Factoring and Pairings are not Necessary for iO:
Circular-Secure LWE Suffices

Zvika Brakerski\textsuperscript{1}, Nico Döttling\textsuperscript{2}, Sanjam Garg\textsuperscript{3}, and Giulio Malavolta\textsuperscript{4}

\textsuperscript{1}Weizmann Institute of Science
\textsuperscript{2}CISPA Helmoltz Center for Information Security
\textsuperscript{3}UC Berkeley
\textsuperscript{4}Max Planck Institute for Security and Privacy

Abstract
We construct indistinguishability obfuscation (iO) solely under circular-security properties of encryption schemes based on the Learning with Errors (LWE) problem. Circular-security assumptions were used before to construct (non-leveled) fully-homomorphic encryption (FHE), but our assumption is stronger and requires circular randomness-leakage-resilience. In contrast with prior works, this assumption can be conjectured to be post-quantum secure; yielding the first provably secure iO construction that is (plausibly) post-quantum secure.

Our work follows the high-level outline of the recent work of Gay and Pass [ePrint 2020], who showed a way to remove the heuristic step from the homomorphic-encryption based iO approach of Brakerski, Döttling, Garg, and Malavolta [EUROCRYPT 2020]. They thus obtain a construction proved secure under circular security assumption of natural homomorphic encryption schemes — specifically, they use homomorphic encryption schemes based on LWE and DCR, respectively. In this work we show how to remove the DCR assumption and remain with a scheme based on the circular security of LWE alone. Along the way we relax some of the requirements in the Gay-Pass blueprint and thus obtain a scheme that is secure under a relaxed assumption. Specifically, we do not require security in the presence of a key-cycle, but rather only in the presence of a key-randomness cycle.

1 Introduction

The goal of program obfuscation [Had00, BGI\textsuperscript{+}01] is to transform an arbitrary circuit $\Pi$ into an unintelligible but functionally equivalent circuit $\tilde{\Pi}$. The aforementioned works showed that strong simulation-based notions of obfuscation were impossible to do for general purpose functionalities. However, the seemingly weaker \textit{indistinguishability obfuscation} (iO) was not ruled out by prior work (and has in fact been shown to be the same as the best possible notion of obfuscation [GR07]). In broad terms, iO requires that if two circuits $\Pi_0$ and $\Pi_1$ are two implementations of the same function, then their obfuscations are computationally indistinguishable.

Garg et al. [GGH13a, GGH\textsuperscript{+}13b] presented the first candidate for general purpose iO, paving the way for numerous other candidates based on a variety of mathematical structures. Although iO appears to be a weak notion of security, it has been shown to be sufficient for numerous cryptographic applications,
including ones that were previously not known to exist under other assumptions (see [SW14,GGHR14,BZ14] for examples). The first realizations of obfuscation relied on a new algebraic object called multilinear maps [GGH13a,CLT13,GGH15], which had only recently been constructed. Furthermore, the security of these objects relied on new (and poorly understood) computational intractability assumptions, or more commonly on plain heuristics. In fact, several attacks on multilinear map candidates [CHL+15,HJ16] and on obfuscation constructions based on multilinear maps [MSZ16,CGH17] were demonstrated. To defend against these attacks, several safeguards have been (e.g., [GMM+16,CVW18,MZ18,BGMZ18,DGG+18]) proposed. Even with these heuristic safeguards, all but the schemes based on the Gentry et al. [GGH15] multilinear maps are known to be broken against quantum adversaries.

Towards the goal of avoiding heuristics and obtaining provably secure constructions, substantial effort was made towards obtaining \( \text{iO} \) while minimizing (with the ultimate goal of removing) the use of multilinear maps [Lin16,LV16,AS17,Lin17,LT17]. These efforts culminated in replacing the use of multilinear maps with just bilinear maps [Agr19,JLMS19,AJL+19], together with an additional pseudorandom generator with special properties. Very recently this last limitation was removed by Jain, Lin and Sahai [JLS20]. Specifically, they obtained \( \text{iO} \) based on the combined (sub-exponential) hardness of the Learning with Errors problem (LWE), a large-modulus variant of the Learning Parity with Noise problem (LPN), the existence of a pseudorandom generator in \( \text{NC}_0 \), and in addition the hardness of the external Diffie-Hellman problem in bilinear groups (SXDH). We note that the use of the pairings makes these construction insecure against quantum adversaries.

A different approach towards provably secure \( \text{iO} \), which is more relevant to this work, was presented by Brakerski et al. [BDGM20]. They showed an \( \text{iO} \) candidate that is based on combining certain natural homomorphic encryption schemes. However, their construction was \textit{heuristic} in the sense that security argument could only be presented in the random oracle model. In a recent work, Gay and Pass [GP20] showed a way to remove the heuristic step and instead rely on a concrete assumption. Their construction is proved secure under the circular security of natural homomorphic encryption schemes — specifically, they use homomorphic encryption schemes based on LWE and Decisional Composite Residuosity (DCR, also known as Paillier’s assumption). In terms of assumptions, their construction assumes sub-exponential security of (i) the Learning with Error (LWE) assumption, (ii) the Decisional Composite Residuosity (DCR) assumption, and (iii) a new notion of security that they call “shielded randomness leakage” (SRL). The latter essentially requires that a fully homomorphic encryption scheme (specifically the GSW encryption scheme [GSW13]) remains secure even in the presence of a key-cycle with the Damgård-Jurik encryption scheme [DJ01]. Moreover, the notion of security is not the standard semantic security, but rather a new notion of security with respect to leakage of ciphertext randomness. We note that this construction is insecure against quantum attackers because of the use of the Damgård-Jurik encryption scheme [DJ01].

In this work, we ask:

\textit{Can we realize provably secure constructions of \( \text{iO} \) based solely on hard problems in lattices?}

1.1 Our results

We obtain a general purpose \( \text{iO} \) construction based solely on the circular security of LWE-based encryption schemes. This is done by presenting a “packed” variant of the dual-Regev LWE-based encryption scheme, and showing novel ways of manipulating ciphertexts of this variant in conjunction with ciphertexts of an FHE scheme. This allows us to remove the need for DCR-based encryption from the construction of [BDGM20, GP20]. Furthermore, our technique allows to relax the SRL security property that is required, so that we no longer need to require SRL security with respect to a key-cycle, but rather only with respect to a key-randomness cycle. We put forth this potentially weaker assumption as an object for further study.

The security of our construction relies thus on a circular-security assumption (either SRL security as in [GP20] or our weaker key-randomness circularity notion). The term “circular security” refers to the notion where the security properties of an encryption scheme are preserved even when given an encryption of the secret key itself, or, as in this case, given a “key cycle” where the key of the first scheme is encrypted.

---

1 Later, [GP20] updated their manuscript to also include a solution based on LWE. See Section 1.3 for additional discussion.
using the second scheme, and the key of the second scheme is encrypted using the first scheme. Circular security assumptions are commonly used in the literature. Notably, they are known to imply such primitives as “unrestricted” (non-leveled) fully homomorphic encryption (FHE) schemes via Gentry’s bootstrapping paradigm [Gen09].

Concretely, the circular assumption made in [GP20], and thus also in this work, is that a scheme (in particular a leveled FHE scheme) which enjoys the property that security is maintained even given some particular kind of leakage on the randomness of the ciphertext. Indeed, standard GSW encryption [GSW13] satisfies SRL security (under the LWE assumption), and the assumption we make is that SLR security holds even when given a key-cycle connecting GSW to another encryption scheme. The second scheme in [GP20] was based on DCR, whereas in our case this second encryption scheme is also based on LWE. While this assumption falls into the category of “circular security assumptions”, similarly to the ones that underlie bootstrapping in FHE, the concrete assumption is quite different. While in the FHE setting it was only assumed that (standard) CPA security is preserved given a key cycle, here we assume that the stronger SRL property remains intact.

Let us now state our results somewhat more precisely.

**Theorem 1.1 (Informal)** Assume the (sub-exponential) hardness of the LWE problem, and the SRL security of GSW in the presence of a 2-key cycle with a packed variant of dual-Regev, then there exists indistinguishability obfuscation for all circuits.

We note that if we further assume that circular security also maintains post-quantum security, then our assumption becomes post-quantum secure; yielding the first provably secure iO construction that is post-quantum secure. Additionally, our techniques yield also a scheme provably secure against the (potentially) weaker assumption of key-randomness SRL security, i.e. that SRL security is retained in the presence of a (packed) dual Regev encryption of the GSW secret key and a GSW encryption of the randomness used in such a ciphertext.

**Theorem 1.2 (Informal)** Assume the (sub-exponential) hardness of the LWE problem, and the SRL security of GSW in the presence of a randomness-key cycle with a packed variant of dual-Regev, then there exists indistinguishability obfuscation for all circuits.

