally-simulatable garbling scheme with $L_{sim}(\lambda, C) = w \cdot \mathsf{poly}(\lambda)$ and $L_{inp}(\lambda, n, m) = \mathsf{poly}(\lambda, n)$, where w is the width of C.*

We will not give a detailed proof here. The scheme is the BLSV blind garbling scheme of [BLSV18], which can be viewed as a slight modification of Yao's garbling scheme. Indeed, the proof given in [AL18] that Yao's garbling scheme is locally-simulatable with the above parameters goes through for the BSLV scheme with only minor modifications.

Let $\mathsf{succGC} = (\mathsf{Setup}, \mathsf{TMEncode}, \mathsf{InpEncode}, \mathsf{Eval})$ be the bounded-runtime succinct garbling construction of Ananth and Lombardi [AL18] instantiated with the locally simulatable garbling scheme Π_{LGC} of Theorem 7.1. We briefly recall the definition of TMEncode and refer the reader to [AL18] for details. Let $C_{M,T}$ be the circuit that computes T steps of M on inputs of the appropriate

[8] We suspect that it is possible to remove both these restrictions and construct a succinct randomized encoding for arbitrary polynomial-time Turing machines. In particular, we suspect that one can remove the small-space restriction by showing that a variant of the Ananth and Lombardi locally-simulatable garbling scheme is blind. And we suspect that one can remove the bounded polynomial runtime restriction by adapting the techniques of Ananth and Lombardi for removing the same restriction in the context of garbling schemes.

length. Let H be a circuit that on input a gate index g of $C_{M,T}$, outputs the corresponding gate of $\Pi_{LGC}.\mathsf{Garble}(\mathsf{gsk}, C_{M,T})$.[9] On input a secret key gsk, time bound T, and T-time Turing machine M, TMEncode outputs $(T, \mathsf{iO}(H))$. In this section, we will consider a modified version of the scheme where TMEncode outputs $(T, \mathsf{iPRO}(H))$ instead of $(T, \mathsf{iO}(H))$. Henceforth we will use $\mathsf{succGC} = (\mathsf{Setup}, \mathsf{TMEncode}, \mathsf{InpEncode}, \mathsf{Eval})$ to refer to the modified scheme. Since we are in the context of randomized encodings, where the input is encoded together with the circuit, we can assume without loss of generality that the input is hardwired into the circuit, and ignore the input encoding algorithm InpEncode.

For $r \in \{0,1\}$, we denote by $M^{(r)}$ a TM such that for all inputs x, $M^{(r)}(x) = M(x) \oplus r$. Given a secret key gsk and time bound T, for $r \in \{0,1\}$, we denote by (T, F_r) the output of TMEncode on input gsk, T, and $M^{(r)}$. We view $\{F_r\}_r$ as a two-element keyed function family.

Construction. On input a Turing machine M and time bound T,[10] our succinct randomized encoding algorithm behaves as follows. It first samples gsk using $\mathsf{Setup}(1^\lambda)$, and samples $r \leftarrow \{0,1\}$. Then it runs TMEncode on input gsk, T, and $M^{(r)}$ to obtain F_r. Then, it computes $\tilde{K} = \mathsf{iPRO.Obf}(\{F_r\}_r, r)$. Finally, it outputs $\tilde{K}$ and r.

On input an encoding $\tilde{K}, r$, the decoding algorithm first uses $\tilde{K}$ to compute the garbling $\langle C^{(r)} \rangle = (\mathsf{iPRO.Eval}(\tilde{K}, g))_g$ of $C^{(r)}$ gate by gate. It then outputs $\Pi_{LGC}.\mathsf{Eval}(\langle C^r \rangle)$.

Correctness is again immediate from the correctness of iPRO and Π_{LGC}. Efficiency follows from the efficiency of succGC, iPRO and Π_{LGC}.

Security. Again, it suffices to show that there exists a polynomial-time simulator Sim such that

$$\tilde{K}, r \approx_c \mathsf{Sim}(1^\lambda, M(x)) \ .$$

We show this by exhibiting a sequence of hybrids such that the final hybrid is efficiently computable given only $M(x)$. Below, all indistinguishabilities hold even against adversaries running in time $\mathsf{poly}(|\mathtt{TT}(F_r)|)$.

Using correctness, we first rewrite $\tilde{K}, r$ as

$$\tilde{K}, r = \tilde{K}, M(x) \oplus \Pi_{LGC}.\mathsf{Eval}((\mathsf{iPRO.Eval}(\tilde{K}, g))_g \tag{7.1}$$

Note that this is efficiently computable from $M(x)$ and $\tilde{K}$, so it remains to show that $\tilde{K}, M(x) \approx_c Y, M(x)$, where Y is efficiently computable from $M(x)$.

As a first step, we show iPRO security can be applied to $\tilde{K}$ in *Eq.* (7.1); that is, we show that $\mathtt{TT}(F_r)$ is pseudorandom given $M(x)$. Indeed, we have

$$M(x), \mathtt{TT}(F_r) = M(x), \Pi_{LGC}.\mathsf{Garble}(\mathsf{gsk}, C^{(r)}) \approx_c M(x), u,$$

because Π_{LGC} is blind, and the output $M(x) \oplus r$ of $C^{(r)}$ is uniformly random given $M(x)$. Having established this, we follow the security proof of Ananth and Lombardi [AL18]. Their security proof consists of showing the indistinguishability of a sequence of hybrids $\mathcal{H}_0, \ldots, \mathcal{H}_N$ for $N \leq \mathsf{poly}(\lambda)$, where each hybrid $\mathcal{H}_i$ has the form[11] $\mathsf{iO}(H_i)$, $H_0 = \tilde{K}$, and H_N is efficiently computable from $M(x)$.

[9] This is possible due to local simulatability. In general, the outputs may be local components of polynomial size, but the notion of "gate" is more intuitive. (And, in the scheme Π_{LGC} the local components do correspond to gates.)

[10] Recall that we assume WLOG that the input is hardwired into M.

[11] In the security proof of [AL18] there is also an input encoding I_i, but for us this is empty, as we have assumed without loss of generality that the input is hardwired into the circuit that we are encoding. We are also ignoring the time bound T.

One can efficiently compute $M(x)$ from each $\mathcal{H}_i$, so we can equivalently write each hybrid in the form

$$\mathsf{iO}(H_i), M(x) \ .$$

so that, if iO is replaced with iPRO, then $\mathcal{H}_0$ is exactly the left-hand side of the desired claim, namely $\tilde{K}, M(x) \approx_c Y, M(x)$, and $\mathcal{H}_N$ matches the right-hand side of the desired claim. Thus, it suffices to argue the indistinguishability of a sequence of hybrids $\mathcal{H}'_0, \ldots, \mathcal{H}'_N$ that are identical to $\mathcal{H}_0, \ldots, \mathcal{H}_N$ except that iO is replaced with iPRO. We go by the following inductive meta-argument. We claim that for all $0 \leq i \leq N$, we have $\mathcal{H}'_0 \approx_c \mathcal{H}'_i$, even against adversaries running in time $\mathsf{poly}(|C_{M,T}|)$. Clearly $\mathcal{H}'_0 = \mathcal{H}'_0$. Now, suppose that $\mathcal{H}'_0 \approx_c \mathcal{H}'_i$. There are two cases. In the first case, the Ananth and Lombardi proof that $\mathcal{H}_i \approx_c \mathcal{H}_{i+1}$ is not an application of iO security, in which case the same proof shows that $\mathcal{H}_i \approx_c \mathcal{H}_{i+1}$. In the second case, it is an application of iO security, and we claim that iPRO security suffices instead. That is, we claim that in $\mathcal{H}'_i$, $\mathtt{TT}(H_i)$ is pseudorandom. But this follows because $\mathcal{H}_0 \approx_c \mathcal{H}_i$ holds even against adversaries running in time polynomial in $\mathtt{TT}(H_i)$. Noting that we can set the parameters of the (subexponential) iPRO and Π_{LGC} primitives sufficiently large so that even after a polynomial-factor loss at each of the $N = \mathsf{poly}(\lambda)$ hybrids, we still obtain $\mathcal{H}'_0 \approx_c \mathcal{H}'_N$, completes the proof.

