An Alternative View of the Graph-Induced Multilinear Maps

Yilei Chen*

February 29, 2016

Abstract

In this paper, we view multilinear maps through the lens of "homomorphic obfuscation". In specific, we show how to homomorphically obfuscate the kernel-test and affine subspace-test functionalities of high dimensional matrices. Namely, the evaluator is able to perform additions and multiplications over the obfuscated matrices, and test subspace memberships on the resulting code. The homomorphic operations are constrained by the prescribed data structure, e.g. a tree or a graph, where the matrices are stored. The security properties of all the constructions are based on the hardness of Learning with errors problem (LWE). The technical heart is to "control" the "chain reactions" over a sequence of LWE instances.

Viewing the homomorphic obfuscation scheme from a different angle, it coincides with the graph-induced multilinear maps proposed by Gentry, Gorbunov and Halevi (GGH15). Our proof technique recognizes several "safe modes" of GGH15 that are not known before, including a simple special case: if the graph is acyclic and the matrices are sampled independently from binary or error distributions, then the encodings of the matrices are pseudorandom.

*Boston University. cheny1@bu.edu. Supported by US NSF grants CNS-1012798, CNS-1012910 and AF-1218461. Part of the research conducted while visiting Tel Aviv University funded by the Check Point Institute for Information Security .

Contents

1 Introduction

This paper focus on the following two cryptographic creatures: (1) Obfuscators for evasive function families; (2) Cryptographic multilinear maps. The motivation is to construct (possibly subclasses of) them based on the hardness of Learning with errors problem.

An *obfuscator* is a compiler that takes a program P, outputs an "opaque" program $\mathsf{Obf}(P)$ which preserves the functionality but is unintelligible. The strongest notion of obfuscation is called "Virtual Black-Box", requiring that the presence of the obfuscated code is no better than given black-box access to the functionality [Had00, BGI+12]. A function family is called *evasive*, if for every input in the domain, all but negligibly many functions evaluate to 0. The study of obfuscators for general evasive functions is initiated in the work of Barak, Bitansky, Canetti, Kalai, Paneth and Sahai [BBC+14].

The concept of *cryptographic multilinear maps* is introduced by Boneh and Silverberg [BS03]. Roughly speaking, it allows the evaluator to perform computations homomorphically over the encoded values, and test zero identities. The minimum security requirement is to hide the plaintexts (i.e. one-wayness). And one may hope to generalize the hard problems in the (bi)linear world to the multilinear setting, such as the Diffie-Hellman problem. The first candidate multilinear maps was constructed by Garg, Gentry and Halevi from ideal lattices [GGH13a] (henceforth GGH13). The realization more-or-less deviates from the "dream version" envisaged by [BS03]; as it is also called "graded encodings", a name that is closer to its nature.

Since the birth of GGH13, we have witnessed general-purpose obfuscators constructed from multilinear maps, most notably a candidate indistinguishability obfuscator [GGH+13b] built upon GGH13. Together with the second candidate multilinear maps proposed by Coron, Lepoint, Tibouchi [CLT13], and more candidate obfuscators [BR14, BGK+14, PST14, BCKP14, AGIS14, Zim15, AB15, GLSW15] based on idealized abstractions or heuristic assumptions over GGH13 or CLT13. The third candidate multilinear maps is proposed by Gentry, Gorbunov and Halevi [GGH15], together with a candidate branching program obfuscator. Although the three candidate multilinear maps have different interfaces, the ideas behind all of these candidate obfuscators are similar, namely to hide the program inside the encodings, let the input select the encoded puzzle pieces, evaluate the pieces of encodings homomorphically according to certain restrictions (typically used to prevent partial evaluation and mixed-input attacks), finally "zero-test" the encoding on the "top level" and get the output.

However, on the security properties of the three candidate multilinear maps, no specific reductions from any ("standard" or "non-standard") cryptographic assumptions were given. Instead, the original papers each explained the intuitions behind the design, and gave several cryptanalyses, demonstrating the (in)effectiveness of some attacks they had thought about. With the developing attentions on these candidates, more vulnerabilities of them are discovered. For GGH13, some weakness given encodings of zero was "confessed" in the original paper [GGH13a]. More attacks to the applications of GGH13 were conducted by Hu and Jia [HJ15]. An attempt of fixing GGH13 was proposed but failed [BGH+15]. CLT13 was conjectured to be safe with encodings of zeros [CLT13], proved wrong by Cheon, Han, Lee, Ryu and Stehlé [CHL+15]; temporally immunized [GGHZ14, BWZ14], exploited [CLT14]; further exploited [CGH+15]; "reobfuscated" [CLT15], "cracked" [CFL+16] ...[1] The implications of the attacks for the candidate obfuscators vary. Some are broken, e.g. the simple variants of [Zim15, AB15]; some don't have candidate multilinear maps to base on, e.g. [GLSW15][2]. The others "cannot sleep well" [Kil88, first paragraph].

[1] "..." absorbs the developing stories on this topic, including the very recent announcements of cryptanalyses of GGH13 [ABD16, CJL16, MSZ16]. See `https://eprint.iacr.org` for the daily updates.

[2] Using the simple versions of [Zim15, AB15] over CLT13 to obfuscate certain functions such as point functions may accumulate enough low-level encodings of zero slots. As a consequence, the attacker may be able to recover all the private parameters of CLT13 [CHL+15, CGH+15]. For [GLSW15], the "subgroup elimination" assumption is not known to be satisfied by any of the candidates. However, it doesn't imply that the concrete instantiation (on one of the candidate multilinear maps) is necessarily broken.

1.1 Our contributions in a nutshell

In this work, we view multilinear maps through the lens of *homomorphic obfuscations*. The main results are obtained via the following steps:

1. We start from obfuscating a special evasive function family, namely the kernel-test and affine subspace-test functionalities for matrices sampled from certain distributions (for example, uniformly random, or binary with sufficient min-entropy in each row).

2. A step further, we show how to obfuscate matrices homomorphically, which enables the evaluator to add and multiply the obfuscated matrices, and perform subspace-tests on the resulting code. The homomorphic operations are constrained by some prescribed data-structures, e.g. a tree or a directed graph, where the matrices are stored.

3. The construction of homomorphic obfuscator coincides with the graph-based multilinear maps [GGH15]. In fact, the proof technique can be used to identify the "safe modes" of GGH15. The easiest-to-state safe mode is when the graph is acyclic and all the matrices are sampled independently from binary or error distributions. In this case the encodings are pseudorandom.

The security of all the constructions above are based on the hardness of Learning with errors, which is known to be as hard as several worst-case lattices problems [Reg05]. The constructions and proof techniques can be used to homomorphically obfuscate ideals of cyclotomic rings, assuming the hardness of RingLWE [LPR13a]. The technical heart is the formalization of "controlled chain reactions" over a sequence of LWE instances.

1.2 How to obfuscate matrix predicates

We start from obfuscating a matrix, and allow the evaluator to test if an input vector lives in the (co)kernel[3] or its affine subspaces. More concretely, a *matrix predicate obfuscator* takes a matrix $\mathbf{M} \in \mathbb{Z}_q^{\ell \times n}$, output $\mathsf{Obf}(\mathbf{M})$, which allows the evaluator to test whether an input vector $\mathbf{s} \in \mathbb{Z}_q^{\ell}$ satisfies $\mathbf{s}^T\mathbf{M} = \mathbf{0}^T$, or $\mathbf{s}^T\mathbf{M} = \mathbf{v}^T$ for a vector $\mathbf{v}$ either chosen by the evaluator or hidden, without learning $\mathbf{M}$.

Linear transformations are essential to the mathematics behind science and engineering. The applicability of "matrices in virtual-black-boxes" is only bounded by the imagination of human beings. If Schrödinger had such a box, then the fate of his cat (a vector in the "ket") would be kept a secret unless being measured (by a linear operator) in the designated way. In fact, a quantum money scheme proposed by Aaronson and Christiano [AC12] explicitly uses VBB obfuscation for subspace membership-testing functionalities.

As another example, obfuscating the "partial decryption" functionality of somewhat homomorphic encryptions is a reasonable approach to obtain multilinear maps, and the decryption algorithms are usually close to linear. In the lattice-based "BGN"-type homomorphic encryption scheme [GHV10], the decryption algorithm is simply performing matrix multiplications; the decryption algorithms of the leveled-homomorphic encryption schemes, e.g. [BV11, GSW13], are computing inner-product of the secret key and the ciphertext plus a threshold test. Similar observations were made in [GVW15] where the authors constructed predicate encryption from inner-product functional encryption and FHE. If the obfuscators for subspace membership predicates are constructed and are composable with the partial decryption algorithms for FHE, then we might be able to get "function-hiding" predicate encryptions.

[3] The scheme syntactically supports cokernel-test but not kernel-test due to our (fortunately or unfortunately) irreversible choice of lattice notations, namely LWE instances with high dimensional secrets are written as $\mathbf{MA} + \mathbf{E}$. Using the transposed notation $\mathbf{A}^T\mathbf{M} + \mathbf{E}^T$ (i.e. transpose $\mathbf{A}, \mathbf{E}$ without changing $\mathbf{M}$) immediately gives kernel-testing. In the rest of the paper we seldom specify the "left" and "right" for the different types of ideals of non-commutative rings.

Our construction from LWE. The starting point of our construction is the point obfuscator (more precisely, the "vector" obfuscator) from learning with errors [GKPV10]. To obfuscate a vector $\mathbf{s} \in \mathbb{Z}_q^n$, let it be the secret of LWE instances, namely sample a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, a small error vector $\mathbf{e}^T \in \chi^m$, compute $\mathbf{y} = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T$. Output $(\mathbf{y}, \mathbf{A})$ as the obfuscated point function $\mathsf{Obf}(\mathsf{Point}_{\mathbf{s}}(\cdot))$. To check if $\mathbf{s}$ is the correct input, compute $\|\mathbf{s}^T\mathbf{A} - \mathbf{y}^T\|$ and output yes if the result is small.

A slight extension yields an obfuscator for matrix predicates. For a matrix $\mathbf{M} \in \mathbb{Z}_q^{\ell \times n}$ to be obfuscated, sample a random $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m}$, error matrix $\mathbf{E} \leftarrow \chi^{\ell \times m}$, compute $\mathbf{A}_1 := \mathbf{M}\mathbf{A}_0 + \mathbf{E}$. The obfuscated matrix predicate $\mathsf{Obf}(P_{\mathbf{M}}(\cdot,\cdot))$ is composed of $\mathbf{A}_1$ and $\mathbf{A}_0$. To test whether a vector $\mathbf{s}$ and a target affine subspace $\mathbf{v}$ satisfies $\mathbf{s}^T\mathbf{M} = \mathbf{v}^T$, compute $\|\mathbf{s}^T\mathbf{A}_1 - \mathbf{v}^T\mathbf{A}_0\|$ and output 1 if the result is small. A caveat here is that the norm of $\mathbf{s}$ has to be small to guarantee correctness. In other words, the LWE-based construction only supports subspace-test for small input vectors.

In terms of security, as long as $\mathbf{M}$ is sampled from a distribution where decisional LWE is hard, $(\mathbf{A}_0, \mathbf{M}\mathbf{A}_0 + \mathbf{E})$ is pseudorandom, i.e. the obfuscator satisfies Virtual Black-Box security. Known distributions on the secrets that are hard for LWE include the uniform distribution over $\mathbb{Z}_q^{\ell \times n}$ [Reg05], error distributions [ACPS09], binary distributions with sufficient min-entropy [GKPV10] in each row, and more (cf. Preliminary). In other words, the applicability of our result is not associated with a specific distribution or form of LWE, but rather depends on the coverage of the *LWE library*. So as the results to be introduced in the next few sections. We also mention some interesting families of matrices that are not covered by our construction (cf. Remark 3.6).

1.3 Obfuscate matrices homomorphically from "cryptographic controlled chain reaction"

We further consider *homomorphic obfuscations*. Namely, the evaluator, given the obfuscated code $\mathsf{Obf}(f)$, is able to produce the obfuscated $\mathsf{Obf}(g \circ f)$ for g of its own choice, without "reverse engineering" $\mathsf{Obf}(f)$. Homomorphism over obfuscated functions is a concept that hasn't been explored much before. A recent work of Ananth, Jain and Sahai [AJS15] on "patchable obfuscation" has a flavor of what they called "semi-private homomorphism". Beyond the ideology of "computing on obfuscated functions rather than encrypted data", homomorphic obfuscators have realistic features that are not implied by the normal variants. In the context of matrix predicate obfuscation, for example, if the evaluator is able to produce $\mathsf{Obf}(P_{\mathbf{M}_1+\mathbf{M}_2})$ from $\mathsf{Obf}(P_{\mathbf{M}_1})$ and $\mathsf{Obf}(P_{\mathbf{M}_2})$, then it is able to test if a vector is the kernel of the sum without learning the kernel of either $\mathbf{M}_1$ or $\mathbf{M}_2$.

For the LWE-based obfuscator of matrix predicate P, the obfuscated code $\mathsf{Obf}(P_{\mathbf{M}})$ is composed of two parts $(\mathbf{A}_1, \mathbf{A}_0)$, which can be thought as the "outer" and "inner" masks of the encoded matrix. Additive homomorphism can be supported if the two matrices are obfuscated with the same inner-mask $\mathbf{A}_0$ (homomorphic addition is performed by adding the outer masks and preserve the inner mask). The potential of supporting multiplicative homomorphism comes from the following observation: the outer mask $\mathbf{A}_1$ is pseudorandom and can be reused as the inner mask of the next matrix. To elaborate, in order to obfuscate $\mathbf{M}_1 \in \mathbb{Z}_q^{\ell_1 \times \ell_0}$, $\mathbf{M}_0 \in \mathbb{Z}_q^{\ell_0 \times n}$ and also obtain the code of $\mathbf{M}_1\mathbf{M}_0$ "for free", one samples $\mathbf{A}_0 \leftarrow U(\mathbb{Z}_q^{n \times m})$, error $\mathbf{E}_0 \leftarrow \chi^{n \times m}$, computes $\mathbf{A}_1 := \mathbf{M}_0\mathbf{A}_0 + \mathbf{E}_0$. Then samples error $\mathbf{E}_1 \leftarrow \chi^{\ell_0 \times m}$, computes $\mathbf{A}_2 := \mathbf{M}_1\mathbf{A}_1 + \mathbf{E}_1$. To see $(\mathbf{A}_2, \mathbf{A}_0)$ is the obfuscation of $P_{\mathbf{M}_1\mathbf{M}_0}$, let's write down the expression of kernel-test as an example:

$$\mathbf{s}^T\mathbf{A}_2 = \mathbf{s}^T\mathbf{M}_1\mathbf{A}_1 + \mathbf{s}^T\mathbf{E}_1 = \mathbf{s}^T\mathbf{M}_1\mathbf{M}_0\mathbf{A}_0 + (\mathbf{s}^T\mathbf{M}_1\mathbf{E}_0 + \mathbf{s}^T\mathbf{E}_1)$$

Correctness holds as long as both $\mathbf{s}$ and $\mathbf{M}_1$ are small. This means that the scheme only supports small left multipliers. Pseudorandomness of $\mathbf{A}_1$ and $\mathbf{A}_2$ holds if $\mathbf{M}_0$ and $\mathbf{M}_1$ are sampled independently from LWE-hard distributions.

The template described above imposes a structure on the allowed homomorphic operation, namely homomorphic operations cannot be performed arbitrarily, but have to follow some rules defined by the structure over

the encoded elements. The most expressive structure that supports the template is a "tree", with the matrices stored on the branches of the tree. The "masks" are stored in the nodes, with inner mask in the parent node, outer mask in the child. Addition can be performed over branches with the same parent node; multiplications can be performed over the branches that are connected and form a longer branch.

The security of the tree-induced obfuscation scheme can be seen as relying on the quality of "chain reactions" over LWE instances. The chain reactions are "controlled" if the prior matrix is "fully obfuscated", thus leaving the outer mask pseudorandom and reusable for the next level; on the other hand, if the prior matrix is sampled from a distribution that is "learnable" with error, then the resulting outer mask is not pseudorandom, thus crashing the "chain reactions" and make the protection void for the subsequent matrices on the subtree.

Our main result states that if the matrices are sampled independently from LWE-hard distributions, organized according to a tree with arbitrary topology, then the construction above achieves virtual black-box security for the matrices "hung" on the tree (cf. Theorem 4.7). When the matrices are sampled correlated, identifying the border of "controlled" and "out-of-control" chain reactions is not easy, and many cases live in the unknown status, according to the current coverage of "LWE library". Instead of attempting to outline some rules that are very limited, we choose to show various examples that are VBB obfuscated, not known to be secure or broken, or clearly insecure if obfuscated by our scheme.

1.4 Relation to the graph-induced multilinear maps

Our tree-induced homomorphic obfuscator is spiritually close to the graph-induced multilinear maps proposed by Gentry, Gorbunov and Halevi [GGH15], in the sense that they both impose some restrictions on the allowed homomorphic operations, and "kernel-test" can be thought as the decomposed zero-test. It turns out that GGH15 can be viewed as a graph-induced homomorphic obfuscation scheme. Our technique of "controlled chain reaction" can be used to identify some "safe modes" of GGH15, for which we now elaborate.

