Return of GGH15: Provable Security Against Zeroizing Attacks*

James Bartusek**, Jiaxin Guan***, Fermi Ma†, and Mark Zhandry‡

Princeton University

Abstract. The GGH15 multilinear maps have served as the foundation for a number of cutting-edge cryptographic proposals. Unfortunately, many schemes built on GGH15 have been explicitly broken by so-called "zeroizing attacks," which exploit leakage from honest zero-test queries. The precise settings in which zeroizing attacks are possible have remained unclear. Most notably, none of the current indistinguishability obfuscation (iO) candidates from GGH15 have any formal security guarantees against zeroizing attacks.

In this work, we demonstrate that all known zeroizing attacks on GGH15 implicitly construct *algebraic relations* between the results of zero-testing and the encoded plaintext elements. We then propose a "GGH15 zeroizing model" as a new general framework which greatly generalizes known attacks.

Our second contribution is to describe a new GGH15 variant, which we formally analyze in our GGH15 zeroizing model. We then construct a new iO candidate using our multilinear map, which we prove secure in the GGH15 zeroizing model. This implies resistance to all known zeroizing strategies. The proof relies on the Branching Program Un-Annihilatability (BPUA) Assumption of Garg et al. [TCC 16-B] (which is implied by PRFs in NC^1 secure against $\mathsf{P/poly}$) and the complexity-theoretic p-Bounded Speedup Hypothesis of Miles et al. [ePrint 14] (a strengthening of the Exponential Time Hypothesis).

1 Introduction

1.1 Motivation

Multilinear maps [2] are a powerful cryptographic tool that have enabled many cryptographic applications, ranging from multiparty key agreement [2] to extremely powerful indistinguishability obfuscation (iO) [3]. There are currently three families of multilinear maps: those of Garg, Gentry, and Halevi [4] (GGH13), those of Coron, Lepoint, and Tibouchi [5] (CLT13), and those of Gentry, Gorbunov, and Halevi [6] (GGH15).

* The full version of this paper is available on the IACR ePrint Archive [1].

** bartusek.james@gmail.com

*** jiaxin@guan.io

† fermima1@gmail.com

‡ mzhandry@princeton.edu

Each of these multilinear map families are based on fully homomorphic encryption (FHE) schemes. However, the FHE schemes are intentionally weakened by providing a broken secret key to allow useful information to be extracted from encrypted values. Because of these broken secret keys, extensive cryptanalysis is required before we can gain confidence that some security remains. In this work, we study the GGH15 multilinear maps. We believe these maps are particularly interesting for a couple reasons:

- In some cases, by specializing the GGH15 construction to certain settings, security can actually be proved based on the well-studied Learning with Errors (LWE) assumption [7]. Notably, the lockable obfuscation constructions of Wichs and Zirdelis [8], of Goyal, Koppula, and Waters [9], and of Chen, Vaikuntanathan, and Wee [10], and the private puncturable PRFs of Canetti and Chen [11] and Chen et. al. [10], are all based in part on the GGH15 multilinear maps, and can be proved secure under LWE.[1] Therefore, the GGH15 multilinear maps seem to be the most promising route to achieving security based on LWE.
- The other two candidate multilinear maps, GGH13 and CLT13, have been shown vulnerable to quantum attacks [13–17]. In contrast, given the positive results above and the fact that LWE appears resistant to quantum attacks, it seems reasonable to expect that GGH15 is quantum immune, at least in certain settings. This leaves GGH15 as the main candidate multilinear map for the post-quantum era.

Despite the above positive results, there is still a large gap between what is provably secure under LWE and what the community hopes to achieve with multilinear maps, namely iO. On the positive side, "direct attacks" on the multilinear maps seem unlikely. Here "direct attacks" refer to attempts to attack the underlying FHE schemes, ignoring the extra information provided through the broken secret key.

Unfortunately, all multilinear map candidates have been subject to very strong "zeroizing" attacks [4, 18, 19] which exploit the broken secret key. These attacks have broken many of the applications which had not been proven secure. Since the original attacks, the field has seen a continual cycle of breaking schemes and fixing them. In the case of GGH15, these attacks [19, 20, 10] have broken many applications, including multiparty key agreement, and several of the iO candidates.

Given the importance of iO, it is important to study the security of multilinear maps even in the setting that lacks a security proof under well-studied assumptions. In order to break free from the cycle above, our aim is to develop a rigorous and formal justification for security, despite the lack of "provable" security.

Recent works have shown how to break the attack-fix-repeat cycle for GGH13 [21] and CLT13 [22] multilinear maps by devising abstract "zeroizing" models that

[1] The lockable obfuscation constructions in [8] and [9] use ideas from prior work of Goyal, Koppula, and Waters [12] which introduced techniques for using GGH15 encodings to encrypt branching programs.

capture and generalize all known zeroizing attack strategies on the maps. These works formally prove security of applications in these models, demonstrating in a rigorous sense that the analyzed schemes are resistant to known zeroizing attacks. Since these works, all subsequent classical polynomial-time attacks have fit the proposed models, demonstrating that these models may reasonably reflect the security of the maps.

Our goal is to extend these works to the GGH15 setting, devising a model that captures and generalizes all known zeroizing attack strategies. For GGH15, however, there are unique challenges that make this task non-trivial:

- The underlying mathematics of the scheme differs from previous schemes, and the details of the attacks are quite different. As such, any attack model will be different.
- There does not appear to be a single unified GGH15 multilinear map in the literature, but instead many variants — the basic GGH15 map, a version with safeguards, a version with commutative plaintexts, etc. Moreover, many applications do not conform to the multilinear map interface, and are instead described directly on the GGH15 implementation. The many variants of GGH15 and applications are accompanied by similarly varied settings for the attacks.
- Additionally, there are some functional limitations of GGH15: plaintexts are required to be "short", by default plaintexts do not commute, and the level structure derives from graphs instead of sets. These present challenges in applying the standard multilinear map tools (such as Kilian randomizing branching programs, straddling sets, etc) to the GGH15 setting. This breaks many of the analysis techniques that have been applied to other multilinear map candidates, and has also led to some ad hoc proposals, such as using diagonal matrices for the plaintexts, multiplying by random scalars to create levels, or Kilian randomizing using special types of matrices.

Therefore, our goal will be to:

Develop an abstract zeroizing attack model that captures all known zeroizing attacks on all variants of GGH15, and develop new techniques for proving security in this model.

Our Results. In this work we devise an abstract attack model that applies to all existing variants of GGH15 and applications built on top of GGH15. We demonstrate that our attack model captures and generalizes all zeroizing attacks.

We then describe a new variant of GGH15, based on several prior works in the area, which we can prove strong security statements about in our model. Our new scheme is flexible enough to support a simple obfuscation scheme which we can prove secure in our model. The result is a scheme that is provably resistant to zeroizing attacks. Before giving our results, we start with a very brief overview of the GGH15 maps and known attacks

1.2 The GGH15 Multilinear Map

GGH15 is a "graph-induced" multilinear map, which departs somewhat from the usual multilinear map notions. Here, we have a connected directed acyclic graph $G = (V, E)$ of d nodes with a single source (labeled 1) and a single sink (labeled d). A "level" is a pair of vertices (u, v) for which there is a path from u to v; we will denote such levels by $u \rightsquigarrow v$ (different paths between u, v will be considered the same level). Plaintexts $\mathbf{S}$ are encoded relative to levels $u \rightsquigarrow v$, and we denote such an encoding as $[\mathbf{S}]_{u \rightsquigarrow v}$.

Given a handful of encodings, the following operations can be performed:

- **Addition:** Two encodings $[\mathbf{S}_0]_{u \rightsquigarrow v}, [\mathbf{S}_1]_{u \rightsquigarrow v}$ relative to the same pair of vertices can be added, obtaining the encoding of the sum $[\mathbf{S}_0 + \mathbf{S}_1]_{u \rightsquigarrow v}$(relative to the same pair of vertices).
- **Multiplication:** Two encodings $[\mathbf{S}_0]_{u \rightsquigarrow v}$, $[\mathbf{S}_1]_{v \rightsquigarrow w}$ whose nodes form a path $u \rightsquigarrow v \rightsquigarrow w$ can be multiplied, obtaining an encoding $[\mathbf{S}_0 \cdot \mathbf{S}_1]_{u \rightsquigarrow w}$ of the product at the level corresponding to concatenating the paths.
- **Zero Testing:** Given an encoding $[\mathbf{S}]_{1 \rightsquigarrow d}$ between the unique source and sink, we can test whether or not $\mathbf{S}$ is equal to 0.

In GGH15, the "plaintexts" are also matrices, rather than scalars, meaning the multiplications above are non-commutative. Moreover, in GGH15, the plaintext matrices are required to be "short".

GGH15 works as follows. Associated to each node u is a matrix $\mathbf{A}_u$. An encoding of $\mathbf{S}$ at level $u \rightsquigarrow v$ is a matrix $\mathbf{D}$ that satisfies $\mathbf{A}_u\mathbf{D} = \mathbf{S}\mathbf{A}_v + \mathbf{E} \bmod q$ where both $\mathbf{D}$ and $\mathbf{E}$ are "short". This encoding is generated using a lattice trapdoor.

Addition is straightforward to verify. For multiplication, suppose $\mathbf{A}_u\mathbf{D}_0 = \mathbf{S}_0\mathbf{A}_v + \mathbf{E}_0 \bmod q$ and $\mathbf{A}_v\mathbf{D}_1 = \mathbf{S}_1\mathbf{A}_w + \mathbf{E}_1 \bmod q$. Then $\mathbf{A}_u\mathbf{D}_0\mathbf{D}_1 = \mathbf{S}_0\mathbf{S}_1\mathbf{A}_w + \mathbf{E}_0\mathbf{D}_1 + \mathbf{S}_0\mathbf{E}_1 \bmod q$.

Since $\mathbf{S}_b$, $\mathbf{D}_b$ and $\mathbf{E}_b$ are short, we can define $\mathbf{E}_2 = \mathbf{E}_0\mathbf{D}_1 + \mathbf{S}_0\mathbf{E}_1$, which is also short, and we see that $\mathbf{D}_0\mathbf{D}_1$ is an encoding of $\mathbf{S}_0\mathbf{S}_1$ relative to the path $u \rightsquigarrow w$.

For zero-testing, we note that if we have an encoding $\mathbf{D}$ of $\mathbf{S}$ relative to $1 \rightsquigarrow d$ and we compute $\mathbf{A}_1\mathbf{D} \bmod q = \mathbf{S}\mathbf{A}_d + \mathbf{E} \bmod q$, the resulting matrix will be "short" relative to q if $\mathbf{S} = 0$, and otherwise, we would expect the result to be large relative to q.

1.3 Zeroizing Attacks on GGH15

As with all current multilinear map candidates, GGH15 is vulnerable to "zeroizing" attacks. These attacks leverage the fact that any time a zero-test actually detects 0, the procedure also produces an equation that holds over the integers.

For GGH15, notice that zero-testing computes $\mathbf{A}_1\mathbf{D} \bmod q = \mathbf{S}\mathbf{A}_d + \mathbf{E} \bmod q$. If $\mathbf{S} = 0$, the result is just $\mathbf{E} \bmod q$, which equals $\mathbf{E}$ since $\mathbf{E}$ is guaranteed to be short relative to q. But recall from the GGH15 description that if $\mathbf{D}$ is the result of several multilinear map operations, $\mathbf{E}$ depends on not just the error

terms of the original encodings, but also on the plaintext values $\mathbf{S}$. Therefore, any successful zero-test will give an equation depending on the original plaintext values, and this equation holds over the integers. These equations can then potentially be manipulated to learn non-trivial information about the underlying plaintexts. This is the heart of all known zeroizing attacks on GGH15.