### 1.2 Technical Overview

We now provide a technical outline of our construction and its properties.

**Obfuscation via Homomorphic Encryption.** The connection between (fully) homomorphic encryption and obfuscation is fairly straightforward. Given a program \( \Pi \) to be obfuscated, we can provide a ciphertext \( c_\Pi \) which encrypts \( \Pi \) under an FHE scheme. This will allow to use homomorphism to derive \( c_x = \text{Enc}(\Pi(x)) \) for all \( x \). Now all that is needed is a way to decrypt \( c_x \) in a way that does not reveal any information on \( \Pi \). Early works (e.g. [GGH+13b] and followups) attempted to use this approach and provide a “defective” version of the secret key of the FHE scheme, but a different approach was suggested in [BDGM20]. Specifically, [BDGM20] considered a homomorphic evaluation that takes \( c_\Pi \) to \( c_{\text{TT}} \), an encryption of the entire truth table of \( \Pi \), i.e. to an encryption of a multi-bit value. By relying on prior generic transformations [LPST16], they showed that one can reduce the task of constructing general-purpose obfuscation to the task of computing a “decryption” hint for \( c_{\text{TT}} \) with the following properties:

- **Succinctness:** The size of the decryption hint must be sublinear in the size of the truth table \( |\text{TT}| \).
- **Simulatability:** The decryption hint should not reveal any additional information besides the truth table \( \text{TT} \).

The reason why this is helpful is that some so-called “packed-encryption” schemes have the property that a short ciphertext-dependent decryption hint suffices in order to decrypt the ciphertext, in a way that does
not seem to leak the secret key of the scheme itself. While standard FHE schemes do not natively support packed encryption, it was shown in [BDGM19] that it is possible to use the so-called key-switching technique to switch from an FHE scheme into a packed-encryption scheme.

Alas, when instantiating the components of the [BDGM20] approach in its simplistic form described above, the decryption hint leaks information that renders the scheme insecure. To counter this issue, [BDGM20] proposed to inject another source of randomness: By adding freshly sampled ciphertexts of the packed-encryption scheme (which in their case was instantiated with the Damgård-Jurik scheme [DJ01]) one can smudge the leakage of the decryption hint. However the size of these fresh ciphertext would largely exceed the size of the truth table $TT$. Therefore, [BDGM20] proposed to heuristically sample them from a random oracle, leveraging the fact that the ciphertexts of [DJ01] are dense, i.e. a uniformly sampled string lies in the support of the encryption algorithm with all but negligible probability. This led to a candidate, but without a proof of security.

A Provably Secure Scheme. In a recent work, Gay and Pass [GP20] observed that for the purpose of constructing obfuscation, it suffices to consider schemes in the common random string (CRS) model where, importantly, the size of the CRS can exceed the size of the truth table. This allowed them to place the Damgård-Jurik ciphertexts in the CRS and therefore avoid relying on random-oracle-like heuristics.

They propose a new method to prove the security of this approach: Leveraging the structural property of the GSW scheme [GSW13]. They showed that adding a GSW encryption of 0 to the evaluated FHE ciphertext (before key-switching to Damgård-Jurik) allows one to program the FHE ciphertext in the security proof. To sample these GSW encryptions of 0, they propose to draw the random coins $r^*$ again from the CRS and let the evaluator recompute the correct ciphertext $c = \text{GSW.Enc}(0; r^*)$.

Taken together, these new ideas allow them to prove their construction secure against the shielded randomness leakage (SRL) security of the resulting FHE scheme. Loosely speaking, SRL security requires that semantic security of an encryption scheme is retained in the presence of an oracle that leaks the randomness $r_f$ of the homomorphic evaluation of the function $f$ over the challenge ciphertext. However the randomness $r_f$ is not revealed in plain to the adversary, instead it is “shielded” by the random coins of a fresh GSW ciphertext $c = \text{GSW.Enc}(0; r^*)$. That is, the adversary is given $(r_f - r^*, c)$. In fact, the adversary can obtain polynomially-many samples from this distribution, for any function $f$, conditioned on the fact that the adversary knows the output of $f(m^*)$, where $m^*$ is the hidden message.

To gain confidence in the veracity of the assumption, [GP20] show that the GSW encryption scheme satisfies SRL security if the (plain) LWE assumption holds. However, their obfuscation scheme requires one to publish a key cycle of GSW and Damgård-Jurik (i.e. an encryption of the GSW secrecy key under Damgård-Jurik and vice versa). Thus their final assumption is that SRL security is retained in the presence of such a key cycle.

Obfuscation from Circular-Secure LWE. We wish to remove the need for the Damgård-Jurik encryption scheme from the above construction paradigm. The major obstacle to overcome consists in designing an LWE-based encryption scheme that simultaneously satisfies three properties.

- **Linear Homomorphism:** In order to key switch the GSW ciphertext into this form, the scheme must satisfy some weak notion of homomorphism. Specifically, it must support the homomorphic evaluation of linear functions.

- **Succinct Randomness:** The scheme must allow us to encrypt a long message string with a short randomness, that can then function as the decryption hint.

- **Dense Ciphertexts:** A uniformly sampled string must lie in the support of the encryption algorithm with all but negligible probability. This will allow us to parse the CRS as a collection of ciphertexts.\(^2\)

\(^2\)Note that for the purpose of constructing the obfuscator, one could make do with a common reference string which can have an arbitrary distribution. However, the string needs to be parsed as a ciphertext with respect to all public-keys. Requiring dense ciphertexts is a simple requirement that implies this property.
Unfortunately all natural lattice-based candidates seem to fail to satisfy all of these properties. In particular, for all LWE-based schemes linear homomorphism seems to be at odds with dense ciphertexts: To ensure that the noise accumulated during the homomorphic evaluation does not impact the decryption correctness, one needs to ensure a gap between the noise bound and the modulus. More concretely, ciphertext are typically of the form \((a, a \cdot s + e + q/2 \cdot m) \in \mathbb{Z}_{q+1}^n\) where \(e \ll q\), which makes them inherently sparse.

**Our Solution: A Packed Variant of Dual-Regev that is also Dense-Friendly.** We show that the above requirements can be relaxed. Our starting point is devising a “packed” version of the dual-Regev encryption scheme [GPV08]. This scheme will not have dense ciphertexts so it does not fit the requirements from previous works. However, we will show how we can define, for the same scheme, a family of ciphertexts which are both “almost dense” and can inter-operate with the non-dense scheme, so as to allow to construct the obfuscator.

Let us start with our packed dual-Regev scheme. To pack a \(k\)-bit plaintext \(m \in \{0, 1\}^k\) in a dual-Regev ciphertext we construct the public key as a matrix \(A \in \mathbb{Z}_q^{m \times n}\), which is statistically close to uniform but is sampled together with a trapdoor \(\tau\) (whose role will be explained below), and another uniformly sampled matrix \(B \in \mathbb{Z}_q^{k \times n}\). The encryption algorithm computes a ciphertext as

\[(A \cdot r + e_0, B \cdot r + q/2 \cdot m + e)\]

where \(r \leftarrow \mathbb{Z}_q^n\) is the encryption randomness and the vectors \(e_0\) and \(e\) are the encryption noises, where the norm of both vectors is bounded by some \(B \ll q\). The property of the trapdoor \(\tau\) is that it allows to recover \(r\) from \(A \cdot r + e_0\). The (semantic) security of the scheme follows directly by definition of LWE. To decrypt, therefore, one can first use the trapdoor \(\tau\) to recover \(r\) from the first \(m\) elements of the ciphertext, and then recompute the mask \(B \cdot r\) and recover each individual bit by rounding to the closest multiple of \(q/2\). Setting the parameters appropriately, we can guarantee that the decryption is always successful. One important property of this scheme is that the random coins \(r \in \mathbb{Z}_q^n\) are sufficient to recover the entire message and furthermore the size of \(r\) is succinct (in particular independent of \(k\)).

In terms of homomorphism, the scheme is straightforwardly additively homomorphic. Furthermore, it supports key switching from any scheme with almost-linear decryption as per [BDGM19]. In particular it is possible to take a (long) message encrypted under an FHE scheme such as GSW and convert it to an encryption of the same message under packed dual-Regev, using precomputed key-switching parameters.

As explained above, this scheme does not have dense ciphertexts. At this point we make two crucial observations that will allow us to bypass this hurdle.

1. In order to construct the obfuscator using the [BDGM20] approach, dense ciphertexts only need to enjoy a very limited form of homomorphism, they only need to support a single addition with a non-dense ciphertext.

