The MMap Strikes Back: Obfuscation and New Multilinear Maps Immune to CLT13 Zeroizing Attacks

Fermi Ma(✉) and Mark Zhandry

Princeton University, Princeton, USA
fermima1@gmail.com, mzhandry@princeton.edu

Abstract. All known multilinear map candidates have suffered from a class of attacks known as "zeroizing" attacks, which render them unusable for many applications. We provide a new construction of *polynomial-degree* multilinear maps and show that our scheme is provably immune to zeroizing attacks under a strengthening of the Branching Program Un-Annihilatability Assumption (Garg et al., TCC 2016-B).

Concretely, we build our scheme on top of the CLT13 multilinear maps (Coron et al., CRYPTO 2013). In order to justify the security of our new scheme, we devise a weak multilinear map model for CLT13 that captures zeroizing attacks and generalizations, reflecting *all known* classical polynomial-time attacks on CLT13. In our model, we show that our new multilinear map scheme achieves *ideal security*, meaning no known attacks apply to our scheme. Using our scheme, we give a new multiparty key agreement protocol that is several orders of magnitude more efficient that what was previously possible.

We also demonstrate the general applicability of our model by showing that several existing obfuscation and order-revealing encryption schemes, when instantiated with CLT13 maps, are secure against known attacks. These are schemes that are actually being implemented for experimentation, but until our work had no rigorous justification for security.

1 Introduction

Cryptographic multilinear maps have proven to be a revolutionary tool. Very roughly, a multilinear map is an encoding scheme where one can blindly compute polynomials over encoded elements, without any knowledge of the underlying elements. They have been used for numerous cutting-edge cryptographic applications, such as multiparty non-interactive key agreement [2], attribute-based encryption for circuits [3], asymptotically optimal broadcast encryption [4], witness encryption [5], functional encryption [6,7], and most notably mathematical program obfuscation [6]. In turn, obfuscation has been used to construct many more amazing applications [8–15] as well as establish interesting connections to other areas of computer science [16,17].

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

A. Beimel and S. Dziembowski (Eds.): TCC 2018, LNCS 11240, pp. 513–543, 2018.
https://doi.org/10.1007/978-3-030-03810-6_19

Unfortunately, all known multilinear maps for degree $d > 2$ [18–20] have suffered from devastating attacks known as "zeroizing" attacks [18,21–23]. These attacks have rendered most of the applications above insecure. In response, many authors introduced "fixes"; these fixes came in many forms, from tweaking how information is extracted from the map [24] to compiling the existing weak multilinear maps into new ones that were presumably stronger [7,25,26]. However, these fixes were largely ad hoc, and indeed it was quickly shown how to generalize the zeroizing attacks to circumvent the fixes [25–29].

Given the many attacks and the speed at which fixes were subsequently broken, researchers have begun attempting to build applications in a sound way using weak maps. The initial observation by Badrinarayanan, Miles, Sahai, and Zhandry [30] is that all (classical polynomial-time[1]) attacks require the ability to obtain an encoding of zero. Miles, Sahai, and Zhandry [36] observed moreover that all zeroizing attacks on the original GGH13 multilinear map have a very similar structure. They define an abstract attack model, called the "annihilating attack model," that encompasses and generalizes all existing zeroizing attacks on these specific maps. Since their initial publication, all subsequent attacks on GGH13 have either relied on quantum procedures [35] or on the specific setting of parameters [34]. On the other hand, zeroizing attacks are inherently *parameter-independent*, as they only depend on the functionality of the zero-testing procedure. It remains the case today that all classical, parameter-independent attacks on GGH13 fit in the "weak model" of Miles et al. [36]. Therefore, this annihilating model appears to be a fully general abstraction of the inherent vulnerabilities of the GGH13 multilinear maps.

Within the weak model, Badrinaryanan et al. [30] constructed a secure witness encryption scheme, while Garg, Miles, Mukherjee, Sahai, Srinivasan, and Zhandry [37] built secure obfuscation and order revealing encryption. Since these constructions have been proven secure in the weak model, they are secure against all known attacks on GGH13. To date, these are the only *direct* applications of multilinear maps that have been proved secure in the weak multilinear map model for GGH13.[2] Moreover, GGH13 is the only multilinear map for which an accurate weak model has been devised. This leads to the following goals:

Devise weak multilinear map models for other multilinear maps — such as CLT13 or GGH15 — that capture all known attack strategies on the maps.