More abstractly, suppose that c plaintext matrices $\mathbf{S}_1, \ldots, \mathbf{S}_c$ are encoded relative to various edges, producing the corresponding encoding matrices $\mathbf{D}_1, \ldots, \mathbf{D}_c$. In all known zeroizing attacks, the adversary adds and multiplies the matrices $\{\mathbf{D}_i\}_i$ honestly (respecting the edge-constraints of the graph) to produce top-level encodings of zero.[2] Let $p_u(\{\mathbf{D}_i\}_i)$ denote the u-th top-level encoding of zero the adversary constructs. Each top-level zero $p_u(\{\mathbf{D}_i\}_i)$ is then zero-tested by multiplying on the left by $\mathbf{A}_1$, successfully obtaining a low-norm matrix of *zero-test results*, which we denote as T_u (in some constructions, T_u is simply a scalar). The current attacks all build a new matrix $\mathbf{W}$ whose entries are plucked from the various T_u matrices (or T_u itself in the case of a scalar). From this point, the known attacks differ in strategy from each other. But at a high level, all of them extract some piece of information from $\mathbf{W}$, such as its kernel or its rank, and use this information to recover non-trivial information about the hidden plaintext matrices $\{\mathbf{S}_i\}_i$.

1.4 Our Zeroizing Model for GGH15

We make the following observation: all known attacks that recover information about the plaintexts $\{\mathbf{S}_i\}_i$ from the $\{T_u\}_u$ set up an *algebraic relation* between the two (we will often refer to this relation as a polynomial). More precisely, this means that implicit in all successful zeroizing attacks on GGH15, there is a non-trivial bounded-degree polynomial Q such that

$$Q(\{T_u\}_u, \{S_{i,j,k}\}_{i,j,k}) = 0$$

holds over the integers, where $S_{i,j,k}$ denotes the (j,k)-th entry of matrix $\mathbf{S}_i$. In known attacks, this Q depends on the matrix $\mathbf{W}$ in some way; however, anticipating potential new avenues for attack, we consider a much more general attack format which assumes as little as possible about the structure of the attacks. Hence, our general condition makes no reference to a matrix $\mathbf{W}$.

While this condition seems simple, it is not a priori obvious that any of the GGH15 zeroizing attacks actually produce such a Q. In theory, an adversary might recover information about the plaintext matrix entries $\{S_{i,j,k}\}_{i,j,k}$ through *any* efficient algorithm taking $\{T_u\}_u$ as input. We certainly cannot hope to re-express any poly-time algorithm as a polynomial over its inputs and outputs. However, we are able to show that all known attacks can be recast as procedures that uncover a Q polynomial.

[2] Technically, the Coron et al. attack on key exchange does not compute top-level encodings of zero, but encodings of the same matrix relative to different source-to-sink paths [19]. However, by connecting a master source node to the original source nodes, we can assume that all GGH15 graphs have a single source. In this case, the Coron et al. attack indeed computes top-level encodings of zero.

Example: The CLLT16 Attack. In Coron et al. [19], the first step of the attack is to construct the matrix $\mathbf{W}$ as above, and then compute a vector $\mathbf{v}$ in the left kernel of $\mathbf{W}$. They show, using the algebraic structure of GGH15, that such a $\mathbf{v}$ in fact gives a relation amongst the plaintext elements only (no error terms). In particular, there is a vector $\mathbf{x}$ of fixed polynomials in the underlying plaintext elements such that $\mathbf{v}$ is orthogonal to $\mathbf{x}$. The attack then proceeds to use this relation amongst the plaintexts to break the scheme.

We observe that an equivalent view of their analysis is that $\mathbf{x}$ is in the column span of $\mathbf{W}$. This means that if we append the column vector $\mathbf{x}$ to $\mathbf{W}$, the rank will be unchanged. Suppose for the moment that $\mathbf{W}$ itself is full rank, and that it is one column shy of being square. Then we can capture the fact that the rank does not increase with a simple algebraic relation: the determinant of $[\ \mathbf{W} \mid \mathbf{x}\ ]$ equals 0. Therefore, in this restricted setting where $\mathbf{W}$ is full rank and almost square, we see that the CLLT16 attack implicitly contains a polynomial Q as desired.

In the actual attack, $\mathbf{W}$ may not be full rank, meaning the determinant may trivially be 0 no matter what $\mathbf{x}$ is; this means Q does not give us a useful relation over the plaintexts. Moreover, $[\ \mathbf{W} \mid \mathbf{x}\ ]$ may not be square, so the determinant may not be defined. With a bit more effort, we can see that a polynomial Q is nonetheless implicit in the attack for general $\mathbf{W}$. Basically, if we knew the rank r of $\mathbf{W}$, we could choose a "random" matrix $\mathbf{R}$ with $r+1$ rows, and a "random" matrix $\mathbf{S}$ with $r+1$ columns. If we compute $\mathbf{R} \cdot [\ \mathbf{W} \mid \mathbf{x}\ ] \cdot \mathbf{S}$, we will obtain an $(r+1) \times (r+1)$ matrix whose rank is (with high probability) identical to the rank of $[\ \mathbf{W} \mid \mathbf{x}\ ]$. Now we can take the determinant of $\mathbf{R} \cdot [\ \mathbf{W} \mid \mathbf{x}\ ] \cdot \mathbf{S}$ to be our algebraic relation. In practice, we do not know r, but we can guess it correctly with non-negligible probability since r is polynomially bounded.

The GGH15 Zeroizing Model. With our observations above in hand, we can define a new zeroizing model for GGH15. Roughly, the model allows the attacker to perform multilinear map operations as explicitly allowed by the multilinear map interface (i.e. following edge constraints). Then, after performing a zero-test, if the encoding actually contained a zero, the adversary obtains a handle to the elements produced by zero-testing (the $\mathbf{E}$ matrix in the discussion above, but potentially a different quantity for different GGH15 variants). Next, the adversary tries to construct an algebraic relation Q between the zero-test results and the original plaintexts. The only restrictions we place on Q are that it must be computable by an efficient algebraic circuit, and that it must have degree that is not too large (e.g. sub-exponential). These restrictions are very conservative, as the known attacks are quite low degree and very efficiently computable.

In the full version, we also discuss how to relax the model even further in two different ways. In one, we allow the adversary to zero-test arbitrary (degree-bounded) polynomials over the encodings, which may not necessarily obey edge restrictions. In the other relaxed model (which is incomparable to the first relaxation), we allow the adversary to zero-test polynomials over handles to *elements* of encodings rather than over handles to the full matrices, as long as the polynomials still follow the edge constraints.

1.5 A New GGH15 Variant

For our next result, we describe a new GGH15 variant. Our goal with this variant is to add safeguards — some of which have been proposed in the literature — in a rigorous way that allows us to formally analyze the effectiveness of these safeguards. Our modifications to GGH15 are as follows:

Tensored Plaintexts. First, we will modify plaintexts as suggested in Chen et al. [10]. Plaintexts will still be matrices $\mathbf{M}$. However, before encoding, we will manipulate $\mathbf{M}$ as follows. First, we will tensor $\mathbf{M}$ with a random matrix $\mathbf{S}$. Then we will also append $\mathbf{S}$ as a block diagonal, obtaining the matrix

$$\mathbf{S}' = \begin{bmatrix} \mathbf{M} \otimes \mathbf{S} & \\ & \mathbf{S} \end{bmatrix}$$

Then we will encode $\mathbf{S}'$ as in plain GGH15. By performing this encoding, we can use the Chen et al. [10] proof to show that direct attacks (those that do not use the broken secret key) are provably impossible, assuming LWE.

Block Diagonal Ciphertexts. Next, after obtaining a plain GGH15 encoding $\mathbf{D}'$ of $\mathbf{S}'$, we append a block diagonal $\mathbf{B}$. Each matrix $\mathbf{B}$ will have "smallish" entries, and will be chosen independently for each encoding. These matrices will multiply independently of the encodings $\mathbf{D}'$. After multiplying to the top level, we will introduce bookend vectors which will combine the products of the $\mathbf{D}'$ and $\mathbf{B}$ matrices together. Since the $\mathbf{B}$ matrices are small, this will not affect zero-testing.

These block diagonals are used to inject sufficient entropy into the encodings, which will be crucial for several parts of our analysis. In particular, these block diagonals will be used to prove that any attack in our zeroizing model will also lead to an attack in a much simpler "GGH15 Annihilation Model", discussed below in Section 1.6. Their role is similar to block diagonals introduced by Garg et al. [21], in the context of GGH13 multilinear maps. However, we note that their role here is somewhat different: our block diagonals are added to the ciphertexts, whereas in [21] they are added to the plaintexts before encoding.

Kilian Randomization. As described so far, the block diagonals $\mathbf{B}$ can simply be stripped off by the adversary, and therefore do not provide any real-world security, despite offering security in our model as discussed in Section 1.6. The reason for this inconsistency is that our model assumes the adversary treats the encoding matrices monolithically, only operating on whole encoding matrices. Such an adversary cannot decompose a block diagonal matrix into its blocks.

We therefore employ the relaxation of our model discussed above, where the adversary can manipulate the individual components of an encoding independently (this is done in the full version [1]). This model captures any adversary's attempts to decompose a block matrix, and potentially much more. In order to maintain security even in this relaxed model, we Kilian-randomize the encodings, which is one of the suggested safeguards from the original GGH15 paper [6].

More precisely, we associate a random matrix $\mathbf{R}_u$ with each node u. Then, when encoding on an edge $u \rightsquigarrow v$, we left-multiply the block diagonal encoding from above by $\mathbf{R}_u^{-1}$, and right multiply by $\mathbf{R}_v$. Note that the inner $\mathbf{R}$ matrices cancel out when multiplying two compatible encodings. Moreover, we include $\mathbf{R}$'s in the bookend vectors to cancel out the outer matrices when zero-testing.

This randomization, intuitively, allows us to bind the matrices $\mathbf{B}$ to $\mathbf{D}'$. We formally prove in our relaxed model that the adversary learns nothing extra if it attempts to manipulate the individual matrix entries; therefore, the adversary might as well just operate monolithically on whole encodings. This allows our analysis from above to go through.

Asymmetric Levels. Finally, we introduce asymmetric levels. In an asymmetric multilinear map, plaintexts are encoded relative to subsets of $\{1, \ldots, \kappa\}$. Encodings relative to the same subset can be added, and encodings relative to disjoint subsets can be multiplied. Encodings relative to the "top" level $\{1, \ldots, \kappa\}$ can be zero-tested.

We do not quite obtain asymmetric multilinear maps from GGH15. Instead, we add the asymmetric level structure on top of the graph structure. That is, there is still a graph on d nodes as well as a set of asymmetric levels. Any plaintext is now encoded relative to a pair $(u \rightsquigarrow v, L)$, where $u \rightsquigarrow v$ is a path in the graph and L is a subset of $\{1, \ldots, \kappa\}$. Encodings can be added as long as both the graph-induced and asymmetric levels are identical, and encodings can be multiplied as long as both sets of levels are compatible. An element can be zero-tested only if it is encoded relative to the source-to-sink path $1 \rightsquigarrow d$, and the "top" asymmetric level $\{1, \ldots, \kappa\}$. Asymmetric levels are useful for creating straddling sets [23] for proving the security of obfuscation.

To achieve this functionality, we use a technique suggested by Halevi [24]. Simply associate a random scalar to each asymmetric level, and divide an encoding by the corresponding subset of level scalars. We choose the level scalars so that they cancel out if and only if they are multiplied together, corresponding to a "top"-level encoding.

We note that it is possible for an adversary to combine elements that do not conform to the asymmetric level structure. For example, an adversary can multiply two encodings with the same asymmetric level. The point is that the adversary will not be able to successfully zero-test such an encoding.

However, the ability to combine illegal elements presents some difficulty for our analysis. Namely, the adversary could combine some illegal elements, and then cancel them out later at some point prior to zero-testing. Such a procedure will generate a valid zero-test, despite being composed of illegal operations. This breaks usual security proofs relying on asymmetric levels, which assume the ability to immediately reject any illegal operations. Essentially what we get then is an "arithmetic model" for the asymmetric levels, due to Miles, Sahai, and Weiss [25]. We will therefore use the techniques from their work in order to prove security in our model.

1.6 An Annihilation Model for Our Scheme

Next, we define a GGH15 Annihilation Model which is much simpler than the zeroizing model described above. This model makes it very easy to evaluate whether a set of plaintexts could possibly lead to an attack.

Up until successful zero-tests, this model is similar to the original model described above: the adversary can combine elements as long as they respect the edges in the underlying graph G. One key difference is that encodings are also associated with an asymmetric level structure. For the asymmetric level structure, we work with the arithmetic model, which allows the adversary to combine arbitrary elements, but any zero-test must be on elements which respect the asymmetric level structure (in addition to respecting the graph level structure).