This is essentially because the obfuscator has the following outline. It starts by considering the dense ciphertext from the CRS (or oracle in the case of the original [BDGM20]), and homomorphically bootstraps it into a non-dense FHE ciphertext by evaluating the decryption circuit. Let \(m\) be the (random) message that is induced by the process. Then, the FHE encryption of \(m\) is processed in order to create a non-dense packed encryption of \(m \oplus TT\), where \(TT\) is the truth table of the program to be obfuscated (or, more accurately, a chunk of this truth table, partitioning into chunks is required in order to allow reusability of the keys). Then a single homomorphic addition between the dense and non-dense ciphertext would imply a packed encryption of the truth table. All of this can be performed by the evaluator of the obfuscated program, so all that is needed is the decryption hint for this final ciphertext, that would allow to recover \(TT\).

We note importantly, that in prior approaches (including the [GP20] blueprint) the aforementioned bootstrapping creates a key cycle, since a packed ciphertext is bootstrapped into an FHE ciphertext, which is afterwards key-switched into a packed ciphertext. However, we notice that it suffices to provide an

---

[3] This is done using the by-now-standard technique of encrypting powers-of-two of the elements of the secret key of the latter scheme, so that it is possible to evaluate any inner product homomorphically.

[4] We note that the key switching parameters are quite long so it is required for our method that they are reusable.
encryption of the (succinct) randomness of the dense ciphertext in order to apply bootstrapping, thus leading to a relaxed key-randomness circular assumption. Interestingly, this observation is not very useful for actual dense ciphertexts (since finding the randomness would require using the key), however, our relaxed notion of density described below will allow to apply it and thus relax the circularity notion as well.

(2) A notion of almost-everywhere density suffices. A ciphertext distribution is almost-everywhere dense if it is dense except for a non-dense part whose length is independent of $k$ (the message length).

The reason that this is sufficient is that the non-dense part of the ciphertext, which we refer to as the header, can be generated by the obfuscator and provided to the evaluator as a part of the obfuscated program. Since the header is short, and in particular the message length $k$ can be selected to be much longer than the header, the effect on the length of the obfuscated program will be minimal.

As hinted above, since the obfuscator generates the header, it in particular also samples the randomness for the final almost-everywhere dense ciphertext. This means that the obfuscator can generate the bootstrapping parameters using this randomness without requiring a key cycle.

**Dense Encryption Mode.** With these observations in mind we describe an alternative encryption mode ($\text{DenseEnc}$) for the packed variant of dual-Regev where the bulk of the ciphertext is dense. On input a message $m \in \{0, 1\}^k$, the encryption algorithm in dense mode computes the following ciphertext

$$(A \cdot r + e_0, B \cdot r + q/2 \cdot m + u)$$

where $r$ and $e_0$ are sampled as before and $u \leftarrow [-q/4,+q/4]^k$. For convenience, we are going to split the ciphertexts into two blocks: The header $h_0 \in \mathbb{Z}_q^m$ and the message carrier $(h_1, \ldots, h_k) \in \mathbb{Z}_q^k$. Foremost, observe that the decryption algorithm as described before still returns the correct message with probability 1, since it recovers the same $r$ from $h_0$. Furthermore, note that (for a fixed header) all vectors $(h_1, \ldots, h_k) \in \mathbb{Z}_q^k$ are in the support of the encryption algorithm. Since $k \gg m$, most of the elements of the ciphertext in the alternative decryption mode are dense.

One can verify that the aforementioned limited form of homomorphism indeed holds, namely that

$$dR.\text{Enc}(m) + dR.\text{DenseEnc}(m') \in dR.\text{DenseEnc}(m_0 \oplus m').$$

This is the case since

$$(A \cdot r + e_0, B \cdot r + q/2 \cdot m + e) + (A \cdot r' + e_0', B \cdot r' + q/2 \cdot m' + u)$$

$$= (A \cdot (r + r') + e_0 + e_0', B \cdot (r + r') + q/2 \cdot (m \oplus m') + e + u)$$

$$= (A \cdot \tilde{r} + e_0, B \cdot \tilde{r} + q/2 \cdot (m \oplus m') + \tilde{u})$$

where $\tilde{u} = e + u \in [-q/4,+q/4]^k$ with all but negligible probability over the random choice of $u$, for an appropriate choice of the parameters.

**Doing Away with the Header.** We notice that given our two observations above, the goal of the header in the obfuscation scheme is quite minimal. The header is not needed for homomorphism, and is only needed for the purpose of extracting the randomness $r$ at decryption time. We then observe that decrypting packed ciphertext is done in two contexts in the scheme. The first is when we bootstrap the almost-everywhere dense ciphertext into an FHE ciphertext, and the other is when the evaluator of the obfuscated program recovers $TT$ from the final ciphertext. For the latter there is no need for a header since the decryption hint, i.e. the respective $r$ value, is provided within the obfuscated program. For the former we do not need a header of a specific structure, but rather simply an encryption of $r$ that allows bootstrapping the almost-dense ciphertext. It therefore suffices to provide $\text{GSW.Enc}(r)$ directly, which makes the header completely redundant.
On the Assumption. Equipped with the newly developed packed version of dual-Regev we can follow the [BDGM20, GP20] approach, with the aforementioned modifications, to construct the obfuscator. The resulting construction can be shown secure against the assumption that the SRL security of GSW is retained in the presence of a key cycle with the packed dual-Regev encryption scheme as presented above.

We then observe that it suffices to assume SRL security with respect to key-randomness cycles, rather than key cycles. We note that this assumption is no-stronger than key-cycle SRL since given a key-cycle it is possible to homomorphically generate a key-randomness cycle, but the converse is not known to be true.

Adding this to our observation about the redundancy of the header, the assumption we require is that SRL security is retained in the presence of a key-randomness cycle between GSW and packed dual-Regev, i.e.

$$(\text{GSW.Enc}(r), \text{dR.Enc}(\text{sk}_{\text{GSW}}; r)).$$

Since dual-Regev is randomness recoverable, this assumption is (potentially) strictly weaker than SRL security in the presence of a key-cycle.

1.3 Other Related Work

Subsequently to the posting of this manuscript online (but concurrently and independently) [GP20] updated their manuscript to include a solution based on LWE in the place of DCR. They do not make the observations that a relaxed notion of density suffices (and is preferable) and thus they explicitly construct an encryption scheme with dense ciphertexts based on the (primal) Regev encryption scheme. The resulting scheme is more involved and in particular requires the circular SRL security of GSW rather than the relaxed key-randomness circularity notion.

Wee and Wichs [WW20], again concurrently, presented another instantiation of the [BDGM20] approach which is arguably post-quantum secure. They rely on an indistinguishability assumption between two distributions and not directly on circular security. However, the underlying machinery developed shares many similarities with our approach. Specifically, while we essentially rely on randomness that is embedded in the CRS by interpreting it as an obliviously sampled ciphertext (which thus corresponds to one encrypted with fresh randomness), their approach is to use a pseudorandom function to transform the CRS into a randomizer for the output hint.

2 Preliminaries

We denote by $\lambda \in \mathbb{N}$ the security parameter. We say that a function $\text{negl}$ is negligible if it vanishes faster than any polynomial. Given a set $S$, we denote by $s \leftarrow S$ the uniform sampling from $S$. We say that an algorithm is PPT if it can be implemented by a probabilistic machine running in time $\text{poly}(\lambda)$. We say that two distributions $(D_0, D_1)$ are computationally (statistically, resp.) indistinguishable if for all PPT (unbounded, resp.) distinguishers, the probability to tell $D_0$ an $D_1$ apart is negligibly close to 1/2. Matrices are denoted by $M$ and vectors are denoted by $v$. We denote the infinity norm of a vector $v$ by $\|v\|_{\infty}$. We recall the smudging lemma [AIK11, AJL+12].

**Lemma 2.1 (Smudging)** Let $B_1 = B_1(\lambda)$ and $B_2 = B_2(\lambda)$ be positive integers and let $e_1 \in [-B_1, B_1]$ be a fixed integer. Let $e_2 \leftarrow [-B_2, B_2]$ chosen uniformly at random. Then the distribution of $e_2$ is statistically indistinguishable to that of $e_2 + e_1$ as long as $B_1/B_2 = \text{negl}(\lambda)$.

2.1 Indistinguishability Obfuscation

We recall the notion of indistinguishability obfuscation (iO) from [GGH+13b].

**Definition 2.2 (Indistinguishability Obfuscation)** A PPT machine $\text{iO}$ is an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ if the following conditions are satisfied:

(Functionality) For all $\lambda \in \mathbb{N}$, all circuit $\Pi \in \mathcal{C}_\lambda$, all inputs $x$ it holds that: $\tilde{\Pi}(x) = \Pi(x)$, where $\tilde{\Pi} \leftarrow \text{iO}(\Pi)$. 