7.3 Succinct Witness Encryption

It is an easy exercise to construct a witness encryption [GGSW13] (without full succinctness) given a dPRO scheme. For a given relation $\mathcal{R}$ and a statement x and a message m, define the circuit $C_{\mathcal{R},x}(K, m, w)$ as follows.

- If $\mathcal{R}(x, w) = 1$, return m.
- Otherwise return $\mathsf{PRF}(K, (x, w))$.

The witness encryption for m is then defined to be $\mathsf{PRO.Obf}(1^\lambda, C_{\mathcal{R},x}, (K, m))$ for a uniformly sampled $K \leftarrow \{0,1\}^\lambda$.

If the relation is not satisfiable, then there exists no valid witness, and therefore the truth-table of the obfuscated circuit is pseudorandom. In other words, we have that

$$\mathsf{ct} = \mathsf{WE.Enc}(1^\lambda, \mathcal{R}, x, m) = \mathsf{PRO.Obf}(1^\lambda, C_{\mathcal{R},x}, (K, m)) \approx_c \left\{u : u \leftarrow \{0,1\}^{|\mathsf{ct}|}\right\}$$

since

$$\{\mathsf{PRF}(K, (x, w))\}_w \approx_c \left\{u_w : u_w \leftarrow \{0,1\}^\lambda\right\}_w$$

for a uniformly sampled K. Note that for this to be the case, it must hold that λ is set in such a way that the pseudorandomness of PRF and PRO holds against distinguishers running in time $2^{|w|}$, and in particular $\lambda > |w|$. Thus the size of the ciphertext, which is $|\mathsf{PRO.Obf}(1^\lambda, C_{\mathcal{R},x}, (K, m))|$, is bounded by $\mathsf{poly}(\lambda, \mathsf{depth}(\mathcal{R}(x, \cdot)), |w|)$ where $\mathsf{depth}(\mathcal{R}(x, \cdot))$ is the depth of the circuit checking the relation $\mathcal{R}$. Since all NP-relations can be checked in poly-logarithmic depth, we can assume without loss of generality that the bound is actually $\mathsf{poly}(\lambda, |w|)$.

As an alternative to the above scheme, another option is to use a (non-doubly) PRO-obfuscation instead, combined with a compute-and-compare obfuscation [WZ17, GKW17]: The PRO obfuscation guarantees semantic security, whereas the compute-and-compare obfuscation makes the ciphertext computationally indistinguishable from random bits. A third option is to use an iO-based

witness encryption and combine it with compute-and-compare obfuscation. Both of the above alternatives would result into slightly worse parameters (specifically, the size of the ciphertext would now depend on $|x|$).

Construction. Now, we show how to remove the dependence on the size of the witness, thus obtaining a *succinct* witness encryption. The main idea is, in essence, to obfuscate a Turing machine (described in Section 6.2) which takes no input and generates a witness encryption ciphertext as in the above naive scheme.

Let $W = W(\lambda)$ be a polynomial upper-bound on the witness length. Let λ_W be the security parameter for the witness encryption described above with witness length $|w| = W$, and let $p_W = p_W(|x|, W, \lambda)$ by a bound on the the number of random bits used be the scheme. Let $G : \{0,1\}^\lambda \to \{0,1\}^{p_W}$ be a uniformly generatable PRG (note that this property can be achieved by instantiating it through PRFs). Consider the following Turing-machine $M_{\mathcal{R},x}$.

- On input a seed $s \in \{0,1\}^\lambda$, and a message m proceed as follows.
- Compute $r \leftarrow G(s) \in \{0,1\}^{p_W}$.
- Return $\mathsf{WE.Enc}(1^{\lambda_W}, \mathcal{R}, x, m; r)$.

We then set the ciphertext to be the obfuscation $\tilde{K} \leftarrow \mathsf{PRO.Obf}^{\mathsf{TM}}(1^\lambda, M_{\mathcal{R},x}, (s,m))$ for a uniformly sampled $s \leftarrow \{0,1\}^\lambda$. The decryption algorithm simply evaluates the obfuscated program to retrieve a witness encryption ciphertext, which can be then decrypted using the witness. Note that, by the efficiency of the PRO for Turing machine, we have that the runtime of the obfuscation procedure is bounded by

$$\left|\mathsf{PRO.Obf}^{\mathsf{TM}}(1^\lambda, M_{\mathcal{R},x}, (s,m))\right| = \mathsf{poly}(\lambda, |\mathcal{R}|, |x|, \log(W), |s|, |m|) \leq \mathsf{poly}(\lambda, |\mathcal{R}|, |x|)$$

whereas the size of the obfuscated program is bounded by a polynomial in the security parameter, in the size of the input $(|s|, |m|)$, and in the depth of the transition function of the underlying Turing machine. The transition function consists of (i) the evaluation of G and (ii) the computation of a witness encryption ciphertext. The depth of the former computation can be made poly-logarithmic in p_W (and thus bounded by some polynomial in λ) by using a low-depth pseudorandom function [BPR12], whereas the depth of the latter is also bounded by $\mathsf{depth}(\mathcal{R}(x,\cdot)) = \mathsf{poly}(\lambda)$, as discussed above. Thus, the size of the obfuscated program is bounded by some fixed $\mathsf{poly}(\lambda, |s|, |m|) = \mathsf{poly}(\lambda)$, as desired.

Security. To argue that the scheme is secure, we first argue that for all false statements x it holds that

$$\left\{\mathsf{WE.Enc}(1^{\lambda_W}, \mathcal{R}, x, m; z) : z \leftarrow \{0,1\}^{p_W}\right\} \approx_c \left\{u : u \leftarrow \{0,1\}^{|\mathsf{ct}|}\right\}$$

against all distinguishers running in time polynomial in 2^W, by the security of the base witness encryption scheme. Clearly, the same holds for all distinguishers running in time polynomial in λ, since $\lambda < \lambda_W$. We now consider distinguishers running in polynomial time (in λ) and we argue

that

$$\left\{\mathsf{WE.Enc}(1^{\lambda_W}, \mathcal{R}, x, m; G(s))\right\} \approx_c \left\{\mathsf{WE.Enc}(1^{\lambda_W}, \mathcal{R}, x, m; z) : z \leftarrow \{0,1\}^{p_W}\right\}$$
$$\approx_c \left\{u : u \leftarrow \{0,1\}^{|\mathsf{ct}|}\right\}$$

where the first implication holds by the security of G and the second implication was discussed above. We can invoke the pseudorandomness of $\mathsf{PRO}^{\mathsf{TM}}$ to conclude that $\tilde{K}$ is computationally indistinguishable from an encryption of 0.

8 Indistinguishability Obfuscation

In this section, we show how to use iPRO to construct general-purpose iO, in conjunction with other assumptions. We begin by recalling some basic definitions.

8.1 Definitions

We recall the definition of iO [BGI+01].

Definition 8.1 (Indistinguishability Obfuscation)**.** An iO scheme for a class of circuits $C : \{0,1\}^\eta \to \{0,1\}$ consists of two PPT algorithm $(\mathsf{Obf}, \mathsf{Eval})$ with the following syntax.

- $\mathsf{Obf}(1^\lambda, C)$: On input the security parameter 1^λ and a circuit C, the probabilistic obfuscation algorithm returns an obfuscated circuit $\tilde{C}$.
- $\mathsf{Eval}(\tilde{C}, x)$: On input an obfuscated circuit $\tilde{C}$ and an index x, the deterministic evaluation algorithm returns an output y.

We require the following properties to hold.