After successful zero-tests, the model changes from above. Instead of trying to compute a polynomial relation Q, the adversary simply tries to compute an annihilating polynomial Q' for the set of zero-test polynomials previously submitted (where each is evaluated over matrices of formal variables). We show that any attack on our scheme in the GGH15 zeroizing model corresponds to an attack in the GGH15 annihilation model, allowing us to focus on proving the security of schemes in the simpler to reason about annihilation model.

1.7 Zeroizing-Proof Obfuscation

We now turn to constructing obfuscation secure against zeroizing attacks. With our new GGH15 construction and models in hand, the construction becomes quite simple. As with the original obfuscator of Garg et al. [3], our obfuscator works on matrix branching programs; such an obfuscator can be "bootstrapped" to a full obfuscator using now-standard techniques (e.g. using FHE as in [3]). Our obfuscator is essentially the obfuscation construction of [26], which in turn is based on [23]. We do have some simplifications, owing to the fact that our multilinear map directly works with matrices.

- We assume the branching program is given as a "dual-input" branching program, following the same restrictions as in [23].[3] Any branching program can be converted into such a dual-input program using simple transformations as described in [23].
- We instantiate our multilinear map with the single path graph G whose length matches the length ℓ of the branching program. We also use the version with asymmetric level structure, using ℓ asymmetric levels.
- We directly encode the branching program matrices. Each matrix is encoded at the asymmetric level corresponding to how it would be encoded in [26]. Its graph-induced level is chosen to be consistent with evaluation order; namely, the branching program matrices in column i are encoded at the i-th edge in G.

[3] Dual-input is necessary to invoke the p-Bounded Speedup Hypothesis for MAX 2-SAT. This arises in the proof of Lemma 7.

We can then easily prove our obfuscator is secure against zeroizing attacks. The following is a sketch of the proof: in our GGH15 annihilation model, following previous analysis of [25], we can show that under the p-Bounded Speedup Hypothesis, the only successful zero-tests the adversary can construct are linear combinations of polynomially many honest branching program evaluations. But then, any annihilation attack gives an annihilating polynomial for branching programs. We then rely on a non-uniform variant of the *Branching Program Un-Annihilatability* Assumption (BPUA) of [21], which conjectures that such annihilating polynomials are computationally intractable. This assumption can be proven true under the very mild assumption that PRFs secure against $\mathsf{P/poly}$ and computable by branching programs exist (in particular, PRFs computable by log-depth circuits suffice).[4]

1.8 Concurrent Work: A Weak Model for CLT13

Ma and Zhandry [28] propose a weak multilinear map model for the CLT13 multilinear maps [5], which they show captures all known zeroizing attacks on CLT13. They prove that an obfuscation scheme of Badrinarayanan, Miles, Sahai, and Zhandry [26] as well as an order revealing encryption construction of Boneh et al. [27] are secure against zeroizing attacks when instantiated with CLT13. They also give a polynomial-degree asymmetric multilinear map "fix" which they prove secure in their model under a new assumption they call the "Vector-Input Branching Program Un-Annihilatability Assumption," a strengthening of the BPUA Assumption.

Due to the substantial differences between the CLT13 and GGH15 multilinear maps, the techniques of Ma and Zhandry do not apply to the GGH15 setting. Most notably, their model captures an attacker's ability to perform a step that leads to factoring the CLT13 modulus. There is no composite modulus in the GGH15 scheme and thus the zeroizing attacks we consider are quite different.

2 Preliminaries

2.1 Notation

Throughout this paper we use capital bold letters to denote a matrix $\mathbf{M}$. Lowercase bold letters denote vectors $\mathbf{v}$. Occasionally, we will use $diag(\mathbf{M}_1, \ldots, \mathbf{M}_k)$ to denote a matrix with block diagonals $\mathbf{M}_1, \ldots, \mathbf{M}_k$. We will often need to distinguish between values and formal variables. For example, in a situation where the variable $x = 2$, it can be difficult to tell when x represents a formal variable or when it represents the number 2. Thus, whenever we want x to denote a formal variable, we explicitly write it as $\widehat{x}$. When an expression over formal variables is identically 0, we write $\equiv$ (or $\not\equiv$ if it is not). Finally, we identify the ring $\mathbb{Z}_q$ with elements $[-q/2, q/2)$.

[4] Using similar arguments, we can adapt the order-revealing encryption (ORE) construction of [27] to our scheme, and prove security under BPUA, analogous to constructing ORE from GGH13 as in [21].

2.2 Background on Lattices

Here, we give a very brief background on lattices. A lattice Λ of dimension n is a discrete additive subgroup of $\mathbb{R}^n$ that is generated by n basis vectors denoted as $\{\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^n\}$. Specifically, we have $\Lambda = \{\sum_{i\in[n]} x_i \cdot \mathbf{b}_i\}$ for integer x_i's. We then have the following useful definitions and lemmas.

Definition 1 (Discrete Gaussian on Lattices). *First, define the Gaussian function on $\mathbb{R}^n$ with center $\mathbf{c} \in \mathbb{R}^n$ and width $\sigma > 0$ as*

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

Then, the discrete Gaussian distribution over an n-dimensional Λ with center $\mathbf{c} \in \mathbb{R}^n$ and width σ is defined as

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

Note that we omit the subscript $\mathbf{c}$ when it is $\mathbf{0}$.

Definition 2 (Decisional Learning with Errors (LWE) [7]). *For $n, m \in \mathbb{N}$ and modulus $q \geq 2$, distributions for secret vectors, public matrices, and error vectors $\theta, \pi, \chi \subseteq \mathbb{Z}_q$, an LWE sample is defined as $(\mathbf{A}, \mathbf{s}^T\mathbf{A} + \mathbf{e}^T \mod q)$ with $\mathbf{s}, \mathbf{A}, \mathbf{e}$ sampled as $\mathbf{s} \leftarrow \theta^n$, $\mathbf{A} \leftarrow \pi^{m\times n}$, and $\mathbf{e} \leftarrow \chi^m$.*

An algorithm is said to solve $\mathsf{LWE}_{n,m,q,\theta,\pi,\chi}$ if it is able to distinguish the LWE sample from one that is uniformly sampled from $\pi^{m\times n} \times U(\mathbb{Z}_q^{m\times 1})$ with probability non-negligibly greater than $1/2$.

Lemma 1 (Hardness of LWE [7]). *Given $n \in \mathbb{N}$, for any $m = \mathsf{poly}(n), q \leq 2^{\mathsf{poly}(n)}$, let $\theta = \pi = U(\mathbb{Z}_q), \chi = D_{\mathbb{Z},\sigma}$ where $\sigma \geq 2\sqrt{n}$. If there exists an efficient (possible quantum) algorithm that breaks $\mathsf{LWE}_{n,m,q,\theta,\pi,\chi}$, then there exists an efficient (possible quantum) algorithm for approximating SIVP and GAPSVP in the ℓ_2 norm, in the worst case, to within $\tilde{O}(nq/\sigma)$ factors.*

Lemma 2 (LWE with Small Public Matrices [29]). *Given n, m, q, σ chosen as in Lemma 1, $\mathsf{LWE}_{n',m,q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$ is as hard as $\mathsf{LWE}_{n,m,q,U(\mathbb{Z}_q),U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma}}$ for $n' \geq 2n\log q$.*

Lemma 3 (Trapdoor Sampling [30]). *There exists a PPT algorithm called $\mathsf{TrapSam}(1^n, 1^m, q)$ that, given any integers $n \geq 1$, prime $q \geq 2$, and sufficiently large $m = O(n\log q)$, outputs $(\mathbf{A}, \tau)$ where $\mathbf{A}$ is statistically close to uniform over $\mathbb{Z}_q^{n\times m}$, and τ is a trapdoor for $\mathbf{A}$. Furthermore, there is another PPT algorithm $\mathsf{SampleD}(\mathbf{A}, \tau, \mathbf{y}, \sigma)$ that outputs a sample of vector $\mathbf{d}$ from $D_{\mathbb{Z}^m,\sigma}$ conditioned on $\mathbf{Ad} = \mathbf{y}$. For sufficiently large $\sigma = O(\sqrt{n\log q})$, with all but negligible probability, we have*

$$\{\mathbf{A}, \mathbf{d}, \mathbf{y} : \mathbf{y} \leftarrow U(\mathbb{Z}_q^n), \mathbf{d} \leftarrow \mathsf{SampleD}(\mathbf{A}, \tau, \mathbf{y}, \sigma)\}$$
$$\approx_s$$
$$\{\mathbf{A}, \mathbf{d}, \mathbf{y} : \mathbf{d} \leftarrow D_{\mathbb{Z}^m,\sigma}, \mathbf{y} = \mathbf{Ad}\}.$$

2.3 Matrix Branching Programs

We introduce dual-input matrix branching programs of the type considered in [31] but with one minor modification. Formally, a dual-input matrix branching program BP of length h, width w, and input length ℓ consists of an input selection function $\mathsf{inp} : [h] \to [\ell] \times [\ell]$ and $4h$ matrices

$$\{\mathbf{M}_{i,b_1,b_2} \in \{0,1\}^{w \times w}\}_{i \in [h]; b_1, b_2 \in \{0,1\}}.$$

BP is evaluated on input $x \in \{0,1\}^\ell$ by checking whether or not

$$\prod_{i \in [h]} \mathbf{M}_{i,x(i)} = 0^{w \times w}$$

where $x(i) := (x_{\mathsf{inp}(i)_1}, x_{\mathsf{inp}(i)_2})$. Note that the definition from [31] includes right and left bookend vectors that are multiplied on either side of the branching program product resulting in a scalar that is either zero or non-zero. We can simply turn each bookend into a matrix by repetition of rows/columns in order to recover the functionality described above. As noted in [31], branching programs of this type can be constructed from any NC^1 circuit with $h = \mathsf{poly}(n)$ and $w = 5$ by Barrington's theorem [32].

2.4 Straddling Sets

Our obfuscator uses the notion of straddling sets in order to enforce input consistency. Please refer to Barak et al. [23] for a simple construction.

Definition 3 (Straddling Set System). *A straddling set system with n entries is a universe set $\mathbb{U}$ and a collection of subsets $\mathbb{S} = \{S_{i,b} \subseteq \mathbb{U}\}_{i \in [n], b \in \{0,1\}}$ such that*

- $\bigcup_{i \in [n]} S_{i,0} = \bigcup_{i \in [n]} S_{i,1} = \mathbb{U}$
- *For any distinct $C, D \subseteq \mathbb{S}$ such that $\bigcup_{S \in C} S = \bigcup_{S \in D} S$, there exists $b \in \{0,1\}$ such that $C = \{S_{i,b}\}_{i \in [n]}$ and $C = \{S_{i,1-b}\}_{i \in [n]}$*

3 GGH15 Zeroizing Model

3.1 Graph-Induced Ideal Model

We discuss the syntax of graph-induced graded encoding schemes and describe an ideal model (also known as a generic multilinear map model) for the graph-induced setting. Note that this is completely analogous to the ideal model for symmetric/asymmetric multilinear maps, which itself is an extension of the generic group model to the multilinear map setting [4, 3].

We consider directed acyclic graphs (DAGs) $G = (V, E)$ where $|V| = d$. We assume the graph has a single source and a single sink. We label the vertices from

1 to d according to some fixed topological ordering, so that all edges/paths in the graph can be written as $j \rightsquigarrow k$ where $j, k \in [d], j < k$. (Note that the precise distinction between paths and edges in graph-induced maps is not important, since the intermediate nodes on a path do not matter).

Formally, the graph-induced ideal model is instantiated with a DAG $G = (V, E)$, a plaintext ring R, and a set of plaintexts $\{M_i, u_i \rightsquigarrow v_i\}_i$. The plaintexts are indexed by i, and plaintext M_i comes with an associated path $u_i \rightsquigarrow v_i$, where $u_i, v_i \in [d], u_i < v_i$.

We describe the model as an interaction between an oracle $\mathcal{M}$ (the "model") and a user $\mathcal{A}$ (the "adversary").