7
For all $\lambda \in \mathbb{N}$, all pairs of circuit $(\Pi_0, \Pi_1) \in \mathcal{C}_\lambda$ such that $|\Pi_0| = |\Pi_1|$ and $\Pi_0(x) = \Pi_1(x)$ on all inputs $x$, it holds that the following distributions are computationally indistinguishable:

$$\text{iO}(\Pi_0) \approx \text{iO}(\Pi_1).$$

**XiO.** We recall a theorem from Lin et al. [LPST16], that states that (assuming the hardness of the LWE problem), constructing an obfuscator for circuits with logarithmically-many input bits suffices to build general-purpose obfuscation.

**Theorem 2.3 (XiO)** Assuming sub-exponentially hard LWE, if there exists a sub-exponentially secure indistinguishability obfuscator (with pre-processing) for $\mathbb{P}^{\log/\text{poly}}$ with non-trivial efficiency, then there exists an indistinguishability obfuscator for $\mathbb{P}/\text{poly}$ with sub-exponential security.

Here $\mathbb{P}^{\log/\text{poly}}$ denotes the class of polynomial-size circuits with inputs of length $\eta = O(\log(\lambda))$ and by non-trivial efficiency we mean that the size of the obfuscated circuit is bounded by $\text{poly}(\lambda, |\Pi|) \cdot 2^{\eta(1 - \epsilon)}$, for some constant $\epsilon > 0$. Note that the above theorem poses no restriction on the runtime of the obfuscator. Furthermore, the theorem allows the obfuscator to access a large uniform random string (the pre-processing) of size even larger than the truth table of the circuit.

### 2.2 Learning with Errors

We recall the definition of the learning with errors (LWE) problem [Reg05].

**Definition 2.4 (Learning with Errors)** The LWE problem is parametrized by a modulus $q$, positive integers $(n, m)$ and an error distribution $\chi$. The LWE problem is hard if the following distributions are computationally indistinguishable:

$$(A, A \cdot s + e) \approx (A, u)$$

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

As shown in [Reg05, PRS17], for any sufficiently large modulus $q$ the LWE problem where $\chi$ is a discrete Gaussian distribution with parameter $\alpha = \sigma q \geq 2\sqrt{n}$ (i.e. the distribution over $\mathbb{Z}$ where the probability of $x$ is proportional to $e^{-\pi(|x|/\sigma)^2}$, is at least as hard as approximating the shortest independent vector problem (SIVP) to within a factor of $\gamma = \tilde{O}(n/\alpha)$ in worst case dimension $n$ lattices. We refer to $\alpha = \sigma/q$ as the modulus-to-noise ratio, and by the above this quantity controls the hardness of the LWE instantiation. Hereby, LWE with polynomial $\alpha$ is (presumably) harder than LWE with super-polynomial or sub-exponential $\alpha$. We can truncate the discrete Gaussian distribution $\chi$ to $\sigma \cdot \omega(\sqrt{\log(\lambda)})$ while only introducing a negligible error. Consequently, we typically omit the actual distribution $\chi$ but only use the fact that it can be bounded by a (small) value $B$.

### 2.3 Public-Key Encryption

We recall the definition of public key encryption in the following.

**Definition 2.5 (Public-Key Encryption)** A public-key encryption scheme consists of the following efficient algorithms.

- **KeyGen$(1^\lambda)$**: On input the security parameter $1^\lambda$, the key generation algorithm returns a key pair $(sk, pk)$.
- **Enc$(pk, m)$**: On input a public key $pk$ and a message $m$, the encryption algorithm returns a ciphertext $c$.
- **Dec$(sk, c)$**: On input the secret key $sk$ and a ciphertext $c$, the decryption algorithm returns a message $m$.  


Correctness and Semantic Security. We recall the standard notions of correctness and semantic security \cite{GM82} for public-key encryption.

**Definition 2.6 (Correctness)** A public-key encryption scheme \((\text{KeyGen}, \text{Enc}, \text{Dec})\) is correct if for all \(\lambda \in \mathbb{N}\), all messages \(m\), all \((sk, pk)\) in the support of \(\text{KeyGen}(1^\lambda)\), and all \(c\) in the support of \(\text{Enc}(pk, m)\) it holds that \(\text{Dec}(sk, c) = m\).

**Definition 2.7 (Semantic Security)** A public-key encryption scheme \((\text{KeyGen}, \text{Enc}, \text{Dec})\) is semantically secure if for all \(\lambda \in \mathbb{N}\), all pairs of message \((m_0, m_1)\), it holds that the following distributions are computationally indistinguishable

\[
(\text{pk}, \text{Enc}(\text{pk}, m_0)) \approx (\text{pk}, \text{Enc}(\text{pk}, m_1))
\]

where \((sk, pk) \leftarrow \text{KeyGen}(1^\lambda)\).

Homomorphic Encryption. We say that a public-key encryption scheme \((\text{KeyGen}, \text{Enc}, \text{Dec})\) is homomorphic for the circuit class \(\{C_\lambda\}_{\lambda \in \mathbb{N}}\) if there exist an efficient deterministic algorithm \(\text{Eval} \) such that for all \(\Pi \in \{C_\lambda\}_{\lambda \in \mathbb{N}}\), all \((sk, pk)\) in the support of \(\text{KeyGen}\), all vectors of messages \((m_1, \ldots, m_\mu)\), all ciphertexts \((c_1, \ldots, c_\mu)\) in the support of \((\text{Enc}(pk, m_1), \ldots, \text{Enc}(pk, m_\mu))\) it holds that

\[
\text{Dec}(sk, \text{Eval}(pk, \Pi, (c_1, \ldots, c_\mu))) = \Pi(m_1, \ldots, m_\mu).
\]

Furthermore, we say that a scheme is fully-homomorphic if it is homomorphic for all polynomial-size circuits.

Circular Security. We say that two encryption schemes \((\text{KeyGen}_0, \text{Enc}_0, \text{Dec}_0)\) and \((\text{KeyGen}_1, \text{Enc}_1, \text{Dec}_1)\) form a key cycle if the distinguisher is given a cross-encryption of the secret keys, i.e. \(\text{Enc}(pk_1, sk_0)\) and \(\text{Enc}(pk_0, sk_1)\). We say that the scheme is 2-circular secure if semantic security is retained in the presence of such a cycle.

**Definition 2.8 (2-Circular Security)** A pair of public-key encryption schemes \((\text{KeyGen}_0, \text{Enc}_0, \text{Dec}_0)\) and \((\text{KeyGen}_1, \text{Enc}_1, \text{Dec}_1)\) is 2-circular secure if for all \(\lambda \in \mathbb{N}\), all pairs of message \((m_0, m_1)\), it holds that the following distributions are computationally indistinguishable

\[
(\text{pk}_0, pk_1, \text{Enc}_0(pk_0, sk_1), \text{Enc}_1(pk_1, sk_0), \text{Enc}_0(pk_0, m_0)) \\
\approx (\text{pk}_0, pk_1, \text{Enc}_0(pk_0, sk_1), \text{Enc}_1(pk_1, sk_0), \text{Enc}_0(pk_0, m_1))
\]

where \((sk_0, pk_0) \leftarrow \text{KeyGen}_0(1^\lambda)\) and \((sk_1, pk_1) \leftarrow \text{KeyGen}_1(1^\lambda)\).

2.4 The GSW Fully-Homomorphic Encryption

In the following we briefly recall the encryption scheme by Gentry, Sahai, and Waters \cite{GSW13} (henceforth, GSW). We denote by \(n = n(\lambda)\) the lattice dimension and by \(q = q(\lambda)\) the modulus (which we assume for simplicity to be even). We set \(m > n \log(q)\) and \(d = d(\lambda)\) as a bound on the depth of the arithmetic circuit to be evaluated.

**KeyGen(1^λ):** Sample a uniform matrix \(A \leftarrow \mathbb{Z}_q^{n \times m}\) and a vector \(s \leftarrow \chi^n\). Set the public key to \((A, b = s^TA + e^T)\), where \(e \leftarrow \chi^m\). The secret key is set to \((-s, 1)\).

**Enc(pk, m):** On input a message \(m \in \{0, 1\}\), sample a uniform \(R \leftarrow \{0, 1\}^{m \times m}\) and compute

\[
C = (A, b) \cdot R + m \cdot G
\]

where \(G = (1, 2, \ldots, 2^{\log(q)})^T \otimes I_{(n+1)}\) and \(I_{(n+1)} \in \{0, 1\}^{(n+1) \times (n+1)}\) denotes the identity matrix.
eval(pk, Π, (c1, . . . , cℓ)): There exists a (deterministic) polynomial-time algorithm that allows one to compute any d-bounded depth arithmetic circuit Π : {0, 1}n → {0, 1} homomorphically over a vector of ciphertexts (c1, . . . , cℓ). For details about this algorithm, we refer the reader to [GSW13]. For the purpose of this work, the only relevant information is that the evaluated ciphertext cΠ ∈ Zq(n+1) is an (n + 1)-dimensional vector. For multiple bits of output, the resulting ciphertext is defined to be the concatenation of the single-bit ciphertexts.