- (ε-Correctness) For all $\lambda \in \mathbb{N}$ and all circuits C, it holds that

  $$\Pr\left[\forall\, x \in \{0,1\}^\eta : C(x) = \mathsf{Eval}(\mathsf{Obf}(1^\lambda, C), x)\right] \geq 1 - \varepsilon$$

  where the probability is taken over the random coins of Obf.
- (Indistinguishability) For all $\lambda \in \mathbb{N}$ and all pairs of functionally-equivalent circuits $C_0 \equiv C_1$, it holds that

  $$\mathsf{Obf}(1^\lambda, C_0) \approx_c \mathsf{Obf}(1^\lambda, C_1).$$

In [LPST16] it is shown that the existence of iO is implied (along with the subexponential LWE assumption) by the notion of *exponentially efficient* iO, denoted by xiO. An xiO $(\mathsf{xiO.Obf}, \mathsf{xiO.Eval})$ scheme is defined identically as above, except that we allow the obfuscation algorithm to run in time polynomial in the size of the truth table (2^η) of C, and we only impose the following requirements:

- (Non-Trivial Efficiency) We require that $|\tilde{C}| \leq 2^{\eta(1-\delta)}$ for some constant $\delta > 0$.
- (Shallowness) We require that $\mathsf{depth}(\mathsf{xiO.Obf}) = \mathsf{poly}(\lambda, |C|)$.

8.2 Bilinear Pairings and Quadratic Functional Encryption

Let $\mathcal{G}$ be a (prime-order) bilinear group generator, that is, $\mathcal{G}$ is an algorithm that takes as an input a security parameter λ and outputs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ is the description of three multiplicative cyclic group, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable bilinear pairing, p is the order of the group which is always a prime number, and g_1, g_2 are generators of the groups $\mathbb{G}_1$, $\mathbb{G}_2$ respectively. Additionally, we denote by $g_t = e(g_1, g_2)$. In this work we assume a bilinear group where elements sampled uniformly at random are indistinguishable from random bitstrings. In other words we require that

$$\{g_2 : g_2 \leftarrow \mathbb{G}_2\} \approx_c \{u : u \leftarrow \{0,1\}^*\}\,.$$

While this is a non-standard property, we remark that examples of such groups are known in the literature [BHKL13, Tib14].

The SXDH assumption postulates that the following distributions are computationally indistinguishable

$$(g_1, g_2, g_b^x, g_b^y, g_b^{xy}) \approx_c (g_1, g_2, g_b^x, g_b^y, g_b^z) \quad \text{for any } b \in \{1,2\}$$

where $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$ and $x, y, z \leftarrow \mathbb{Z}_p$.

We recall the definition of quadratic functional encryption (FE), specialized to the case of bilinear groups and with a very weak security notion, that is however sufficient for our purposes.

Definition 8.2 (Quadratic Functional Encryption)**.** A quadratic FE scheme consists of four PPT algorithm $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax.

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, it outputs a pair of master public and secret keys $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{KeyGen}(\mathsf{msk}, f)$: On input a master secret key msk and a quadratic function $f : \mathbb{Z}_p^n \times \mathbb{Z}_p^n \to \mathbb{Z}_p$, it outputs a functional secret key sk_f.
- $\mathsf{Enc}(\mathsf{mpk}, m)$: On input a master public key mpk and a message $m \in \mathbb{Z}_p^n \times \mathbb{Z}_p^n$, outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$: On input a functional secret key sk_f and a ciphertext ct, returns a group element in g_t.

We require the following properties to hold.

- (Correctness) For all $\lambda \in \mathbb{N}$, all $f : \mathbb{Z}_p^n \times \mathbb{Z}_p^n \to \mathbb{Z}_p$, and all $m \in \mathbb{Z}_p^n \times \mathbb{Z}_p^n$, it holds that

  $$\Pr\left[\mathsf{Dec}(\mathsf{KeyGen}(\mathsf{msk}, f), \mathsf{Enc}(\mathsf{mpk}, m)) = g_t^{f(m)}\right] = 1$$

  where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
- (Succinctness) We require that $|\mathsf{mpk}| = \mathsf{poly}(\lambda)$, $|\mathsf{sk}_f| = \mathsf{poly}(\lambda)$, and $|\mathsf{ct}| = O(n) \cdot \mathsf{poly}(\lambda)$.
- (Simulation Security) There exists a simulator Sim such that for all $\lambda \in \mathbb{N}$, all $q \in \mathsf{poly}(\lambda)$, all quadratic functions $\{f_i\}_{i \in \{1,\ldots,q\}}$, and all messages $m \in \mathbb{Z}_p^n \times \mathbb{Z}_p^n$, it holds that the following distributions are computationally indistinguishable:

  $$\left(\mathsf{mpk}, \mathsf{ct}, \{\mathsf{sk}_{f_i}\}_{i \in \{1,\ldots,q\}}\right) \approx_c \mathsf{Sim}\left(1^\lambda, \left\{g_2^{f_i(m)}\right\}_{i \in \{1,\ldots,q\}}\right)$$

where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{sk}_{f_i} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_i)$, and $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, m)$

A quadratic FE satisfying the above syntax and security can be constructed from the DLIN assumption in asymmetric bilinear groups [Wee20].

8.3 Construction of xiO

We present our construction of xiO scheme $(\mathsf{xiO.Obf}, \mathsf{xiO.Eval})$ in the following. We consider any circuit $C : \{0,1\}^\eta \to \{0,1\}$ and, for notational convenience, we assume that η is divisible by 3 and we denote by $n = 2^{\eta/3}$. In addition to a bilinear group and a quadratic FE as defined above, we will use an xiPRO scheme $(\mathsf{xPRO.Obf}, \mathsf{xPRO.Eval})$ with *indistinguishable security*.

- $\mathsf{xiO.Obf}(1^\lambda, C)$:
  - Sample a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2) \leftarrow \mathcal{G}(1^\lambda)$.
  - Sample a key pair $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
  - Sample n vectors $(\mathbf{x}_i, \mathbf{y}_i) \leftarrow \mathbb{Z}_p^n \times \mathbb{Z}_p^n$.
  - Compute $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{mpk}, (\mathbf{x}_i, \mathbf{y}_i))$.
  - For $j \in \{1, \ldots, n^2\}$, let f_j be the j-th unit vector. I.e., f_j is the quadratic function that returns the j-th monomial of $\mathbf{x} \otimes \mathbf{y}$. Compute $\mathsf{sk}_{f_j} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_j)$.
  - Let $C_i : (\mathbb{Z}_p^n \times \mathbb{Z}_p^n \times \{0,1\}^{|C|}) \times \{0,1\}^{2\eta/3} \to \mathbb{G}_2$ for $i \in \{0,1\}^{\eta/3}$ be the circuit defined as follows:
    - It is keyed by the vectors $(\mathbf{x}_i, \mathbf{y}_i)$ and the circuit C, which is hardwired.
    - Takes as input some $j \in \{1, \ldots, n^2\}$, and defines $x = (i, j)$.
    - Returns $g_2^{C(x)} \cdot g_2^{f_j(\mathbf{x}_i, \mathbf{y}_i)}$.
  - For all $i \in \{1, \ldots, n\}$, obfuscate $\tilde{K}_i \leftarrow \mathsf{xPRO.Obf}(1^\lambda, C_i, (\mathbf{x}_i, \mathbf{y}_i, C))$.
  - Output $\tilde{C} = (\{\tilde{K}_i, \mathsf{ct}_i\}_i, \{\mathsf{sk}_j\}_j)$.
- $\mathsf{xiO.Eval}(\tilde{C}, x)$:
  - Denote by (i, j) be the decomposition of x, where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, n^2\}$.
  - The output bit is defined to be 0 if

$$e\left(g_1, \mathsf{xPRO.Eval}(C_i, \tilde{K}_i, j)\right) = \mathsf{Dec}(\mathsf{sk}_{f_j}, \mathsf{ct}_i)$$

    and 1 otherwise.