- *Instance Generation.* The model $\mathcal{M}$ is instantiated with the graph G, plaintext ring R and the set $\{M_i, u_i \rightsquigarrow v_i\}_i$. For each i, the model $\mathcal{M}$ generates a handle $\widehat{C}_i$, stores a pointer from $\widehat{C}_i$ to M_i, and releases $(\widehat{C}_i, u_i \rightsquigarrow v_i)$ publicly.

$\mathcal{A}$ can only interact with the handles $\widehat{C}_i$, which in the ideal setting leak no information about M_i. The model provides the following interfaces for $\mathcal{A}$:

- *Addition.* Addition on two handles $\widehat{C}_i$, $\widehat{C}_j$ is permitted only if their corresponding paths $u_i \rightsquigarrow v_i$, $u_j \rightsquigarrow v_j$ are the same. The model $\mathcal{M}$ looks up the corresponding plaintexts M_i, M_j, and returns a newly generated handle $\widehat{C}_k$ to the sum $M_i + M_j$, along with the path $u_i \rightsquigarrow v_i$.
- *Multiplication.* Multiplication on two handles $\widehat{C}_i$, $\widehat{C}_j$ is permitted only if the path $u_i \rightsquigarrow v_i$ ends where path $u_j \rightsquigarrow v_j$ begins ($v_i = u_j$). The model $\mathcal{M}$ looks up the corresponding plaintexts M_i, M_j, and returns a newly generated handle $\widehat{C}_k$ to the product $M_i \cdot M_j$, along with the combined path $u_i \rightsquigarrow v_j$.
- *Zero-Test.* $\mathcal{A}$ can request a zero-test on a handle $\widehat{C}$. $\mathcal{M}$ responds with "zero" if the corresponding plaintext is 0, and the corresponding path is the source-to-sink path. Otherwise, the result is "not zero."

Implicit in this model is the assumption that the adversary cannot learn anything beyond what the interfaces explicitly allow. In particular, it can only learn the bits returned by zero-testing honestly generated source-to-sink encodings, and nothing more.

Zero-Test Circuits Observe that addition, multiplication, and zero-testing can be handled in a single interface. Here, $\mathcal{A}$ simply submits an arithmetic circuit p that computes a polynomial over the handles $\{\widehat{C}_i\}_i$. Any handle that results in a successful zero-test in the above model can be represented as a polynomial-size circuit over $\{\widehat{C}_i\}_i$ where each arithmetic gate respects the addition and multiplication restrictions enforced by the graph structure.

However, we can relax the restriction on the arithmetic circuit so that the individual gates may not necessarily respect the graph constraints, but the resulting polynomial still computes a valid source-to-sink encoding (for example, if terms that violate graph constraints cancel out in the final evaluation). Looking ahead to our GGH15 Zeroizing Model, we will require this relaxed constraint on arithmetic circuits, which only makes the model more conservative.

3.2 GGH15 Variants

There are a number of GGH15 variants in the literature that modify the original GGH15 construction at a number of key points. We identify several points in which the various schemes differ, and establish standard notation before introducing our model.

Pre-Processing. In the original GGH15 construction [6], an encoding of a plaintext matrix $\mathbf{M}$ at path $u \rightsquigarrow v$ is the matrix $\mathbf{D}$ satisfying $\mathbf{A}_u \cdot \mathbf{D} = \mathbf{M} \cdot \mathbf{A}_v + \mathbf{E}$.

A number of works have proposed performing additional pre-processing to $\mathbf{M}$ before sampling the matrix $\mathbf{D}$. For example, the $\gamma_{\otimes\text{diag}}$-GGH15 encodings of Chen et al. [10] encode a plaintext matrix $\mathbf{M}$ by first sampling a random $\mathbf{P}$ (in the notation of [10], this is the $\mathbf{S}_{i,b}$ matrix) and constructing the matrix $\mathit{diag}(\mathbf{M} \otimes \mathbf{P}, \mathbf{P})$ where $\otimes$ denotes the tensor product (Kronecker product).

Then the encoding $\mathbf{D}$ is the matrix satisfying

$$\mathbf{A}_u \cdot \mathbf{D} = \begin{bmatrix} \mathbf{M} \otimes \mathbf{P} & \\ & \mathbf{P} \end{bmatrix} \cdot \mathbf{A}_v + \mathbf{E}.$$

As other GGH15 variants perform different pre-processing steps on the initial plaintext $\mathbf{M}$, we denote the result of pre-processing as $\mathbf{S}$. If there is no pre-processing step, then $\mathbf{S} = \mathbf{M}$. In the example above $\mathbf{S} = \mathit{diag}(\mathbf{M} \otimes \mathbf{P}, \mathbf{P})$.[5] The encoding is then computed as $\mathbf{A}_u \cdot \mathbf{D} = \mathbf{S} \cdot \mathbf{A}_v + \mathbf{E}$.

Post-Encoding. The original GGH15 paper [6] as well as Halevi [24] discuss various steps intended to safeguard the scheme against attacks (sometimes called "GGH15 with safeguards"). These steps essentially perform operations on the matrix $\mathbf{D}$ generated from the standard GGH15 encoding procedure to produce a "final" encoding $\mathbf{C}$. We will adopt this notation, and set $\mathbf{C}$ to be the result of the overall encoding process. If there is no post-encoding step, then $\mathbf{C} = \mathbf{D}$.

Zero-Testing. In the original GGH15 construction, zero-testing a source-to-sink encoding is done by computing a matrix from the public parameters and the encodings $\mathbf{C}$, and testing if this matrix is small. Ideally, only the bit of information (whether or not the result is small) is useful to the adversary. Of course, the zeroizing attacks on GGH15 show that this assumption is false, and that the actual matrix resulting from the zero-test can provide useful information to the adversary [19, 20, 10]. This matrix will be referred to as the "result" of zero-testing. To avoid confusion, the 0/1 bit learned from the zero-test will be referred to as a bit rather than the result.

In certain GGH15 variants, the result of zero-testing is not a matrix. For example in "GGH15 with safeguards" [6, 24], the result of zero-testing is a scalar. We will use the letter T to generically denote the result of zero-testing (noting that T may represent a matrix depending on the scheme, even though it might not be written in bold).

[5] Essentially, $\mathbf{S}$ is the result of the γ functions in the notation of [10]. However, the $\mathbf{S}$ notation is more natural for our setting, especially when referring to entries of these matrices.

GGH15 Algorithms. Unlike the graph-induced ideal model, our GGH15 Zeroizing Model is defined with respect to a specific GGH15 scheme/variant in mind. For example, in the ideal setting, a zero-test is successful if and only if the product of the plaintexts is zero. In our GGH15 Zeroizing Model, the model explicitly maintains encodings corresponding to each plaintext, and whether a zero-test is successful is determined by performing computations on the encodings and public parameters corresponding to an actual GGH15 variant.

To specify our model, we let the scheme be denoted by G. For example, G may be the original GGH15 construction [6], the "GGH15 with safeguards" [24], etc. To be a valid GGH15 scheme, we require G to have the following algorithms (in the literature, $\mathsf{PreProcess}$ is usually implicit):

- $\mathsf{G.KeyGen}(1^\lambda, G, R, \mathsf{aux})$: Takes the security parameter, a description of a graph G with source 1 and sink d, a ring R, and potential auxiliary information aux, and produces public parameters pp and secret parameters sp.
- $\mathsf{G.PreProcess}(\mathsf{sp}, \mathbf{M})$: Converts the input plaintext $\mathbf{M}$ into a pre-encoding $\mathbf{S}$. For many schemes (including the original GGH15 construction), $\mathbf{S} = \mathbf{M}$.
- $\mathsf{G.Enc}(\mathsf{sp}, \mathbf{S}, u_i \rightsquigarrow v_i)$: Encodes $\mathbf{S}$ on the path $u_i \rightsquigarrow v_i$.
- $\mathsf{G.Add}(\mathsf{pp}, \mathbf{C}_1, \mathbf{C}_2)$: Takes an encoding $\mathbf{C}_1$ of $\mathbf{M}_1$ at path $u_1 \rightsquigarrow v_1$ and an encoding $\mathbf{C}_2$ of $\mathbf{M}_2$ at path $u_2 \rightsquigarrow v_2$. If $u_1 = u_2$ and $v_1 = v_2$, this produces an encoding $\mathbf{C}_3$ of $\mathbf{M}_1 + \mathbf{M}_2$ at path $u_1 \rightsquigarrow v_1$.
- $\mathsf{G.Mult}(\mathsf{pp}, \mathbf{C}_1, \mathbf{C}_2)$: Takes an encoding $\mathbf{C}_1$ of $\mathbf{M}_1$ at path $u_1 \rightsquigarrow v_1$ and an encoding $\mathbf{C}_2$ of $\mathbf{M}_2$ at path $u_2 \rightsquigarrow v_2$. If $v_1 = u_2$, this produces an encoding $\mathbf{C}_3$ of $\mathbf{M}_1 \cdot \mathbf{M}_2$ at path $u_1 \rightsquigarrow v_2$.
- $\mathsf{G.ZeroTest}(\mathsf{pp}, \mathbf{C})$: Takes an encoding $\mathbf{C}$, computes a result T, and returns (T, b). If $\mathbf{C}$ is an encoding of 0 relative to path $1 \rightsquigarrow d$, then T is "small" and $b = 1$ (indicating successful zero-test). Otherwise, $b = 0$ with overwhelming probability.

3.3 GGH15 Zeroizing Model

Initialize Parameters. $\mathcal{M}$ is initialized with a security parameter λ, a graph $G = (V, E)$, a ring R, potential auxiliary information aux, and a graph-induced encoding scheme G. It runs $\mathsf{G.KeyGen}(1^\lambda, G, R, \mathsf{aux})$ to generate the public and secret parameters $(\mathsf{pp}, \mathsf{sp})$, which it stores.

Initialize Elements. $\mathcal{M}$ is given a set of initial plaintext elements $\{\mathbf{M}_i, u_i \rightsquigarrow v_i\}_i$ where each plaintext is indexed by i, and i-th plaintext $\mathbf{M}_i$ is associated with path $u_i \rightsquigarrow v_i$. The model applies a pre-processing procedure to the plaintext (recall in the standard GGH15 construction, this procedure does nothing):

$$\mathbf{S}_i \leftarrow \mathsf{G.PreProcess}(\mathsf{sp}, \mathbf{M}_i).$$

Then it computes the encoding $\mathbf{C}_i$ from the pre-encoding $\mathbf{S}_i$:

$$\mathbf{C}_i \leftarrow \mathsf{G.Enc}(\mathsf{sp}, \mathbf{S}_i, u_i \rightsquigarrow v_i).$$

Each tuple $(\mathbf{S}_i, \mathbf{C}_i, u_i \rightsquigarrow v_i)$ is stored in the *pre-zero-test table*. For each encoding $\mathbf{C}_i$, the model generates a corresponding handle $\widehat{C}_i$ that contains no information about $\mathbf{C}_i$ or $\mathbf{S}_i$. The handle is released, along with the corresponding encoding level $u_i \rightsquigarrow v_i$, and the model internally stores a mapping between the handle $\widehat{C}_i$ and the tuple $(\mathbf{S}_i, \mathbf{C}_i, u_i \rightsquigarrow v_i)$. While the encoding $\mathbf{C}_i$ is a matrix, the adversary is given a single handle $\widehat{C}_i$ to the entire matrix.

Zero-Testing. The adversary generates a polynomial p (represented as a $\mathsf{poly}(\lambda)$-size arithmetic circuit), over the handles $\widehat{C}_i$ and submits it to the model. Note that since the handles correspond to non-commutative encodings, p must be treated as a polynomial over non-commuting variables.

The model verifies that p computes an edge-respecting polynomial, meaning that each monomial is a product of encodings corresponding to a source-to-sink path. If p is not edge-respecting, the model returns $\bot$. If p is edge-respecting, the model $\mathcal{M}$ evaluates p on the encodings $\mathbf{C}_i$, producing a matrix $p(\{\mathbf{C}_i\}_i)$ that corresponds to a valid source-to-sink encoding (or a linear combination of source-to-sink encodings). Finally, $\mathcal{M}$ zero-tests $p(\{\mathbf{C}_i\}_i)$, obtaining $(T, b) \leftarrow \mathsf{G.ZeroTest}(\mathsf{pp}, p(\{\mathbf{C}_i\}_i))$. If the zero-test is successful ($b = 1$), the model stores the value T (possibly a matrix, vector, or scalar) and generates a handle $\widehat{T}_\ell$ to each element of T. Otherwise, the model returns $\bot$.