Give new applications of multilinear maps that can be constructed and proven secure in weak multilinear map models.

[1] Sub-exponential [31–33], parameter-dependent [34], and quantum attacks [35] have been discovered on multilinear map candidates. In this work we will focus on classical adversaries, and will not consider quantum attacks. We will also not consider parameter-dependent or sub-exponential attacks as a break, since they can be defeated by increasing the security parameter.

[2] Most other applications are possible by using obfuscation as a building block.

A weak multilinear map model for CLT13 is especially important, as it is currently the most efficient multilinear map known [38], and therefore most likely to eventually become usable in practice.

In addition, the vulnerabilities of existing multilinear map candidates lead to a natural third goal:

Construct multilinear maps that are not vulnerable to zeroizing attacks.

There has been some initial progress toward this goal. Several authors [39,40] have shown how to construct a version of multilinear maps from obfuscation, which can in turn be built from weak multilinear maps using the aforementioned constructions. This gives a compelling proof of concept that multilinear maps immune to zeroizing attacks should be possible. However, as obfuscation is currently incredibly inefficient, such multilinear map constructions are entirely impractical today. Moreover, obfuscation can be used to directly achieve most applications of multilinear maps, so adding a layer of multilinear maps between obfuscation and application will likely compound the efficiency limitations. Therefore, it is important to build multilinear maps without obfuscation.

1.1 Our Work: New Multilinear Maps

In this paper, we make additional progress on all three goals above. We revisit the idea of "fixing" multilinear maps by using weak maps to build strong maps. We do not use obfuscation, though our scheme is inspired by obfuscation techniques. Unlike the fixes discussed above that were quickly broken, we develop our fix in a methodical way that allows us to formally argue our fix is immune to generalizations of zeroizing attacks. Specifically, our results are the following:

Weak CLT13 Model. First, we need a framework in which to argue security against zeroizing attacks. Our first result is a new weak multilinear map model for CLT13 maps. We demonstrate that this model naturally captures all known attack strategies on CLT13. The model is somewhat different from the model for the GGH13 maps, owing to the somewhat different technical details of the attacks. Unlike the GGH13 case, where the common thread amongst all the attacks was rather explicit[3], the common features of CLT13 attacks are a bit more nebulous, and require additional effort to pull out and formalize.

Model Conversion Theorem. To aid the analysis of schemes in our model, we prove that an attack in the weak CLT13 model requires the existence of a certain type of "annihilating polynomial," analogous to the annihilating polynomials in the weak GGH13 model, but simpler. This "plain annihilating model" makes it very easy to test if a particular usage of CLT13 is safe. For example, it is immediate from known results that an *existing* class of obfuscation constructions [30,41] is secure in our plain annihilating model, under the same algebraic complexity

[3] Namely, all attacks compute the ideal $\langle g \rangle$ generated by some element g.

assumption as in [37]. Hence, these schemes are also secure in our weak CLT13 model. This is the first rigorous argument for security of these schemes. We note that these obfuscation constructions are currently being implemented [38], so justifying their security is important.

Note that [30,41] are *not* secure in the weak GGH13 model, indicating that the weak CLT13 model may be somewhat more useful.

New Multilinear Map Scheme. Armed with a weak model for CLT13, we devise a new *polynomial-degree* multilinear map scheme built on top of CLT13. We then prove that within our weak CLT13 model, *there are no attacks on our new scheme* under a new "Vector-Input Branching Program Un-Annihilatability" Assumption. That is, under our new assumption, any attack at all on our scheme will yield an attack that does not fit in our CLT13 model, and hence gives a brand new attack technique on CLT13 maps. Our scheme is based on obfuscation techniques, but avoids building a full obfuscation scheme, making our scheme significantly more efficient than obfuscation-based multilinear maps, at least for simple settings.

Concretely, to implement a 4-Party Non-Interactive Key Exchange with 80 bits of security, our scheme requires approximately 2^{31} CLT13 encodings with degree 281. For comparison, the most efficient obfuscation-based approach requires at least 2^{44} CLT13 encodings from a much higher-degree map [42]. These estimates are derived in the full version of this paper [1].