Similar to GGH13 and CLT13 which evolve from somewhat homomorphic encryption schemes (GGH13 from NTRU [HPS98], CLT13 from [vDGHV10]), GGH15 is originated from the homomorphic encryption scheme proposed by Gentry, Sahai and Waters [GSW13]. Briefly recall the trace of evolution from GSW to GGH15:

$$\mathbf{a}^T\mathbf{D} = \mu\mathbf{a}^T + \mathbf{e}^T \quad \Rightarrow \quad \mathbf{AD} = \mathbf{MA} + \mathbf{E} \quad \Rightarrow \quad \mathbf{A}_1\mathbf{D} = \mathbf{Y} = \mathbf{MA}_0 + \mathbf{E}$$

where the first equation represents the format of GSW FHE: the ciphertext is the matrix $\mathbf{D}$, the secret-key $\mathbf{a}$ is the approximate eigenvector, and the plaintext μ corresponds to the approximate eigenvalue. GGH15 twists the FHE by increasing the dimension of the secret-key and plaintext. Namely the eigenvector becomes an approximate eigenspace, and the encoded value becomes a matrix $\mathbf{M}$. In addition, they use different $\mathbf{A}$ matrices on the left and right hand side (denote as $\mathbf{A}_1$, $\mathbf{A}_0$), sampled independently with trapdoors. To encode $\mathbf{M}$ w.r.t. $\mathbf{A}_1$ and $\mathbf{A}_0$, calculate $\mathbf{Y} = \mathbf{MA}_0 + \mathbf{E}$ and use the trapdoor of $\mathbf{A}_1$ to sample a small matrix $\mathbf{D}$ s.t. $\mathbf{A}_1\mathbf{D} = \mathbf{Y}$. Then publish $\mathbf{D}$ as the "official" encoding of $\mathbf{M}$.

The template above can be extended to encode a bunch of matrices and allow homomorphic computations. Unlike in GSW FHE that every ciphertexts can add and multiply the others (as long as the noise doesn't overflow), here, due to the "separation" of $\mathbf{A}$ matrices, correctness of additions and multiplications are only guaranteed when they are encoded according to certain topology. The structure presented by GGH15 is a directed acyclic graph, where matrices to be encoded are stored on the paths, the $\mathbf{A}$ matrices are sitting in the vertices. To some extent, readers can think of graphs as being obtained by "sticking" the leaves of the tree. Addition can be performed for the paths with the same tail and head, multiplication is allowed when two paths are connected and form a longer path. These operations are instantiated by simply adding and multiplying the $\mathbf{D}$ matrices. Zero-test is enabled by publishing the $\mathbf{A}$ matrix stored on the tail of the path. The evaluator simply calculates $\mathbf{Y} = \mathbf{AD}$ and see if it is small.

Note that the zero-test step reduces the computation back to the tree structure, namely $\mathbf{Y}$ is exactly the outer mask. By publishing all the $\mathbf{A}$ matrices, one can recover all the $\mathbf{Y}$ matrices thus performing the subspace-test functionality as was in the tree variant. One can think of the graph-induced "multilinear maps" and "homomorphic obfuscators" as two different ways of viewing the same object.

Security. Involving the trapdoor sampling techniques makes it more complicated when arguing security, since the resulting $\mathbf{D}$ matrices may leak information about trapdoors and secrets. The original paper of [GGH15] didn't give any claim of security, even for the one-wayness of the encodings. In fact, they gave an explicit example that is insecure, namely the encoding of zero matrices are "weak" trapdoors.

However, it is still hopeful that for some "nice" distributions of matrices, the construction is secure. Our main contribution to the graph-induced construction is to handle the trapdoors in the chaining technique. For some distributions, it is possible to indistinguishably "turns off" the trapdoors along the paths in the graph. More concretely, we sequentially applying the discrete Gaussian sampling lemma developed in the work of Gentry, Peikert and Vaikuntanathan [GPV08], which says if the target is uniform, then discrete Gaussian sampling can be simulated without knowing the trapdoor. The easiest-to-state safe mode is that for directed acyclic graph with arbitrary topology, the matrices on the graph is obfuscatable if they are sampled independently from LWE-hard distributions (cf. Theorem 5.6). We also give some examples where the matrices are sampled correlated, covering "controllable", "broken down", and "unknown" cases.

A brief comparison of trees and graphs; cyclic or acyclic. We give a brief comparison of tree and graph-induced homomorphic obfuscator, in terms of their expressibility and security (note that without considering security, graphs are strictly more expressive than trees, since a tree itself is an acyclic graph). Roughly speaking, the advantage of graphs is that they support random branching programs with arbitrary polynomial depth and width, where trees are very limited on the depth. On the other hand, trees support public sampling of encodings with some utilities, but for graphs we don't know how to publicly sample encodings safely. For a concrete comparison see Section 5.3.

In [GGH15], the graphs are stated as acyclic. However, the construction itself supports cycles, including self-loops. The original paper [GGH15] didn't explain the reasons to avoid cycles. The possible reasons are[4]: (1) For security concerns, since cycles involves circularities in the trapdoor sampling. (2) For practicality, since in the two candidate applications in [GGH15], cycles are not used. In our presentation, we allow cycles in the syntax, though our chaining technique is not known to apply to any cycles based on the current LWE library. The reason to keep cycles as an option is that cycles expand the expressibility of graphs, and not all the distributions on cycles are known to be insecure. We semi-formally discuss this question in Section 6.1, where a related lattice problem called *Learning with errors with shadow* is defined and studied.

1.5 Implications for the landscape of multilinear maps and obfuscations

On GGH15 and its applications. Our proof technique implies that the encoding of GGH15 is pseudorandom if the underlying distributions of matrix and the topology of the graph satisfy certain constrains. However, it hasn't covered the two applications mentioned in [GGH15]. For the key-exchange protocol, the encoded elements on the chains are "irresolvably correlated". Moreover, the protocol requires public sampling which is not safely realizable by our current proof technique. In fact, Coron et al [CLLT15] gave an explicit attack for the basic version of GGH15 key-exchange.

[4] Partially due to the talks and personal communication with Shai Halevi.

Neither do we know how to base the security of the branching program obfuscator presented in [GGH15] on learning with errors, due to (1) the weakness of permutation matrices (or other kinds of matrix branching programs with low-entropic rows) as LWE secrets without using Kilian's randomization techniques [Kil88]; (2) the appearance of unresolvable correlations when using Kilian's technique and other "safeguards". An interesting question is to justify (or disprove) the effectiveness of Kilian's randomization technique (with or without other safeguards) by proposing a related LWE assumption (resp. an explicit attack).

A recent work of Brakerski et al [BVWW16] is the first to build high-level obfuscators using GGH15 and a clearly stated assumption of RingLWE. They show that entropic conjunction functionality can be obfuscated based on an encoding technique, which is rooted from the work of Brakerski and Rothblum [BR13] based on GGH13. They "transplant" the encoding onto GGH15, and propose a clearly stated assumption called "entropic RingLWE" to handle a correlated case in the encoding, though it hasn't been clear how to base the new assumption on standard RingLWE (or break it). Their result can be interpreted as an example of applying the chaining lemma with an extended LWE library. See an exposure in Section 5.3.

On the other multilinear maps. Our proof technique doesn't seem to apply to GGH13 or CLT13, due to the differences of the design principles and algebraic backgrounds among these candidates. Most of the applications of multilinear maps in the literature uses GGH13 or CLT13 as the bases for candidate instantiations. It is worthwhile to state them in the language of GGH15 (or the simplified tree variant) and see if the chain reactions can be controlled over the underlying encoding. The candidate key-exchange and branching program obfuscator in [GGH15] are such attempts. The conjunction obfuscator mentioned above [BVWW16] is another one.

There are some other multilinear maps proposed based on the assumption that indistinguishability obfuscators exist. These work includes self-bilinear map from iO and factoring [YYHK14], "multilinear jigsaw puzzles" from VGB or iO [BCKP14, PS15], multilinear maps from iO and algebraic assumptions [AFH+16]. They are usually with cleaner interfaces and stronger security guarantees (conditioned on the existence of iO). Also, they use different design principles and may be of independent interest. The security and implications of these constructions will be better understood if the "core obfuscators" in these works (e.g. the obfuscation of "modular exponentiation" in [YYHK14]) can be based on better understood hardness assumptions, or when they "accidentally" imply something that doesn't exist at all, thus possibly killing iO.

1.6 Additional related work on matrix predicates and other evasive function obfuscators

Obfuscators for point functions and their natural generalizations were constructed in [Can97, Wee05, CD08, GKPV10, BC14] via various hardness assumptions. In fact, many of them are additive homomorphic.

An obfuscator for inner-product predicate was proposed by Canetti, Rothblum and Varia [CRV10] based on a DDH-type assumption. A natural question is if it generalizes to subspace memberships. The answer is probably "yes", especially when the entries are sampled independently and uniformly random. The hardness is to analyze the cases with non-uniform entries. Related DDH-type assumptions over "matrices in the exponents" are studied in the work of Boneh et al, Escala et al and Marillo et al [BHHO08, EHK+13, MRV15]. The potential DLOG-based construction may cover distributions that are disjoint with the coverage of the LWE-based solution. At least for the one-dimensional special case which is inner-product predicate, the LWE-based construction doesn't seem to apply whereas [CRV10] works naturally. We are not aiming to go further with DLOG-based constructions in this article since we don't know how to make the potential DLOG-based obfuscator multiplicatively homomorphic.

In the motivation scenario of quantum money[5] [AC12], they require an obfuscation of membership test functionalities of subspace $\mathbf{M} \subseteq \mathbb{F}_2^n$ and its orthogonal $\mathbf{M}^\perp$. They propose candidate obfuscators based on the hardness of solving (noisy or noiseless) multivariate polynomials. A heuristic algorithm that threatens the noiseless version is given by Pena, Faugère and Perret [PFP15].

Can our scheme handle this case? In general, for a subspace $\mathbf{S} \subseteq \mathbb{Z}_q^n$ and its dual $\mathbf{S}^*$, one can set (the bases of) them as the secrets of independent LWE instances. But we don't know (even a special case) where the security of obfuscating such related matrices can be based on learning with errors. In fact, adapting our obfuscator into $\mathbb{F}_2^n$ (i.e. using Learning parity with noise) results into an insecure scheme, as was pointed out in the original paper [AC12, Claim 35].

Obfuscators can also be obtained from functional encryption schemes that are public-key and function-hiding, as was pointed out by [BRS13a]. As far as we know, none of the existing function-hiding functional encryption schemes (including [BRS13b]) covers the function families/distributions discussed in this paper.

There are several candidate obfuscators covering more or all evasive circuits based on the heuristic assumptions [BBC+14] or idealized models of multilinear maps [BMSZ15, BCKP14]. However, even these obfuscators from "utopian" assumptions are not necessary homomorphic.

1.7 The lattice nature of the obfuscated object

In a former proposal of the title *Obfuscate Lattices by Lattices, Homomorphically*, the first "Lattices" refers to the objects being obfuscated, since both high dimensional matrices and elements in $2n^{th}$ cyclotomic rings define bases of "crypto-friendly" lattices. However, the results in this paper haven't used the lattice nature of the obfuscated object. The exciting future of surfing over "obfuscated lattices" is left to the readers.

In fact, the "lattice nature of the secret" is used implicitly in the recent work of Bishop et al [BHW15] (followed by the very recent work [AP16, KW16]). They construct counterexamples for circular secure public-key encryption schemes, by "obfuscating" lattices with trapdoors in the LWE secrets of each other. The very recent work [AP16, KW16] can also be thought as counterexamples for a variant of "learning with errors with shadows" (cf. Def. 6.1) for commutative "secret space".

1.8 Organizations and reader's guide

The rest of the paper starts from obfuscating one matrix (Section 3), to homomorphically obfuscate matrices hung on a tree (Section 4), to homomorphically obfuscate matrices stored in a directed graph (Section 5). For readers who are not familiar with the rich library of LWE, it is easier to begin with the cases where all the matrices (including each row of them) are sampled independently from binary or error distributions, and focus on the controlled chain reactions in the proof. Then turn to the examples at the end of each section for the alternative distributions and correlated cases.

Construction 5.3 is isomorphic to the one in [GGH15, Section 3], except some subtle details explained in Section 5.4 of this paper. Readers who would like to view GGH15 from the perspective of multilinear maps (or simply for the transposed lattice notations) are highly recommended to turn to [GGH15] for reference.

Several future directions are proposed in Section 6. In particular, plausible extension packages of LWE, including "LWE with shadow", "Uncertainty LWE", are proposed in Section 6.1 and may be of independent interest beyond being building blocks for obfuscations.

We present the main results for the ring of integer matrices. Most of the results can be applied seamlessly to the other rings where the corresponding RingLWE problem is hard.

[5] This gives another excuse for not going further with DLOG-based solutions, as they won't be safe in the quantum world [Sho99].

2 Preliminaries

Notations and terminologies. The notation conventions in this paper are mainly adapted from [Pei07, GPV08]. Denote $\mathbb{R}$, $\mathbb{Z}$, $\mathbb{N}$ as the sets of real numbers, integers and positive integers. For $n \in \mathbb{N}$, $[n] := \{1, ..., n\}$. A vector in $\mathbb{R}^n$ is represented in column form, and written as a bold lower-case letter, e.g. $\mathbf{v}$. For a vector $\mathbf{v}$, the i^{th} component of $\mathbf{v}$ will be denoted by v_i. A matrix is written as a bold capital letter, e.g. $\mathbf{A}$. The i^{th} column vector of $\mathbf{A}$ is denoted $\mathbf{a}_i$.

For $p \in [1, \infty]$, the length of a vector is the norm $\|\mathbf{v}\|_p = (\sum v_i^p)^{1/p}$. For any $\mathbf{v} \in \mathbb{R}^n$ and $p \in [2, \infty]$, we have $n^{1/p-1/2}\|\mathbf{v}\|_2 \leq \|\mathbf{v}\|_p \leq \|\mathbf{v}\|_2$. The length of a matrix is the norm of its longest column: $\|\mathbf{A}\|_p = \max_i \|\mathbf{a}_i\|_p$. In this article the default norm is ℓ_2-norm, but in some places we switch to ℓ_∞-norm for convenience (e.g. when defining the domain of "small" vectors). For $\mathbf{A} \in \mathbb{R}^{\ell \times m}$, $\mathbf{B} \in \mathbb{R}^{m \times n}$, $\|\mathbf{A} \cdot \mathbf{B}\|_2 \leq \|\mathbf{A}\|_2 \cdot \|\mathbf{B}\|_2$, $\|\mathbf{A} \cdot \mathbf{B}\|_\infty \leq \max\{\ell, m, n\}\|\mathbf{A}\|_\infty \cdot \|\mathbf{B}\|_\infty$.

The security parameter is denoted as λ. An algorithm is "efficient" if it runs in (probabilistic) polynomial time. When a vector or matrix is called "small", we refer to its norm.

Definition 2.1 (Evasive collections [BBC+14])**.** *Let $\mathcal{F} = \{f_k : \{0,1\}^{\ell(\lambda)} \to \{0,1\}\}_{\lambda \in \mathbb{N}}$ be a function collection, we say $\mathcal{F}_\lambda$ is* evasive *if there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $x \in \{0,1\}^{\ell(\lambda)}$:*

$$\Pr_k[f_k(x) \neq 0] \leq \mathsf{negl}(\lambda)$$

2.1 Obfuscation

The obfuscators considered in this article preserve the functionality with all but negligible probability, and cause a polynomial blow-up on the size of the function description. To be precise, for the function family $\mathcal{F} = \{f : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{m(\lambda)}\}_{f \in \mathcal{F}_\lambda}$, a probabilistic algorithm Obf is an obfuscator, if

1. The string $\mathsf{Obf}(f)$ describes a circuit that computes the same function as f;
2. There is a polynomial $B(\cdot)$ such that $|\mathsf{Obf}(f)| \leq B(|f|)$.

The security property is defined below.

Definition 2.2 (Virtual-Black-Box Obfuscation [BGI+12])**.** Obf *is a Virtual-Black-Box obfuscator (VBB) for $\mathcal{F}$ if for any feasible adversary A, there is a p.p.t. simulator S and a negligible function $\mathsf{negl}(\cdot)$ such that for all but negligibly many functions $f_\lambda \in \mathcal{F}_\lambda$:*

$$\left|\Pr[A(\mathsf{Obf}(f_\lambda)) = \pi(f_\lambda)] - \Pr[S^{f_\lambda}(1^{|f_\lambda|}) = \pi(f_\lambda)]\right| \leq \mathsf{negl}(|f_\lambda|)$$

for any predicate $\pi : \mathcal{F}_\lambda \to \{0,1\}$. The probability is over the randomness of distinguisher, A, S and Obf.

Remark 2.3 (Worst-case/average-case, with/without auxiliary input)**.** *The definition above slightly relax the "worst-case" VBB defined in [BGI+12], by requiring that security holds for "all but negligibly many" functions in the family instead of "all".*

We also adapt the term "average-case" VBB where f_λ is drawn from a distribution on $\mathcal{F}_\lambda$, and is included in the probability in the expression above. In fact, the "mixed" situations can be defined and can often be the cases in this paper. Namely, part of the function is fixed and part is sampled from a distribution, therefore the whole can be described by distributions in general. In the rest of the paper we omit the terms of "worst-case" or "average-case" and instead be precise about the underlying distribution that is covered.

By default we consider the notions of obfuscation without related auxiliary input. There are some distributions where our results hold against related auxiliary inputs.