We index the successful zero-tests by the letter u, so T_u will denote the result of the u-th successful zero-test, $\widehat{T}_u$ will be the corresponding handles, and p_u will be the polynomial submitted for the u-th successful zero-test.[6]

Post-Zero-Test. In the post-zero-test stage, the adversary submits a polynomial Q of degree at most $2^{o(\lambda)}$ over the handles $\{\widehat{T}_u\}_u$ and pre-encoding elements $\{\widehat{S}_{i,j,k}\}_{i,j,k}$ where $\widehat{S}_{i,j,k}$ is a handle to the (j,k)-th entry of the i-th pre-encoding matrix $\mathbf{S}_i$. For the sake of readability, we will frequently drop the outer subscripts and denote these sets as $\{\widehat{T}_u\}$ and $\{\widehat{S}_{i,j,k}\}$. The model $\mathcal{M}$ checks the following:

1. $Q(\{T_u\}, \{S_{i,j,k}\}) = 0$
2. $Q(\{T_u\}, \{\widehat{S}_{i,j,k}\}) \not\equiv 0$
3. $Q(\{\widehat{T}_u\}, \{S_{i,j,k}\}) \not\equiv 0$

If all three checks pass, the model returns "Win", and otherwise it returns $\bot$. In Section 3.4, we explain how we derive these conditions, and in Section 3.5 we justify how these conditions capture the known attacks. We note that $\mathcal{A}$ is free to submit as many polynomials Q as it wants as long as it remains polynomial time. If any such Q causes $\mathcal{M}$ to return "Win" then the adversary is successful.

Note that in reality, a zeroizing attack that succeeds with non-negligible probability is indeed considered successful. Thus, we will allow the adversary to be possibly randomized, and we define a successful adversary to be one that can obtain a "Win" with non-negligible probability (over the randomness of the model and the adversary).

[6] Although we denote each zero-test result as T_u, an adversary is not required to use T_u monolithically. For example, an adversary can extract a single entry of T_u in the case when T_u are matrices.

3.4 Deriving the Post-Zero-Test Win Condition

All known zeroizing attacks on GGH15 exclusively rely on the results of zero-tests to recover information about the hidden plaintexts [19, 20, 10]. In our model, this can be viewed as using the values $\{T_u\}$ to learn something about the values $\{S_{i,j,k}\}$. Furthermore, we claim that all attacks that do this recover information that can be expressed as an *algebraic relation* (we justify this claim in Section 3.5).

More precisely, underneath all successful zeroizing attacks on GGH15, there is a non-trivial bounded-degree polynomial Q (the algebraic relation) such that

$$Q(\{T_u\}, \{S_{i,j,k}\}) = 0$$

holds over the integers.

This corresponds to the intuition that in a zeroizing attack, the adversary can learn something about the pre-encoding entries $S_{i,j,k}$ by plugging the results of zero-testing $\{T_u\}$ into the above relation. While not every algebraic relation is solvable, we take the conservative route and model any non-trivial relation the adversary can construct as a win.

Now we formalize what it means for Q to be non-trivial. If the adversary can indeed plug in the results of zero-testing to learn something about the $S_{i,j,k}$, then the expression must not be identically zero over the $\widehat{S}_{i,j,k}$ terms (taken as formal variables), when the $\{T_u\}$ values are plugged in. Thus, we have the condition

$$Q(\{T_u\}, \{\widehat{S}_{i,j,k}\}) \not\equiv 0.$$

We also want to ensure that the zeroizing attack uncovers information about the pre-encodings beyond what the adversary can learn honestly. Note that if the adversary obtains a successful zero-test, it learns that some function of the pre-encoding entries $\widehat{S}_{i,j,k}$ evaluates to 0. As a simple example, if the adversary learns from an honest zero-test that matrix $\mathbf{S}_{i'}$ is the 0 matrix, then $\mathbf{S}_{i',j',k'} = 0$ for any choice of j', k'. The formal polynomial $Q = \widehat{S}_{i',j',k'}$ for any j', k' would then satisfy both of the above conditions. However, we should not consider this a successful zeroizing "attack," as it does not use the zero-test results to derive information about the pre-encodings.

To ensure that what the adversary learns about the pre-encodings relies on T_u in a non-trivial way, we enforce a third condition

$$Q(\{\widehat{T}_u\}, \{S_{i,j,k}\}) \not\equiv 0.$$

Roughly, this condition states that the relation is not always satisfied regardless of what the $\{T_u\}$ values are, and thus the attack "uses" the zero-test leakage.

3.5 Algebraic Relations in Known Attacks

We now describe in detail how in the Coron et al. [19] attack (henceforth CLLT16) on multiparty key exchange over GGH15, we can derive an algebraic

relation Q satisfying our three win conditions with non-negligible probability. For the analogous description of the other major zeroizing attacks ([20, 10]), refer to the full version of this work [1], and for a review of the settings of these attacks, refer to the full version or the original papers [19, 20, 10].

Step 1: Compute Top-Level Encodings of Zero. The CLLT16 attack on GGH15 key exchange does not explicitly compute encodings of zero in the original exposition. Instead, the attack computes encodings of the same plaintext on two different source-to-sink paths (starting from different sources), and subtracts the encodings. In our setting we enforce without loss of generality that all graphs must have a single source, which can be generically achieved by connecting a "super" source node to the original source nodes of the graph, and encoding a 1 (or identity matrix) on edges leading into the original sources.

The encodings used in the key exchange are $\mathbf{C}_{i,0}$ for $1 \leq i \leq 3$ (which we introduce to connect the super source node) and $\mathbf{C}_{i,i',l}$ for $1 \leq i, i' \leq 3, 1 \leq l \leq N$ (for some large enough N). Then for $\{\mathbf{C}\} = \{\mathbf{C}_{i,0}\}_{i\in\{1,2,3\}} \cup \{\mathbf{C}_{i,i',l}\}_{i,i'\in\{1,2,3\},l\in[N]}$, the polynomial

$$p_{j,k}(\{\mathbf{C}\}) = \mathbf{C}_{2,0} \cdot \mathbf{C}_{2,1,1} \cdot \mathbf{C}_{2,2,j} \cdot \mathbf{C}_{2,3,k} - \mathbf{C}_{3,0} \cdot \mathbf{C}_{3,1,k} \cdot \mathbf{C}_{3,2,1} \cdot \mathbf{C}_{3,3,j}$$

is an encoding of $s_{3,1} \cdot s_{1,j} \cdot s_{2,k} - s_{2,k} \cdot s_{3,1} \cdot s_{1,j} = 0$ for all choices of $j \in [J], k \in [K]$, where for this attack $J = K = N$ (N is a parameter in the key exchange construction). Recall the key exchange construction uses a GGH15 variant that supports a commutative plaintext space, so this is always an encoding of 0.

Step 2: Zero-Test and Build W Matrix. Zero-test each of these top-level encodings, and let the result of zero-testing $p_{j,k}(\{\mathbf{C}\})$ be $T_{j,k}$. Construct a $J \times K$ matrix $\mathbf{W}$ where the (j,k)-th entry $W_{j,k}$ is derived from $T_{j,k}$. In all current attacks, the matrix $\mathbf{W}$ has the following properties:

- $\mathbf{W}$ factors into $\mathbf{X} \times \mathbf{Y}$ where the rows of $\mathbf{Y}$ are linearly independent over the integers (with high probability).
- There exists a column of $\mathbf{X}$ that is in the column space of a $J \times \eta$ dimensional matrix $\mathbf{M}$, for some η that we specify below for each attack. Each entry of $\mathbf{M}$ is a polynomial over the entries of pre-encoding matrices $\{\mathbf{S}\}$.

In the CLLT16 setting (augmented with our "super" source $\mathcal{S}$), we zero-test by multiplying $\mathbf{A}_{\mathcal{S}}$ with $p_{j,k}(\{C\})$ evaluated over the encodings. This gives a zero-test result $T_{j,k}$ as a vector. Coron et al. observe that the first element of this vector can be written as a dot product $\mathbf{x}_j \cdot \mathbf{y}_k$ where the entries of $\mathbf{x}_j$ depend only on the encodings corresponding to user 1 (and the fixed encodings) and the entries of $\mathbf{y}_k$ depend only on the encodings corresponding to user 2 (and the fixed encodings). Moreover, the first element of $\mathbf{x}_j$ is the pre-encoding $s_{1,j}$. Coron et al. also argue that arranging many column vectors $\mathbf{y}_k$ into a square matrix $\mathbf{Y}$ results in $\mathbf{Y}$ being invertible with high probability. Thus we take $W_{j,k}$ to be the first element of $T_{j,k}$, $\mathbf{X}$ to consist of the row vectors $\mathbf{x}_1, ..., \mathbf{x}_J$, and $\mathbf{M}$ to simply be the column vector $[s_{1,1}\ s_{1,2}\ \cdots\ s_{1,J}]^\top$ (of dimension $J \times \eta$ where $\eta = 1$).

Step 3: Deriving an Algebraic Relation. To show how the CLLT16 attack is captured by our model, we demonstrate that this $\mathbf{W}$ matrix is already sufficient

to come up with a Q satisfying our post-zero-test win condition (with non-negligible probability). For this it suffices to give a polynomial-time procedure (the adversary) that extracts a Q satisfying our win condition.

To win in our model, the adversary will pick the parameter K so that $\mathbf{Y}$ turns out to be square and thus invertible and the parameter $J \geq K + \eta$ (where η is specified in step 2 by the setting we are in). $\mathbf{Y}$ being invertible implies that every column of $\mathbf{X}$ is in the column space of $\mathbf{W}$, so in particular we have a column of $\mathbf{X}$ that is in both the column space of $\mathbf{W}$ and the column space of $\mathbf{M}$. Intuitively, if we are able to combine the columns of $\mathbf{W}$ and $\mathbf{M}$ into a square matrix, we are guaranteed that the determinant of this matrix will be zero. We just have to ensure that the columns from $\mathbf{W}$ and the columns from $\mathbf{M}$ are each linearly independent so that the determinant polynomial is not identically zero when either set of variables is substituted in. The adversary mounts the attack as follows, where the parameter β is taken to be exponential in the security parameter λ that the underlying scheme was initialized with, and $\xleftarrow{\text{U}}$ denotes "drawing uniformly at random."

To start, the adversary forms the matrix $\mathbf{W}$ of handles to honest zero-test results and the matrix $\mathbf{M}$ of pre-encoding handles where $\mathbf{W} \in \mathbb{Z}^{J \times K}$ and $\mathbf{M} \in \mathbb{Z}^{J \times \eta}$. The adversary then guesses the ranks $r_{\mathbf{M}}$ of $\mathbf{M}$ and $r_{\mathbf{W}}$ of $\mathbf{W}$ uniformly at random. The adversary guesses the correct ranks with probability $1/(K\eta)$.

The adversary then draws four random matrices $\mathbf{U}, \mathbf{U}' \xleftarrow{\text{U}} \mathbb{Z}_\beta^{(r_{\mathbf{M}}+r_{\mathbf{W}}) \times J}$, $\mathbf{V} \xleftarrow{\text{U}} \mathbb{Z}_\beta^{\eta \times r_{\mathbf{M}}}$, $\mathbf{V}' \xleftarrow{\text{U}} \mathbb{Z}_\beta^{K \times r_{\mathbf{W}}}$, and constructs

$$\mathbf{M}' = \mathbf{U} \cdot \mathbf{M} \cdot \mathbf{V} \text{ , and } \mathbf{W}' = \mathbf{U}' \cdot \mathbf{W} \cdot \mathbf{V}'.$$

Note that $\mathbf{M}' \in \mathbb{Z}^{(r_{\mathbf{M}}+r_{\mathbf{W}}) \times r_{\mathbf{M}}}$, and $\mathbf{W}' \in \mathbb{Z}^{(r_{\mathbf{M}}+r_{\mathbf{W}}) \times r_{\mathbf{W}}}$. Lastly, the adversary constructs a square $(r_{\mathbf{M}} + r_{\mathbf{W}}) \times (r_{\mathbf{M}} + r_{\mathbf{W}})$ matrix $\mathbf{A} = [\ \mathbf{M}' \ \mid \ \mathbf{W}'\ ]$ by concatenating $\mathbf{M}'$ and $\mathbf{W}'$. Note that the entries of $\mathbf{A}$ are over handles to the zero-test results and the pre-encodings. The adversary takes the determinant polynomial Q of this matrix and submits Q as the post-zero-test polynomial.