Dec(sk, c): We assume without loss of generality that the input ciphertext c ∈ Zq(n+1) is the output of the evaluation algorithm. Such a ciphertext defines a linear function ℓc such that

\[ ℓ_c(sk) = q/2 \cdot m + e \]

where |c| ≤ B = (m + 1)d m B. The message m is recovered by returning the most significant bit of the output.

Note that the decryption routine of GSW consists of the application of a linear function, followed by a rounding and we refer to this property as to almost-linear decryption. In a slight abuse of notation, we sometimes write KeyGen(1^λ; q) to denote the above key generation algorithm with a fixed modulus q.

Alternate Encryption. For convenience we also define a modified encryption algorithm, where the output ciphertexts consists of a single column vector. An additional difference is that we sample the randomness with norm B = 2^λ · B.

ColEnc(pk, m): On input a message m, sample a uniform r ← [−B, +B]m and compute

\[ c = (A, b) \cdot r + m. \]

The multi-bit version of such an algorithm is defined accordingly to output the concatenation of independently sampled ciphertexts. This algorithm is going instrumental for our scheme, although ciphertexts in this form no longer support the homomorphic evaluation of arbitrary circuits. We now recall a useful Lemma from [GP20].

Lemma 2.9 (GSW Smudging) Let B = 2^λ · B. For all λ ∈ N, for all (sk, pk) in the support of KeyGen(1^λ), for all messages (m1, . . . , mℓ), for all depth-d circuit (Π1, . . . , Πτ), the following distributions are statistically indistinguishable

\[
\begin{align*}
\text{Eval}(pk, Π, (c1, . . . , cℓ), (r1, . . . , rτ)) + \text{ColEnc}(pk, 0; r1), . . . , \text{ColEnc}(pk, 0; rτ) & \\
\approx & \left( (c1, . . . , cℓ), (r1 - r1, . . . , rτ - rτ), \text{ColEnc}(pk, Π1(m1, . . . , mℓ); r1), . . . , \text{ColEnc}(pk, Πτ(m1, . . . , mℓ); rτ) \right)
\end{align*}
\]

where ci ← Enc(pk, mi), rτ ← [−B, +B]m, and rτ is the randomness of the i-th evaluated ciphertext Eval(pk, Π, (c1, . . . , cℓ)).

Shielded Randomness Leakage. The notion of shielded randomness leakage (SRL) security [GP20] says that the scheme is semantically secure even in the presence of an oracle that leaks some information about the randomness for evaluated ciphertext. The circuit to be evaluated homomorphically are fixed ahead of time (although they may depend on the challenge ciphertext) and the adversary is constrained to know the output of the evaluation ahead of time. We present a formal definition in the following.

Definition 2.10 (SRL Security) A homomorphic encryption scheme (KeyGen, Enc, Eval, Dec) is SRL secure if for all λ ∈ N, all pairs of message (m0, m1), all (α1, . . . , ατ), all circuits (Π1, . . . , Πτ) such that
for all $i = 1 \ldots \tau$ it holds that $\Pi_i(m_0, \cdot) = \Pi_i(m_0, \cdot) = \alpha$, the following distributions are computationally indistinguishable

$$(pk, c = \text{Enc}(pk, m_0), c'_1, \ldots, c'_\tau, r_1^i - r_1, \ldots, r_\tau^i - r_\tau)$$

$$\approx (pk, c = \text{Enc}(pk, m_1), c'_1, \ldots, c'_\tau, r_1^i - r_1, \ldots, r_\tau^i - r_\tau)$$

where $(sk, pk) \leftarrow \text{KeyGen}(1^\lambda)$, $r_i^i \leftarrow [-\hat{B}, +\hat{B}]$, $c_i^* = \text{ColEnc}(0; r_i^i)$, and $r_i$ is the randomness of the $i$-th evaluated ciphertext $\text{Eval}(pk, \Pi_i(\cdot, c_i^*), c)$.

In [GP20], the authors showed that the GSW scheme satisfies the notion of SRL security if the LWE problem is hard. For completeness, we recall the theorem statement in the following.

Theorem 2.11 (SRL Security of GSW) If the LWE assumption holds, then the GSW encryption scheme satisfies the notion of SRL security.

If SRL security is retained in the presence of a key cycle, then we say that the scheme satisfies the notion of 2-circular SRL security. Specifically, we define this notion as above, except that the challenge ciphertext $c$ encrypts $m_0 || sk$ and the distinguisher is additionally given $\text{Enc}(pk, sk)$, where $(sk, pk)$ is an independently sampled key pair of a (possibly different) public-key encryption scheme.

3 Packed Encryption from LWE

In the following we describe a packed version of the dual-Regev encryption scheme [GPV08]. We denote by $n = n(\lambda)$ the lattice dimension, by $q = q(\lambda)$ the modulus (which we assume for simplicity to be even), and by $k = k(\lambda)$ the expansion factor.

KeyGen($1^\lambda$, $1^k$): Sample a uniform $k \times n$ matrix $B \leftarrow \mathbb{Z}_q^{k \times n}$ and a key pair of a regular public-key encryption scheme $(sk_{PKE}, pk_{PKE}) \leftarrow \text{PKE.KeyGen}(1^\lambda)$. The public key consists of $(B, pk_{PKE})$ and the secret key is set to $sk_{PKE}$.

Enc($pk, m$): To encrypt a $k$-bit message $m \in \{0, 1\}^k$, sample a uniform randomness vector $r \leftarrow \mathbb{Z}_q^k$ a noise vector $e \leftarrow \chi^k$ and return the ciphertext

$$c = (\text{PKE.Enc}(pk_{PKE}, r), B \cdot r + q/2 \cdot m + e).$$

Dec($sk, c$): Parse $c$ as $(c_{PKE}, c_1, \ldots, c_k)$ and recover the random coins $r = \text{PKE.Dec}(sk_{PKE}, c_{PKE})$. Let $b_i$ be the $i$-th row of $B$. For $i = 1 \ldots k$, compute $m_i = \text{MSB}(c_i - b_i \cdot r)$, where MSB returns the most significant bit of the input integer. Output $m = (m_1, \ldots, m_k)$.

Clearly, the scheme is perfectly correct since

$$(\text{MSB}(c_1 - b_1 \cdot r), \ldots, \text{MSB}(c_k - b_k \cdot r)) = (\text{MSB}(q/2 \cdot m_1 + e_1), \ldots, \text{MSB}(q/2 \cdot m_k + e_k))$$

$$= (\text{MSB}(q/2 \cdot m_1), \ldots, \text{MSB}(q/2 \cdot m_k))$$

$$= (m_1, \ldots, m_k) = m.$$

Extended Encryption. It is not hard to see that the scheme presented above is (bounded) additively homomorphic over $\mathbb{Z}_q^k$. To lift the class of computable functions to all linear functions, we adopt the standard trick of encrypting the message multiplied by all powers of two $(1, 2, \ldots, 2^{\text{log}(q)})$. For convenience, we define the following augmented encryption algorithm.
**ExtEnc** \((pk, m)\): On input an \(\ell\)-dimensional message \(m \in \mathbb{Z}_q^\ell\), define \(m^{(i,j)}\) as the \(k\)-dimensional vector 
\[(0, \ldots, 0, m_j, 0, \ldots, 0)^T\] 
that contains \(m_j\) in the \(i\)-th position and 0 everywhere else. Define 
\[
M = \begin{bmatrix}
m^{(1,1)} & \cdots & m^{(1,1)} & 2 & \cdots & 0^k \\
0^k & \cdots & 0^k \\
\vdots & \ddots & \ddots & \vdots \\
0^k & \cdots & 0^k & m^{(k,\ell)} & 2^\log(q)
\end{bmatrix} \in \mathbb{Z}_q^{k \times \ell \cdot k \cdot \log(q)}.
\]

Sample a uniform randomness matrix \(R \leftarrow \mathbb{Z}_q^{n \times k \cdot \log(q)}\) and a uniform noise matrix \(E \leftarrow \chi^{k \times \ell \cdot k \cdot \log(q)}\). Compute
\[
C = B \cdot R + M + E
\]
and return the ciphertext \((PKE.\text{Enc}(pk_{PKE}, R), C)\).

Decryption works, as before, by recovering \(R\) from the public-key encryption scheme and then decrypting \(m\) component-wise.

**Almost-Everywhere Dense Encryption.** For convenience, we also define an alternative encryption algorithm in the following. Note that the encryption algorithm does not take as input any message, instead it encrypts a uniform \(k\)-bit binary vector.