2.2 Lattices

An n-dimensional lattice Λ is a discrete additive subgroup of $\mathbb{R}^n$. Given n linearly independent basis vectors $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n \in \mathbb{R}_n\}$, the lattice generated by $\mathbf{B}$ is $\Lambda(\mathbf{B}) = \Lambda(\mathbf{b}_1, ..., \mathbf{b}_n) = \{\sum_{i=1}^{n} x_i \cdot \mathbf{b}_i, x_i \in \mathbb{Z}\}$. We have the quotient group $\mathbb{R}^n/\Lambda$ of cosets $\mathbf{c} + \Lambda = \{\mathbf{c} + \mathbf{v}, \mathbf{v} \in \Lambda\}$, $\mathbf{c} \in \mathbb{R}^n$. The dual of a lattice Λ in $\mathbb{R}^n$, denoted as Λ^*, is the lattice given by the set of all vectors $\mathbf{y} \in \mathbb{R}^n$ satisfying $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}$ for all vectors $\mathbf{x} \in \Lambda$. For any $\mathbf{B} \in \mathbb{R}^{n \times n}$, $\Lambda(\mathbf{B})^* = \Lambda((\mathbf{B}^{-1})^T)$. Let $\tilde{\mathbf{B}}$ denote the Gram-Schmidt orthogonalization of $\mathbf{B}$.

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

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

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

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

If $\mathbf{x}$ is distributed according to $D_{s,\mathbf{c}}$ and we condition on $\mathbf{x} \in \Lambda$, the conditional distribution of $\mathbf{x}$ is $D_{\Lambda+\mathbf{c},s}$.

We recall some facts of discrete Gaussian over lattices.

Lemma 2.4 ([PR06, MR07]). *Let $\mathbf{B}$ be a basis of an m-dimensional lattice Λ, and let $s \geq \|\tilde{\mathbf{B}}\| \cdot \omega(\log n)$, then $\Pr_{\mathbf{x} \leftarrow D_{\Lambda,s}}[\|\mathbf{x}\| \geq s \cdot \sqrt{m} \vee \mathbf{x} = \mathbf{0}] \leq \mathsf{negl}(n)$.*

Gentry, Peikert and Vaikuntanathan [GPV08] show how to sample statistically close to discrete Gaussian distribution in polynomial time for s bigger than smoothing parameter. The sampler is upgraded in [BLP+13] so that the output is distributed "exactly" as a discrete Gaussian. (However the statistical version suffices for our purposes.)

Lemma 2.5 ([GPV08, BLP+13]). *There is a p.p.t. algorithm that, given a basis $\mathbf{B}$ of an n-dimensional lattice $\Lambda(\mathbf{B})$, $\mathbf{c} \in \mathbb{R}^n$, $s \geq \|\tilde{\mathbf{B}}\| \cdot \sqrt{\ln(2n+4)/\pi}$, outputs a sample from $D_{\Lambda+\mathbf{c},s}$.*

Learning with errors. Recall the learning with errors (LWE) problem due to Regev [Reg05]. For positive integer dimension n, modulus $q \geq 2$, an "error" probability distribution χ over $\mathbb{Z}_q$. Define the LWE distribution $\mathbf{A}_{\mathbf{s},\chi}$ to be the distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by fixing a vector $\mathbf{s} \in \mathbb{Z}_q^n$, choosing a vector $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, an error term $\mathbf{e} \leftarrow \chi$, and outputting $(\mathbf{a}, \mathbf{b} = \langle \mathbf{s}, \mathbf{a} \rangle + \mathbf{e} \mod q)$.

We say that an algorithm solves (search) LWE with modulus q and error distribution χ if, for any $\mathbf{s} \in \mathbb{Z}_q^n$, given an arbitrary number of independent samples from $\mathbf{A}_{\mathbf{s},\chi}$, it outputs $\mathbf{s}$ with high probability. It is shown by Regev [Reg05] that for certain moduli q and Gaussian error distributions, LWE is as hard as solving several standard worst-case lattice problems using a quantum algorithm.

Proposition 2.6 ([Reg05]). *Let q be an integer, $0 < s < q$ such that $s > 2\sqrt{n}$. If there exists an efficient (possibly quantum) algorithm that solves $\mathsf{LWE}_{n,q,D_{\mathbb{Z},s}}$, then there exists an efficient quantum algorithm for approximating SIVP and GapSVP in the ℓ_2 norm, in the worst case, to within $\tilde{O}(nq/s)$ factors.*

Peikert [Pei07] extended the result in ℓ_p norm, $2 \leq p \leq \infty$, with almost the same approximation factors. Classical hardness results are shown by [Pei09, BLP+13] based on GapSVP.

The decisional variant of LWE (DLWE), parameterized by n, q, m, χ, ask if the result of m samples $\mathbf{Y} \in \mathbb{Z}_q^{n \times m}$ is from LWE distribution $\mathbf{A}_{\mathbf{s},\chi}$ or uniform distribution.

The LWE library. Most of the results in this paper are not associated with LWE for a specific distribution, but rather applies to all the hard distributions in the *LWE library*. We call a set of vectors $\mathcal{V} \subseteq \mathbb{Z}_q^n$ (resp. a distribution $\mathcal{D}$ over $\mathbb{Z}_q^n$) *LWE-hard* if worst-case (resp. average-case) decisional-LWE problem is hard for that set (resp. distribution). For the convenience of notation, we include the corresponding set or distribution in the subscript of the problem, e.g. $\mathsf{DLWE}_{\mathcal{V}}$ or $\mathsf{DLWE}_{\mathcal{D}}$.

Examples of sets (or distributions) of secrets that are LWE-hard based on the worst-case (possibly quantum) hardness of SIVP or GapSVP:

- $\mathbb{Z}_q^n$ (standard LWE): Worst-case/average-case equivalence is shown by [Reg05]; worst-case search to average-case decision for polynomial q by [Reg05]; for general modulus by [MP12, BLP+13].
- From error distribution χ^n ("Hermit normal form" LWE): for prime power modulus by [ACPS09], for general modulus by [BLP+13].
- Binary vectors with high min-entropy and one-way auxiliary inputs ("Robust" LWE): due to [GKPV10].

Some other variants of DLWE are studied in the literature, and proved (at least for some parameters) to be as hard as standard DLWE, including but are not limited to LWE with hints on the error [OPW11], LWE with hints of the kernel [LPSS14], LWE that can be tampered [Pie12], LWE with small parameters [MP13, DMQ13], LWE with "somewhere error-free" [FMR13]. A subset of the formers are improved by [BLP+13]. See more in the survey due to Peikert [Pei15]. Readers are recommend to check the original papers for the precise parameters.

The term "LWE-hard" generalizes to matrices when the hardness on each secret vector composes, e.g. when all the vectors are sampled independently (equivalent to standard LWE by a hybrid argument), or allowing dependences such as for Robust LWE.

Trapdoor notions for lattices.

Lemma 2.7 ([Ajt99, AP11, MP12])**.** *There is a p.p.t. algorithm* $\mathsf{TrapSam}(1^n, 1^m, q)$ *that, given* $n > 1$, $q > 2$ *and sufficiently large* $m = \Omega(n \log q)$*, outputs* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *with a trapdoor* τ*.*

The type of trapdoor considered in the work of [Ajt99, AP11] is a small full rank matrix $\mathbf{T} \in \mathbb{Z}^{m \times m}$ s.t. $\mathbf{AT} = 0 \bmod q$ is a trapdoor of $\mathbf{A}$. The trapdoor notion introduced in [MP12] is different, but they can be turned into each other for all the applications in this paper.

Below is a corollary of Lemma 2.5 and 2.7.

Corollary 2.8 ([GPV08])**.** *There is a p.p.t. algorithm* $\mathsf{PreimgSam}(\mathbf{A}, \tau, \mathbf{y}, s)$ *that with all but negligible probability over* $(\mathbf{A}, \tau) \leftarrow \mathsf{TrapSam}(1^n, 1^m, q)$*, for sufficiently large* $s = \Omega(\sqrt{n \log q})$*, the following distributions are within negligible statistical distance:*

$$\{\mathbf{A}, \mathbf{x}, \mathbf{y} : \mathbf{x} \leftarrow D_{\mathbb{Z}^m, s}, \mathbf{y} = \mathbf{Ax}\} \approx \{\mathbf{A}, \mathbf{x}, \mathbf{y} : \mathbf{y} \leftarrow U(\mathbb{Z}_q^n), \mathbf{x} \leftarrow \mathsf{PreimgSam}(\mathbf{A}, \tau, \mathbf{y}, s)\}$$

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

In the "semi-formal discussion" section (cf. Section 6) we use one of the "Bonsai techniques" introduced in [CHKP12]. Namely an owner of lattice $\mathbf{A}$ with trapdoor $\mathbf{T}$ can derive a trapdoor of an arbitrary higher-dimensional extension $\mathbf{A}||\mathbf{A}'$.

Lemma 2.9 (Extending control [CHKP12])**.** *There is a deterministic polynomial-time algorithm* ExtBasis *with the following properties: given an arbitrary* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *whose columns generate the entire group* $\mathbb{Z}_q^n$*, an arbitrary basis* $\mathbf{T} \in \mathbb{Z}^{m \times m}$ *of* $\Lambda^{\perp}(\mathbf{A})$*, and an arbitrary* $\mathbf{A}' \in \mathbb{Z}_q^{n \times m'}$*,* $\mathsf{ExtBasis}(\mathbf{T}, \mathbf{A}'' = \mathbf{A}||\mathbf{A}')$ *outputs a basis* $\mathbf{T}''$ *of* $\Lambda^{\perp}(\mathbf{A}'') \subseteq \mathbb{Z}^{m+m'}$ *s.t.* $\|\tilde{\mathbf{T}}''\| = \|\tilde{\mathbf{T}}\|$.

Corollary 2.8 and Lemma 2.9 also applies to other trapdoor notions (such as the one introduced in [MP12]), but the basis-trapdoor suffices for our purpose.

Ideal lattices and ring LWE. Most of the concepts of hard problems over integer lattices has their correspondence over ideal lattices. For simplicity, readers can assume we are working with the $2n^{th}$ cyclotomic ring $R := \mathbb{Z}[X]/(X^n+1)$ with n being a power of two, and a RingLWE sample is of the form $(a, sa+e) \in R_q \times R_q$ (to be precise, the secret is from the dual $R_q^{\vee}$ and the RingLWE samples are from $R_q \times R_q^{\vee}$; however, for $2n^{th}$ cyclotomic rings, they are equivalent, since $R^{\vee} = n^{-1}R$). Lyubashevsky, Peikert and Regev show that the hardness of solving decisional RingLWE over cyclotomic rings can be based on the worst-case quantum hardness of approximate SIVP over ideal lattices [LPR13a]. The concepts of HNF-RingDLWE, trapdoors, discrete Gaussian sampling and "bonsai techniques" also exist in ideal lattices (or by themselves general to any lattices). We refer the readers to [LPR13a, LPR13b] for details, and we adapt their choice of measurement (namely canonical embedding) without formally defining it.

For the main results in this paper, the constructions and proof techniques can be adapted seamlessly from the ring of integers to the ring of polynomials. Working with cyclotomic rings gives a commutative variant of plaintext space (i.e. "secret space" for the (Ring)LWE instances), thus enjoying some unique features that are not provided by the ring of integer matrices.

3 How to Obfuscate a Matrix

For positive integers ℓ, n, and $q \geq 2$, we study the problem of obfuscating a matrix $\mathbf{M} \in \mathbb{Z}_q^{\ell \times n}$ and allowing the evaluator to test whether an input vector $\mathbf{s} \in \mathbb{Z}_q^{\ell}$

1. is the cokernel of $\mathbf{M}$ (s.t. $\mathbf{s}^T\mathbf{M} = \mathbf{0}^T$);
2. lives in a public affine subspace of the cokernel (s.t. $\mathbf{s}^T\mathbf{M} = \mathbf{c}^T$, $\mathbf{c} \in \mathbb{Z}_q^n$ is chosen by the evaluator);
3. lives in a hidden affine subspace of the cokernel (s.t. $\mathbf{s}^T\mathbf{M} = \mathbf{r}^T$, $\mathbf{r} \in \mathbb{Z}_q^n$ is hidden).

These functionalities can be represent by *matrix predicates*.

Definition 3.1 (Matrix predicates)**.** *For matrices* $\mathbf{M} \in \mathbb{Z}_q^{\ell \times n}$*,* ℓ*,* n*,* $q \in \mathbb{N}$ *and* $q \geq 2$*. An affine subspace predicate* $P : \mathbb{Z}_q^{\ell} \setminus \{\mathbf{0}\} \times \mathbb{Z}_q^n \to \{0,1\}$ *takes a vector* $\mathbf{s}$ *and a target subspace* $\mathbf{v}$ *as the input, computes*

$$P_{\mathbf{M}}(\mathbf{s}, \mathbf{v}) = \begin{cases} 1, & \text{if } \mathbf{s}^T\mathbf{M} = \mathbf{v}^T \\ 0, & \text{elsewhere} \end{cases}$$

(cokernel-test can be performed by selecting $\mathbf{v} = \mathbf{0}$*.)*

A hidden affine subspace predicate is defined as $H : \mathbb{Z}_q^{\ell} \to \{0,1\}$*,*

$$H_{\mathbf{M},\mathbf{v}}(\mathbf{s}) = \begin{cases} 1, & \text{if } \mathbf{s}^T\mathbf{M} = \mathbf{v}^T \\ 0, & \text{elsewhere} \end{cases}$$

When q^{-n} is negligible, both predicates are evasive.

3.1 Obfuscate matrix predicates from learning with errors

We show how to obfuscate the matrix predicates defined above under certain choices of parameters q, ℓ, n, and the underlying set $\mathcal{M} \subseteq \mathbb{Z}_q^{\ell \times n}$ or a distribution $\mathcal{D}$ over $\mathcal{M} \subseteq \mathbb{Z}_q^{\ell \times n}$. In addition, the construction only allows to test input vectors with small norm, say $\|\mathbf{s}\|_\infty \leq B$ for some small $B \in \mathbb{R}^+$ whose precise bound to be determined later. Let $\mathcal{B} = \{x \mid x \in \mathbb{Z}, |x| \leq B\}$. In the rest of the section (in fact, this article) we restrict the domain of P to be $\mathcal{B}^\ell \setminus \{\mathbf{0}\} \times \mathbb{Z}_q^n$, the domain of H as $\mathcal{B}^\ell$. The construction is based on the hardness of decisional LWE over $\mathcal{M}$ or distribution $\mathcal{D}$.

Theorem 3.2 (VBB obfuscation for matrix predicates)**.** *Assuming* $\mathsf{DLWE}_{\mathcal{M}}$*, for* $\mathbf{M} \leftarrow \mathcal{M}$*, there is a VBB obfuscator for affine-subspace-predicate* $P_{\mathbf{M}}$*. Additionally, assuming* $\mathsf{DLWE}_{\mathcal{V}}$*, for* $\mathbf{v} \leftarrow \mathcal{V}$*, there is a VBB obfuscator for hidden-affine-subspace-predicate* $H_{\mathbf{M},\mathbf{v}}$*.*

Remark 3.3 (Worst-case or average-case)**.** *Following Remark 2.3, whether it is worst-case or average-case VBB immediately follows whether the hardness of decisional LWE applies to the family or the distribution. To simplify the presentation we choose to omit the terms in the theorem statement.*

Construction 3.4 (LWE-based obfuscator for matrix predicates)**.** *An LWE-based matrix predicate obfuscator* Obf *takes as inputs the parameters* $1^\lambda, q, 1^n, 1^\ell, B$ *(the precise bounds to be specified later) and* $\mathbf{M} \in \mathbb{Z}_q^{\ell \times n}$*, the matrix to be obfuscated. It derives parameters* m*,* z*, where* z *is threshold for correctness. Sample* $\mathbf{A}_0 \leftarrow U(\mathbb{Z}_q^{n \times m})$*, a matrix from error distribution* $\mathbf{E} \leftarrow \chi^{\ell \times m}$*. Compute* $\mathbf{A}_1 := \mathbf{M}\mathbf{A}_0 + \mathbf{E}$*.*

1. *To support affine subspace test, publish* $\mathbf{A}_1$ *and* $\mathbf{A}_0$*, i.e.* $\mathsf{Obf}(P_{\mathbf{M}}) = (\mathbf{A}_1, \mathbf{A}_0)$*. The evaluation algorithm is defined as*

$$\mathsf{Eval}(\mathsf{Obf}(P_{\mathbf{M}}), \mathbf{s}, \mathbf{v}) = \begin{cases} 1, & \text{if } \|\mathbf{s}^T\mathbf{A}_1 - \mathbf{v}^T\mathbf{A}_0\|_\infty \leq z \\ 0, & \text{elsewhere} \end{cases}$$

2. *To support hidden affine subspace test, let* $\mathbf{v} \in \mathbb{Z}_q^n$ *be the target vector to be hided. Compute* $\mathbf{y}^T = \mathbf{v}^T\mathbf{A}_0 + \mathbf{e}^T$*. Publish* $\mathbf{A}_1$ *and* $\mathbf{y}$*, i.e.* $\mathsf{Obf}(H_{\mathbf{M},\mathbf{v}}) = (\mathbf{A}_1, \mathbf{y})$*. To evaluate, compute*

$$\mathsf{Eval}(\mathsf{Obf}(H_{\mathbf{M},\mathbf{v}}), \mathbf{s}) = \begin{cases} 1, & \text{if } \|\mathbf{s}^T\mathbf{A}_1 - \mathbf{y}^T\|_\infty \leq z \\ 0, & \text{elsewhere} \end{cases}$$

The setting of parameters should satisfy the following constrains:

1. For the hardness of LWE (based on the worst-case hardness of GapSVP for polynomial approximation factors): Let $n \geq \lambda$, $s \geq 2\sqrt{n}$, $nq/s < \mathsf{poly}(\lambda)$, $\chi = D_{\mathbb{Z},s}$.
2. For preserving the functionality of P and H w.h.p.: $m = \Theta(n \log q)$, $(B\ell + 1)s\sqrt{n} < z < q^{3/4}$.