Assume the adversary has guessed the two ranks correctly, which happens with non-negligible probability since $K, \eta = \mathsf{poly}(\lambda)$. We now show that Q will satisfy the following three win conditions in our model with non-negligible probability.

1. $Q(\{T_{j,k}\}, \{S_{i,j,k}\}) = 0$
2. $Q(\{T_{j,k}\}, \{\widehat{S}_{i,j,k}\}) \not\equiv 0$
3. $Q(\{\widehat{T}_{j,k}\}, \{S_{i,j,k}\}) \not\equiv 0$

First, $Q(\{T_{j,k}\}, \{S_{i,j,k}\}) = 0$ since we have explicitly introduced a linear dependency among the columns of $\mathbf{A}$. Now we argue that with high probability, $\mathbf{M}'$ has an $r_{\mathbf{M}} \times r_{\mathbf{M}}$ dimensional submatrix of rank $r_{\mathbf{M}}$ which implies that its columns are linearly independent and thus that $Q(\{\widehat{T}_{j,k}\}, \{S_{i,j,k}\}) \not\equiv 0$. The same argument applies to $\mathbf{W}'$ implying that $Q(\{T_{j,k}\}, \{\widehat{S}_{i,j,k}\}) \not\equiv 0$. This follows from an application of the following lemma (with proof in the full version [1]), noting that in our case, β is exponential in λ and the dimensions of $\mathbf{M}$ and $\mathbf{W}$ are polynomial in λ.

Lemma 4. *Suppose an* $\boldsymbol{M} \in \mathbb{Z}_\beta^{n \times m}$ *has rank* r*. Draw uniformly random* $\mathbf{U} \leftarrow \mathbb{Z}_\beta^{r \times n}, \mathbf{V} \leftarrow \mathbb{Z}_\beta^{m \times r}$*. Then* $\mathbf{M}' := \mathbf{U} \cdot \mathbf{M} \cdot \mathbf{V}$ *is full rank with probability at least* $1 - \frac{2r}{\beta}$*.*

3.6 Limitations of Our Model

Our model does not permit a number of common operations that might arise in standard lattice cryptanalysis. For example, we naturally disallow any modular reductions or rounding on the results of zero-testing, since the relation would no longer be algebraic. This may at first appear problematic, since it means our model does not capture many simple attack strategies such as LLL [33].

We stress, however, that this is a common feature of many abstract attack models defined in the literature. For example, the random oracle model does not allow for differential cryptanalysis, despite it being a powerful way to attack hash functions. This is usually considered okay, since schemes are tuned (say, by increasing the number of rounds) to make such attacks useless. Similarly, the generic group model is often applied to elliptic curves, even though the model does not allow for known attacks such as the MOV attack [34]. Instead, these models capture things the adversary can do no matter how parameters are chosen.

Our setting is similar, as most lattice attacks can be defeated by tuning parameters. The most devastating attacks on schemes such as GGH15 are zeroizing attacks, as they are present *no matter how parameters are chosen*. Therefore, we devise a model that accurately captures how zeroizing attacks are performed, and tune parameters to block all other attacks.

4 Towards Zeroizing Resistance: New Models and Constructions

4.1 Section Overview

In this section we construct a graph-induced encoding scheme with two desirable properties.

Property 1: Asymmetric Levels. In asymmetric multilinear maps such as GGH13 and CLT13, plaintexts are encoded relative to subsets $\ell \subseteq [\kappa]$, where κ is a positive integer. Two encodings can be added if and only if they are encoded at the same level set and can be multiplied if and only if they are encoded at disjoint level sets. Only top level $[\kappa]$ encodings can be zero-tested. In certain settings such as obfuscation, it is desirable to enforce restrictions based on these asymmetric levels (for example, to implement straddling sets which prevent "mixed-input" attacks [23, 25]). Unfortunately, the GGH15 edge restrictions do not immediately give us the same capabilities of asymmetric level restrictions. Thus, we require a notion of "Graph-Induced Multilinear Maps with Asymmetric Levels", which simultaneously associates every encoding with a graph path $u_i \rightsquigarrow v_i$ as well

as a level set $\ell \subseteq [\kappa]$ (first described by Halevi [24]). Addition, multiplication, and zero-test operations are only allowed as long as both the graph-induced restrictions and the asymmetric level set restrictions are satisfied.

We naturally redefine our GGH15 Zeroizing Model for this new notion, calling the resulting model the "Level-Restricted GGH15 Zeroizing Model". This model is identical to the GGH15 Zeroizing Model, except the adversary is now forced to additionally respect the asymmetric level restrictions when computing a top-level encoding of zero.

Property 2: Semantic Security of Encodings. Recent techniques of Chen et al. [10] show how to produce GGH15 encodings that achieve provable semantic security from LWE via a new construction they call "γ-GGH15 encodings". We give the formal security statement and show how to adapt their security proof to our setting in the full version [1]. Note that this semantic security guarantee is orthogonal to what our GGH15 Zeroizing Model captures. Semantically secure encodings ensure that the encodings themselves do not leak information, but only in the setting where successful zero-tests are computationally unachievable. On the other hand, our GGH15 Zeroizing Model captures adversaries who attack using the zero-test leakage but only under the idealized assumption that the encodings themselves leak nothing.

A New GGH15 Variant We integrate these two new techniques into a new construction we call γ-GGH15-AL (γ-encodings and asymmetric levels). We enforce asymmetric levels using a simple trick of dividing by random scalars due to Halevi [24]. We show that security of our γ-GGH15-AL construction in the GGH15 Zeroizing Model implies security in a (more restrictive) Level-Restricted GGH15 Zeroizing Model. In other words, we prove that an attack on γ-GGH15-AL that is free to disobey the asymmetric level restrictions has no more power than an attack that obeys the asymmetric level restrictions. The proof proceeds from applications of the Schwartz-Zippel lemma, which allow us to argue that a top-level encoding that disobeys level restrictions will not give a successful zero-test (with overwhelming probability). To achieve semantic security guarantees, we incorporate the γ-GGH15 encoding strategy of [10] into our γ-GGH15-AL construction.

We note that semantic security is only a heuristic statement in our setting. The semantic security proofs of [10] hold when the adversary cannot successfully zero-test, but in our construction, zero-testing can be achieved using a right bookend vector. Thus, our construction only has semantic security when this bookend vector is hidden from the adversary. The intuition is that when the right bookend vector is not hidden, security is lost because of zeroizing attacks, at which point we appeal to our GGH15 Zeroizing Model.

At the end of this section, we introduce a third model we call the "GGH15 Annihilation Model." We show that any successful zeroizing attacks in the GGH15 Zeroizing Model on our γ-GGH15-AL construction imply the existence of a successful adversary in the GGH15 Annihilation Model (by first going through the

Level-Restricted GGH15 Zeroizing Model). An adversary in the GGH15 Annihilation Model will correspond to a polynomial-complexity arithmetic circuit that *annihilates* the zero-test polynomials submitted by the adversary.

4.2 A Graph-Induced Encoding Scheme with Asymmetric Levels

Overview To encode a plaintext matrix $\mathbf{M}$ on an edge $i \rightsquigarrow j$ with level set $L \subseteq [\kappa]$ we first generate a random matrix $\mathbf{P}$ in order to apply the $\gamma_{\otimes \text{diag}}$ function of [10]. The resulting pre-encoding $diag(\mathbf{M} \otimes \mathbf{P}, \mathbf{P})$ is encoded via the ordinary GGH15 encoding procedure to obtain an encoding $\mathbf{D}$. The next step is to draw a random $k \times k$ matrix $\mathbf{B}$ and append it on along the diagonal. This matrix $\mathbf{B}$ ensures each final encoding matrix $\mathbf{C}$ has sufficient entropy (used in Lemma 5), and is crucial for Lemma 6. The next step is to multiply by Kilian-randomization matrices (drawn by KeyGen for each vertex), and then divide by level scalars $\prod_{\ell \in L} z_\ell$. The resulting encoding is

$$\mathbf{C} = (\prod_{\ell \in L} z_\ell)^{-1} \cdot \mathbf{R}_i^{-1} \cdot \begin{bmatrix} \mathbf{D} & \\ & \mathbf{B} \end{bmatrix} \cdot \mathbf{R}_j.$$

To ensure that zero-testing works, we construct our right bookend vector $\mathbf{w}$ to contain the product $(\prod_{\ell \in [\kappa]} z_\ell)$, which cancels out the level scalars in the encoding as long as it is at the top level $[\kappa]$. The left and right bookends also contain Kilian-randomization matrices $\mathbf{R}_1$ and $\mathbf{R}_d^{-1}$ multiplied in to cancel out the Kilian-randomization on the encodings. The bookends contain additional components $\mathbf{b}_v$ and $\mathbf{b}_w^\top$ which multiply with the $\mathbf{B}$ random matrices during zero-testing. This has the effect of adding the products of random matrices (with two random bookends) to the result of any zero-test (this will be crucial for our obfuscation security proof, where it will have the effect of adding a random branching program evaluation). The remaining bookend components are essentially set to be the bookends required by the γ-GGH15 encodings. However, we also multiply them by randomly sampled vectors $\mathbf{v}'$ and $\mathbf{w}'$ to simplify dimensions.

Construction γ-$\mathsf{GGH15\text{-}AL.KeyGen}(1^\lambda, G, R = \mathbb{Z}, \kappa, \beta, k)$:[7]

Parameter Generation

- Label the nodes of G in topological order as $1, \ldots, d$ where node 1 is the unique source and node d is the unique sink.
- Choose parameters $n, w, n', m, q, \sigma, \chi, B$ where $n = wn' + n'$ according to the remark below. All operations happen over $\mathbb{Z}_q$. Plaintexts have dimension $w \times w$ with entries bounded by β, pre-encodings have dimension $n \times n$ with entries bounded (with high probability) by $\beta \cdot \sigma \cdot \sqrt{n}$, and encodings have

[7] κ is the number of asymmetric levels, β is a bound on the size of plaintext entries, and k is the dimension of the block diagonal matrices we append during the encoding procedure.

dimension $(m+k)\times(m+k)$ with entries bounded by $\nu = 2^\lambda$. We draw error matrices under distribution $(\chi)^{n\times m}$ and set B to be the zero-test bound.

Instance Generation

- (GGH15 matrices and trapdoors) For each vertex $i \in V$, sample $(\mathbf{A}_i, \tau_i) \leftarrow \mathsf{TrapSam}(1^n, 1^m, q)$.
- (Kilian-randomization matrices) For each vertex $i \in V$, sample a random invertible $\mathbf{R}_i \in \mathbb{Z}_q^{(m+k)\times(m+k)}$.
- (Asymmetric level scalars) For each level $\ell \in [\kappa]$, sample a random invertible $z_\ell \in \mathbb{Z}_q$.