**DenseEnc** \((pk)\): Sample a uniform randomness vector \(r \leftarrow \mathbb{Z}_q^n\) and return the ciphertext 
\[
c = (c_{PKE}, c_1, \ldots, c_k) = (PKE.\text{Enc}(pk_{PKE}, r), B \cdot r + u).
\]

where \(u \leftarrow \mathbb{Z}_q^k\).

We highlight two facts about this algorithm that are going to be important for our later construction: (i) The decryption algorithm works for both \(\text{Enc}\) and \(\text{DenseEnc}\) algorithms, where the plaintext of \(\text{DenseEnc}\) corresponds to \((\text{MSB}(u_1), \ldots, \text{MSB}(u_k))\). In fact, the scheme satisfies perfect correctness in both cases. (ii) The domain of the elements \((c_1, \ldots, c_k)\) is dense, i.e. the support of the scheme spans the entire vector space \(\mathbb{Z}_q^k\). Since the element \(c_{PKE}\) is small (i.e. independent of \(k\)) for an appropriate choice of the public-key encryption scheme, we refer to such a property as *almost-everywhere* density.

### 3.1 Analysis

Here we argue that our scheme as described above satisfies a few properties of interest and we discuss some suitable instantiations for the underlying building blocks.

**Semantic Security.** First we argue that the scheme satisfies a strong form of semantic security, i.e. the honestly computed ciphertexts are computationally indistinguishable from uniform vectors in \(\mathbb{Z}_q^k\). Semantic security for the extended encryption \(\text{ExtEnc}\) and the dense encryption \(\text{DenseEnc}\) follows along the same lines.

**Theorem 3.1 (Semantic Security)** If \((\text{PKE.KeyGen}, \text{PKE.Enc}, \text{PKE.Dec})\) is semantically secure and the LWE assumption holds, then for all \(\lambda \in \mathbb{N}\) and all messages \(m\) it holds that the following distributions are computationally indistinguishable
\[
(pk, \text{Enc}(pk, m)) \approx (pk, \text{PKE.Enc}(pk_{PKE}, z), u).
\]

where \((sk, pk) \leftarrow \text{KeyGen}(1^\lambda, 1^k), z \leftarrow \mathbb{Z}_q^n\), and \(u \leftarrow \mathbb{Z}_q^k\).

**Proof:** The security of the scheme follows routinely by an invocation of semantic security of the public-key encryption scheme and an invocation of the LWE assumption. \(\square\)
Circuit Privacy. We require that the underlying public-key encryption scheme supports the homomorphic evaluation of linear functions, however we pose no compactness requirements for the evaluated ciphertexts. Instead, we require that the scheme satisfies the following notion of circuit privacy.

Definition 3.2 (Circuit Privacy) A public-key encryption scheme \((\text{PKE.KeyGen, PKE.Enc, PKE.Dec})\) is circuit-private for the circuit class \(\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}\) if for all \(\Pi \in \mathcal{C}_\lambda\), all \((\text{sk}_{\text{PKE}}, \text{pk}_{\text{PKE}})\) in the support of \(\text{PKE.KeyGen}\), and all messages \(m\), it holds that the following distributions are statistically indistinguishable

\[
(\text{pk}, \text{PKE.Enc}(\text{pk}_{\text{PKE}}, \Pi(m))) \approx (\text{pk}, \text{PKE.Eval}(\text{pk}_{\text{PKE}}, \Pi, \text{PKE.Enc}(\text{pk}_{\text{PKE}}, m))).
\]

Many fully-homomorphic encryption schemes are known to satisfy such a notion (see e.g. [BDGM19, DB18, DGIOPP14]). As we do not require compactness, we can even instantiate this building block using a two-round oblivious transfer with statistical sender privacy [PVW08, BD18, BDGM19] and information-theoretic garbled circuits [Kil88, AIK04, IP07].

4 Constructing XiO

In the following we present the construction of XiO from the GSW scheme \((\text{GSW.KeyGen, GSW.Enc, GSW.Eval, GSW.Dec})\) and the packed version of the dual-Regev encryption \((\text{dR.KeyGen, dR.Enc, dR.Eval, dR.Dec})\) as described in Section 3. The construction and the analysis is largely unchanged from [GP20] (which in turn is based on the blueprint of [BD18]), except that we include some extra elements in the XiO scheme to account for the fact that some parts of the dual-Regev ciphertext are not dense.

4.1 Construction

The scheme assumes a long uniform string that is, for convenience, split in two chunks:

- A sequence of randomization vectors \((r_1^1, \ldots, r_{\eta - \log(k)}^n)\) for the GSW scheme \(\text{GSW.PubCoin}\), where each \(r_i \in [\bar{B}, +\bar{B}]^{m \cdot k}\).
- A sequence of dense ciphertexts \(((h_1,1, \ldots, h_1,k), \ldots, (h_{\eta - \log(k),1}, \ldots, h_{\eta - \log(k),k}))\) for packed dual-Regev scheme \(\text{dR.PubCoin}\), where each \((h_{i,1}, \ldots, h_{i,k}) \in \mathbb{Z}_q^k\).

On input the security parameter \(1^\lambda\) and the circuit \(\Pi : \{0,1\}^n \rightarrow \{0,1\}\), the obfuscator proceeds as follows.

Setting the Public Keys: Sample a dual-Regev key pair \((\text{sk}, \text{pk}) \leftarrow \text{dR.KeyGen}(1^\lambda, 1^k)\) and GSW key pair \((\text{sk}, \text{pk}) \leftarrow \text{GSW.KeyGen}(1^\lambda, q)\), where \(q\) is the modulus defined by the dual-Regev scheme. Compute a GSW encryption \(c_{\Pi} \leftarrow \text{GSWEnc}(\text{pk}, \Pi)\) of the binary representation of the circuit \(\Pi\).

Compute a Key Cycle: Compute a GSW encryption of the dual-Regev secret key \(c_{\text{sk}} = \text{GSW.Enc}(\text{pk}, \text{sk})\) and a dual-Regev extended encryption of the GSW secret key

\[
(\tilde{c}_{\text{sk}}, \tilde{C}_{\text{sk}}) = \text{dR.ExtEnc}(\text{pk}, \text{sk}; S),
\]

where \(\text{sk} \in \mathbb{Z}_q^{n+1}\) and \(S \leftarrow \mathbb{Z}_q^{n \times \log(q) \cdot k \cdot (n+1)}\).

Decryption Hints: For all indices \(i \in \{0,1\}^{-\log(k)}\), do the following.

Evaluate the Circuit: Let \(\Phi_i : \{0,1\}^{\Pi} \rightarrow \{0,1\}^k\) be the universal circuit that, on input a circuit description \(\Pi\), returns the \(i\)-th block (where each block consists of \(k\) bits) of the corresponding truth table. Compute

\[
c_i \leftarrow \text{GSW.Eval}(\text{pk}, \Phi_i, c_{\Pi}).
\]

Compute the Encryption Header: Sample a uniform \(r_i \leftarrow \mathbb{Z}_q^n\) and set \(h_{i,0} = \text{PKE.Enc}(\text{pk}_{\text{PKE}}, r_i)\), where \(\text{pk}_{\text{PKE}}\) is the public-key encryption scheme defined by the key generation of dual-Regev.
Compute the Low-Order Bits: Parse the $i$-th block of dR.PubCoin as 
\[(h_{i,1}, \ldots, h_{i,k}) = B \cdot r_i + (u_{i,1}, \ldots, u_{i,k}) \in \mathbb{Z}_q^k\]
for some $(u_{i,1}, \ldots, u_{i,k}) \in \mathbb{Z}_q^k$. Let $\Psi_i : \{0,1\}^\lambda \to \{0,1\}^k$ be circuit that, on input the dual-Regev secret key, computes the decryption of $(h_{i,0}, h_{i,1}, \ldots, h_{i,k})$, i.e.
\[
\Psi_i(pk) = \text{dR.Dec}(sk, (h_{i,0}, h_{i,1}, \ldots, h_{i,k})).
\]
Compute homomorphically the $k$-bit ciphertext $c_{i,\text{MSB}} = \text{GSW.Eval}(pk, \Psi_i, c_{sk})$.