Proof of Theorem 3.2. Correctness (of both P and H) follows from the setting of parameters, which implies the "injective mode" of LWE by setting $m = \Theta(n \log q)$, and the calculation of the threshold:

$$\max\{\|\mathbf{s}^T\mathbf{A}_1 - \mathbf{v}^T\mathbf{A}_0\|_\infty, \|\mathbf{s}^T\mathbf{A}_1 - \mathbf{y}^T\|_\infty\} \leq \|\mathbf{s}^T\mathbf{M}\mathbf{A}_0 + \mathbf{s}^T\mathbf{E} - \mathbf{v}^T\mathbf{A}_0\|_\infty + \|\mathbf{e}^T\|_\infty \leq (B\ell + 1)s\sqrt{n}$$

VBB security (of both P and H) follows the hardness of decisional LWE over the underlying distribution of $\mathbf{M}$ and $\mathbf{v}$, i.e. $(\mathbf{A}_0, \mathbf{A}_1, \mathbf{y}) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^{\ell \times m} \times \mathbb{Z}_q^m$ is pseudorandom. In other words, the simulator simply outputs random strings in $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^{\ell \times m} \times \mathbb{Z}_q^m$. □

We mention several observations.

1. Publishing $\mathbf{A}_0$, $\mathbf{A}_1$, $\mathbf{y}$ together allows the evaluator to test both P and H. To support cokernel-test without providing other functionalities, publish $\mathbf{A}_1$ only. However, the latter is a claim of "the minimum component" needed for cokernel-test, rather than claiming "the evaluator can do nothing other than testing cokernel".

2. When $\ell \leq n$, the cokernel/affine subspace test functionality is in the injective mode w.h.p., i.e. the only small input that is in the cokernel is the zero vector. However, nontrivial solutions for affine subspace tests may exist. When $\ell > n$, or more precisely, when nontrivial small cokernels exist with non-negligible probability, hardness of finding the small kernels given $\mathbf{A}_1$ is also justified by the one-wayness of Ajtai function [Ajt96]. (Same to the hardness of hitting a designated affine subspace.)

3. The evaluator can test if $\mathbf{s}$ and μ is a pair of small (left) eigenvector and the corresponding eigenvalue of $\mathbf{M}$, by setting $\mathbf{v} = \mu\mathbf{s}$ and compute $\mathsf{Eval}(\mathsf{Obf}(P_{\mathbf{M}}), \mathbf{s}, \mathbf{v})$.

Example 3.5. *Examples of families/distributions of matrices that are obfuscatable by our scheme: all the families/distributions that are LWE-hard as mentioned in the preliminary, e.g. for family $\mathbb{Z}_q^{\ell \times n}$ (worst-case), error distributions $\chi^{\ell \times n}$ (average-case), when the rows are sampled independently. Dependence can be supported when the distribution is known to be LWE-hard given related auxiliary inputs, such as the "robust-LWE" distribution [GKPV10].*

The families/distributions on different rows can be different, e.g. the first row from error distribution χ^n, the second row from uniform distribution $U(\mathbb{Z}_q^n)$, etc. In fact the scheme is "family/distribution private".

Example 3.6. *Examples of distributions that are clearly not obfuscated, or not known to be covered by this scheme given the current library of LWE (i.e. Decisional LWE is easy or at least not known to be hard over these distributions).*

1. *Permutation matrices are not obfuscated by this scheme since it simply permutes $\mathbf{A}_0$ (adding small noise doesn't help);*

2. *Triangle matrices include some rows with very few non-zero entries, therefore are not covered by our construction;*

3. *Symmetric matrices have dependence across the rows thus not known to be LWE-hard in general.*

4. *Inner-product predicate, the seemingly one-dimensional special case, is not covered by our scheme.*

3.2 Obfuscate ideals by ideal lattices

The template can be adapted to ideal lattices. To obfuscate an element s in the ring $R_q := \mathbb{Z}_q[X]/(X^n+1)$, set s as the RingLWE secret and give out m samples $\mathbf{A}, s\mathbf{A} + \mathbf{E} \in R_q^{1\times m} \times R_q^{1\times m}$. It allows to test membership of ideals, say $P : R_q \times R_q \to \{0, 1\}$:

$$P_s(v, t) = \begin{cases} 1, & \text{if } vs = t \\ 0, & \text{elsewhere} \end{cases}$$

but only for "small" v, for the same reason as in the ring of integer matrices. It can also be extend to testing hidden target ideals.

4 The Arboriculture of Obfuscating Matrices

We show that the obfuscators for matrix predicates (cf. Construction 3.4) can be constructed in a way that supports constrained addition and multiplication over the obfuscated code.

4.1 Tree-induced homomorphic obfuscator for matrices

We extend the gadgets introduced in Section 1.3 into a general homomorphic obfuscator. We partially adapt the graph-based terminology used in [GGH15], that is, the matrices to be obfuscated sit on the path of a directed graph, the masks ($\mathbf{A}$ matrices) are computed on the vertices.

The most expressive structure that supports the gadgets is *tree*, a.k.a. an acyclic graph of out-degree at most 1. Additions can be performed over the branches that share the same parent node, multiplications can be performed over branches that are connected and form a longer branch (we consider branches as "paths" in a graph, not like edges). Note that the rules of evaluation in the tree-induced homomorphic obfuscator is different from the ones in the graph-induced one in GGH15. The difference will be clearer when we compare trees and graphs in Section 5.

To simplify the presentation, we denote matrices as $\mathbf{M}_{c,p}$ where c and p are the IDs for the child and parent nodes of the branch. We only describe the construction for square matrices with the same dimension, but the construction can be extended to various dimensions. The matrices live in LWE-hard distributions. To begin with, readers can think of them as being sampled independently from binary or error distributions (even achieving this case is not known before). The proof strategy we describe will cover the general (possibly correlated) cases.

Definition 4.1 (Tree-induced data structure)**.** *A tree-induced data structure* $\mathsf{Y} = (R, \mathcal{N}, \mathcal{J}) \in \mathcal{Y}$ *is parameterized by a ring* R *(where the elements stored in the tree belong to). It contains a set of nodes* $\mathcal{N} = \{\mathsf{node}\}$*, a set of branches* $\mathcal{J} = \{\mathsf{branch}\}$*. The data structures of the nodes and branches are:*

1. *Each node has an ID:* $\mathsf{node} = (\mathsf{id})$*;*
2. *Each branch stores the IDs of its child, parent, and an element* $\mathsf{ele} \in R$*:* $\mathsf{branch} = (\mathsf{child}, \mathsf{parent}, \mathsf{ele})$.

All the natural parameters (depth, width, topology, etc.) and efficient operations on a tree (initialize, insert, delete, search, etc.) are supported and efficiently computable for our tree-based data structure. The details are not important to this work so we skip the descriptions. By default $R = \mathbb{Z}_q^{n \times n}$.

Next we define tree-induced homomorphic obfuscator for matrix predicate P (cf. Def. 3.1). The construction for H is derivable from P and we omit it. The homomorphic obfuscator we define doesn't hide the topology of the tree. In contrast it must preserve the topology which determines the type of allowed homomorphic operations.

Definition 4.2 (Tree-induced homomorphic obfuscator for matrix predicates)**.** *A tree-induced homomorphic obfuscator* Obf *takes a security parameter* λ*, matrices stored in a tree* $\mathsf{Y} = (R, \mathcal{N}, \mathcal{J})$*, outputs* $\mathsf{Obf}(\mathsf{Y}) = (\mathsf{Obf}(\mathcal{N}), \mathsf{Obf}(\mathcal{J}))$.

The data structures of obfuscated nodes and branches $\mathsf{Obf}(\mathcal{N}), \mathsf{Obf}(\mathcal{J})$*:*

1. *Each node stores its ID and some obfuscated code:* $\mathsf{Obf}(\mathsf{node}) = (\mathsf{id}, \mathsf{code})$*;*
2. *Each branch stores the IDs of its child, parent, and some obfuscated code:* $\mathsf{Obf}(\mathsf{branch}) = (\mathsf{child}, \mathsf{parent}, \mathsf{code})$.

The following (deterministic) polynomial time algorithms are supplied over the encodings:

1. *Negation can be performed on any branch:* $\mathsf{Obf}(-\mathsf{branch}) := \mathsf{Neg}(\mathsf{Obf}(\mathsf{branch}))$.
2. *Addition can be performed if two branches have the same parent: if* $\mathsf{Obf}(\mathsf{branch}_1).\mathsf{parent} = \mathsf{Obf}(\mathsf{branch}_2).\mathsf{parent}$, $\mathsf{Obf}(\mathsf{branch}_{1+2}) := \mathsf{Add}(\mathsf{Obf}(\mathsf{branch}_1), \mathsf{Obf}(\mathsf{branch}_2))$.
3. *Multiplication can be performed if the parent of one is the child of the other: if* $\mathsf{Obf}(\mathsf{branch}_1).\mathsf{parent} = \mathsf{Obf}(\mathsf{branch}_2).\mathsf{child}$, $\mathsf{Obf}(\mathsf{branch}_{1\times 2}) := \mathsf{Mult}(\mathsf{Obf}(\mathsf{branch}_1), \mathsf{Obf}(\mathsf{branch}_2))$.
4. *The evaluator* Eval *for subspace-test predicate* P *takes an obfuscated branch* $\mathsf{Obf}(\mathsf{branch})$ *which encodes* $\mathbf{M}$*, and inputs* $\mathbf{s}$*,* $\mathbf{v}$*, outputs*

$$P.\mathsf{Eval}(\mathsf{Obf}(\mathsf{branch}), \mathbf{s}, \mathbf{v}) = \begin{cases} 1, & \text{if } \mathbf{s}^T\mathbf{M} = \mathbf{v}^T \\ 0, & \text{elsewhere} \end{cases}$$

In addition, Obf *is a VBB obfuscator for* $\mathcal{Y}$ *if for any feasible adversary* A*, there is a p.p.t. simulator* S *and a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$|\Pr[A(\mathsf{Obf}(\mathsf{Y}_\lambda)) = \pi(\mathsf{Y}_\lambda)] - \Pr[S(\mathsf{Y}.\mathsf{Topology}) = \pi(\mathsf{Y}_\lambda)]| \leq \mathsf{negl}(|\mathsf{Y}_\lambda|)$$

for any predicate $\pi : \mathcal{Y}_\lambda \to \{0,1\}$*. The probability is taken over the randomness of distinguisher,* A*,* S*,* Obf*, and the sampler of* Y_λ*.*

Next we describe the tree-induced homomorphic obfuscator for matrix predicates from LWE.

Construction 4.3. *An LWE-based tree-induced homomorphic obfuscator takes a security parameter* 1^λ*, a tree of matrices* $\mathsf{Y} = (R, \mathcal{N}, \mathcal{J})$*, generates the obfuscated tree* $\mathsf{Obf}(\mathsf{Y}) = (\mathsf{Obf}(\mathcal{N}), \mathsf{Obf}(\mathcal{J}))$ *as follows:*

1. *Compute (parse) the parameters* $n, m, q \in \mathbb{N}$*,* $s \in \mathbb{R}^+$*,* $B \in \mathbb{R}^+$ *(the bound on the norm of vector* $\|\mathbf{s}\|_\infty$*),* $C \in \mathbb{R}^+$ *(the bound on the norm of matrices to be obfuscated* $\|\mathbf{M}\|_\infty$*),* $0 < z < q$ *(the threshold for correctness). The constrains on the choices of parameters are discussed later.*
2. *Go to the root of the tree* root*, sample* $\mathbf{A}_{\mathsf{root}}$ *from* $U(\mathbb{Z}_q^{n\times m})$*.* $\mathsf{Obf}(\mathsf{root}).\mathsf{code} := \mathbf{A}_{\mathsf{root}}$.
3. *Visit every branches by traversing the tree from the root (either breadth-first or depth-first suffices): For the visited branch* branch*, let the ID of its parent be* u*, child be* v*,* $\mathbf{M}_{v,u}$ *be the element in the branch. Sample* $\mathbf{E}_{v,u} \leftarrow \chi^{n\times m}$*. Compute the obfuscated code of the child node as* $\mathsf{Obf}(\mathsf{node}_v).\mathsf{code} := \mathbf{A}_v := \mathbf{M}_{v,u}\mathbf{A}_u + \mathbf{E}_{v,u}$ *The obfuscated code of the branch as* $\mathsf{Obf}(\mathsf{branch}).\mathsf{code} := (\mathsf{Obf}(\mathsf{node}_v).\mathsf{code}, \mathsf{Obf}(\mathsf{node}_u).\mathsf{code}) = (\mathbf{A}_v, \mathbf{A}_u)$.

The functionalities over the obfuscated code are instantiated as follows:

1. $\mathsf{Obf}(-\mathsf{branch}) \leftarrow \mathsf{Neg}(\mathsf{Obf}(\mathsf{branch}))$*: Parse the ID of the child and parent of* branch *as* v*,* u*. Parse* $\mathsf{Obf}(\mathsf{branch}).\mathsf{code} = (\mathbf{A}_v, \mathbf{A}_u)$*. Compute* $\mathsf{Obf}(-\mathsf{branch}).\mathsf{child} := v$*,* $\mathsf{Obf}(-\mathsf{branch}).\mathsf{parent} := u$*,* $\mathsf{Obf}(-\mathsf{branch}).\mathsf{code} = (-\mathbf{A}_v, \mathbf{A}_u)$.
2. $\mathsf{Obf}(\mathsf{branch}_{1+2}) \leftarrow \mathsf{Add}(\mathsf{Obf}(\mathsf{branch}_1), \mathsf{Obf}(\mathsf{branch}_2))$*: Parse the ID of the common parent of these two branches as* u*, their children's IDs as* v*,* w *respectively. Parse* $\mathsf{Obf}(\mathsf{branch}_1).\mathsf{code} = (\mathbf{A}_v, \mathbf{A}_u)$*,* $\mathsf{Obf}(\mathsf{branch}_2).\mathsf{code} = (\mathbf{A}_w, \mathbf{A}_u)$*. Compute* $\mathsf{Obf}(\mathsf{branch}_{1+2}).\mathsf{parent} := u$*,* $\mathsf{Obf}(\mathsf{branch}_{1+2}).\mathsf{child} := \perp$*,* $\mathsf{Obf}(\mathsf{branch}_{1+2}).\mathsf{code} := (\mathbf{A}_v + \mathbf{A}_w, \mathbf{A}_u)$.

3. $\mathsf{Obf}(\mathsf{branch}_{1\times 2}) \leftarrow \mathsf{Mult}(\mathsf{Obf}(\mathsf{branch}_1), \mathsf{Obf}(\mathsf{branch}_2))$*: Let w, v be the IDs of the child and parents of the first branch and v, u be the ones for the second. Let* $\mathsf{Obf}(\mathsf{branch}_1).\mathsf{code} = (\mathbf{A}_w, \mathbf{A}_v)$, $\mathsf{Obf}(\mathsf{branch}_2).\mathsf{code} = (\mathbf{A}_v, \mathbf{A}_u)$. *Compute* $\mathsf{Obf}(\mathsf{branch}_{1\times 2}).\mathsf{parent} := u$, $\mathsf{Obf}(\mathsf{branch}_{1\times 2}).\mathsf{child} := w$, $\mathsf{Obf}(\mathsf{branch}_{1\times 2}).\mathsf{code} := (\mathbf{A}_w, \mathbf{A}_u)$.

4. *Predicate-tester P takes an obfuscated branch* $\mathsf{Obf}(\mathsf{branch}_{i,j})$ *(which obfuscates $\mathbf{M}_{i,j}$, with child node i, parent node j) and inputs $\mathbf{s}, \mathbf{v} \in \mathcal{B}^n \setminus \{\mathbf{0}\} \times \mathbb{Z}_q^n$, parses the code of* $\mathsf{Obf}(\mathsf{branch}_{i,j})$ *as $\mathbf{A}_i, \mathbf{A}_j$. Evaluate as*

$$\mathsf{Eval}(\mathsf{Obf}(P_{\mathbf{M}_{i,j}}), \mathbf{s}, \mathbf{v}) = \begin{cases} 1, & \text{if } \|\mathbf{s}^T\mathbf{A}_i - \mathbf{v}^T\mathbf{A}_j\|_\infty \leq z \\ 0, & \text{elsewhere} \end{cases}$$

Correctness, setting of parameters Correctness follows the gadget operations and the control of the noise budget. Gadget operations: for negation, $-\mathbf{s}^T\mathbf{A}_1 = -\mathbf{s}^T\mathbf{M}\mathbf{A}_0 - \mathbf{s}^T\mathbf{E}$; for addition, $\mathbf{s}^T(\mathbf{A}_{1,0} + \mathbf{A}_{2,0}) = \mathbf{s}^T(\mathbf{M}_{1,0} + \mathbf{M}_{2,0})\mathbf{A}_0 + (\mathbf{s}^T\mathbf{E}_{1,0} + \mathbf{s}^T\mathbf{E}_{2,0})$; for multiplication, $\mathbf{s}^T\mathbf{A}_2 = \mathbf{s}^T\mathbf{M}_{2,1}\mathbf{M}_{1,0}\mathbf{A}_0 + (\mathbf{s}^T\mathbf{M}_{2,1}\mathbf{E}_{1,0} + \mathbf{s}^T\mathbf{E}_{2,1})$. Analysis of the noise budget: denote $\eta(\mathsf{Obf}(\mathsf{branch}))$ as the magnitude of noise accumulated in an obfuscated branch, measured by its infinite norm. The initial noise level for the encodings are $\eta \leq s\sqrt{n}$. Negation doesn't increase the noise. Addition of two encodings causes an additive blow-up: $\eta(a+b) \leq \eta(a) + \eta(b)$; multiplication of two encodings also depends on the norm of the left matrix: $\eta(a \times b) \leq \eta(a) + nC\eta(b)$.