The scheme is correct since, by the correctness of the xiPRO scheme and the quadratic FE we have that for all $(i, j) = x \in \{0,1\}^\eta$, it holds that

$$\begin{aligned} e\left(g_1, \mathsf{xPRO.Eval}(C_i, \tilde{K}_i, j)\right) &= e\left(g_1, g_2^{C(x)} \cdot g_2^{f_j(\mathbf{x}_i, \mathbf{y}_i)}\right) \\ &= g_t^{C(x)} \cdot g_t^{f_j(\mathbf{x}_i, \mathbf{y}_i)} \\ &= g_t^{C(x)} \cdot \mathsf{Dec}(\mathsf{sk}_{f_j}, \mathsf{ct}_i). \end{aligned}$$

Furthermore, the scheme is compact since

$$\begin{aligned}|\tilde{C}| &= n|\tilde{K}| + n|\mathsf{ct}| + n^2|\mathsf{sk}_f| \\ &= 2^{\eta/3} \cdot (2^{2\eta/3})^{2/3} \cdot \mathsf{poly}(\lambda) + 2^{\eta/3} \cdot 2^{\eta/3} \cdot \mathsf{poly}(\lambda) + 2^{2\eta/3} \cdot \mathsf{poly}(\lambda) \\ &= 2^{7/9\eta} \cdot \mathsf{poly}(\lambda)\end{aligned}$$

assuming that the xiPRO has a compression factor of $2/3$ (like the one built in Section 4). We are now ready to show that the construction satisfies the standard notion of indistinguishability.

Theorem 8.3. *Assume that* $(\mathsf{xPRO.Obf}, \mathsf{xPRO.Eval})$ *is a pseudorandom xiPRO scheme,* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is a secure quadratic FE, and SXDH is hard. Then* $(\mathsf{xiO.Obf}, \mathsf{xiO.Eval})$ *satisfies indistinguishability.*

Proof. We begin with an obfuscation of the circuit C_0 and we gradually change it to an obfuscation of some C_1, where $C_1 \equiv C_0$, in a series of hybrid distributions. In the i-th hybrid, we switch $\tilde{K}_i$ to hardwire C_1 (denote this by $\tilde{K}_i^{(1)}$) instead of C_0 (denote this by $\tilde{K}_i^{(0)}$). Since this is the only place where the circuit is used, if we can show that each hybrid is computationally indistinguishable from the previous distribution, then we complete the proof. Fix any index i, it suffices to show that

$$C_0, C_1, \tilde{K}_i^{(0)}, \mathsf{ct}_i, \{\mathsf{sk}_j\}_j \approx_c C_0, C_1, \tilde{K}_i^{(1)}, \mathsf{ct}_i, \{\mathsf{sk}_j\}_j$$

since all other elements $\tilde{K}_{\neq i}$ and $\mathsf{ct}_{\neq i}$ are publicly and efficiently computable given the above variables and therefore their presence cannot increase the advantage of the distinguisher. By the security of the quadratic FE, we know that

$$C_0, C_1, \tilde{K}_i^{(b)}, \mathsf{ct}_i, \{\mathsf{sk}_j\}_j \approx_c C_0, C_1, \tilde{K}_i^{(b)}, \mathsf{Sim}\left(1^\lambda, \left\{g_2^{f_j(\mathbf{x}_i, \mathbf{y}_i)}\right\}_j\right)$$

for both $b \in \{0, 1\}$. Now, observe that the output of the simulator is publicly and efficiently computable given $\tilde{K}_i^{(b)}$ and C_b: We can simply evaluate the obfuscation $\tilde{K}_i^{(b)}$ on all $j \in \{1, \ldots, n^2\}$ to obtain as an output

$$g_2^{C_b(x)} \cdot g_2^{f_j(\mathbf{x}_i, \mathbf{y}_i)} \leftarrow \mathsf{xPRO.Eval}(C_i, \tilde{K}_i^{(b)}, j)$$

where $x = (i, j)$. Dividing by $g_2^{C_b(x)} = g_2^{C_0(x)} = g_2^{C_1(x)}$, we obtain all $\{g_2^{f_j(\mathbf{x}_i, \mathbf{y}_i)}\}_j$, which we can feed as input to the simulator Sim, which is by assumption also efficiently computable. Once again, since this is an efficient computation that can be done publicly on $\tilde{K}_i^{(b)}$, it cannot increase the distinguishing advantage of any efficient distinguisher, and therefore we can without loss of generality prove that

$$C_0, C_1, \tilde{K}_i^{(0)} \approx_c C_0, C_1, \tilde{K}_i^{(1)}$$

to obtain the desired implication. Note that the two circuits are functionally equivalent, and thus by the pseudorandomness of xiPRO, it suffices to show that the function table

$$\left\{\mathsf{xPRO.Eval}(C_i, \tilde{K}_i^{(b)}, j)\right\}_j = \left\{g_2^{C_b(x)} \cdot g_2^{f_j(\mathbf{x}_i, \mathbf{y}_i)}\right\}_j \approx_c \{u_j : u_j \leftarrow \mathbb{G}_2\}_j$$

is computationally indistinguishable from uniform. By the SXDH assumption, we have that

$$\left\{g_2^{C_b(x)} \cdot g_2^{f_j(\mathbf{x}_i, \mathbf{y}_i)}\right\}_j \approx_c \left\{g_2^{C_b(x)} \cdot g_2^{z_j}\right\}_j$$

where $z_j \leftarrow \mathbb{Z}_p$, since $f_j(\mathbf{x}_i, \mathbf{y}_i)$ is the j-th monomial of $\mathbf{x}_i \otimes \mathbf{y}_i$. The proof is concluded by observing that the RHS is uniformly distributed in $\mathbb{G}_2$ and therefore indistinguishable from uniform. □

8.4 Construction of iO in the Pseudorandom Oracle Model

The pseudorandom oracle model (PROM) is a recently introduced model [JLLW23] which captures all the properties of a random oracle, while also allowing one to make non black-box use of it, for instance by obfuscating the corresponding random function. We recall the definition in the following.

Definition 8.4 (PROM [JLLW23])**.** Let PRF be a PRF. The pseudorandom oracle model (PROM) for PRF is a model with an oracle that implements a random permutation $\mathsf{HMap} : \{0,1\}^\lambda \to \{0,1\}^\lambda$ and that answers to two different types of queries:

- $\mathcal{O}(\mathsf{Gen}, k) = \mathsf{HMap}(k)$.
- $\mathcal{O}(\mathsf{Eval}, h, x) = \mathsf{PRF}(\mathsf{HMap}^{-1}(h), x)$.

We sketch how (x)iPRO implies the existence of (x)iO in the PROM, without additional assumptions: Query $\mathcal{O}(\mathsf{Gen}, k)$ on a uniformly sampled k to obtain a handle h. Then use the (x)iPRO to obfuscate the circuit C' defined as:

$$C'(x) = C(x) \oplus \mathsf{PRF}(k, x).$$

Return $(\tilde{C}', h)$ as the obfuscation. On input x, the evaluator can recover the value $C(x)$ by evaluating the obfuscated circuit $\tilde{C}'(x)$ and querying the oracle $\mathcal{O}(\mathsf{Eval}, h, x)$ using the handle h.

To show indistinguishability, we first observe that we can simulate the oracle $\mathcal{O}(\mathsf{Eval}, h, x)$ by computing:

$$\tilde{C}'(x) \oplus C_0(x) = \tilde{C}'(x) \oplus C_1(x) = \mathsf{PRF}(k, x).$$

Thus all information in the view of the distinguisher is efficiently computable using $\tilde{C}'$, which means that it suffices to show that the (x)iPRO obfuscation of C_0 is computationally indistinguishable from the (x)iPRO obfuscation of C_1 (ignoring other variables in the view of the distinguisher). By the pseudorandomness of (x)iPRO, it suffices to show that the truth-table of C' is computationally close to uniform, which follows immediately by the (sub-exponential) security of PRF.

9 On the impossibility of PRO for all PRF families

Following the ideas of [BCC+14, Theorem 4.1], one can show that there exist PRF families for which there does not exist an average-case universal virtual black-box simulator (with or without auxilliary input). For completeness, we recap their counterexample adapted to our setting.

9.1 Counterexample in the presence of auxillary input

In this section, we show that for *all* pseudorandom circuit families, there exists an auxiliary information with respect to which the family is not PRO obfuscatable.