The lack of a zeroizing attack means we can be more liberal in the types of encodings that are made public. This allows for greatly enhanced functionality compared to existing multilinear map schemes: for example, we can encode arbitrary ring elements and give out encodings of zero.

The notable limitation of our construction and analysis is that our security proof relies on the Vector-Input Branching Program Un-Annihilatability Assumption, a new algebraic complexity assumption about annihilating polynomials. The assumption is similar to the Branching Program Un-Annihilatability Assumption used in [37], though our new assumption is somewhat stronger and less justified than theirs.

While our scheme is far from practical, we believe this result is a proof of concept that multilinear maps without zeroizing attacks are possible without first building obfuscation. Hopefully, future work will be able to streamline our construction to obtain much more efficient multilinear maps.

Applications. Despite some minor functionality limitations, our multilinear maps can still be used to solve problems that were not previously possible without first building obfuscation. For example, we show how to use it for multiparty non-interactive key exchange (NIKE) for a polynomial number of users. This is the most efficient scheme for $n > 3$ users that is immune to known attacks. Hopefully, our maps can be used to make other applications much more efficient as well.

Ideal Multilinear Maps from GGH13. We note that our techniques for constructing multilinear maps from CLT13 can be combined with the techniques of Garg et al. [37] to give new multilinear maps in the weak model for GGH13.

1.2 Techniques

Weak CLT13 Model. In CLT13, there is a composite modulus $N = \prod_i p_i$.[4] An encoding s is an integer mod N. Let $s_i = s \bmod p_i$ be the vector of Chinese Remainder Theorem components. Each component s_i encodes a component m_i of the plaintext element. An element in the plaintext space can therefore be interpreted as a vector of integers. Each encoding is associated to a level, which is a subset of $\{1, \ldots, d\}$, where d is the multilinearity of the map. Encodings can be added and multiplied, following certain level-restrictions, until a "top-level" encoding is obtained, which is an encoding relative to the set $\{1, \ldots, d\}$. For singleton sets, we drop the set notation and let level $\{i\}$ be denoted as level i.

In CLT13, if s is a top-level encoding of zero — meaning all components of the plaintext are 0 — then one can obtain from it $t = \sum_i \gamma_i s_i$, where γ_i is a rational number, and equality holds over the rationals. The γ_i are unknown, but global constants determined by the parameters of the scheme. That is, for each s, the derived t will use the same γ_i.

All known attacks on the CLT13 multilinear maps follow a particular form. First, public encodings are combined to give top-level encodings, *using operations explicitly allowed by the maps.* If these top-level encodings are zero, then one obtains a t term. The next step in the attack is to solve a polynomial equation Q where the coefficients are obtained from the t terms. In current attacks, the polynomial equation is the characteristic polynomial of a matrix whose entries are rational functions of the zeros.

The next step is to show that the solutions to Q isolate the various s_i components of the initial encodings. Then by performing some GCD computations, one is then able to extract the prime factors p_i, which leads to a complete break of the CLT13 scheme. We show how to capture this attack strategy, and in fact much more general potential strategies, in a new abstract attack model for CLT13. Our model is defined as follows:

- Denote the set of encodings provided to the adversary as $\langle s \rangle$, where each s encodes some plaintext element.
- The adversary is allowed to combine the encodings as explicitly allowed by the multilinear map. Operations are performed component-wise on the underlying plaintext elements.
- If the adversary ever gets a top-level encoding of zero — meaning that the plaintext element is zero in all coordinates — this zero is some polynomial p in the underlying plaintext elements. The adversary obtains a handle to the corresponding element $t = \sum_i \gamma_i p(\langle s \rangle_i)$. Here, $\langle s \rangle_i$ represents the collection of ith components of the various encodings provided.

[4] In [19], this modulus is referred to as x_0.

- The adversary then tries to construct a polynomial Q_i that isolates the ith component for some i. The way we model a successful isolation is that Q_i is a polynomial in two sets of variables: T variables — which correspond to the t terms obtained above — and S variables — which correspond to the ith components of the s encodings. The adversary's goal is to devise a polynomial Q_i such that Q evaluates to 0 when S is substituted for $\langle s\rangle_i$ and T is substituted for the set of t obtained above. We say the adversary wins if she finds such a Q.