For the hardness of LWE: Let $n \geq \lambda$, $s \geq 2\sqrt{n}$, $nq/s < 2^{n/\lambda}$, $\chi = D_{\mathbb{Z},s}$.

To preserve the functionality of P w.h.p.: $m = \Theta(n \log q)$, and the noise doesn't overcome $z < q^{3/4}$.

Examples of setting of parameters for a tree of depth d: The maximum noise blow-up will be $nB \times (1 + nC)^{d-1} \times s\sqrt{n}$, where C depends on the choice of distribution (e.g. $C = 1$ if the matrices are binary). If d is super-constant then q/s must be super-polynomial, which means we must rely on the hardness of super-polynomial approximation of the worst-case lattice problems.

Note that the parameter above for security is necessary but not sufficient. The sufficient conditions will be explored in Sections 4.2 and 4.4.

Remark 4.4 (On the syntax and the construction of branches)**.** *Following the syntax defined above, the resulting branches from additions are not the ones in the existing data structure. Their parent is defined but child is left as $\perp$, which means the matrices on these branches cannot right-multiply anyone else. However, it doesn't limit the allowed computation on a tree since one can first multiply then add. In our definition, homomorphic operations doesn't create new nodes. One can also define and construct in a way that new nodes are created so that all the branches have children, but it complicates the presentation so we choose not to do it.*

As was implied by the data structure, syntax and construction, there are no pairs of branches that share a same child in the initial data structure. However, branches derived from the homomorphic evaluation can share the same child (e.g. the ones derived from multiplications with the left multipliers.)

Remark 4.5 (Duplicated storage of obfuscated code)**.** *In the definition, obfuscated codes exist both in nodes and branches, this is without loss of generality. In the construction, obfuscated codes are indeed "redundantly" stored. Either storing them in the nodes (then the branch can access them via the index of the node) or in the branches suffices. We choose to present "the duplicated version" for the clarity of the relations of the codes in nodes and branches.*

Algorithm 4.6 (Public sampling)**.** *The evaluator can publicly sample encodings of matrices with truncated functionalities. To sample an encoding of matrix $\mathbf{C} \in \mathbb{Z}_q^{n\times n}$ associated on a new branch with existing parent node u, compute $\mathbf{A}_v := \mathbf{C}\mathbf{A}_u$. There is no need to add noises if the purpose is for the evaluator to play by itself. In this case $\mathbf{C}$ doesn't have to be sampled from LWE-hard distributions. One can add noise if*

the application is, for example, to delegate the tree to a 3rd party. The obfuscated code of new branch is $(v, u, (\mathbf{A}_v, \mathbf{A}_u))$*. A new node* v *is created (with* $v.\mathsf{code} = \mathbf{A}_v$*).*

As implied by the topology, the encoding of $\mathbf{C}$ *can be added with the branches that shares the same parent node* u*, or left multiply on the parent branch, but cannot right multiply on the existing encodings since no branches are created on* v*. To be consistent with the correctness requirement, the norm of* $\mathbf{C}$ *has to be small if* $\mathbf{A}_u$ *is not the root of the tree.*

4.2 Security proof for independent distributions

Not all the distributions of tree-induced matrices are obfuscatable by our construction (the distribution is joint over the topology of the tree, and the sub-distributions where the individual matrices are sampled from). A sufficient condition for a distribution to be obfuscatable by our scheme is: if there is a path to traverse the tree such that one can "filp" the $\mathbf{A}$ matrices into random one-by-one based on decisional LWE. The proof technique reflects the *controlled chain reactions of learning with errors*.

We don't know how to efficiently identify such a path (or whether there is such a path) for arbitrary distributions on $\mathcal{Y}$. Instead, we choose to demonstrate several examples on both sides. To start with, we show that if all the matrices are sampled independently from LWE-hard distributions, then no matter what the topology of the tree is, it can be VBB obfuscated by construction 4.3 (i.e. the chain reaction is controlled throughout the tree).

Theorem 4.7. *For* Y *with arbitrary topology, if the matrices are sampled independently from LWE-hard distributions, then* $\mathsf{Obf}(\mathsf{Y})$ *based on construction 4.3 satisfies VBB security (cf. Def. 4.2).*

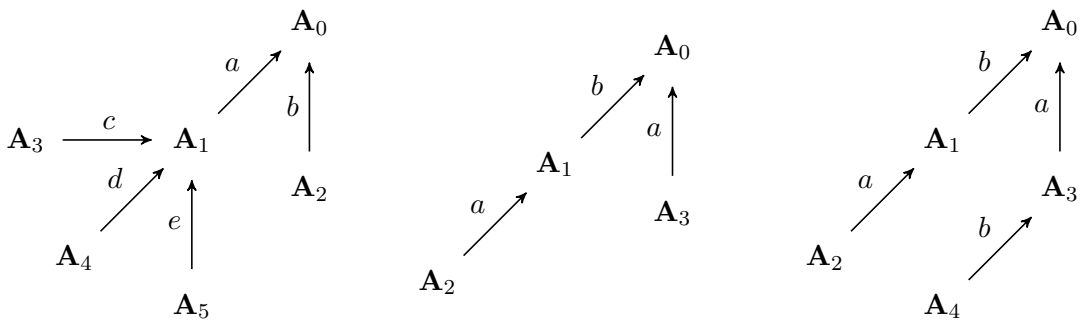
Proof. We proof by a hybrid argument. In hybrid 0, all the $\mathbf{A}$ matrices are sampled according to the real distribution. Setup a counter $\mathsf{cnt} = 0$. Traverse the obfuscated tree from the root (either breadth-first or depth-first suffices): (1) $\mathsf{cnt} \leftarrow \mathsf{cnt} + 1$; (2) For the visited branch $\mathsf{branch}_{\mathsf{cnt}}$, parse the code as $(\mathbf{A}_i, \mathbf{A}_j)$ where $\mathbf{A}_i = \mathbf{M}_{i,j}\mathbf{A}_j + \mathbf{E}_{i,j}$. Replace $\mathbf{A}_i$ by a uniformly random matrix from $\mathbb{Z}_q^{n\times m}$, and sample the rest of the $\mathbf{A}$ matrices as was in Hybrid $\mathsf{cnt} - 1$. Record it as Hybrid cnt. There are $|\mathcal{J}| + 1$ hybrids, and Hybrid $|\mathcal{J}|$ is the output of the simulator.

Next we show the adjacent hybrids are indistinguishable following DLWE: Note that $\mathbf{A}_j$ is uniformly random in the base case (i.e. if j is the root), and indistinguishable from random by induction (node j must have been visited by the order of traverse). Given that $\mathbf{M}_{i,j}$ is sampled from LWE-hard distributions, and is independent from the other matrices in the tree, $\mathbf{A}_i$ is pseudorandom following the hardness of decisional LWE. The traverse can be completed for a Y with arbitrary topology. Therefore the simulator indistinguishably outputs random $\mathbf{A}$ matrices according to the topology, i.e. Y is VBB obfuscated. □

4.3 Obfuscate ideal lattices by ideal lattices, homomorphically

We can homomorphically obfuscate ring elements ordered in the tree structure, for the rings where RingDLWE is hard, e.g. $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ where n is a power of 2. Security holds when the chain reactions over RingLWE can be applied, for example, when all the elements are sampled independently from R_q. Correctness requires the secrets to be small, e.g. when they are sampled from error distributions (HNF-RingLWE is as hard as standard RingLWE [LPR13b, Lemma 2.4]).

This give a variant where the plaintext space is commutative. Still, multiplications over obfuscated code can only be performed non-commutatively according to the prescribed order constrained by the tree.



(a) Y_1 (b) Y_2 (c) Y_3

Figure 4.1: Examples for trees

4.4 More examples of (in)admissible distributions

We illustrate more examples of distributions over a tree of matrices where the chain rule applies or doesn't apply. For simplicity, our measurement of security only considers whether the resulting encoding is pseudorandom or not (there are cases where pseudorandomness doesn't hold but one-wayness still hold). If one of the matrix in the tree is sampled from distributions where DLWE is easy, such as those in Examples 3.6, then the entire encoding is certainly not pseudorandom, and the chain reactions of LWE breaks at the branch where the "LWE-easy" matrix is stored. If all of them are sampled from LWE-hard distributions, then security depends on how correlated they are and their ordering.

Below we give examples. In all the examples, elements being obfuscated are "small" enough s.t. correctness holds, and are sampled from (Ring)LWE-hard distributions so that they are not trivially distinguishable from random.

Algebraic and graphic expressions for tree-induced homomorphic obfuscators. We introduce informal algebraic and graphic expressions to represent the ring elements stored on the tree, where the ring R is typically $\mathbb{Z}_q^{n\times n}$ as an example for non-commutative plaintext space or $\mathbb{Z}_q[X]/(X^n+1)$ with n being a power of 2 for a commutative case (the ring/dimension of $\mathbf{A}$ will be adjusted correspondingly). Denote $[t]_{v,u}$ as an encoding of $t \in R$ stored in branch $v \rightsquigarrow u$, and a set of them as a tree. For example, $\mathsf{Y}_1 = \{[a]_{1,0}, [b]_{2,0}, [c]_{3,1}, [d]_{4,1}, [e]_{5,1}\}$ represents the tree in Figure 4.1a. If all the elements are sampled independently from $\chi^{n\times n}$ in Y_1, then all the $\mathbf{A}$ matrices are pseudorandom by Theorem 4.7 with the proof of security works by traversing the tree in the order of $[a]_{1,0}, [b]_{2,0}$; $[c]_{3,1}, [d]_{4,1}, [e]_{5,1}$[6].

Next we give some examples where the elements are sampled correlated. The tree in Figure 4.1b can be represented by algebraic expression $\mathsf{Y}_2 = \{[a]_{2,1}, [b]_{1,0}, [a]_{3,0}\}$. The resulting encoding is pseudorandom, given chain reactions in the order of $[b]_{1,0}$; $[a]_{2,1}, [a]_{3,0}$. Namely, we first show $\mathbf{A}_1$ is indistinguishable from random since the reduction can pick its own a and produce the rest of the matrices; then show $\mathbf{A}_2$ and $\mathbf{A}_3$ are pseudorandom, since the sample can be thought as a "longer" LWE sample with the same secret over $\mathbf{A}_1||\mathbf{A}_0$.

The tree in Figure 4.1c can be represented by $\mathsf{Y}_3 = \{[a]_{2,1}, [b]_{1,0}, [a]_{3,0}, [b]_{4,3}\}$. If R is non-commutative (e.g. $\mathbb{Z}_q^{n\times n}$) then we don't know whether $\mathsf{Obf}(\mathsf{Y}_3)$ is distinguishable from random or not; if R is commutative then it is clearly distinguishable from random, since both $\mathbf{A}_2$ and $\mathbf{A}_4$ will be close to $ab\mathbf{A}_0$.

More examples are given when we compare the tree-induced with the graph-induced variant. See Section 5.3.

[6] We use ";" to separate the chain reactions that must happen in orders, "," for the ones that can be triggered in the same step.

5 Graph-Induced Homomorphic Obfuscators and the Safe Modes of GGH15

In this section we "rephrase" the graph-induced multilinear maps constructed by Gentry, Gorbunov and Halevi [GGH15] in the language of homomorphic obfuscation. GGH15 uses the lattice trapdoor sampling techniques to generate the obfuscated code, and the resulting topology is different from the tree variant. An overview of the difference: in the the tree-induced (i.e. trapdoorless) variant, the obfuscated code of $\mathbf{M}_{s,t}$ is $(\mathbf{A}_s, \mathbf{A}_t)$ where $\mathbf{A}_s = \mathbf{M}_{s,t}\mathbf{A}_t + \mathbf{E}_{s,t}$. In GGH15, the code is $\mathbf{D}_{s,t}$, a small m by m matrix s.t.

$$\mathbf{A}_s\mathbf{D}_{s,t} = \mathbf{Y}_{s,t} = \mathbf{M}_{s,t}\mathbf{A}_t + \mathbf{E}_{s,t}$$

where $\mathbf{A}_s$, $\mathbf{A}_t$ are stored in the tail and head vertices, $\mathbf{D}_{s,t}$ is stored in the path. Essentially, computing $\mathbf{A}_s\mathbf{D}_{s,t}$ reveals $\mathbf{Y}_{s,t}$ which is exactly the outer mask "$\mathbf{A}_s$" in the tree variant. However, the slight modification yields a different rule for homomorphic operations. Additions are allowed among the paths with the same tail and head, unlike in the tree where it has only to be the same head. Rules for multiplication are the same, namely if connecting two paths forms a longer path. The evaluation algorithms for predicate-test are also slightly different. As mentioned in the introduction, one can think of graphs as being obtained by "sticking" the leaves of trees.

Another difference is the order of sampling the encodings (during the obfuscation procedure). In the tree-induced variant, the $\mathbf{A}$ matrices are sampled from the root to the leaves, following the route of traversal. In the graph-induced variant, the $\mathbf{A}$ matrices (also stored in the nodes) can be sampled ahead of time. Then the small $\mathbf{D}$ matrices are sampled according to the underlying matrices stored on the paths.

However, the "small solutions" sampled using the trapdoor may leak information about the trapdoor and the secret. As discussed in [GGH15, Section 4.1], sampling the encodings of zero matrices reveals a "weak" trapdoor. On the other hand, to show security for some distributions, we have to identify some criteria under which trapdoored samplings are harmless. To this end, we observe that if the topology of the graph together with the samplers of the matrices satisfy some conditions, then the statistical properties of the discrete Gaussian sampling technique (cf. Lemma 2.8) introduced in the work of Gentry, Peikert and Vaikuntanathan [GPV08] can be used to "turn off" the trapdoors one-by-one.

The rest of this section is organized as follows: First we formalize the syntax and give the construction of graph-induced homomorphic obfuscator for matrix predicates; next we prove security for independent LWE-hard distributions, then give more examples for correlated cases and a comparison with the tree variant.

5.1 Graph-induced homomorphic obfuscator for matrices

First we define the object to be obfuscated, which is a set of matrices stored in a directed graph.

Definition 5.1 (Graph-induced data structure)**.** *A (directed) graph-induced data structure* $\mathsf{G} = (R, \mathcal{V}, \mathcal{P}) \in \mathcal{G}$ *is parameterized by a ring* R*. It contains a set of vertices* $\mathcal{V} = \{\mathsf{vertex}\}$*, a set of paths* $\mathcal{P} = \{\mathsf{path}\}$*. The data structures of the vertices and paths are:*

1. *Each vertex has an ID:* $\mathsf{vertex} = (\mathsf{id})$*;*
2. *Each path stores the IDs of its tail, head, and an element* $\mathsf{ele} \in R$*:* $\mathsf{path} = (\mathsf{tail}, \mathsf{head}, \mathsf{ele})$.

Similar to the previous section, we assume all the natural parameters and efficient operations on a graph are supported and efficiently computable for the graph-induced data structure.

Next we define graph-induced homomorphic obfuscator for matrix predicate P (cf. Def. 3.1). The obfuscator preserves the topology of the graph, which defines the rules of homomorphic operations.

Definition 5.2 (Graph-induced homomorphic obfuscator for matrix predicates)**.** *A graph-induced homomorphic obfuscator* Obf *takes a security parameter* λ*, matrices stored in a graph* $\mathsf{G} = (R, \mathcal{V}, \mathcal{P})$*, outputs* $\mathsf{Obf}(\mathsf{G}) = (\mathsf{Obf}(\mathcal{V}), \mathsf{Obf}(\mathcal{P}))$.

The data structures of obfuscated vertices and paths $\mathsf{Obf}(\mathcal{V}), \mathsf{Obf}(\mathcal{P})$*:*

1. *Each vertex stores its ID and some obfuscated code:* $\mathsf{Obf}(\mathsf{vertex}) = (\mathsf{id}, \mathsf{code})$*;*
2. *Each path stores the IDs of its tail and head, and some obfuscated code:* $\mathsf{Obf}(\mathsf{path}) = (\mathsf{tail}, \mathsf{head}, \mathsf{code})$.