Theorem 9.1 (Adapted from [BCC+14, Theorem 4.1])**.** *Assuming the existence of subexponentially secure instance-hiding witness encryption*[12] *for* NP*, for every class of circuit families C satisfying*

$$\{C(K,x)\}_{x\in\{0,1\}^\eta} \approx_c \left\{u_x : u_x \leftarrow \{0,1\}^\ell\right\}_{x\in\{0,1\}^\eta},$$

[12]This property means that for $x, x' \notin L$, $\mathsf{WE.Enc}(x, b) \approx_c \mathsf{WE.Enc}(x', b)$. This property is achieved by witness encryptions constructed via null-iO.

there exists a dependent auxiliary input such that the pre-condition of PRO *is satisfied, but the post-condition is not.*

Proof. Suppose the description of $C(K,\cdot) \leftarrow \mathcal{C}_\lambda$ has size $p(\lambda)$, and suppose there exists a PRO obfuscator such that $|\mathsf{PRO}(C)| \leq q(|C|)$ for some polynomial q. We use the notation $C(K, I)$ for $I \subseteq \{0,1\}^\ell$ to indicate the ordered list of evaluations $\{C(K,i)\}_{i\in I}$.

Consider the following NP language:

$$\mathcal{L}_{k,h} = \{\{y_1,\ldots,y_k\} \mid \text{there exists a circuit } C \text{ of size at most } h \text{ such that } C(i) = y_i\} \tag{9.1}$$

where h some polynomial.

Let $h = h(\lambda) = q \circ p(\lambda)$ and $k = k(\lambda) = h(\lambda)^2$, and sample the auxiliary information for key K as

$$\mathsf{aux}_K \leftarrow (\mathsf{WE.Enc}(C(K,[k]),0), \mathsf{WE.Enc}(C(K,[k]),1))$$

where WE is witness encryption for the language $\mathcal{L}_{k,h}$ corresponding to the statement $C(K,[k])$[13].

Analyzing the post-condition. Clearly, given $\tilde{C} = \mathsf{PRO.Obf}(C_K)$, and aux_K, one can use $\tilde{C}$ to decrypt each witness encryption of aux_K to obtain the bits 0 and 1 respectively, with probability close to 1 (by the correctness of the obfuscation). On the other hand, if an adversary is only given the witness encryptions without $\tilde{C}$ (or any succinct representation of C) we have the following observation:

Lemma 9.2. *For all ppt adversaries $\mathcal{A}$,*

$$\left|\Pr_K[\mathcal{A}(\mathsf{WE.Enc}(C(K,[k])),0)=1] - \Pr_K[\mathcal{A}(\mathsf{WE.Enc}(C(K,[k]),1)=1]\right| \leq \mathsf{negl}(\lambda).$$

Proof. We argue this following the fact that $C(K,\cdot)$

- Hybrid $\mathcal{H}_0$: $\mathcal{A}$ is given $\mathsf{WE.Enc}(C(K,[k])),0)$.
- Hybrid $\mathcal{H}_1$: For all i, switch $C(K,i)$ with $u_i \leftarrow \{0,1\}^\ell$. In particular, the witness encryption is sampled as
$$\mathsf{WE.Enc}((u_1,\ldots,u_h),0),$$
This follows from the subexponential PRF security of the family $C(K,\cdot) \leftarrow \mathcal{C}_\lambda$.
- Hybrid $\mathcal{H}_2$: Switch the encryption to be an encryption of 1.
Note that with high probability, $(u_1,\ldots,u_k)$ for uniformly u_i corresponds to a no-instance of $\mathcal{L}_{k,h}$ by a counting argument. This is because, there are 2^h possible circuits of size h, and there are $2^k = 2^{h^2}$ possible strings $(u_1,\ldots,u_k)$, and hence with probability $2^k/2^{h^2} \leq 2^{-\lambda}$. Therefore, one can invoke the security of witness encryption to switch the bit to 1.
- Hybrid $\mathcal{H}_3$: Switch $(u_1,\ldots,u_k)$ to $C(K,[k])$. Once again, we invoke the subexponential PRF security of the family $C(K,\cdot)$.

□

[13]Recall that $[k]$ is shorthand for the set $\{1,2,\ldots,k\}$

In particular, note that

$$|\Pr[\mathsf{WE.Dec}(\mathsf{ct}_0, \tilde{u}) = b] - \Pr[\mathsf{WE.Dec}(\mathsf{ct}_1, \tilde{u}) = b]| = \mathsf{negl}(\lambda).$$

Therefore, given $\tilde{u}$ and $\mathsf{aux}_K = (\mathsf{ct}_0, \mathsf{ct}_1)$, if one were to compute $b_0 \leftarrow \mathsf{WE.Dec}(\mathsf{ct}_0, \tilde{u})$ and $\mathsf{WE.Dec}(\mathsf{ct}_1, \tilde{u})$, we would have $(b_0, b_1) \neq (0, 1)$ with non-negligible probability. (This is because the above shows that $(1, 0)$ is almost as probable as $(0, 1)$, so that $(0, 1)$ cannot occur with $1 - \mathsf{negl}(\lambda)$ probability.) Therefore, this gives us a distinguisher for the case where $\tilde{C} = \mathsf{PRO.Obf}(C_K)$ and $\tilde{u}$.

Analyzing the pre-condition. However, we claim that the pre-condition of Definition 3.1 holds with respect to this choice of aux_K.

Lemma 9.3. *Assuming that $C(K, \cdot)$ is a subexponentially secure PRF family*

$$\{C(K, x)\}_{x \in \{0,1\}^\eta}, \mathsf{aux}_K \approx_c \left\{u_x : u_x \leftarrow \{0,1\}^\ell\right\}_{x \in \{0,1\}^\eta}, \mathsf{aux}_K. \tag{9.2}$$

Proof. We proceed in a few hybrids.

- Hybrid $\mathcal{H}_0$: The distribution is as in the left-hand side of the (9.2).
- Hybrid $\mathcal{H}_1$: For all i, switch $C(K, i)$ with $u_i \leftarrow \{0,1\}^\ell$. In particular, also switch aux_K to be sampled as
$$\mathsf{aux}_K := \mathsf{WE.Enc}((u_1, \ldots, u_h), 0), \mathsf{WE.Enc}((u_1, \ldots, u_h), 1),$$
This follows from the subexponential PRF security of the family $C(K, \cdot) \leftarrow \mathcal{C}_\lambda$.
- Hybrid $\mathcal{H}_2$: Switch aux_K to be sampled from an independent distribution:
$$\mathsf{aux}_K := \mathsf{WE.Enc}((u'_1, \ldots, u'_k), 0), \mathsf{WE.Enc}((u'_1, \ldots, u'_k), 1)$$
where $u'_i \leftarrow \{0,1\}^\ell$ independent from u_i.
Note that with high probability, both $(u_1, \ldots, u_k)$ and $(u'_1, \ldots, u'_k)$ correspond to a no-instance of $\mathcal{L}_{\lambda,h}$ by a counting argument. Therefore, by the instance hiding property of the witness encryption, $\mathcal{H}_1$ and $\mathcal{H}_2$ are indistinguishable.
- Hybrid $\mathcal{H}_3$: Replace $u'_1, \ldots, u'_k$ with $(C(K, 1), \ldots, C(K, k))$ for random $C(K, \cdot) \leftarrow \mathcal{C}_\lambda$. This follows from the subexponential PRF security of the family $C(K, \cdot) \leftarrow \mathcal{C}_\lambda$. This is exactly the RHS of (9.2).

This completes the proof. □

Hence, this shows that for any subexponentially secure PRF family, there exists an auxillary input with respect to which the pre-condition holds, but the post-condition does not hold. □

9.2 Counterexample without any auxiliary input

In this section, we show that PRO obfuscation is not possible in general, even in the absence of auxiliary input.

The idea is similar to the construction in the previous section. Intuitively, we wish to produce the same counterexample even in the absence of auxiliary input by modifying the circuit so that certain of its outputs correspond to the witness encryptions in the previous counterexample.