Rerandomize the Ciphertext: Parse the $i$-th block of GSW.PubCoin as $r_i^* \in [-\tilde{B}, +\tilde{B}]^{m-k}$ and compute
\[
c_{i,\text{MSB}}' = c_{i,\text{MSB}} + \text{GSW.ColEnc}(pk, 0^k; r_i^*).
\]

Proxy Re-Encrypt: Define $d_i$ as the GSW ciphertext resulting from the homomorphic sum of $c_{i,\text{MSB}}'$ and $c_i$, i.e. $d_i = c_{i,\text{MSB}}' + c_i$. Observe that $d_i$ is a $k$-bit ciphertext and let $\ell_{i,j}$ be the linear function associated with the decryption of the $j$-th bit. Define $\ell_i = (\ell_{i,1}, \ldots, \ell_{i,k})$ and compute
\[
\bar{c}_i = \bar{C}_{sk} \cdot \text{Bit}(\ell_i) + (h_{i,1}, \ldots, h_{i,k}) \in \mathbb{Z}_q^k
\]
where the function $\text{Bit} : \mathbb{Z}_q^k(n+1) \to \{0,1\}^{\log(q) \cdot k(n+1)}$ is the bit decomposition operator.

Release Hint: Compute the $i$-th decryption hint as
\[
\rho_i = S \cdot \text{Bit}(\ell_i) + r_i \in \mathbb{Z}_q^n.
\]

Output: The obfuscated circuit consists of the public keys $(pk, pk)$, the key cycle $(C_{sk}, C_{sk})$, the GSW encryption of the circuit $c_{\Pi}$, the encryption headers $(h_1, \ldots, h_{\eta - \log(k)})$, and the decryption hints $(\rho_1, \ldots, \rho_{\eta - \log(k)})$.

To evaluate the obfuscated circuit on input $x$, let $i$ be the index of the block of the truth table of $\Pi$ that contains $\Pi(x)$. The evaluator computes $\bar{c}_i$ as specified above (note that all the operations are public, given the information included in the obfuscated circuit) and recovers $\Pi^{(i)}$ (the $i$-th block of the truth table of $\Pi$) by computing
\[
\Pi^{(i)} = \text{MSB}(c_i - B \cdot \rho_i)
\]
where MSB returns the most significant bit of each component of the input vector.

Correctness. To see why the evaluation algorithm is correct, recall that
\[
\bar{c}_i = \bar{C}_{sk} \cdot \text{Bit}(\ell_i) + (h_{i,1}, \ldots, h_{i,k}).
\]
First observe that $(h_{i,0}, h_{i,1}, \ldots, h_{i,k})$ is a ciphertext in the support of $\text{dR.DenseEnc}(pk)$, and in particular
\[
(h_{i,0}, h_{i,1}, \ldots, h_{i,k}) = (\text{PKE.Enc}(pk_{\text{PKE}}, r_i), B \cdot r_i + (u_{i,1}, \ldots, u_{i,k}))
\]
\[
= (\text{PKE.Enc}(pk_{\text{PKE}}, r_i), B \cdot r_i + u_i).
\]
Furthermore, observe that
\[
d_i = c_{i,\text{MSB}}' + c_i
= c_{i,\text{MSB}}' + \text{GSW.ColEnc}(pk, \Pi^{(i)})
= \text{GSW.ColEnc}(pk, (\text{MSB}(u_1), \ldots, \text{MSB}(u_k))) + \text{GSW.ColEnc}(pk, \Pi^{(i)})
= \text{GSW.ColEnc}(pk, (\text{MSB}(u_1) \oplus \Pi_1^{(i)}, \ldots, \text{MSB}(u_k) \oplus \Pi_k^{(i)}))
\]

14
by the evaluation correctness of GSW. By the almost-linear decryption of GSW, it follows that

\[ C_{sk} \cdot \text{Bit}(\ell_i) = B \cdot \tilde{s}_i + \xi_i + \zeta_i + q/2 \cdot \left( \text{MSB}(u_1) \oplus \Pi_1, \ldots, \text{MSB}(u_k) \oplus \Pi_k \right) \]

where \( \xi_i \) is the decryption noise of the packed dual-Regev scheme (i.e. the subset sum of the noise terms of \( C_{sk} \)) and \( \zeta_i \) is the decryption noise of the GSW ciphertext. It follows that \( \|\xi_i\|_\infty \leq B \cdot \log(q) \cdot k \cdot (n + 1) \) and, by Lemma 2.1, \( \|\zeta_i\|_\infty \leq B \) with all but negligible probability. Note that, by linearity we have that \( \tilde{s}_i = S \cdot \text{Bit}(\ell_i) \). Consequently, it holds that

\[
\begin{align*}
\tilde{c}_i &= B \cdot \tilde{s}_i + \xi_i + \zeta_i + q/2 \cdot \left( \text{MSB}(u_1) \oplus \Pi_1, \ldots, \text{MSB}(u_k) \oplus \Pi_k \right) + B \cdot r_i + u_i \\
&= B \cdot (\tilde{s}_i + r_i) + \xi_i + \zeta_i + q/2 \cdot \left( \text{MSB}(u_1) \oplus \Pi_1, \ldots, \text{MSB}(u_k) \oplus \Pi_k \right) + u_i \\
&= B \cdot (\tilde{s}_i + r_i) + q/2 \cdot \Pi(i) + v_i \\
&= B \cdot \rho_i + q/2 \cdot \Pi(i) + v_i
\end{align*}
\]

where \( v_i = u_i + q/2 \cdot \text{MSB}(u_i) + \xi_i + \zeta_i \) and \( \|v_i\|_\infty < q/4 \) with all but negligible probability, over the random choice of \( u_i \). This is because \( d_i \) is statistically close to a fresh GSW encryption of \( (\text{MSB}(u_1), \ldots, \text{MSB}(u_k)) \oplus \Pi(i) \), by Lemma 2.9. Therefore we have that

\[
\begin{align*}
\text{MSB}(c_i - B \cdot \rho_i) &= \text{MSB}(B \cdot \rho_i + q/2 \cdot \Pi(i) + v_i - B \cdot \rho_i) \\
&= \text{MSB}(q/2 \cdot \Pi(i) + v_i) \\
&= \text{MSB}(q/2 \cdot \Pi(i)) \\
&= \Pi(i)
\end{align*}
\]

with the same probability.

4.2 Analysis

In the following we show that our XiO scheme satisfies the notion of security for indistinguishability obfuscation.

**Theorem 4.1 (XiO Security)** If the GSW scheme \((\text{GSW.KeyGen}, \text{GSW.Enc}, \text{GSW.Eval}, \text{GSW.Dec})\) and the packed dual-Regev scheme \((\text{dR.KeyGen}, \text{dR.Enc}, \text{dR.Eval}, \text{dR.Dec})\) are 2-circular SRL secure, then the XiO scheme as described above is secure.

**Proof:** We prove the scheme via a series of hybrid experiments. The proof follows closely the argument of [GP20] and it is reported here for completeness.

- **Hybrid \( H_0 \):** This is the original obfuscation of the circuit \( \Pi_0 \).
- **Hybrid \( H_1 \):** This hybrid is identical to the previous one, except that for all \( i \in \{0, 1\}^{n-\log(k)} \) we sample \( c'_{i,\text{MSB}} \) as

\[
\begin{align*}
c'_{i,\text{MSB}} &= \text{GSW.ColEnc}(pk, (\text{MSB}(u_{i,1}), \ldots, \text{MSB}(u_{i,k})); r^*_{i})
\end{align*}
\]

where \( r^*_{i} \leftarrow [-\tilde{B}, +\tilde{B}]^{m \cdot k} \). Let \( r_{\Psi,i} \) be the random coins of \( c_{i,\text{MSB}} \) (as computed in the original protocol). We additionally set the \( i \)-the block of the GSW.PubCoin to \( r^*_{i} - r_{\Psi,i} \). Statistical indistinguishability with respect to the previous hybrid follows from an invocation of Lemma 2.9.
• Hybrid $\mathcal{H}_2$: This hybrid is identical to the previous one, except that for all $i \in \{0,1\}^{n-\log(k)}$ we set
\[
(h_{i,1}, \ldots, h_{i,k}) = B \cdot r_i + (u_{i,1}, \ldots, u_{i,k})
\]
where $(u_{i,1}, \ldots, u_{i,k}) \leftarrow \mathbb{Z}_q^k$. The only difference with respect to the previous hybrid is that the obfuscator knows the values $(u_{i,1}, \ldots, u_{i,k})$ ahead of time. However, the two distributions are identical to the eyes of the distinguisher and therefore the change here is only syntactical.