The following (deterministic) polynomial time algorithms are supplied over the encodings:

1. *Negation can be performed on any path:* $\mathsf{Obf}(-\mathsf{path}) := \mathsf{Neg}(\mathsf{Obf}(\mathsf{path}))$.
2. *Addition can be performed if two paths that have the same tail and head: if* $\mathsf{Obf}(\mathsf{path}_1).\mathsf{tail} = \mathsf{Obf}(\mathsf{path}_2).\mathsf{tail}$ *and* $\mathsf{Obf}(\mathsf{path}_1).\mathsf{head} = \mathsf{Obf}(\mathsf{path}_2).\mathsf{head}$*,* $\mathsf{Obf}(\mathsf{path}_{1+2}) := \mathsf{Add}(\mathsf{Obf}(\mathsf{path}_1), \mathsf{Obf}(\mathsf{path}_2))$.
3. *Multiplication can be performed if the head of one is the tail of the other: if* $\mathsf{Obf}(\mathsf{path}_1).\mathsf{head} = \mathsf{Obf}(\mathsf{path}_2).\mathsf{tail}$*,* $\mathsf{Obf}(\mathsf{path}_{1\times 2}) := \mathsf{Mult}(\mathsf{Obf}(\mathsf{path}_1), \mathsf{Obf}(\mathsf{path}_2))$.
4. *The evaluator* Eval *for* P *takes an obfuscated path* $\mathsf{Obf}(\mathsf{path})$ *which encodes* $\mathbf{M}$ *and inputs* $\mathbf{s}$*,* $\mathbf{v}$*, outputs*

$$\mathsf{Eval}(\mathsf{Obf}(\mathsf{path}), \mathbf{s}, \mathbf{v}) = \begin{cases} 1, & \text{if } \mathbf{s}^T\mathbf{M} = \mathbf{v}^T \\ 0, & \text{elsewhere} \end{cases}$$

In addition, Obf *is a VBB obfuscator for predicate* P *over* $\mathcal{G}$ *if for any feasible adversary* A*, there is a p.p.t. simulator* S *and a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$|\Pr[A(\mathsf{Obf}(\mathsf{G}_\lambda)) = \pi(\mathsf{G}_\lambda)] - \Pr[S(\mathsf{G}_\lambda.\mathsf{Topology}) = \pi(\mathsf{G}_\lambda)]| \leq \mathsf{negl}(|\mathsf{G}_\lambda|)$$

for any predicate $\pi : \mathcal{G}_\lambda \to \{0, 1\}$*. The probability is taken over the randomness of distinguisher,* A*,* S*,* Obf*, and the sampler of* G_λ.

Next we describe the LWE-based construction for the default ring $R = \mathbb{Z}_q^{n\times n}$.

Construction 5.3. *A lattice-based graph-induced homomorphic obfuscator takes a security parameter* 1^λ*, a graph of matrices* $\mathsf{G} = (R, \mathcal{V}, \mathcal{P})$*, obfuscate the graph and generate* $\mathsf{Obf}(\mathsf{G}) = (\mathsf{Obf}(\mathcal{V}), \mathsf{Obf}(\mathcal{P}))$ *as follows:*

1. *Compute (parse) the parameters* $n, m, q \in \mathbb{N}$*,* $s \in \mathbb{R}^+$*,* $B \in \mathbb{R}^+$ *(the bound on the norm of vector* $\|\mathbf{s}\|_\infty$*),* $C \in \mathbb{R}^+$ *(the bound on the norm of matrices to be obfuscated* $\|\mathbf{M}\|_\infty$*),* $0 < z < q$ *(the threshold for correctness). The constrains on the choices of parameters are discussed later.*
2. $\forall \mathsf{vertex} \in \mathcal{V}$*, sample* $(\mathbf{A}_{\mathsf{vertex.id}}, \tau_{\mathsf{vertex.id}}) \leftarrow \mathsf{TrapSam}(1^n, 1^m, q)$*. Assign* $\mathsf{Obf}(\mathsf{vertex}).\mathsf{code} := \mathbf{A}_{\mathsf{vertex.id}}$*. Keep all the trapdoors as secret parameters.*
3. $\forall \mathsf{path} \in \mathcal{P}$*, parse* $\mathsf{path.tail.id}$ *as* v*,* $\mathsf{path.head.id}$ *as* u*,* $\mathsf{path.ele}$ *as* $\mathbf{M}_{v,u}$*. Assign the tail and head of the obfuscated path as* v*,* u*. Sample* $\mathbf{E}_{v,u} \leftarrow \chi^{n\times m}$*, compute* $\mathbf{Y}_{v,u} := \mathbf{M}_{v,u}\mathbf{A}_u + \mathbf{E}_{v,u}$*. Compute the obfuscated code on the path as*

$$\mathsf{Obf}(\mathsf{path}).\mathsf{code} := \mathbf{D}_{v,u} \leftarrow \mathsf{PreimgSam}(\mathbf{A}_v, \tau_v, \mathbf{Y}_{v,u}, s)$$

The functionalities are instantiated as follows:

1. $\mathsf{Obf}(-\mathsf{path}) \leftarrow \mathsf{Neg}(\mathsf{Obf}(\mathsf{path}))$*: Parse the ID of the tail and head of* path *as* v, u*. Parse* $\mathsf{Obf}(\mathsf{path}).\mathsf{code} = \mathbf{D}$*. Compute* $\mathsf{Obf}(-\mathsf{path}).\mathsf{tail} := v$*,* $\mathsf{Obf}(-\mathsf{path}).\mathsf{head} := u$*,* $\mathsf{Obf}(-\mathsf{path}).\mathsf{code} = -\mathbf{D}$*.*

2. $\mathsf{Obf}(\mathsf{path}_{1+2}) \leftarrow \mathsf{Add}(\mathsf{Obf}(\mathsf{path}_1), \mathsf{Obf}(\mathsf{path}_2))$*: Parse the IDs of the common tail and head of these two paths as* v, u*. Parse* $\mathsf{Obf}(\mathsf{path}_1).\mathsf{code} = \mathbf{D}_1$*,* $\mathsf{Obf}(\mathsf{path}_2).\mathsf{code} = \mathbf{D}_2$*. Compute* $\mathsf{Obf}(\mathsf{path}_{1+2}).\mathsf{head} := u$*,* $\mathsf{Obf}(\mathsf{path}_{1+2}).\mathsf{tail} := v$*,* $\mathsf{Obf}(\mathsf{path}_{1+2}).\mathsf{code} := \mathbf{D}_1 + \mathbf{D}_2$*.*

3. $\mathsf{Obf}(\mathsf{path}_{1\times 2}) \leftarrow \mathsf{Mult}(\mathsf{Obf}(\mathsf{path}_1), \mathsf{Obf}(\mathsf{path}_2))$*: Let* w*,* v *be the IDs of the tail and head of the first path and* v*,* u *be the ones for the second. Let* $\mathsf{Obf}(\mathsf{path}_1).\mathsf{code} = \mathbf{D}_1$*,* $\mathsf{Obf}(\mathsf{path}_2).\mathsf{code} = \mathbf{D}_2$*. Compute* $\mathsf{Obf}(\mathsf{path}_{1\times 2}).\mathsf{head} := u$*,* $\mathsf{Obf}(\mathsf{path}_{1\times 2}).\mathsf{tail} := w$*,* $\mathsf{Obf}(\mathsf{path}_{1\times 2}).\mathsf{code} := \mathbf{D}_1\mathbf{D}_2$*.*

4. *Predicate-tester* P *takes an obfuscated path* $\mathsf{Obf}(\mathsf{path}_{i,j})$ *and inputs* $\mathbf{s}, \mathbf{v} \in \mathcal{B}^n \setminus \{\mathbf{0}\} \times \mathbb{Z}_q^n$*, parse the code of* $\mathsf{Obf}(\mathsf{path}_{i,j})$ *as* $\mathbf{D}_{i,j}$*, parse the code stored in the tail* (i) *as* $\mathbf{A}_i$*, in the head* (j) *as* $\mathbf{A}_j$*. Compute* $\mathbf{A}_i\mathbf{D}_{i,j} = \mathbf{Y}_{i,j}$*. Finally, evaluate as*

$$\mathsf{Eval}(\mathsf{Obf}(P_{\mathbf{M}_{i,j}}), \mathbf{s}, \mathbf{v}) = \begin{cases} 1, & \text{if } \|\mathbf{s}^T\mathbf{Y}_{i,j} - \mathbf{v}^T\mathbf{A}_j\|_\infty \leq z \\ 0, & \text{elsewhere} \end{cases}$$

Remark 5.4 (Cyclic or acyclic?)**.** *The construction (same as was in [GGH15]) supports cycles. However, [GGH15] presented in a way that the graphs are acyclic. In this paper, we leave cycles possible in the syntax and construction, although all the examples we show that are secure (based on the current library of LWE) in this paper are acyclic. More discussions on cycles see Section 6.1.*

Correctness, setting of parameters. Correctness and setting of parameters follow the original settings in [GGH15, Appendix A]. Here we present them as an immediate extension of the tree-induced variant. Note that calculating $\mathbf{A}_v\mathbf{D}_{v,u}$ yields $\mathbf{Y}_{v,u}$ which is the outer mask in the tree variant. So the correctness of predicate-test, homomorphic operations, noise blow-up, parameters for the hardness of LWE, immediately follow the tree variant. The parameter m will be slightly larger than the tree-variant, as a result of trapdoor sampling.

Remark 5.5 (Generalize to ideal lattices)**.** *The notion of trapdoor and discrete Gaussian sampling in [GPV08] is defined for general lattices. The constructions and analysis in [GPV08, MP12, LPR13b] apply to the setting of RingLWE.*

5.2 Security proof for independent distributions

Similar to the tree-based variant, not all the distributions of graph-induced matrices are obfuscatable by this construction. In fact, explicit examples of insecure distributions are given in [GGH15, Section 4.1]. The examples include encoding zero matrices.

On the secure side, an "upgraded" technique for controlling chain reactions is needed to handle the lattice trapdoors. In particular, we use the discrete Gaussian sampling technique developed in [GPV08] to "turn off" the trapdoors, if the underlying topology and distribution of matrices satisfy certain constrains.

We start with the case where the graph is acyclic and the matrices are sampled independently from LWE-hard distributions. In this case the graph can be VBB obfuscated by construction 5.3.

Theorem 5.6. *If* G *is acyclic, the matrices on the paths are sampled independently from LWE-hard distributions, then Construction 5.3 satisfies VBB security (cf. Def. 5.2).*

Proof. For each path (marked as $v \rightsquigarrow u$), add a field to indicate whether the $\mathbf{D}_{v,u}$ matrix on the path is sampled by the trapdoor of $\mathbf{A}_v$ or is "simulated". In the real instantiation (marked as Hybrid 0), all the $\mathbf{D}$ matrices are sampled by trapdoors. Next we define simulated distributions (gradually) and show that they are indistinguishable from the real one.

1. Initialize the counter $\mathsf{cnt} := 0$;
2. Find a vertex with out-degree 0, or a vertex s.t. all the $\mathbf{D}$ matrices on the paths projected from it are "simulated":
   (a) $\mathsf{cnt} \leftarrow \mathsf{cnt} + 1$;
   (b) Parse its ID as u, its code as $\mathbf{A}_u$;
   (c) For all paths whose head is u:
      i. Parse the ID its tail as v.
      ii. Sample $\{\mathbf{d}_i \leftarrow D_{\mathbb{Z}^m,s}\}_{i\in[m]}$, let $D_{v,u} := [\mathbf{d}_1||...||\mathbf{d}_m]$. Compute $\mathbf{Y}_{v,u} = \mathbf{A}_v\mathbf{D}_{v,u}$.
      iii. Mark the path $v \rightsquigarrow u$ as "simulated".
   (d) Sample the rest of the obfuscated code in the same way as in Hybrid $\mathsf{cnt} - 1$. Mark the current obfuscated code as what is obtained in Hybrid cnt.
3. Repeat step 2 until there are no $\mathbf{D}$ matrices in the paths sampled with trapdoors.

Lemma 5.7. *If the directed graph is acyclic, then step 2 in the algorithm above finds a new vertex every round and the algorithm composed of steps 1-3 above terminates with all the vertices visited.*

Proof. We prove that for every round there is a new vertex satisfies the condition for repetition. For hybrid $\mathsf{cnt} \in \{0, ..., |\mathcal{V}| - 1\}$, start from any vertex w that hasn't been visited. Either the trapdoor τ_w is not used in sampling any $\mathbf{D}$ matrices in the hybrid, or τ_w is still used in sampling say $\mathbf{D}_{w,v}$. If it is the first case, then visit vertex w; if not, then v is not visited, as was implied by step 2.(c).ii. Repeat the checking process above from v, until finding a vertex whose trapdoor is not used. Since the graph is acyclic, such a vertex exists every round and can be find efficiently. By induction, the algorithm above traverses the graph. □

Lemma 5.8. *For* $\mathsf{cnt} \in \{1, ..., |\mathcal{V}| - 1\}$, *Hybrid* $\mathsf{cnt} - 1$ *is indistinguishable from Hybrid* cnt.

Proof. The proof consists of two steps. For vertex u visited in hybrid cnt, we first show that for all the paths that project to u (denote one of them as $v \rightsquigarrow u$), $\mathbf{Y}_{v,u} = \mathbf{M}_{v,u}\mathbf{A}_u + \mathbf{E}_{v,u}$ is indistinguishable from random $\mathbf{U}_{v,u}$; then apply Lemma 2.8 to argue that $\mathbf{D}_{v,u}$ can be sampled without the trapdoor of $\mathbf{A}_v$, indistinguishably[7].

The first step is based on the hardness of DLWE. Assume by contradiction, we use the distinguisher of the graph as an attacker to break DLWE: Given LWE challenge $\mathbf{A} \in \mathbb{Z}^{n\times m}, \mathbf{Y} \in \mathbb{Z}^{n\times m}$ either from LWE samples $\mathbf{A}, \mathbf{MA} + \mathbf{E}$ or random $\mathbf{A}, \mathbf{U}$, we plant the challenge on path $v \rightsquigarrow u$, namely, let $\mathbf{A}_u := \mathbf{A}$ and $\mathbf{Y}_{v,u} := \mathbf{Y}$, sample the rest of the obfuscated code according to Hybrid $\mathsf{cnt} - 1$. The distribution is simulatable since the trapdoor of $\mathbf{A}_u$ (a.k.a. $\mathbf{A}$) is not used in Hybrid $\mathsf{cnt} - 1$, and the rest of the matrices in the graph are independent from the potential LWE secret $\mathbf{M}_{v,u}$.

In the second step, where $\mathbf{Y}_{v,u}$ is uniformly random, $\{\mathbf{A}_v, \mathbf{Y}_{v,u}, \mathbf{D}_{v,u} : \mathbf{D}_{v,u} \leftarrow \mathsf{PreimgSam}(\mathbf{A}_v, \tau_v, \mathbf{Y}_{v,u}, s)\}$ is statistically close to $\{\mathbf{A}_v, \mathbf{Y}_{v,u}, \mathbf{D}_{v,u} : \{\mathbf{d}_i \leftarrow D_{\mathbb{Z}^m,s}\}_{i\in[m]}, \mathbf{D}_{v,u} := [\mathbf{d}_1||...||\mathbf{d}_m], \mathbf{Y}_{v,u} := \mathbf{A}_v\mathbf{D}_{v,u}\}$, by Lemma 2.8. □

[7] Note that there are $\mathsf{in.degree}(u)$-ly many such paths. Technically, the proof can be more precise by introducing $\mathsf{in.degree}(u)$ more sub-hybrids, or relying on DLWE with $\mathsf{in.degree}(u)$ secrets. Since all the secrets are sampled independently, these two views are the same and the sub-hybrid arguments are fairly standard so we choose to absorb these sub-hybrids.

Combining the two lemmas, if the graph is acyclic and all the matrices are sampled independently from LWE-hard distributions, then the obfuscation satisfies VBB security (the simulator outputs samples in Hybrid $|\mathcal{V}| - 1$, include the $\mathbf{A}$ matrices, each independently from $U(\mathbb{Z}_q^{n\times m})$; and the $\mathbf{D}$ matrices, with each row independently from $D_{\mathbb{Z}^m,s}$.) □

5.3 More examples, and a comparison with the tree variant

Next we give examples of graphs with correlated ring elements. The examples are designed with the same principles as was in Section 4.4, that is, the elements are small and sampled from (Ring)LWE-hard distributions. We give examples together with a comparison with the tree variant.