However, this naive approach does not work. Indeed, consider fixing any value of h, the upper bound on the witness size permitted by the witness encryption scheme. Then, the witness encryption ciphertexts may have bitlength larger than h. But this means that any attempt to hardcode them into the circuit will render the circuit, thus certainly the obfuscated circuit, too large to be a witness.

So, instead of a circuit, we will construct a *Turing machine* such that certain of its outputs correspond to the witness encryptions in the previous counterexample. The resulting construction will be a counterexample to PRO for Turing machines. However, since we have shown that PRO (for circuits) implies PRO for Turing machines, the counterexample also rules out PRO.

The reason this avoids the problem mentioned above is simple. A Turing machine need not scale with the size of the witness encryptions in order to produce them as output. Instead, only h needs to be hardcoded into the Turing machine. As the description of h is only $O(\log h)$ bits, it suffices to set h large enough so that $h \geq p(|\mathrm{TM}| + \mathrm{O}(\log \mathrm{h}))$, where p is a polynomial upper bound on the blowup of the PRO scheme, and $|TM|$ denotes the bit length of the description of a base Turing machine (which takes h as an input). This is of course possible. We now turn to the precise statement and proof.

Theorem 9.4 (Adapted from [BCC+14, Corollary 4.3])**.** *Suppose there exists a subexponentially secure witness encryption scheme for* NP *with pseudorandom ciphertexts, which we denote by* WE*, satisfying the following uniformity condition: There exists a polynomial-time Turing machine that on input 1^λ, (a description of) the relation circuit $\mathcal{R}$, an instance x, and a message m, outputs* $\mathsf{ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathcal{R}, x, m)$.

Then, for any polynomial p, there exists a polynomial-time Turing machine TM *computing a function family $f_p : \{0,1\}^{\mathfrak{h}} \times \{0,1\}^\eta \to \{0,1\}^\ell$ that, for random $K \leftarrow \{0,1\}^\eta$, satisfies the* PRO *precondition*

$$\{f_p(K,x)\}_{x\in\{0,1\}^\eta}, \approx_c \left\{u_x : u_x \leftarrow \{0,1\}^\ell\right\}_{x\in\{0,1\}^\eta} .$$

and such that from any obfuscation (circuit) C of size at most $p(\lambda)$ computing $f_p(K,\cdot)$, one can efficiently recover the key K and thus distinguish C from random.

In particular, a pseudorandom obfuscation scheme for all Turing machines does not exist.

Proof. We will denote by $\mathsf{PRF} : \mathcal{K} \times \{0,1\}^\ell \to \{0,1\}^\eta$ a subexponentially secure PRF, also computable by a polynomial-time Turing machine, with $|\mathcal{K}| = \mathsf{poly}(\ell)$ large enough that for random $K_{\mathsf{PRF}} \leftarrow \mathcal{K}$,

$$\{\mathsf{PRF}(K_{\mathsf{PRF}}, x)\}_{x\in\{0,1\}^\eta}, \approx_c \left\{u_x : u_x \leftarrow \{0,1\}^\ell\right\}_{x\in\{0,1\}^\eta} .$$

(The existence of such a PRF follows from our assumption regarding the existence of witness encryption.) Let $h = \lambda^c$, where c will be chosen later, and as before, let $k(\lambda) = h(\lambda)^2$. Let $p_h = p_h(\lambda)$ be the (polynomial) length of witness encryptions under WE, where the language is $\mathcal{L}_{k,h}$ (from Eq. (9.1)), the instance is of the form $(z_1, \ldots, z_k)$ for $z_i \in \{0,1\}^\eta$, and the message has length $\mathfrak{h}$. We may assume without loss of generality that η divides p_h.

Define a Turing machine TM' as follows. On input $c \in \mathbb{N}$, $K \in \{0,1\}^{\mathfrak{h}}$ and $x \in \{0,1\}^{\ell}$, TM interprets its key K as a tuple of values (K_{PRF}, r), where K_{PRF} is a key for PRF, and r is a string of random bits of the length needed for witness encryption. It then sets h, k, p_h as above starting with $h := \lambda^c$, and computes

$$\mathsf{ct} := \mathsf{WE.Enc}(\mathcal{L}_{k,h}, (\mathsf{PRF}(K_{\mathsf{PRF}}, 1), \ldots, \mathsf{PRF}(K_{\mathsf{PRF}}, k)), K; r) \ ,$$

and encodes it as a sequence of $N := p_h/\eta$ values $\mathsf{ct}^{(1)}, \ldots, \mathsf{ct}^{(N)} \in \{0,1\}^{\eta}$. Finally, it outputs

$$\mathrm{TM}'(K, x) := \begin{cases} \mathsf{PRF}(K_{\mathsf{PRF}}, x) & x < 2^{\ell} - N \\ \mathsf{ct}^{(i)} & x = 2^{\ell} - N + i - 1 \ . \end{cases}$$

We then define TM to be a Turing machine which has c hardcoded and on input K, x, simulates TM' on input c, K, x and outputs whatever TM' outputs. To finish the construction, we choose c large enough so that $h := \lambda^c \geq p$ and $|\mathrm{TM}| \leq \mathrm{h}(\lambda) = \lambda^{\mathrm{c}}$. The latter is possible as $|\mathrm{TM}| = \mathrm{O}(\log \mathrm{c}) + |\mathrm{TM}'|$, and $|\mathrm{TM}'|$ is independent of c.

We now argue that TM has the properties required by the theorem statement. The fact that TM satisfies the PRO precondition follows immediately from the facts that PRF is subexponentially secure, and that WE has pseudorandom ciphertexts. For the impossibility of obfuscation, let C be any circuit of size at most $p(\lambda)$ computing $f_p(K, \cdot)$. By evaluating C on inputs $2^{\ell} - N + i - 1$ for $i \in [N]$, the adversary obtains the witness encryption

$$\mathsf{ct} := \mathsf{WE.Enc}(\mathcal{L}_{k,h}, (\mathsf{PRF}(K_{\mathsf{PRF}}, 1), \ldots, \mathsf{PRF}(K_{\mathsf{PRF}}, k)), K; r) \ .$$

But the circuit C is itself a witness for $(\mathsf{PRF}(K_{\mathsf{PRF}}, 1), \ldots, \mathsf{PRF}(K_{\mathsf{PRF}}, k))$, as $|C| \leq p(\lambda) \leq h(\lambda)$, and $C(i) = \mathsf{PRF}(K_{\mathsf{PRF}}, i)$ for all $i \leq k$. Hence, the adversary may run

$$K := \mathsf{WE.Dec}(\mathcal{L}_{k,h}, C, (\mathsf{PRF}(K_{\mathsf{PRF}}, 1), \ldots, \mathsf{PRF}(K_{\mathsf{PRF}}, k)), \mathsf{ct})$$

to recover K.

For the particular statement, note that the adversary can check the validity of $K = (K_{\mathsf{PRF}}, r)$ by checking that $\mathsf{PRF}(K_{\mathsf{PRF}}, i) = C(i)$ for all $i \leq [k]$. As this will almost certainly not hold for random (description of) C, this allows the description of C to be distinguished from random, violating the PRO postcondition. □

Acknowledgements

We would like to thank Rahul Ilango for helpful discussions on notions of pseudorandom obfuscation and impossibilities.

G.M. and P.B. are supported by the European Research Council through an ERC Starting Grant (Grant agreement No. 101077455, ObfusQation). G.M. and V.V. are partially supported by a Misti Germany grant. S.M. is partially supported by Jane Street. S.P. is supported in part by the NSF under Grants Nos. CCF-2122230 and CCF-2312296, a Packard Foundation Fellowship, and a generous gift from Google.

V.V. and S.M. are supported in part by NSF CNS-2154149 and a Simons Investigator Award.