Bookend Generation

- (Left bookend matrix from γ-GGH15 encodings) Sample a random $\mathbf{J}' \leftarrow \{0,1\}^{n'\times wn'}$ and define
$$\mathbf{J} := [\mathbf{J}' \mid \mathbf{I}^{n'\times n'}].$$
- (Encoding matrix used in right bookend) Sample a uniform $\mathbf{A}^* \leftarrow \mathbb{Z}_q^{n\times m}$, an error matrix $\mathbf{E}^* \leftarrow (\chi)^{n\times m}$, and compute
$$\mathbf{D}^* \leftarrow \mathsf{SampleD}(\mathbf{A}_d, \tau_d, \begin{bmatrix}\mathbf{I}^{wn'\times wn'} & \\ & \mathbf{0}^{n'\times n'}\end{bmatrix}\cdot \mathbf{A}^* + \mathbf{E}^*, \sigma)$$
This encoding serves to cancel out the lower random block diagonals on pre-encodings and enables zero-testing on the actual plaintexts.
- (Random bookend vectors) Sample $\mathbf{v}' \leftarrow D^{n'}_{\mathbb{Z},\sigma}, \mathbf{w}' \leftarrow D^{m}_{\mathbb{Z},\sigma}$.
- (Final bookend vectors) Sample uniform $\mathbf{b}_v \in \mathbb{Z}_\nu^k, \mathbf{b}_w \in \mathbb{Z}_\nu^k$ and compute the final bookends
$$\mathbf{v} = [\mathbf{v}'\cdot\mathbf{J}\cdot\mathbf{A}_1|\mathbf{b}_v]\cdot\mathbf{R}_1,\ \mathbf{w} = (\prod_{\ell\in[\kappa]} z_\ell)\cdot\mathbf{R}_d^{-1}\cdot\begin{bmatrix}\mathbf{D}^*\cdot\mathbf{w}'^\top \\ \mathbf{b}_w^\top\end{bmatrix}.$$

Output

- Public parameters $\mathsf{pp} = \{n, w, n', m, k, q, \sigma, \chi, B, \mathbf{v}, \mathbf{w}\}$
- Secret parameters $\mathsf{sp} = \{\mathbf{A}_i, \tau_i, \mathbf{R}_i\}_{i\in[d]}, \{z_\ell\}_{\ell\in[\kappa]}$

γ-GGH15-AL.Enc$(\mathsf{sp}, \mathbf{M}\in\mathbb{Z}_\beta^{w\times w}, i \rightsquigarrow j, L\subseteq[\kappa])$:

- Draw $\mathbf{P} \leftarrow D^{n'\times n'}_{\mathbb{Z},\sigma}$ and $\mathbf{E}\leftarrow(\chi)^{n\times m}$
- Compute $\mathbf{D} \leftarrow \mathsf{SampleD}(\mathbf{A}_i, \tau_i, \begin{bmatrix}\mathbf{M}\otimes\mathbf{P} & \\ & \mathbf{P}\end{bmatrix}\cdot\mathbf{A}_j + \mathbf{E}, \sigma)$
- Draw uniform $\mathbf{B}\leftarrow\mathbb{Z}_\nu^{k\times k}$ and output the encoding
$$\mathbf{C} = (\prod_{\ell\in L} z_\ell)^{-1}\cdot\mathbf{R}_i^{-1}\cdot\begin{bmatrix}\mathbf{D} & \\ & \mathbf{B}\end{bmatrix}\cdot\mathbf{R}_j$$

γ-GGH15-AL.ZeroTest$(\mathsf{pp}, \mathbf{C})$:

- Return zero if $|\mathbf{v} \cdot \mathbf{C} \cdot \mathbf{w}^\top| \leq B$, and not zero otherwise.

Parameters. First, we derive an additional security parameter $\lambda_{\mathrm{LWE}} = \mathsf{poly}(\lambda)$ which determines the hardness of LWE instances associated with the construction. We set the encoding bound $\nu = 2^\lambda$ and choose $n, w, n', m, q, \sigma, \chi = D_{\mathbb{Z},s}$ where $n = wn' + n'$, $m = \Theta(n \log q)$ and $\sigma = \Theta(\sqrt{n \log q})$ for trapdoor functionality and $n' = \Theta(\lambda_{\mathrm{LWE}} \log q)$ and $s = \Omega(\sqrt{n'})$ for LWE security.[8] Set the zero-test bound $B := (m \cdot \beta \cdot \sigma \cdot \sqrt{n})^{d+1} + (k \cdot \nu)^{d+1}$ and choose $q \geq B \cdot \omega(\mathsf{poly}(\lambda))$ such that $q \leq (\sigma/\lambda_{\mathrm{LWE}}) \cdot (2^{\lambda_{\mathrm{LWE}}})^{1-\epsilon}$ for some $\epsilon \in (0, 1)$.

In the full version of this paper, we show that these constraints can be satisfied with $\lambda_{\mathrm{LWE}} = \mathsf{poly}(\lambda)$, and furthermore that this setting of parameters satisfies correctness [1].

4.3 Level-Restricted GGH15 Zeroizing Model

In order to define this model, we need the following definition.

Definition 4 (Level-Respecting Encodings). *Fix a universe of levels $[\kappa]$. Let L_i be the set of levels associated with encoding $\mathbf{C}_i$. Let m be a monomial over encodings $\{\mathbf{C}_i\}$ which contains the j encodings $\mathbf{C}_1, ..., \mathbf{C}_j$. Then m is level-respecting if $L_1, ..., L_j$ are disjoint and $\bigcup_{i=1}^{j} L_i = [\kappa]$. A polynomial p over encodings $\{\mathbf{C}_i\}$ is level-respecting if and only if each of its monomials is.*

We only mention the differences between this model and the GGH15 Zeroizing Model. Here we expect that the GGH15 variant G that the model is initialized with supports asymmetric levels, namely that $\mathsf{G.Enc}$ additionally takes as input a level set $L \subseteq [\kappa]$.

Initialize Parameters. The model $\mathcal{M}$ in addition takes a parameter κ denoting the number of asymmetric levels.

Initialize Elements. $\mathcal{M}$ is additionally given a level set $L_i \subseteq [\kappa]$ along with each plaintext $\mathbf{M}_i$ and path $u_i \rightsquigarrow v_i$. $\mathcal{M}$ computes the corresponding pre-encoding $\mathbf{S}_i$ (from $\mathsf{G.PreProcess}$), and computes the encoding

$$\mathbf{C}_i \leftarrow \mathsf{G.Enc}(\mathsf{sp}, \mathbf{S}_i, u_i \rightsquigarrow v_i, L_i).$$

$\mathcal{M}$ stores $(\mathbf{S}_i, \mathbf{C}_i, u_i \rightsquigarrow v_i, L_i)$ in a *pre-zero-test table*.

Zero-test. When the adversary submits a polynomial p, $\mathcal{M}$ additionally checks that it is level-respecting, and if it is not, $\mathcal{M}$ returns $\perp$.

Lemma 5. *Let $\mathcal{A}$ be a successful adversary in the GGH15 Zeroizing Model instantiated with γ-GGH15-AL. Then there exists a successful adversary $\mathcal{A}'$ in the Level-Restricted GGH15 Zeroizing Model instantiated with γ-GGH15-AL.*

See the full version [1] for a proof of the above lemma, which relies on a simple application of the Schwartz-Zippel lemma applied to polynomials over formal variables corresponding to the level scalars.

[8] Following Chen et. al. [10]

4.4 GGH15 Annihilation Model

We turn to describing a new model which has properties that are much easier to reason about when proving security. Instead of requiring the adversary to find an algebraic relation in the post-zero-test stage, we instead require the adversary to find an annihilating polynomial for the set of successful zero-test polynomials it previously obtained. More specifically, this polynomial must annihilate the zero-test polynomials when evaluated on square matrices of formal variables of some dimension k. This k affects the difficulty of winning in the model, since matrices of larger dimension will be harder to annihilate. The advantage of having this model is that we have a notion of winning that corresponds more directly to the underlying plaintexts encoded with the scheme. Namely, if we are able to encode plaintexts (taking advantage of asymmetric levels) in such a way that annihilating successful zero-test polynomials is hard, we can immediately obtain security in this model.

We describe the differences between this model and the Level-Restricted GGH15 Zeroizing Model. First, there is no computational bound on the adversary — it can submit as many zero-test queries as it wants and can take as much computation as it wants in the post-zero-test stage. However, each post-zero-test polynomial it submits must be implemented with a polynomial size circuit. The other modifications are described below.

Initialize Parameters. The model $\mathcal{M}$ takes in an additional 'tuning' parameter k, which determines in some sense how strong the win condition will be.

Post-zero-test. At this point the adversary has submitted a set $\{p_u\}_u$ of successful zero-test polynomials which we associate with a set of formal variables $\{\widehat{p}_u\}_u$. The adversary now submits a *polynomial sized* circuit $\bar{C}$ that implements a polynomial $\bar{Q}(\{\widehat{p}_u\}_u)$ over these formal variables. The model $\mathcal{M}$ associates a set of $k \times k$ matrices $\{\widehat{\mathbf{C}}_i\}_i$ of formal variables with the set of encodings $\{\mathbf{C}_i\}_i$ and considers two additional k-dimensional vectors $\widehat{\mathbf{v}}$ and $\widehat{\mathbf{w}}$ of formal variables. Note that each individual entry of each of these matrices and vectors is a distinct formal variable. $\mathcal{M}$ returns "Win" if the following hold:

1. The degree of $\bar{Q}$ is $2^{o(\lambda)}$
2. $\bar{Q}(\{\widehat{p}_u\}_u) \not\equiv 0$
3. $\bar{Q}(\{\widehat{\mathbf{v}} \cdot p_u(\{\widehat{\mathbf{C}_i}\}_i) \cdot \widehat{\mathbf{w}}^\top\}_u) \equiv 0$

Lemma 6. *Fix any $k \in \mathbb{N}$. Let $\mathcal{A}$ be a successful adversary in the Level-Restricted GGH15 Zeroizing Model instantiated with γ-GGH15-AL where* KeyGen *receives the parameter k. Then there exists a successful adversary $\mathcal{A}'$ in the GGH15 Annihilation Model with tuning parameter k.*

A proof of the above can also be found in the full version [1]. It again relies on the Schwartz-Zippel lemma, this time applied to polynomials over formal variables corresponding to the elements of the block diagonals added during γ-GGH15-AL.Enc.

5 An iO Candidate with Zeroizing Resistance

We design our obfuscator to invoke the Branching Program Un-Annihilatability (BPUA) Assumption of Garg et al. [21]. Roughly, this assumption states that no polynomial-size circuit can annihilate the evaluations of every matrix branching program, provided we consider branching programs whose input bits are read many times and in interleaved layers.

Thus, the first step of our obfuscator is to pad the input branching program in order to satisfy the requirement of the BPUA Assumption. To facilitate this, one of the inputs to our obfuscator is the parameter $t = t(\ell, \lambda) \geq 4\ell^4$ which specifies the minimum number of layers required. Note that the resulting padded program may have length greater than t, so we use a separate variable d to denote the actual length of the branching program after padding. We also enforce that each *pair* of input bits is read together in many layers, which is required to invoke the p-Bounded Speedup Hypothesis of [25].

To encode the matrices with γ-GGH15-AL, we pick asymmetric level sets from a straddling set system. The sets are assigned precisely to enforce that evaluations respect the input read structure of the padded branching program. The encoding edges are picked so that the branching program evaluations are naturally computed by traversing a path graph.

5.1 Construction

Input. The input to the obfuscator is the security parameter λ and a dual-input branching program BP (defined in Section 2.3) of length h, width w, and input length ℓ. BP consists of the matrices $\{\mathbf{M}_{i,b_1,b_2}\}_{i\in[h],b_1,b_2\in\{0,1\}}$ and input selection function $\mathsf{inp} : [h] \to [\ell] \times [\ell]$ which satisfies the following requirements:

- For each $i \in [h] : \mathsf{inp}(i)_1 \neq \mathsf{inp}(i)_2$, where $\mathsf{inp}(i)_1, \mathsf{inp}(i)_2$ denote the first and second slots of $\mathsf{inp}(i)$, respectively.
- For each pair $j \neq k \in [\ell]$, there exists $i \in [h]$ such that $\mathsf{inp}(i) \in \{(j,k),(k,j)\}$.

BP is evaluated on input $x \in \{0,1\}^\ell$ by checking whether

$$\prod_{i\in[h]} \mathbf{M}_{i,x(i)} = 0^{w\times w}$$

where we abbreviate $x(i) := (x_{\mathsf{inp}(i)_1}, x_{\mathsf{inp}(i)_2})$.

Step 1: Pad the branching program. We pad the branching program with identity matrices until it has $d \geq t$ layers to ensure the following conditions:

- Each pair of input bits (j,k) is read in at least $4\ell^2$ different layers.
- There exist layers $i_1 < i_2 < \cdots < i_t$ such that $\mathsf{inp}(i_1)_1, \ldots, \mathsf{inp}(i_t)_1$ cycles t/ℓ times through $[\ell]$.