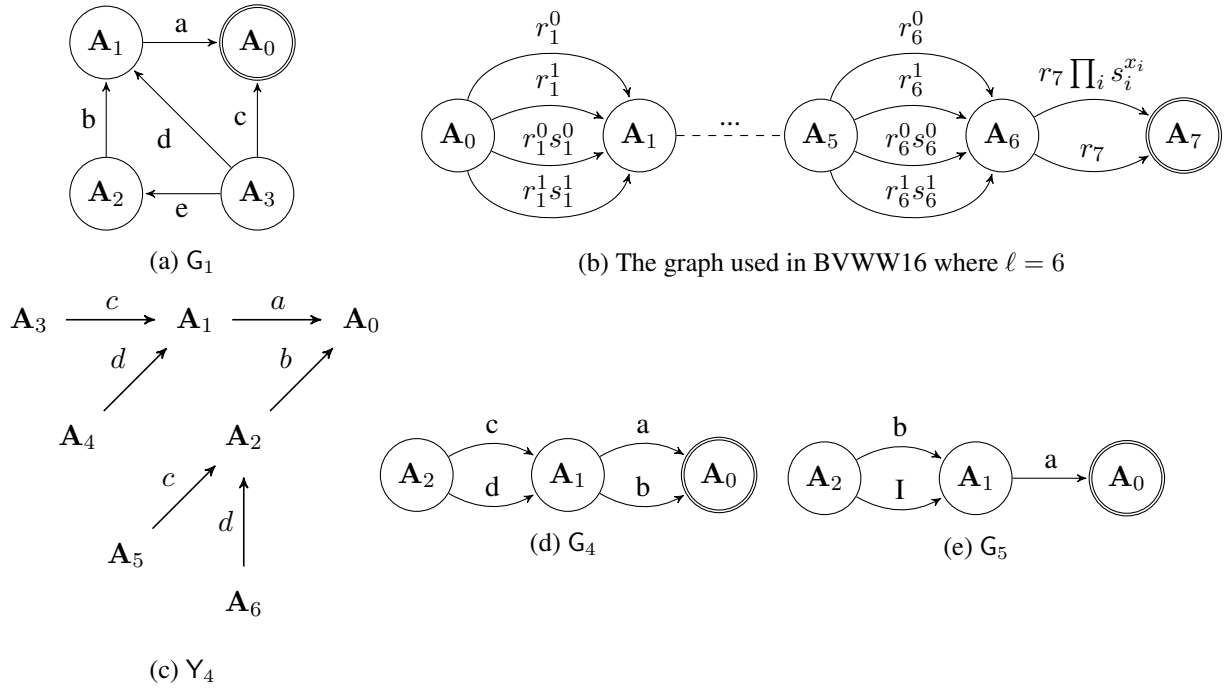
Algebraic and graphic expressions for graph-induced homomorphic obfuscators. Some informal algebraic and graphic expressions are introduced to represent the ring elements stored on the graph. The notations are slightly different from the tree. In the graphic notation, vertices with out-degree 0 are marked with double circle. For the algebraic notation, denote $[\![t]\!]_{v,u}$ as the encoding of $t \in R$ stored in path $v \rightsquigarrow u$, and a set of them forms a graph. For example, $\mathsf{G}_1 = \{[\![a]\!]_{1,0}, [\![b]\!]_{2,1}, [\![c]\!]_{3,0}, [\![d]\!]_{3,1}, [\![e]\!]_{3,2}\}$ represents the graph in Figure 5.1a. If all the elements are sampled independently from $\chi^{n\times n}$ in G_1, then the graph is VBB obfuscated according to Theorem 5.6 with the proof of security going through by traversing the graph in the order of $[\![a]\!]_{1,0}; [\![b]\!]_{2,1}, [\![d]\!]_{3,1}; [\![c]\!]_{3,0}, [\![e]\!]_{3,2}$.

It turns out that the best example to demonstrate the correlated case is the graph used in [BVWW16]. [BVWW16] obfuscates conjunctions with ℓ literals. We simplify the example by taking out the semantics of "conjunctions", ignoring "wildcards", and focus on the encoding itself (see Figure 5.1b for the graph with $\ell = 6$). Assume the elements $\{r_i^b, s_i^b\}_{i\in[\ell],b\in\{0,1\}}, r_{\ell+1}$ are sampled independently from $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ with n being a power of 2. x is sampled from $\{0,1\}^\ell$ with sufficient min-entropy and $\prod_i s_i^{x_i}$ is the subset product determined by x.

To prove security, the hope is to start from $\mathbf{A}_{\ell+1}$ and "turn off" the trapdoor of $\mathbf{A}_\ell$, and move backwards till $\mathbf{A}_0$. However, we don't know how to argue that $\prod_i s_i^{x_i}\mathbf{A}_{\ell+1} + \mathbf{E}$ is indistinguishable from random given the related pieces $\{s_i^b\}_{i\in[\ell]}$ in the air. To address the problem, [BVWW16] proposes a plausible assumption called "entropic RingLWE" which explicitly assume this is pseudorandom as long as x has enough entropy. The rest of the encodings, though still correlated, can be resolved following standard RingLWE.

The applications proposed in GGH15. Our techniques hasn't covered the two applications mentioned in [GGH15, Section 5] based on the current LWE library. For the candidate key-exchange protocol, the encoded elements on the chains are correlated in an "irresolvable" manner. Moreover, the protocol requires public sampling which is not safely realizable by our current proof technique. In fact, an explicit attack on the basic version of GGH15 key-exchange is shown by Coron et al [CLLT15]. For the candidate branching program obfuscator presented in [GGH15], our proof technique fails again due to the irresolvable correlated elements introduced by the underlying matrices themselves and (any subsets of) the "safeguards". We refer the readers to the original paper [GGH15] and the attack by Coron et al [CLLT15] for the details and pictures of these candidates.

Comparisons with the tree variant. We compare trees and graphs through their "expressibility", i.e. we partially answer the question of "Are there any algebraic expressions that can be securely implemented by graphs but not by trees?" and vice versa. Note that without considering security, graphs are strictly more



(a) G_1

(b) The graph used in BVWW16 where $\ell = 6$

(c) Y_4

(d) G_4

(e) G_5

Figure 5.1: Graphs, and graphs versus trees

expressive since trees are acyclic graphs. The comparisons are made based on the current understanding of the applicability of LWE-library and the resulting proof techniques.

Roughly speaking, the branching programs can be instantiated on graph efficiently but not on trees. For example, a sequence of width-w branching program of depth d with independent elements costs wd encodings on a graph (which is optimal); if instantiated by a tree, it has to duplicate the next level and uses roughly w^{d-1} encodings, which means a tree can only instantiate a random branching program with constant depth (or logarithmic depth if the width is constant). See Figures 5.1c and 5.1d for examples of safely instantiating depth-2 width-2 random branching programs. Similarly, we don't know how to express the encodings in [BVWW16] by trees for more than $O(\log(\lambda))$ literals.

The advantage of trees is allowing "public sampling" (though with truncated functionalities), as illustrated by Algorithm 4.6. For graphs we only know how to publicly sample random encodings w.r.t. a specific path (by publishing a lot of encodings of random elements on that path plus discrete-Gaussian leftover hash lemma [AGHS13, AR13]), but haven't yet found any safe ways to publicly sample encodings of "specific" elements.

Also, trees are more expressive on "subset sums". For example, $\mathsf{Y}_5 = \{[b]_{2,1}, [a]_{1,0}\}$ (figure omitted) supports affine combination of ba and a (namely $c_2ba + c_1a + c_0$ with small $c_2, c_1, c_0 \in R$ chosen by the evaluator). For graphs, to safely encode even a limited arithmetic expression (say $ba+a$) requires a quite delicate construction, such as by G_5 in Figure 5.1e. Note that the identity I is not a typical "innocent" element to be encoded in the graph. In $\mathsf{G}_5 = \{[\![b]\!]_{2,1}, [\![I]\!]_{2,1}, [\![a]\!]_{1,0}\}$, it is safe since the product of identity (either in the ring of integer matrix or the cyclotomic ring) and $\mathbf{A}_1$ is random, therefore the simulator can sample $\mathbf{A}_1$ by first sample $\mathbf{A}_2$, $\mathbf{D}_I$ and compute $\mathbf{A}_1 := \mathbf{A}_2\mathbf{D}_I$ (this trick is not used in the proof of Theorem 5.6). However, if there are two identities projected to the same vertex (not necessarily started from the same source), then we don't know how to simulate in general.

As a final remark, trees and graphs can be mix used to make the best use of their advantages and overcome their limitations. One can locally detach the tails of two paths in the graph and make them the leaves of a

subtree. The candidate branching program obfuscator in [GGH15] can be seen as graph-induced but zero-test in a tree manner.

5.4 APIs for multilinear maps

We informally translate the tree and graph-induced homomorphic obfuscators into the language of multilinear maps, together with a comparison to the original presentation in [GGH15].

- Generate public/secret parameters. In the tree-induced multilinear maps, public parameters are n, m, q, s, B, C; there are no secret parameters. In the graph variant, additionally sample the $\mathbf{A}$ matrices with trapdoors. Include all the $\mathbf{A}$ matrices in the public parameters and set the trapdoors as secret parameters.

  In [GGH15], not all the $\mathbf{A}$ matrices are published but only those where zero-test is used. Here we adapt this option but remark that this is a statement of "minimum components needed for zero-test at certain vertex", rather than claiming "all the unpublished $\mathbf{A}$ matrices are hided". In fact [GGH15, Section 4.2] shows that in some cases, unpublished $\mathbf{A}$ matrices can be recovered. On the other hand, for all the secure examples in this paper, publishing $\mathbf{A}$ matrices or not doesn't affect the proof of security.

- Privately sampling encodings is equivalent to the obfuscation algorithms.

- Public sampling is not known to be secure in the graph variant. The tree supports secure public sampling with truncated functionalities. See Algorithm 4.6 for the details.

- Negation, addition and multiplication are the homomorphic operations in the obfuscators.

- "Zero-test" in the tree variant is equivalent to checking if the outer-mask is small in the tree variant. In the graph, recover $\mathbf{Y}$ by computing $\mathbf{AD}$ then reduce to the tree case. The ideal-testing functionalities in the homomorphic obfuscators corresponds to "zero-testing" and "random-testing" each row in a matrix or each coordinate of the element in the cyclotomic rings (via canonical embedding).

The corresponding security notion we obtain is literally "pseudorandomness of the encodings" when the ring elements are sampled from certain "nice" distributions (such as independently from LWE-hard distributions). We haven't found any convincing analog of "multilinear-DDH" that is sufficient for non-interactive multiparty key-exchange (which requires public sampling, and possibly commutative ciphertext space).

6 Fear and Desire

In this section we semi-formally discuss future directions.

6.1 Learning with errors with new friends

We propose several extensions of learning with errors and discuss their plausibility. We assume the readers are familiar with the typical useful and nontrivial settings of parameters on the magnitude of noise and the distributions of secrets, as we will sometimes be imprecise of them in this section.

The first extension (and possibly the strongest throughout this section) is made to address the safety of loops (esp. self-loops) in the graph structure. Informally, denote $\mathbf{S} \in R$ as the secret, where the typical rings are $R = \mathbb{Z}_q^{n \times n}$ as an example of non-commutative secret space or $R = \mathbb{Z}_q[X]/(X^n + 1)$ with n being a power of

2 as a commutative variant. We ask if given $\mathbf{A}, \mathbf{SA}+\mathbf{E}$, and $\mathbf{D} \leftarrow \mathsf{PreimgSam}(\mathbf{A}, \tau, \mathbf{SA}+\mathbf{E}, s)$, are they indistinguishable from random? If loops can be proved secure, it would dramatically expand the coverage of LWE library and, as a "byproduct", the coverage of the graph-based homomorphic obfuscators (multilinear maps). Just to name a few implications:

1. Consider "LWE with power" for small $\mathbf{S}$, where the challenge is to distinguish $\mathbf{A}$, $\mathbf{SA}+\mathbf{E}_1$, $\mathbf{S}^2\mathbf{A}+\mathbf{E}_2$, ... from random $\mathbf{A}$, $\mathbf{U}_1$, $\mathbf{U}_2$, ... (the exact number of powers depends on the setting of the norm of $\mathbf{S}$ and $\mathbf{E}$ relative to q). LWE with power (for proper choices of parameters) is implied by self-loops since the samples can be generated by $\mathbf{S}$ encoded in the loop.

2. Bounded depth homomorphic obfuscator that hides the computation. Given that encoded elements in the self-loops can add and multiply themselves and the other encodings in the same loop.

3. If the same plaintext can be encoded for both directions and forms a loop, it can be thought as supporting commutative ciphertext space for multilinear maps,

Formally, we introduce *Learning with errors with shadow*, where the adversary is given LWE samples $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{Y} = \mathbf{SA}+\mathbf{E} \in \mathbb{Z}_q^{n\times m}$, together with a "small solution" $\mathbf{D} \leftarrow \mathsf{PreimgSam}(\mathbf{A}, \tau, \mathbf{Y}, s)$ (for the dimension to match secrets must be a square matrix $\mathbf{S} \in \mathbb{Z}_q^{n\times n}$). $\mathbf{D}$ is called the *shadow* of $\mathbf{Y}$.

Definition 6.1 (Learning with errors with shadow (LWE-shadow))**.** *For $(\mathbf{A}, \tau) \leftarrow \mathsf{TrapSam}(1^n, 1^m, q)$. Given $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{Y} = \mathbf{SA}+\mathbf{E}$, and a small matrix $\mathbf{D} \leftarrow \mathsf{PreimgSam}(\mathbf{A}, \tau, \mathbf{Y}, s)$. The search LWE-shadow problem asks to find $\mathbf{S}$ given $\mathbf{A}$, $\mathbf{D}$. The decisional LWE-shadow problem asks to distinguish whether $\mathbf{Y}$ is from LWE sample or uniformly random in $\mathbb{Z}_q^{n\times m}$, given $\mathbf{A}$, $\mathbf{D} \leftarrow D_{\Lambda^{\perp}_{\mathbf{Y}}(\mathbf{A}),s}$. (In both problems $\mathbf{Y}$ is computable from $\mathbf{A}$ and $\mathbf{D}$ so it doesn't have to be given out.)*

When the error magnitude is above the smoothing parameter of $\Lambda(\mathbf{A})$, the two distributions are statistically close due to Lemma 2.8, but the LWE problem itself will be statistically unsolvable thus with no utility. The interesting case will be when the error magnitude is same to the ones in the typical LWE problems. For the "useful" error magnitude, the indistinguishability is not implied by Lemma 2.8. Also it is not clear whether sampling $\mathbf{D}$ using the trapdoor will leak more information about the trapdoor τ itself or the secret $\mathbf{S}$.

The definition naturally generalizes to RingLWE with shadow. However, it turns out to be quite easy to show counterexamples with commutative secret spaces.

Example 6.2 (Counterexample of RingLWE-shadow for small secrets)**.** *Consider small secrets s_1 and s_2 (say, both sampled from error distribution) both encoded with respect to $\mathbf{A} \rightsquigarrow \mathbf{A}$. Denote their encodings as $\mathbf{D}_1$, $\mathbf{D}_2$, then $\mathbf{AD}_1\mathbf{D}_2$ and $\mathbf{AD}_2\mathbf{D}_1$ are close to $s_1s_2\mathbf{A}$, therefore it is certainly distinguishable from random. Moreover, with high probability, subtracting the encodings yields a non-trivial encoding of zero $\mathbf{D}_1\mathbf{D}_2 - \mathbf{D}_2\mathbf{D}_1$, a.k.a. a "weak" trapdoor of $\mathbf{A}$. The consequence of leaking weak trapdoor see the discussions in [GGH15, Section 4.1].*

On the other hand, we show that a variant of LWE-shadow, called *Learning with errors with joint shadow*, is hard. Roughly speaking, it considers giving the joint shadow of 2 independent LWE samples. This variant (both for commutative and non-commutative secret spaces) does not seem to be useful for multiplicative homomorphism in the loops. However, the problem and technique might be of independent interest so we choose to present it for reference.

Definition 6.3 (LWE with Joint Shadow)**.** $(\mathbf{A}_1||\mathbf{A}_2, \tau_{1,2}) \leftarrow \mathsf{TrapSam}(1^n, 1^{2m}, q)$. *Given $\mathbf{A}_1||\mathbf{A}_2 \in \mathbb{Z}_q^{n\times 2m}$, $\{\mathbf{Y}_i = \mathbf{S}_i\mathbf{A}_i + \mathbf{E}_i\}_{i=1,2}$, and a small matrix $\mathbf{D} \leftarrow \mathsf{PreimgSam}(\mathbf{A}_1||\mathbf{A}_2, \tau_{1,2}, \mathbf{Y}_1||\mathbf{Y}_2, s)$. Search LWE-joint-shadow: find $\mathbf{S}_1$, $\mathbf{S}_2$ given $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{D}$. Decisional LWE-joint-shadow: distinguish whether $\mathbf{Y}_1||\mathbf{Y}_2$ is from LWE samples or uniformly random in $\mathbb{Z}_q^{n\times 2m}$, given $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{D} \leftarrow D_{\Lambda^{\perp}_{\mathbf{Y}_1||\mathbf{Y}_2}(\mathbf{A}_1||\mathbf{A}_2),s}$.*

Theorem 6.4. *Decisional LWE with joint shadow (*$\mathsf{DLWE}_{n,q,2m,\chi,\mathcal{V}}$*-joint-shadow) is as hard as standard Decisional LWE* $\mathsf{DLWE}_{n,q,m,\chi,\mathcal{V}}$.

Proof idea: instead of sampling the shadow using the full-fledged trapdoor of $\mathbf{A}_1||\mathbf{A}_2$ directly, which could be "explicit" enough to reveal both the trapdoor of $\mathbf{A}_1$ and $\mathbf{A}_2$, we use the trapdoor of one of them (say $\mathbf{A}_1$) and extend it to the trapdoor of $\mathbf{A}_1||\mathbf{A}_2$ using the "Bonsai technique" [CHKP12] (cf. Lemma 2.9). Note that this doesn't give a trapdoor of $\mathbf{A}_2$ and in fact doesn't affect the pseudorandomness of LWE instances on $\mathbf{A}_2$ (unless the lattice-based cryptography collapses). We then "plant" the LWE challenges on the safe side.