In a real attack, roughly, the adversary takes Q, plugs in the values of t, and then solves over the rationals for $\langle s\rangle_i$. Then by taking $GCD(N, s - s_i)$ for some encoding s, she obtains the prime factor p_i. The real adversary does this for every i until she completely factors N. In general, solving Q for $\langle s\rangle_i$ is a computationally intractable task and may not yield unique solutions. The attacks in the literature build a specific Q that can be solved efficiently. In our model, we conservatively treat *any* Q the adversary can find, even ones that are intractable to solve, as a successful attack.

We note that the attacks described in the literature actually build a Q that is a rational function. However, such rational functions can readily be converted into polynomial functions. We indeed demonstrate that such a polynomial Q is implicit in all known attack strategies.

Next, we prove a "model conversion theorem" that implies any attack in our CLT13 model actually yields an attack in a much simpler model that we call the "plain annihilating model." Here, the adversary still constructs polynomials p of the underlying encodings, trying to find a top-level zero. However now, instead of trying to find a Q_i as above, the adversary simply tries to find a polynomial R that annihilates the p polynomials. That is, $R(\{p(\langle S\rangle)\}_p)$ is identically zero as a polynomial over $\langle S\rangle$, where $\langle S\rangle$ are now treated as formal variables.

With this simpler model in hand, we immediately obtain the VBB-security of existing obfuscation constructions [30,41] based on branching programs. Those works show that the only top-level zeros that can be obtained correspond to the evaluations of branching programs. Therefore, relying on the Branching Program Un-Annihilatability Assumption (BPUA) of [37], we find that it is impossible to find an annihilating polynomial R, and hence a polynomial Q_i. This gives security in our CLT13 model.

New Multilinear Maps. We now turn to developing a new multilinear map scheme that we can prove secure in our weak model for CLT13. Guided by our annihilation analysis, we design the scheme to only release encodings for which the successful zero-test polynomials cannot be annihilated by polynomial-size circuits; our model conversion theorem shows that such encodings will be secure in the weak CLT13 model.

In this work, we focus on building an asymmetric scheme, where levels are subsets of $\{1, \ldots, d\}$, and elements can only be multiplied if they belong to disjoint levels. It is straightforward to extend to symmetric multilinear maps. The scheme will be based heavily on obfuscation techniques, plus some new techniques that we develop; however, we will not build a full obfuscation scheme. Therefore, we expect that our multilinear maps will be much more efficient than those that can be built using obfuscation.

Our starting point is Garg, Gentry, Halevi, and Zhandry [7][5], which offered a potential fix to block zeroizing attacks that we call the GGHZ16 fix. The fix was quickly broken, but we show how to further develop the idea into a complete fix. Garg et al. define a level-i "meta-encoding" of x to be a matrix of level-i CLT13 encodings, obtained by encoding component-wise matrices of the form:

$$R \cdot \begin{pmatrix} x & 0 & \dots & 0 \\ 0 & \$ & \cdots & \$ \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \$ & \cdots & \$ \end{pmatrix} \cdot R^{-1}$$

where \$ represent plaintexts drawn at random[6], and R is a random matrix of plaintext elements.

Such meta-encodings can be added and multiplied just like CLT13 encodings, since the matrices R cancel out. However, due to the R matrices, it is no longer possible to isolate the upper-left corner to perform a zero-test on x. Instead, also handed out are "bookend" vectors s, t which encode the plaintext vectors

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix} \cdot R^{-1} \text{ and } R \cdot \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix}^T \quad ,$$

respectively. Now by multiplying a meta-encoding by the bookend vectors on the left and right, one obtains a CLT13 encoding of the plaintext x, which can then be zero-tested.

Next, Garg et al. include with the public parameters meta-encodings of various powers of 2, as well as many meta-encodings of 0. Powers of 2 allow for anyone to encode arbitrary elements, and the encodings of 0 allow for re-randomizing encodings. Unfortunately, as shown in [27], this fix does not actually protect against zeroizing attacks: with a bit more work, the meta-encodings of 0 can be used just like regular encodings of zero in the attacks to break the scheme.