N.D. is funded by the European Union (ERC, LACONIC, 101041207). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

[AB06] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2006. 54

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. 12

[Agr19] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 191–225, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland. 4

[AIK11] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 120–129, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press. 24

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 308–326, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Berlin, Heidelberg, Germany. 7, 14, 21, 51

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 483–501, Cambridge, UK, April 15–19, 2012. Springer, Berlin, Heidelberg, Germany. 24

[AL18] Prabhanjan Ananth and Alex Lombardi. Succinct garbling schemes from functional encryption through a local simulation paradigm. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 455–472, Panaji, India, November 11–14, 2018. Springer, Cham, Switzerland. 9, 58, 59, 60

[AP20] Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 110–140, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland. 4

[AS17] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017,*

Part I, volume 10210 of *Lecture Notes in Computer Science*, pages 152–181, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland. 4

[AWY20] Shweta Agrawal, Daniel Wichs, and Shota Yamada. Optimal broadcast encryption from LWE and pairings in the standard model. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 149–178, Durham, NC, USA, November 16–19, 2020. Springer, Cham, Switzerland. 16, 25, 26

[BCC+14] Nir Bitansky, Ran Canetti, Henry Cohn, Shafi Goldwasser, Yael Tauman Kalai, Omer Paneth, and Alon Rosen. The impossibility of obfuscation with auxiliary input or a universal simulator. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 71–89, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Berlin, Heidelberg, Germany. 8, 67, 70

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 52–73, San Diego, CA, USA, February 24–26, 2014. Springer, Berlin, Heidelberg, Germany. 12

[BDGM20] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate iO from homomorphic encryption schemes. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 79–109, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland. 4, 6, 11, 15, 16

[BDGM22] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for IO: Circular-secure LWE suffices. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *ICALP 2022: 49th International Colloquium on Automata, Languages and Programming*, volume 229 of *LIPIcs*, pages 28:1–28:20, Paris, France, July 4–8, 2022. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. 4, 6, 12, 15, 16, 19

[BdMW16] Florian Bourse, Rafaël del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 62–89, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Berlin, Heidelberg, Germany. 26

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 533–556, Copenhagen, Denmark, May 11–15, 2014. Springer, Berlin, Heidelberg, Germany. 20, 26

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian,

editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Berlin, Heidelberg, Germany. 4, 12, 63

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 221–238, Copenhagen, Denmark, May 11–15, 2014. Springer, Berlin, Heidelberg, Germany. 4

[BGK+23] James Bartusek, Vipul Goyal, Dakshita Khurana, Giulio Malavolta, Justin Raizes, and Bhaskar Roberts. Software with certified deletion. Cryptology ePrint Archive, Paper 2023/265, 2023. https://eprint.iacr.org/2023/265. 4

[BGL+15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 439–448, Portland, OR, USA, June 14–17, 2015. ACM Press. 4, 7, 9, 58

[BHKL13] Daniel J. Bernstein, Mike Hamburg, Anna Krasnova, and Tanja Lange. Elligator: elliptic-curve points indistinguishable from uniform random strings. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 967–980, Berlin, Germany, November 4–8, 2013. ACM Press. 64

[BIJ+20] James Bartusek, Yuval Ishai, Aayush Jain, Fermi Ma, Amit Sahai, and Mark Zhandry. Affine determinant programs: A framework for obfuscation and witness encryption. In Thomas Vidick, editor, *ITCS 2020: 11th Innovations in Theoretical Computer Science Conference*, volume 151, pages 82:1–82:39, Seattle, WA, USA, January 12–14, 2020. LIPIcs. 4

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 535–564, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland. 9, 19, 21, 48, 49, 50, 58, 59, 82

[BMSZ16] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 764–791, Vienna, Austria, May 8–12, 2016. Springer, Berlin, Heidelberg, Germany. 4

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737, Cambridge, UK, April 15–19, 2012. Springer, Berlin, Heidelberg, Germany. 62

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a Nash equilibrium. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 1480–1498, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press. 4

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 1–25, San Diego, CA, USA, February 24–26, 2014. Springer, Berlin, Heidelberg, Germany. 4

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 264–302, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland. 20, 26

[BUW24] Chris Brzuska, Akin Unal, and Ivy K. Y. Woo. Evasive lwe assumptions: Definitions, classes, and counterexamples. Springer-Verlag, 2024. 13, 14, 27, 28

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 97–106, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press. 9

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 171–190, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press. 7, 14, 21, 51

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 480–499, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Berlin, Heidelberg, Germany. 4

[CLLZ21] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. Hidden cosets and applications to unclonable cryptography. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 556–584, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland. 4

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 468–497, Warsaw, Poland, March 23–25, 2015. Springer, Berlin, Heidelberg, Germany. 8, 13, 55, 56, 57

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of

Lecture Notes in Computer Science, pages 577–607, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland. 4

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976. 4

[DJ01] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Cheju Island, South Korea, February 13–15, 2001. Springer, Berlin, Heidelberg, Germany. 15

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. 7

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press. 4

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science*, pages 464–479, Singer Island, Florida, October 24–26, 1984. IEEE Computer Society Press. 29

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 467–476, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. 7, 9, 61

[GJ18] Craig Gentry and Charanjit S. Jutla. Obfuscation using tensor products. *IACR Cryptol. ePrint Arch.*, page 756, 2018. 4

[GK05] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *46th Annual Symposium on Foundations of Computer Science*, pages 553–562, Pittsburgh, PA, USA, October 23–25, 2005. IEEE Computer Society Press. 8

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 555–564, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. 21, 82

[GKPV10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In Andrew Chi-Chih Yao, editor, *ICS 2010: 1st Innovations in Computer Science*, pages 230–240, Tsinghua University, Beijing, China, January 5–7, 2010. Tsinghua University Press. 24

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, *58th Annual Symposium on Foundations of Computer Science*, pages 612–621, Berkeley, CA, USA, October 15–17, 2017. IEEE Computer Society Press. 9, 12, 13, 61

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd Annual ACM Symposium on Theory of Computing*, pages 736–749, Virtual Event, Italy, June 21–25, 2021. ACM Press. 4, 6, 12, 15, 16

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press. 15, 16, 17, 25, 26

[GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 194–213, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Berlin, Heidelberg, Germany. 4

[GS18] Sanjam Garg and Akshayaram Srinivasan. A simple construction of iO for Turing machines. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 425–454, Panaji, India, November 11–14, 2018. Springer, Cham, Switzerland. 9

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Berlin, Heidelberg, Germany. 9, 15, 28, 82

[Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 443–457, Kyoto, Japan, December 3–7, 2000. Springer, Berlin, Heidelberg, Germany. 4

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. 29

[HJL21] Samuel B. Hopkins, Aayush Jain, and Huijia Lin. Counterexamples to new circular security assumptions underlying iO. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 673–700, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland. 16

[HLL23] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 415–434. IEEE, 2023. 14

[ILW23] Rahul Ilango, Jiatu Li, and R. Ryan Williams. Indistinguishability obfuscation, range avoidance, and bounded arithmetic. In Barna Saha and Rocco A. Servedio, editors, *55th Annual ACM Symposium on Theory of Computing*, pages 1076–1089, Orlando, FL, USA, June 20–23, 2023. ACM Press. 4

[Imp11] Russell Impagliazzo. Lecture notes in advanced complexity, Spring 2011. 54

[JJ22] Abhishek Jain and Zhengzhong Jin. Indistinguishability obfuscation via mathematical proofs of equivalence. In *63rd Annual Symposium on Foundations of Computer Science*, pages 1023–1034, Denver, CO, USA, October 31 – November 3, 2022. IEEE Computer Society Press. 9

[JLLS23] Aayush Jain, Huijia Lin, Paul Lou, and Amit Sahai. Polynomial-time cryptanalysis of the subspace flooding assumption for post-quantum $i\mathcal{O}$. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part I*, volume 14004 of *Lecture Notes in Computer Science*, pages 205–235, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland. 16

[JLLW23] Aayush Jain, Huijia Lin, Ji Luo, and Daniel Wichs. The pseudorandom oracle model and ideal obfuscation. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part IV*, volume 14084 of *Lecture Notes in Computer Science*, pages 233–262, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland. 6, 8, 10, 67

[JLMS19] Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials over $\mathbb{R}$ to build $i\mathcal{O}$. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 251–281, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland. 4