Step 2: Form straddling sets. For each input index $i \in [\ell]$, let r_i be the number of layers in which the bit i is read, and create a straddling set system with universe $\mathbb{U}^{(i)}$ and subsets $\{S^{(i)}_{j,b}\}_{j\in[r_i],b\in\{0,1\}}$. Let $\mathbb{U} := \bigcup_{i\in[\ell]} \mathbb{U}^{(i)}$.

Step 3: Encode with γ-GGH15-AL. Let G be a path graph with $d+1$ nodes $1, ..., d+1$ and initialize the γ-GGH15-AL construction[9]

$$\mathsf{pp}, \mathsf{sp} \leftarrow \gamma\text{-GGH15-AL.KeyGen}(1^\lambda, G, \mathbb{Z}, |\mathbb{U}|, \max_{i,b_1,b_2}\{||\mathbf{M}_{i,b_1,b_2}||_\infty\}, k = 5).$$

For $i \in [d]$ and $b \in \{1, 2\}$, define $j_b(i)$ to be the number of times $\mathsf{inp}(i)_b$ has been read after reading i columns of the branching program, and compute

$$\mathbf{C}_{i,b_1,b_2} \leftarrow \gamma\text{-GGH15.Enc}(\mathsf{sp}, \mathbf{M}_{i,b_1,b_2}, i \rightsquigarrow i+1, S^{\mathsf{inp}(i)_1}_{j_1(i),b_1} \cup S^{\mathsf{inp}(i)_2}_{j_2(i),b_2}).$$

5.2 Security

In order to state the p-Bounded speedup hypothesis, we recall the following definition of Miles et al. [25].

Definition 5 (X-Max-2-SAT Solver). *Consider a set $X \subseteq \{0,1\}^\ell$. We say that an algorithm $\mathcal{A}$ is an X-Max-2-SAT solver if it solves the Max-2-SAT problem restricted to inputs in X. Namely given a 2-CNF formula ϕ on ℓ variables, $\mathcal{A}(\phi) = 1$ iff $\exists x \in X$ that satisfies at least a $7/10$ fraction of ϕ's clauses.*

Assumption 1. *(p-Bounded Speedup Hypothesis, introduced in [25]). Let $p : \mathbb{N} \to \mathbb{N}$. Then for any X-Max-2-SAT solver that has size $t(\ell)$, $|X| \leq p(\mathsf{poly}(t(\ell)))$.*

The assumption essentially states that the NP-complete problem Max-2-SAT is still hard even for restricted sets of variable assignments. This hardness is parameterized by p, and in its strongest form, p is taken to be a polynomial. In this form, the assumption states that no polynomial time algorithm can solve X-Max-2-SAT on an X of super-polynomial size. However, we can also take p to be $2^{\mathsf{polylog}(n)}$ and obtain meaningful results as we discuss in the full version of this work [1].

We now state a non-uniform variant of the BPUA, but first we need the following definition from [21].

Definition 6. *A matrix branching program BP is L-bounded for $L \in \mathbb{N}$ if every intermediate value computed when evaluating BP on any input is at most L. In particular all of BP's outputs and matrix entries are at most L.*

[9] We set $k = 5$ so that the dimension of the random block diagonals added during encoding match the dimension of matrix branching programs obtained from Barrington's theorem.

Assumption 2. *(Non-uniform variant of the BPUA assumption of [21]) Let* $t = \mathsf{poly}(\ell, \lambda)$ *and let* $\mathcal{X} \subseteq \{0,1\}^\ell$ *have* $\mathsf{poly}(\lambda)$ *size and* Q *be a* $\mathsf{poly}(\lambda)$*-size* $2^{o(\lambda)}$*-degree polynomial over* $\mathbb{Z}$*. Then for all* ℓ*, sufficiently large* λ*, and all primes* $2^\lambda < p < 2^{\mathsf{poly}(\lambda)}$*, there exists a* 2^λ*-bounded dual-input matrix branching program* $BP : \{0,1\}^\ell \to [2^\lambda]$ *of length* t *whose first input selection function* (inp_1) *iterates over the* ℓ *input bits* t/ℓ *times, such that* $Q(\{BP(x)\}_{x\in\mathcal{X}}) \neq 0 \pmod p$*.*

Note that this statement is a very mild strengthening of the original BPUA assumption stated in [21]. Their assumption is required to hold for any Q of bounded degree generated by a polynomial-time algorithm, whereas our assumption must hold for any Q of polynomial size and bounded degree. However, we note that Garg et al. [21] justify their assumption by showing it is implied by the existence of PRFs in NC^1 secure against $\mathsf{P/poly}$. With a minor tweak to their proof, we can show our non-uniform BPUA is also implied by the existence of PRFs in NC^1 secure against $\mathsf{P/poly}$. We simply modify the non-uniform adversary used in [Theorem 2, [21]] to take the polynomial-size Q as advice.

Finally, we use the following definition in our security proof.

Definition 7 (Input-Respecting Polynomial). *Given a branching program* $\{\boldsymbol{M}_{i,b_1,b_2}\}_{i\in[h],b_1,b_2\in\{0,1\}}$ *with input selection function* $\mathsf{inp} : [h] \to [\ell] \times [\ell]$*, a polynomial* p *over the matrices (or elements of matrices) is input-respecting if no monomial involves two encodings* $\{\boldsymbol{M}_{i,b_1^{(i)},b_2^{(i)}}\}, \{\boldsymbol{M}_{j,b_1^{(j)},b_2^{(j)}}\}$ *(or entries of encodings) such that* $\mathsf{inp}(i)_1 = \mathsf{inp}(j)_1$ *and* $b_1^{(i)} \neq b_1^{(j)}$ *or* $\mathsf{inp}(i)_2 = \mathsf{inp}(j)_2$ *and* $b_2^{(i)} \neq b_2^{(j)}$*.*

Theorem 1 (Main Theorem). *Assuming the* p*-Bounded Speedup Hypothesis and the non-uniform BPUA Assumption (implied by the existence of PRFs in* NC^1 *secure against* $\mathsf{P/poly}$*), our obfuscator is secure in the GGH15 Zeroizing Model.*

Proof. It suffices to prove security in the GGH15 Annihilation Model with parameter 5 (since we set $k = 5$ in the obfuscation construction). Suppose an adversary $\mathcal{A}$ wins in this model instantiated with our obfuscator. We argue that every successful zero-test polynomial submitted by $\mathcal{A}$ is a linear combination of polynomially many branching program evaluations and thus that the existence of a Q used to win in the GGH15 Annihilation Model would violate Assumption 2. We know that every successful zero-test polynomial submitted by $\mathcal{A}$ in this model is level-respecting, so by construction of straddling sets, we can conclude that every polynomial is input-respecting. A polynomial that is both edge-respecting (so each monomial contains exactly one branching program matrix from each layer) and input-respecting, is a linear combination of branching program evaluations. However, we have no bound on the number of terms in the linear combination. We now rely on the analysis techniques of Miles, Sahai, and Weiss [25] to show that each polynomial is in fact a linear combination of polynomially many branching program evaluations, assuming the p-Bounded Speedup Hypothesis. A proof of the following lemma is available in the full version [1].

Lemma 7. *(adapted from [25]) Consider an adversary $\mathcal{A}$ interacting with our obfuscation candidate in the GGH15 Annihilating Model. Assuming the p-Bounded Speedup Hypothesis, any edge-respecting and input-respecting polynomial submitted by $\mathcal{A}$ is a linear combination of polynomially-many branching program evaluations.*

With this lemma in hand, we inspect the Q submitted by $\mathcal{A}$ that resulted in the model outputting "Win". Notice that the $\{\widehat{\mathbf{C}}_i\}_i$ are in the shape of a dual-input branching program of width 5 (without the bookends), so by Lemma 7, every $\widehat{\mathbf{v}} \cdot p_u(\{\widehat{\mathbf{C}}_i\}_i) \cdot \widehat{\mathbf{w}}^\top$ is actually a linear combination of polynomially many honest branching program evaluations. Since there are only polynomially many p_u's (since Q is implemented with a polynomial size circuit), and since Q is identically zero over these evaluations, Q contradicts Assumption 2, and we can conclude that $\mathcal{A}$ could not have won in the GGH15 Annihilation model and thus in the GGH15 Zeroizing Model except with negligible probability.

□

References

1. Bartusek, J., Guan, J., Ma, F., Zhandry, M.: Return of GGH15: Provable security against zeroizing attacks. Cryptology ePrint Archive, Report 2018/511 (2018) https://eprint.iacr.org/2018/511.
2. Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. Contemporary Mathematics **324** (2003) 71–90
3. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. (2013) 40–49
4. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. (2013) 1–17
5. Coron, J.S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. (2013) 476–493
6. Gentry, C., Gorbunov, S., Halevi, S.: Graph-induced multilinear maps from lattices. (2015) 498–527
7. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. (2005) 84–93
8. Wichs, D., Zirdelis, G.: Obfuscating compute-and-compare programs under LWE. (2017) 600–611
9. Goyal, R., Koppula, V., Waters, B.: Lockable obfuscation. (2017) 612–621
10. Chen, Y., Vaikuntanathan, V., Wee, H.: GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. (2018) 577–607
11. Canetti, R., Chen, Y.: Constraint-hiding constrained PRFs for NC^1 from LWE. (2017) 446–476
12. Goyal, R., Koppula, V., Waters, B.: Separating semantic and circular security for symmetric-key bit encryption from the learning with errors assumption. (2017) 528–557
13. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. (1994) 124–134

14. Albrecht, M.R., Bai, S., Ducas, L.: A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. (2016) 153–178
15. Cramer, R., Ducas, L., Peikert, C., Regev, O.: Recovering short generators of principal ideals in cyclotomic rings. (2016) 559–585
16. Biasse, J.F., Song, F.: Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. (2016) 893–902
17. Pellet-Mary, A.: Quantum attacks against indistinguishablility obfuscators proved secure in the weak multilinear map model. (2018) 153–183
18. Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. (2015) 3–12
19. Coron, J.S., Lee, M.S., Lepoint, T., Tibouchi, M.: Cryptanalysis of GGH15 multilinear maps. (2016) 607–628
20. Chen, Y., Gentry, C., Halevi, S.: Cryptanalyses of candidate branching program obfuscators. (2017) 278–307
21. Garg, S., Miles, E., Mukherjee, P., Sahai, A., Srinivasan, A., Zhandry, M.: Secure obfuscation in a weak multilinear map model. (2016) 241–268
22. Ma, F., Zhandry, M.: New multilinear maps from CLT13 with provable security against zeroizing attacks. Cryptology ePrint Archive, Report 2017/946 (2017) http://eprint.iacr.org/2017/946.
23. Barak, B., Garg, S., Kalai, Y.T., Paneth, O., Sahai, A.: Protecting obfuscation against algebraic attacks. (2014) 221–238
24. Halevi, S.: Graded encoding, variations on a scheme. Cryptology ePrint Archive, Report 2015/866 (2015) http://eprint.iacr.org/2015/866.
25. Miles, E., Sahai, A., Weiss, M.: Protecting obfuscation against arithmetic attacks. Cryptology ePrint Archive, Report 2014/878 (2014) http://eprint.iacr.org/2014/878.
26. Badrinarayanan, S., Miles, E., Sahai, A., Zhandry, M.: Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. (2016) 764–791
27. Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., Zimmerman, J.: Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. (2015) 563–594
28. Ma, F., Zhandry, M.: The mmap strikes back: Obfuscation and new multilinear maps immune to CLT13 zeroizing attacks. In: TCC 2018. (2018)
29. Boneh, D., Lewi, K., Montgomery, H.W., Raghunathan, A.: Key homomorphic PRFs and their applications. (2013) 410–428
30. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the fortieth annual ACM symposium on Theory of computing, ACM (2008) 197–206
31. Garg, S., Miles, E., Mukherjee, P., Sahai, A., Srinivasan, A., Zhandry, M.: Secure obfuscation in a weak multilinear map model. Cryptology ePrint Archive, Report 2016/817 (2016) http://eprint.iacr.org/2016/817.
32. Barrington, D.A.M.: Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. (1986) 1–5
33. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**(4) (1982) 515–534
34. Menezes, A.J., Okamoto, T., Vanstone, S.A.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Transactions on Information Theory **39**(5) (Sept 1993) 1639–1646