To help motivate our new scheme, think of the GGHZ16 fix as follows: arrange the matrices in a grid where the columns of the grid correspond to the levels, and the matrices for level i are listed out in column i in an arbitrary order. We will call a "monomial" the product of one meta-encoding from each level, in level order (e.g. the level-1 encoding comes first, then level-2, etc.). Such monomials correspond to an iterated matrix product that selects one matrix from each column. We re-interpret these monomials as evaluations of a certain branching program. In this branching program, there are t inputs, and each input is not a bit, but a digit from 0 to $k-1$ where k is the number of matrices in each

[5] We actually need the version of [7] dated November 12, 2014 from https://eprint.iacr.org/eprint-bin/versions.pl?entry=2014/666. More recent versions and the proceedings version removed the CLT13 fix that we start from.

[6] Actually, in [7], some of the zeros are also set to be random elements, but the above form suffices for our discussion and more naturally leads to our construction.

column. Each input digit selects the matrix from the corresponding column, and the result of the computation is the result of the corresponding iterated matrix product.

Note that this branching program is *read-once*, and this is fundamentally why the fix does not succeed. One way to see this is through the lens of our model conversion theorem: a read-once branching program can be annihilated, in the sense that it is possible to construct a set of inputs and an annihilating polynomial Q such that Q always evaluates to zero on the set of branching program outputs. For example, one can partition the input bits into two sets, and select subsets S and T of partial inputs from each half of the input partition. Evaluate the branching program on all points in the combinatorial rectangle defined by S, T, and arrange as a matrix. The rank of this matrix is at most the width of the branching program. Therefore, as long as the number of partial inputs is larger than the width, this branching program will be annihilated by the determinant. This is true for arbitrary branching programs, not just the branching programs derived above.

One possible way to block the attack above is to make the branching program so wide that even if the adversary queries on the entire domain, the matrix obtained above is still full rank. While it is possible to do this to build a *constant degree* multilinear map over CLT13, the map will be of little use. Roughly, the reason is that the branching program is now so wide that adding a random subset-sum of zero encodings is insufficient to fully re-randomize.

Instead, we turn to Garg et al. [37]'s obfuscator, which blocks this annihilating attack for obfuscation by explicitly requiring the branching program being obfuscated to read each input many times. By reading each input multiple times, the rank of the matrix above grows exponentially in the number of reads, blocking determinant-style attacks. Moreover, under the assumption that there are PRFs that can be computed by branching programs, such read-many programs cannot be annihilated in general. Garg et al. therefore conjecture a branching program un-annihilatability assumption, which says that read-many branching programs cannot be annihilated. Under this assumption, Garg et al. prove security in the weak GGH13 model.

Inspired by this interpretation and by techniques used to prove security of obfuscation, we modify GGHZ16 to correspond to a read-many branching program. This will allow us to block determinant-style attacks without increasing the width, allowing for re-randomization. Toward that end, we associate each meta-level i with ℓ different CLT13 levels, interleaving the levels for different i. This means that for a d-level meta-multilinear map, we will need $d\ell + 2$ CLT13 levels (the extra 2 levels for the bookends). An encoding at level i will be a sequence of ℓ different matrices of encodings, where the ℓ matrices are encoded at the ℓ corresponding CLT13 levels. The matrices (in the 3×3 case) have the form:

$$R_{i-1}\begin{pmatrix} x & 0 & 0 \\ 0 & \$ & \$ \\ 0 & \$ & \$ \end{pmatrix}R_i^{-1}, R_{d+i-1}\begin{pmatrix} 0 & 0 & 0 \\ 0 & \$ & \$ \\ 0 & \$ & \$ \end{pmatrix}R_{d+i}^{-1}, \cdots, R_{(\ell-1)d+i-1}\begin{pmatrix} 0 & 0 & 0 \\ 0 & \$ & \$ \\ 0 & \$ & \$ \end{pmatrix}R_{(\ell-1)d+i}^{-1}$$

Essentially, each of our meta-encodings is a list of GGHZ16 meta-encodings, where the first meta-encoding encodes x, and the rest encode 0. Our bookend vectors have the form $\left(1\ \$\ \cdots\ \$\right) \cdot R_0^{-1}$ and $R_{d\ell} \cdot \left(1\ \$\ \cdots\ \$\right)^T$ and are encoded, respectively, in the two remaining CLT13 levels. Unlike GGHZ16, we will not have the \$ terms be random, but instead chosen more carefully (details below). Note that we choose different randomizing matrices R in each position; this corresponds to the randomizing matrices used for Kilian [43] randomization of branching programs in obfuscation. Such randomization forces matrices to be multiplied in order as in a branching program.