Proof. We elaborate the proof in a sequence of hybrid experiments. Descriptions of the hybrids:

1. Hybrid 1 refers to the case where the adversary is given the "LWE" side of the DLWE-joint-shadow challenge.
2. In Hybrid 2, instead of sampling the trapdoor $\tau_{1,2}$ with $\mathbf{A}_1||\mathbf{A}_2$, we sample $\mathbf{A}_1$ with a trapdoor τ_1 and extend it to a working trapdoor for $\mathbf{A}_1||\mathbf{A}_2$ by $\tau_{1\to1,2} \leftarrow \mathsf{ExtBasis}(\tau_1, \mathbf{A}_1||\mathbf{A}_2)$. Continue with $\tau_{1\to1,2}$ in the rest of the experiment identical to Hybrid 1.
3. Hybrid 3 is almost the same with Hybrid 2 except that $\mathbf{Y}_2$ is sampled from $U(\mathbb{Z}_q^{n\times m})$ instead of $\mathbf{S}_2\mathbf{A}_2 + \mathbf{E}_2$. The shadow is computed in the same way as in Hyb 2 except with the random $\mathbf{Y}_2$: $\mathbf{D} \leftarrow \mathsf{PreimgSam}(\mathbf{A}_1||\mathbf{A}_2, \tau_{1\to1,2}, \mathbf{Y}_1||\mathbf{Y}_2, s)$.
4. In Hybrid 4, we sample $\mathbf{A}_2$ with trapdoor τ_2 instead of $\mathbf{A}_1$. Then extend it to a working trapdoor for $\mathbf{A}_1||\mathbf{A}_2$ by $\tau_{2\to1,2} \leftarrow \mathsf{ExtBasis}(\tau_2, \mathbf{A}_2||\mathbf{A}_1)$. Continue with $\tau_{2\to1,2}$ in the rest of the experiment in the same way to Hybrid 3.
5. Hybrid 5 is almost the same with Hybrid 4 except that $\mathbf{Y}_1$ is sampled from $U(\mathbb{Z}_q^{n\times m})$ instead of $\mathbf{S}_1\mathbf{A}_1 + \mathbf{E}_1$. The shadow is computed in the same way as in Hybrid 4 except with the random $\mathbf{Y}_1$, as $\mathbf{D} \leftarrow \mathsf{PreimgSam}(\mathbf{A}_1||\mathbf{A}_2, \tau_{2\to1,2}, \mathbf{Y}_1||\mathbf{Y}_2, s)$.
6. Hybrid 6 refers to the case the case where the adversary is given the "uniform" side of the DLWE-joint-shadow challenge. The difference of Hybrid 5 and 6 is that Hybrid 6 use a general trapdoor $\tau_{1,2}$ sampled together with $\mathbf{A}_1||\mathbf{A}_2$.

Lemma 6.5. *The distributions Hybrid 1 and 2, 3 and 4, 5 and 6 are statistically close.*

Proof. By Lemma 2.9 of Bonsai technique, the "quality" of $\tau_{1,2}$, $\tau_{1\to1,2}$ and $\tau_{2\to1,2}$ are the same. In addition, by Lemma 2.8, the shadow $\mathbf{D}$ sampled between Hybrids 1 and 2, 3 and 4, 5 and 6 are statistically close. □

Lemma 6.6. *Assuming DLWE, the distributions in Hybrid 2 and 3, 4 and 5 are computationally indistinguishable.*

Proof. Suppose by contradiction, there's a distinguisher for Hybrid 2 and 3, we use it to break DLWE. Upon receiving the DLWE challenge $\mathbf{A}$, $\mathbf{Y} = \mathbf{S}\mathbf{A} + \mathbf{E}$ or uniform, plant it at position 2. Namely, sample $\mathbf{A}_1$ with trapdoor τ_1, use Bonsai technique to extend it to a trapdoor τ^* for $\mathbf{A}_1||\mathbf{A}$. Pick a secret $\mathbf{S}_1$, an error matrix $\mathbf{E}_1 \leftarrow \chi^{n\times m}$, sample the shadow of $(\mathbf{S}_1\mathbf{A}_1 + \mathbf{E}_1)||\mathbf{Y}$ on $\mathbf{A}_1||\mathbf{A}$ using the trapdoor τ^*. When $\mathbf{Y}$ is the LWE sample, the distribution is statistically close to what defined in Hybrid 2; otherwise it is statistically close to the one in Hybrid 3. The distinguisher of Hybrid 2 and 3 can be immediately turned into an attacker of DLWE. Indistinguishability of Hybrid 4 and 5 can be obtained symmetrically. □

The proof completes by combining the above lemmas. □

The same idea applies to RingLWE with joint shadow. Note that it doesn't contradict to Counterexample 6.2.

The negative and positive results above haven't ruled out or proved LWE-with-shadow for non-commutative secret space, e.g. $\mathbb{Z}_q^{n\times n}$.

Narrowing down the questions by not giving out the "shadow", we propose a simpler LWE extension with related secrets, called *Uncertainty LWE*: for "small" $\mathbf{M}_1, \mathbf{M}_2$ sampled independently from LWE-hard distributions in $\mathbb{Z}_q^{n\times n}$, the adversary is given $\mathbf{A}$, $\mathbf{M}_1\mathbf{M}_2\mathbf{A}+\mathbf{E}_1$, $\mathbf{M}_2\mathbf{M}_1\mathbf{A}+\mathbf{E}_2$, and is asked to find $\mathbf{M}_1$, $\mathbf{M}_2$, or distinguish the samples from random. Uncertainty LWE is implied by LWE with shadow with 2 samples living in the same loop. The tree in Figure 4.1c with $a, b \in \mathbb{Z}_q^{n\times n}$ can be seen as "Uncertainty LWE" plus more hints. We don't know how to distinguish them from random either.

6.2 Potential future directions for homomorphic obfuscators and multilinear maps

In this paper, homomorphic obfuscators and multilinear maps happened to encode the same type of objects, namely matrices or elements in the rings of cyclotomic polynomials (in general, bases of "crypto-friendly" lattices). However, the personal opinion of the author is that it is a coincidence rather than being necessarily linked. Obfuscators will be more useful if they cover more functionalities, whereas multilinear maps will be easier to use if the elements being encoded are more elementary, e.g. integers in $\mathbb{Z}_q$.

Homomorphism over obfuscations is a new concept yet to be explored. If we take a step back, even without considering homomorphism, we don't have many constructions for obfuscators with elementary but "non-trivial" functionalities (except from candidate iOs or relying on utopian assumptions over candidate graded encoding). For example, obfuscator for permutations (its representation and interface to be determined by the potential application) is a very natural target that we currently don't have a candidate. And studying how to obfuscate permutation matrices homomorphically may shed lights on general branching program obfuscations.

For multilinear maps, our work shows that the security of GGH15 over some distributions on the plaintexts can be based on standard cryptographic assumptions. However, this candidate is limited due to the special plaintext spaces and the non-commutative ciphertext space (unless loop is proved a good idea). They don't pose an obstacle to instantiate branching program obfuscations, but are not "user friendly" to other potential applications. It remains an open problem to propose a reliable candidate for "general" plaintext distributions (or provide more evidences that GGH13 and CLT13 are still good templates towards achieving this goal).

Finally, we remark that the "reasonable" approach of obtaining multilinear maps mentioned in the introduction, which is via obfuscating the partial decryption algorithm of somewhat homomorphic encryption schemes, hasn't been justified in this article and remains a plausible future direction.

Acknowledgments

I am grateful to Vinod Vaikuntanathan for his wonderful course of Lattices (6.876J). Great thanks to Shai Halevi for his tireless updates of the status of candidate multilinear maps, among those exciting and helpful talks I was fortunate to attended four (in ICERM, Berkeley, Charles River Crypto day and Sde Boker). I also thank Justin Holmgren for introducing the work of quantum money, and Ryo Nishimaki for pointing me the work of patchable obfuscation.

References

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In *Theory of Cryptography*, pages 528–556. Springer, 2015.

[ABD16] Martin Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched ntru assumptions: Cryptanalysis of some fhe and graded encoding schemes. Cryptology ePrint Archive, Report 2016/127, 2016.

[AC12] Scott Aaronson and Paul Christiano. Quantum money from hidden subspaces. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 41–60. ACM, 2012.

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

[AFH+16] Martin R Albrecht, Pooya Farshim, Dennis Hofheinz, Enrique Larraia, and Kenneth G Paterson. Multilinear maps from obfuscation. In *Theory of Cryptography*, pages 446–473. Springer, 2016.

[AGHS13] Shweta Agrawal, Craig Gentry, Shai Halevi, and Amit Sahai. Discrete gaussian leftover hash lemma over infinite domains. In *Advances in Cryptology-ASIACRYPT 2013*, pages 97–116. Springer, 2013.

[AGIS14] Prabhanjan Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding barrington's theorem. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 646–658. ACM, 2014.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Patchable obfuscation. Cryptology ePrint Archive, Report 2015/1084, 2015.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC*, pages 99–108, 1996.

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

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

[AP16] Navid Alamati and Chris Peikert. Three's compromised too: Circular insecurity for any cycle length from (ring-)lwe. Cryptology ePrint Archive, Report 2016/110, 2016.

[AR13] Divesh Aggarwal and Oded Regev. A note on discrete gaussian combinations of lattice vectors. *arXiv preprint arXiv:1308.2405*, 2013.

[BBC+14] Boaz Barak, Nir Bitansky, Ran Canetti, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Obfuscation for evasive functions. In Lindell [Lin14], pages 26–51.

[BC14] Nir Bitansky and Ran Canetti. On strong simulation and composable point obfuscation. *J. Cryptology*, 27(2):317–357, 2014.

[BCKP14] Nir Bitansky, Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On virtual grey box obfuscation for general circuits. In Garay and Gennaro [GG14b], pages 108–125.

[BGH+15] Zvika Brakerski, Craig Gentry, Shai Halevi, Tancrde Lepoint, Amit Sahai, and Mehdi Tibouchi. Cryptanalysis of the quadratic zero-testing of ggh. Cryptology ePrint Archive, Report 2015/845, 2015.

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

[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, *EUROCRYPT*, volume 8441 of *LNCS*, pages 221–238. Springer, 2014.

[BHHO08] Dan Boneh, Shai Halevi, Mike Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In *Advances in Cryptology–CRYPTO 2008*, pages 108–125. Springer, 2008.

[BHW15] Allison Bishop, Susan Hohenberger, and Brent Waters. New circular security counterexamples from decision linear and learning with errors. In *Advances in Cryptology–ASIACRYPT 2015*, pages 776–800. Springer, 2015.

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

[BMSZ15] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: The case of evasive circuits. Cryptology ePrint Archive, Report 2015/167, 2015.

[BR13] Zvika Brakerski and Guy N Rothblum. Obfuscating conjunctions. In *Advances in Cryptology–CRYPTO 2013*, pages 416–434. Springer, 2013.

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Lindell [Lin14], pages 1–25.

[BRS13a] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In *Advances in Cryptology–CRYPTO 2013*, pages 461–478. Springer, 2013.

[BRS13b] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private subspace-membership encryption and its applications. In *Advances in Cryptology-ASIACRYPT 2013*, pages 255–275. Springer, 2013.

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

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

[BVWW16] Zvika Brakerski, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Obfuscating conjunctions under entropic ring LWE. In Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 147–156. ACM, 2016.

[BWZ14] Dan Boneh, David J. Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. *IACR Cryptology ePrint Archive*, 2014:930, 2014.

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *Advances in CryptologyCRYPTO'97*, pages 455–469. Springer, 1997.

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Obfuscating point functions with multibit output. In *Advances in Cryptology–EUROCRYPT 2008*, pages 489–508. Springer, 2008.

[CFL+16] Jung Hee Cheon, Pierre-Alain Fouque, Changmin Lee, Brice Minaud, and Hansol Ryu. Cryptanalysis of the new clt multilinear map over the integers. Cryptology ePrint Archive, Report 2016/135, 2016.

[CG13] Ran Canetti and Juan A. Garay, editors. *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *LNCS*. Springer, 2013.

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrede Lepoint, Hemanta K Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New mmap attacks and their limitations. In *Advances in Cryptology–CRYPTO 2015*, pages 247–266. Springer, 2015.

[CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of cryptology*, 25(4):601–639, 2012.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *LNCS*, pages 3–12. Springer, 2015.

[CJL16] Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for cspr problems and cryptanalysis of the ggh multilinear map without an encoding of zero. Cryptology ePrint Archive, Report 2016/139, 2016.

[CLLT15] Jean-Sebastien Coron, Moon Sung Lee, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of ggh15 multilinear maps. Cryptology ePrint Archive, Report 2015/1037, 2015.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Canetti and Garay [CG13], pages 476–493.

[CLT14] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of two candidate fixes of multilinear maps over the integers. *IACR Cryptology ePrint Archive*, 2014:975, 2014.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *LNCS*, pages 267–286. Springer, 2015.

[CRV10] Ran Canetti, Guy N Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In *TCC*, volume 10, pages 72–89. Springer, 2010.

[DMQ13] Nico Döttling and Jörn Müller-Quade. Lossy codes and a new variant of the learning-with-errors problem. In *Advances in Cryptology–EUROCRYPT 2013*, pages 18–34. Springer, 2013.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Rafols, and Jorge Villar. An algebraic framework for diffie-hellman assumptions. In *Advances in Cryptology–CRYPTO 2013*, pages 129–147. Springer, 2013.

[FMR13] Benjamin Fuller, Xianrui Meng, and Leonid Reyzin. Computational fuzzy extractors. In *Advances in Cryptology-ASIACRYPT 2013*, pages 174–193. Springer, 2013.

[GF05] Harold N. Gabow and Ronald Fagin, editors. *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*. ACM, 2005.

[GG14a] Juan A. Garay and Rosario Gennaro, editors. *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *LNCS*. Springer, 2014.

[GG14b] Juan A. Garay and Rosario Gennaro, editors. *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, volume 8617 of *LNCS*. Springer, 2014.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Johansson and Nguyen [JN13], pages 1–17.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49. IEEE Computer Society, 2013.

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

[GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Functional encryption without obfuscation. Cryptology ePrint Archive, Report 2014/666, 2014.

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple bgn-type cryptosystem from LWE. In Gilbert [Gil10], pages 506–522.

[Gil10] Henri Gilbert, editor. *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *LNCS*. Springer, 2010.

[GKPV10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In Andrew Chi-Chih Yao, editor, *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 230–240. Tsinghua University Press, 2010.

[GLSW15] Craig Gentry, Allison Bishop Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 151–170. IEEE, 2015.

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

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Canetti and Garay [CG13], pages 75–92.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from lwe. In *Advances in Cryptology–CRYPTO 2015*, pages 503–523. Springer, 2015.

[Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *LNCS*, pages 443–457. Springer, 2000.

[HJ15] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. *IACR Cryptology ePrint Archive*, 2015:301, 2015.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. Ntru: A ring-based public key cryptosystem. In *Algorithmic number theory*, pages 267–288. Springer, 1998.

[JN13] Thomas Johansson and Phong Q. Nguyen, editors. *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *LNCS*. Springer, 2013.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In Janos Simon, editor, *STOC*, pages 20–31. ACM, 1988.

[KW16] Venkata Koppula and Brent Waters. Circular security counterexamples for arbitrary length cycles from lwe. Cryptology ePrint Archive, Report 2016/117, 2016.

[Lin14] Yehuda Lindell, editor. *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, volume 8349 of *LNCS*. Springer, 2014.

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

[LPR13b] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In Johansson and Nguyen [JN13], pages 35–54.

[LPSS14] San Ling, Duong Hieu Phan, Damien Stehlé, and Ron Steinfeld. Hardness of k-lwe and applications in traitor tracing. In Garay and Gennaro [GG14a], pages 315–334.

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

[MP13] Daniele Micciancio and Chris Peikert. Hardness of sis and lwe with small parameters. In *Advances in Cryptology–CRYPTO 2013*, pages 21–39. Springer, 2013.

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

[MRV15] Paz Morillo, Carla Ràfols, and Jorge L. Villar. Matrix computational assumptions in multilinear groups. Cryptology ePrint Archive, Report 2015/353, 2015.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over ggh13. Cryptology ePrint Archive, Report 2016/147, 2016.

[OPW11] Adam ONeill, Chris Peikert, and Brent Waters. Bi-deniable public-key encryption. In *Advances in Cryptology–CRYPTO 2011*, pages 525–542. Springer, 2011.

[Pei07] Chris Peikert. Limits on the hardness of lattice problems in ℓ_p norms. In *22nd Annual IEEE Conference on Computational Complexity (CCC 2007), 13-16 June 2007, San Diego, California, USA*, pages 333–346. IEEE Computer Society, 2007.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 333–342. ACM, 2009.

[Pei15] Chris Peikert. A decade of lattice cryptography. *IACR Cryptology ePrint Archive*, 2015:939, 2015.

[PFP15] Marta Conde Pena, Jean-Charles Faugère, and Ludovic Perret. Algebraic cryptanalysis of a quantum money scheme the noise-free case. In *Public-Key Cryptography–PKC 2015*, pages 194–213. Springer, 2015.

[Pie12] Krzysztof Pietrzak. Subspace lwe. In *TCC*, volume 7194, pages 548–563. Springer, 2012.

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

[PS15] Omer Paneth and Amit Sahai. On the equivalence of obfuscation and multilinear maps. *IACR Cryptology ePrint Archive*, 2015:791, 2015.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Garay and Gennaro [GG14a], pages 500–517.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Gabow and Fagin [GF05], pages 84–93.

[Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Gilbert [Gil10], pages 24–43.

[Wee05] Hoeteck Wee. On obfuscating point functions. In Gabow and Fagin [GF05], pages 523–532.

[YYHK14] Takashi Yamakawa, Shota Yamada, Goichiro Hanaoka, and Noboru Kunihiro. Self-bilinear map on unknown order groups from indistinguishability obfuscation and its applications. In Garay and Gennaro [GG14b], pages 90–107.

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *LNCS*, pages 439–467. Springer, 2015.