• Hybrid $\mathcal{H}_4$: In this hybrid we generate, for all $i \in \{0,1\}^{n-\log(k)}$, $\tilde{c}_i$ as follows
\[
\tilde{c}_i = B \cdot t_i + \xi_i + \zeta_i + q/2 \cdot \left( \text{MSB}(u_1) \oplus \Pi_1(i), \ldots, \text{MSB}(u_k) \oplus \Pi_k(i) \right) + u_i
\]
where $t_i \leftarrow \mathbb{Z}_q^n$. Here $\xi_i$ and $\zeta_i$ denote the decryption noises of $\tilde{C}_{sk}$ and $d_i$, respectively. Furthermore, we set $(h_{i,1}, \ldots, h_{i,k}) = \tilde{c}_i - C_{sk} \cdot \text{Bit}(\ell_i)$ and $h_{i,0} = \text{PKE.Enc}(\text{pk}_{\text{PKE}}, t_i - S \cdot \text{Bit}(\ell_i))$. Note that $c_{i,\text{MSB}}$ is fixed (and in particular is independent of $h_{i,0}$) and thus the above variables are always well defined. In fact, observe that
\[
(h_{i,1}, \ldots, h_{i,k}) = \tilde{c}_i - C_{sk} \cdot \text{Bit}(\ell_i)
\]
where $t_i \leftarrow \mathbb{Z}_q^n$ and $w_i \leftarrow [-q/4, +q/4]^k$. Note that we can bound $\|\xi\|_\infty \leq B \cdot \log(q) \cdot k \cdot (n + 1)$. Furthermore, we have that $\|\zeta\|_\infty \leq B$ with all but negligible probability over the random choice of $r_i^*$, by Lemma 2.1. Statistical indistinguishability follows from another application of Lemma 2.1.

• Hybrid $\mathcal{H}_5$: Here we define, for all $i \in \{0,1\}^{n-\log(k)}$, the circuit $\Gamma_i : \mathbb{Z}_q^{n \times (n+1) \cdot k \cdot \log(q)} \rightarrow \mathbb{Z}_q^n$ as the following circuit
\[
\Gamma_i(x) = t_i - X \cdot \text{Bit}(\ell_i).
\]
Then we set $h_{i,0} = \text{PKE.Enc}(\text{pk}_{\text{PKE}}, \Gamma_i, \tilde{c}_{sk})$. To see why the hybrids are statistically close, recall that $\tilde{c}_{sk} = \text{PKE.Enc}(\text{pk}_{\text{PKE}}, S)$, where $S$ are the random coins used in the encryption of $sk$. Then
\[
\text{PKE.Enc}(\text{pk}_{\text{PKE}}, \Gamma_i, \text{PKE.Enc}(\text{pk}_{\text{PKE}}, S)) \approx \text{PKE.Enc}(\text{pk}_{\text{PKE}}, t_i - S \cdot \text{Bit}(\ell_i))
\]
by the statistical circuit privacy of the public-key encryption scheme. Jumping ahead to our modified scheme, note that the same hybrid step would have succeeded (with the same argument) if we were given a GSW encryption of $S$.

• Hybrid $\mathcal{H}_6$: In this hybrid we compute $c_0$ as an encryption of $\Pi_1$ instead of $\Pi_0$. Note that we no longer use the secret key of either of the encryption schemes to obfuscate the circuit, besides the encrypted key cycle. The random coins $\text{GSW PubCoin}$ (which we set to $r_{r_{\Psi,i}}$) can be modelled as an SRL leakage and therefore security follows from a standard reduction to the 2-circular SRL security of GSW and packed dual-Regev: The experiment is defined with respect to the set of circuits $(\Psi_1, \ldots, \Psi_{\eta-\log(k)})$ (as defined in $\mathcal{H}_4$) and the reduction uses the extra ciphertexts $(c_{1,1}, \ldots, c_{\eta-\log(k)})$ to compute $c_{i,\text{MSB}}$ to be $c_i + (0^n, \text{MSB}(u_{i,1}), \ldots, 0^n, \text{MSB}(u_{i,k}))$. The rest of the information is already given by the SRL experiment or recomputed as in $\mathcal{H}_4$.

• Hybrids $\mathcal{H}_7 \ldots \mathcal{H}_{11}$: In this series of hybrids we undo all changes that we did in $\mathcal{H}_5 \ldots \mathcal{H}_4$. The statistical indistinguishability follows by the same arguments as before. Note that $\mathcal{H}_{11}$ is the original obfuscation of $\Pi_1$. This concludes our proof.
4.3 Parameters

The analysis of our scheme sets two constraints on the noise growth of the encryption schemes. The first application of the smudging Lemma requires that the noise bound $B$ is exponentially larger than the noise bound on evaluated GSW ciphertexts $\hat{B}$, i.e. \( 1 \leq \hat{B} \leq 2^B \). The second application requires that \( \frac{q}{4} \geq 2^\lambda \cdot B \) and \( \frac{q}{4} \geq 2^\lambda \cdot B \cdot \log(q) \cdot k \cdot (n+1) \), to ensure that the term $u_i$ (minus its most significant bit) properly floods the noise terms of the GSW and dual-Regev ciphertexts. Note that, for an appropriate choice of parameters of the GSW scheme, condition (3) is already implied by conditions (1) and (2) and therefore all we need ensures is that these two conditions are satisfiable. It is not hard to see that the above pair of constraints can be satisfied by setting the modulo-to-noise ratio of the LWE assumption to be super-polynomial.

To make the XiO scheme compressing, we set $k = 2^{\eta/4}$. We analyze the size of each component of the obfuscated circuit in the following:

- The size of the public keys \((pk, pk)\) is linear in $k$ and thus can be bounded by $2^{n/4} \cdot \text{poly}(\lambda)$.
- The size of the key cycle \((C_{sk}, c_{sk})\) can be bounded by $k^2 \cdot \text{poly}(\lambda) = 2^{\eta/2} \cdot \text{poly}(\lambda)$.
- The GSW encryption of the circuit $c_{\Pi}$ is of size $|\Pi| \cdot \text{poly}(\lambda)$.
- The size of an encryption header $h_i$ is $\text{poly}(\lambda)$ (and in particular independent of $k$ by the almost-everywhere density of dual-Regev) and the size of each decryption hint $\rho_i$ is also $\text{poly}(\lambda)$ (by the randomness succinctness of dual-Regev). It follows that the total size of the encryption headers and the decryption hints can be bounded by $2^{\eta \cdot \log(k)} \cdot \text{poly}(\lambda) = 2^{\eta k/4} \cdot \text{poly}(\lambda)$.

The summation of the above terms is sublinear in $\text{poly}(|\Pi|, \lambda) \cdot 2^{\eta(1-\varepsilon)}$ (for $\varepsilon > 0$) and therefore the XiO scheme satisfies non-trivial efficiency. Note that we omitted the public parameters $\text{GSW}, \text{PubCoin}$ and $\text{dR}, \text{PubCoin}$ from the analysis since they are uniformly at random and thus can be computed in the pre-processing stage of the obfuscator.

4.4 Randomness-Key Circularity

As it is described above, our scheme assumes the 2-circular SRL security of GSW and the variant of packed dual-Regev (presented in Section 3). While SRL security can be based on the plain LWE assumption for GSW alone, we conjecture that it is retained also in the presence of a 2-key cycle.

It is instructive to discuss why a key cycle is needed: On the one hand we need to key-switch the GSW encryption into a packed dual-Regev ciphertext, which has succinct randomness and therefore short decryption hints. This requires us to encrypt the GSW secret key under the dual-Regev encryption. On the other hand, we also need to recover the most significant bits of the vector $u_i$ (component-wise) to make sure that the smudging noise does not interfere with the correctness of the scheme. This is done by parsing \((h_{i,0}, h_{i,1}, \ldots, h_{i,k})\) as a dense ciphertext of packed dual-Regev and computing the decryption homomorphically (thus the need to encrypt the dual-Regev secret key under GSW).

We observe that we can slightly tweak the XiO scheme to weaken the circularity assumption. Recall that the packed dual-Regev decryption works by decrypting the randomness $r_i$ from the term $h_{i,0}$ and uses it to recover the message. Our idea is to bypass the first step by simply including in the obfuscated circuit a GSW encryption of the randomnesses \((r_1, \ldots, r_{\eta \cdot \log(q)})\). With this modification we can omit from the obfuscated circuit both the GSW encryption of the secret key $\text{GSW.Enc}(pk, sk)$ and the encryption headers \((h_{1,0}, \ldots, h_{\eta \cdot \log(q), 0})\). However, we need to add a GSW encryption of $S$ (the randomness used to compute $C_{sk}$) in order to simulate the GSW encryption of $r_i$ in the proof. Thus, the size of the obfuscated circuit is (asymptotically) identical. The security argument follows along the same lines of what already shown, except that the computational step boils down to a randomness-key circularity assumption. More precisely, we assume that SRL security holds in the presence of the following cycle:

\[
(\text{GSW.Enc}(pk, S), C_{sk})
\]
where \((\bar{c}_{sk}, \bar{C}_{sk}) = dR.\text{ExtEnc}(pk, sk; S)\). We stress that such an assumption is \textit{strictly weaker} than assuming SRL security of a 2-key cycle, since we could have used the homomorphism of GSW and the randomness recoverability of dual-Regev to compute a GSW encryption of \(S\) homomorphically.

References


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