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd Annual ACM Symposium on Theory of Computing*, pages 60–73, Virtual Event, Italy, June 21–25, 2021. ACM Press. 4, 7, 12

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 670–699, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland. 4, 7, 12

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for Turing machines with unbounded memory. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 419–428, Portland, OR, USA, June 14–17, 2015. ACM Press. 9

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology*

– *EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 28–57, Vienna, Austria, May 8–12, 2016. Springer, Berlin, Heidelberg, Germany. 4

[Lin17] Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 599–629, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland. 4

[LPST16] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 9615 of *Lecture Notes in Computer Science*, pages 447–462, Taipei, Taiwan, March 6–9, 2016. Springer, Berlin, Heidelberg, Germany. 6, 9, 14, 19, 31, 63

[LT17] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 630–660, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland. 4

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th Annual Symposium on Foundations of Computer Science*, pages 11–20, New Brunswick, NJ, USA, October 9–11, 2016. IEEE Computer Society Press. 4

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer, Berlin, Heidelberg, Germany. 17, 23, 25, 26

[MPV24] Surya Mathialagan, Spencer Peters, and Vinod Vaikuntanathan. Adaptively sound zero-knowledge SNARKs for UP. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part X*, volume 14929 of *Lecture Notes in Computer Science*, pages 38–71, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland. 6, 7, 11, 27, 28

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages 372–381, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press. 23

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 629–658, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Berlin, Heidelberg, Germany. 4

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Berlin, Heidelberg, Germany. 14, 28

[PRS17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th Annual ACM Symposium on Theory of Computing*, pages 461–473, Montreal, QC, Canada, June 19–23, 2017. ACM Press. 27

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 500–517, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Berlin, Heidelberg, Germany. 4

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th Annual Symposium on Foundations of Computer Science*, pages 859–870, Paris, France, October 7–9, 2018. IEEE Computer Society Press. 6, 19, 20, 21, 43, 44, 82

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press. 6, 13, 24, 27, 29

[Ros98] Sheldon M. Ross. *A First Course in Probability*. Prentice Hall, Upper Saddle River, N.J., fifth edition, 1998. 16

[Rot11] Ron Rothblum. Homomorphic encryption: From private-key to public-key. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 219–234, Providence, RI, USA, March 28–30, 2011. Springer, Berlin, Heidelberg, Germany. 55

[RVV24] Seyoon Ragavan, Neekon Vafa, and Vinod Vaikuntanathan. Indistinguishability obfuscation from bilinear maps and LPN variants. *Cryptology ePrint Archive*, 2024. 4

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press. 4, 7, 12

[Tib14] Mehdi Tibouchi. Elligator squared: Uniform points on elliptic curves of prime order as uniform random strings. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *FC 2014: 18th International Conference on Financial Cryptography and Data Security*, volume 8437 of *Lecture Notes in Computer Science*, pages 139–156, Christ Church, Barbados, March 3–7, 2014. Springer, Berlin, Heidelberg, Germany. 64

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 535–559, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland. 6, 7, 11, 13, 14, 27

[VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-IO from evasive LWE. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part I*, volume 13791 of *Lecture Notes in Computer Science*, pages 195–221, Taipei, Taiwan, December 5–9, 2022. Springer, Cham, Switzerland. 6, 7, 8, 9, 11, 13, 14, 27

[Wee20] Hoeteck Wee. Functional encryption for quadratic functions from k-lin, revisited. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 210–228, Durham, NC, USA, November 16–19, 2020. Springer, Cham, Switzerland. 11, 65

[Wee22] Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 217–241, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland. 6, 13, 14, 27

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part III*, volume 12698 of *Lecture Notes in Computer Science*, pages 127–156, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland. 4, 6, 12, 15, 16

[WWW22] Brent Waters, Hoeteck Wee, and David J. Wu. Multi-authority ABE from lattices without random oracles. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022: 20th Theory of Cryptography Conference, Part I*, volume 13747 of *Lecture Notes in Computer Science*, pages 651–679, Chicago, IL, USA, November 7–10, 2022. Springer, Cham, Switzerland. 14

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In Chris Umans, editor, *58th Annual Symposium on Foundations of Computer Science*, pages 600–611, Berkeley, CA, USA, October 15–17, 2017. IEEE Computer Society Press. 9, 12, 13, 61

[Zha19] Mark Zhandry. On ELFs, deterministic encryption, and correlated-input security. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 3–32, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland. 4

A Blind LFE from Blind AB-LFE

We will briefly sketch how the blind AB-LFE scheme discussed in the technical outline (Section 1.2) can be upgraded into a fully-fledged blind LFE scheme. We rely on the same transformation

as [QWW18], but using slightly stronger components. Since this is fairly a standard transformation (see e.g. [GKP+13]), we will only provide a sketch.

- The input ct is an FHE encryption of the input K. By relying on an FHE scheme with pseudorandom keys and ciphertexts, such as the GSW scheme [GSW13], x is pseudorandom.
- Instead of using the circuit C directly, we use a circuit C' such that $C'(ct, i, b)$ where $i \in \{1, \ldots, m\}$ and $b \in \{0, 1\}$ outputs a bit $ct'_i \oplus b$, where $ct' \in \{0,1\}^m$ (for some $m = \mathsf{poly}(\lambda)$) is the bit-representation of the resulting from FHE evaluation of the circuit C on the ciphertext ct.
- The blind AB-LFE encryption algorithm takes as input x, and index i, a bit b and a string μ.
- To encrypt a key K, we proceed as follows.
  - Generate a fresh GSW key pair $(\mathsf{pk}, \mathsf{sk})$
  - Encrypt K into a GSW ciphertext ct
  - Garble the GSW decryption circuit $\mathsf{Dec}(\mathsf{sk}, \cdot)$ with the secret key sk hardwired using the blind garbling scheme of [BLSV18]. This yields a garbled circuit $\tilde{\mathsf{Dec}}$ as well as labels $(\mu_{1,0}, \mu_{1,1}), \ldots, (\mu_{m,0}, \mu_{m,1})$.
  - For $i \in \{1, \ldots, m\}$ and $b \in \{0, 1\}$ compute blind AB-LFE ciphertexts $c_{i,b} = \mathsf{Enc}(h, (ct, i, b), \mu_{i,b})$.
  - The blind LFE ciphertext consists of $\mathsf{pk}, ct, \tilde{\mathsf{Dec}}$ and $\{c_{i,0}, c_{i,1}\}_{i \in \{1,\ldots,m\}}$.
- To decrypt such a ciphertext, proceed as follows.
  - Recompute the ciphertext ct' by homomorphically evaluating C on ct.
  - Decrypt the c_{i,ct_i} to μ_{i,ct_i} using the LFE decryption algorithm with the circuit C'.
  - Evaluate the garbled circuit $\tilde{\mathsf{Dec}}$ on the $\mu_{1,ct_1}, \ldots, \mu_{m,ct_m}$ to obtain the output $C(K)$.

To argue blindness of this LFE scheme, we proceed in the following hybrid steps:

- By the blindness of the AB-LFE scheme, the ciphertexts $\{c_{i,b}\}_{i,b}$ are computationally indistinguishable from uniformly chosen values subject to the c_{i,ct'_i} decrypting to μ_{i,ct'_i}.
- By the blindness of the garbling scheme, $\tilde{\mathsf{Dec}}$ and the $\mu_{1,ct'_1}, \ldots, \mu_{m,ct'_m}$ are uniformly random subject to $\tilde{\mathsf{Dec}}$ evaluating to $C(K)$ on $\mu_{1,ct'_1}, \ldots, \mu_{m,ct'_m}$
- By the pseudorandom keys and ciphertexts property of GSW, pk and ct are pseudorandom.
- Finally, by the pseudorandomness of C it holds that $C(K)$ is pseudorandom.

Hence, it follows that the entire LFE ciphertext is pseudorandom (given C, the hash h etc.), i.e. the above LFE scheme is blind. In the main body (Section 5) we provide more direct, but conceptually slightly more involved construction of blind LFE based on the *direct construction of function hiding LFE* given in [QWW18].