Addition is component-wise. For this discussion, we will only allow a pairing operation that goes directly to the top level; this is the kind of multilinear map envisioned by [2]. We explain how to give intermediate levels below.

The pairing operation takes one meta-encoding for each meta-level, and arranges all the matrices in branching-program order. Then, roughly, it multiplies the matrices together, along with the bookends. The result is a single top-level CLT13 encoding, which can be zero-tested as in CLT13. We have to slightly tweak the procedure scheme for this to work, as multiplying all the matrices of a top-level encoding will always give an encoding of zero, owing to most of the GGHZ16 meta-encodings containing 0. Instead, we add an offset vector midway through the pairing operation to make the CLT13 encoding an encoding of the correct value; see Sect. 4 for details. For the purposes of this discussion, however, this tweak can be ignored.

The good news is that if one restricts to adding and pairing encodings as described, this blocks the zeroizing attack on GGH16 meta-encodings, assuming ℓ is large enough. We would now like to prove our scheme is actually secure, using the branching program un-annihilatability assumption as was done in obfuscation. Unfortunately, there are several difficulties here:

- First, we need to force the adversary to follow the prescribed pairing procedure. While the pairing operation is basically just a branching program evaluation, this ends up being quite different than in the setting of obfuscation. For example, in obfuscation, forcing input consistency can be done with the level structure of the underlying multilinear map. In our case, this appears impossible. The reason is that we want to be able to add two encodings at the same meta-level before pairing, meaning the underlying encodings must be at the same CLT13 level. In obfuscation, the different encodings for a particular input are encoded at different levels. There are other ways to force input consistency [6,44], but they appear to run into similar problems.
- Second, the ability to add encodings means we cannot quite interpret allowed operations as just evaluations of a branching program. For example, if one adds two meta-encodings, and then multiplies them by a third, the result is a linear combination of iterated matrix products containing *cross terms* of the branching program that mix inputs. This is a result of the degree of zero-testing being non-linear. Therefore, prior means of forcing input consistency will be too restrictive for our needs.

We overcome these issues by developing several new techniques:

- First, we prove a generalization of a lemma by Badrinarayanan et al. [30] which tightly characterizes the types of iterated matrix products that an adversary is allowed to create. Our lemma works in far more general settings so as to be applicable to our scheme.
- Second, we re-interpret the allowed operations not as branching program evaluations, but as *vector-input* branching program evaluations, a new notion we define. In a vector-input branching program, inputs are no longer digits, but a list of vectors. The vectors specify a linear combination. To evaluate, for each column apply the corresponding linear combination, and then multiply all the results together. By being able to take linear combinations of the input matrices, we now capture the ability of an adversary to add encodings.
- Finally, we introduce new "enforcing" matrices that we place in the $ entries. The goal of our enforcing matrices is to force the adversary's operations to correspond to vector-input branching program evaluations.

Using our enforcing matrices and our new analysis techniques, we show that the adversary is limited to producing linear combinations of vector-input branching program evaluations. Therefore, by our model conversion theorem, if the adversary can attack in our weak CLT13 model, it can find an annihilating polynomial for vector-input branching programs. We therefore formulate a concrete conjecture that, like regular branching programs, vector-input branching programs cannot be annihilated. Under this assumption, no zeroizing attacks exist on our scheme.

Discussion. We now discuss some limitations of our construction above.

- We do not know how to justify our vector-input branching program assumption based on PRFs, unlike the corresponding assumption for standard branching programs. The reasons are twofold:

  - Most importantly, we do not know of any PRFs that can be evaluated by vector-input branching programs.
  - In our analysis, the adversary can produce a linear combination of *exponentially many* vector-input branching program evaluations. Therefore, an annihilating polynomial annihilates exponentially-many inputs, and therefore would not correspond to a polynomial-time attack on a PRF, even if one were computable by vector-input branching programs. We note that, if we were to ignore the first issue, it is straightforward to overcome the second issue using a sub-exponentially secure PRF, since a sub-exponential time algorithm can potentially query a PRF on the entire domain and construct exponential-sized linear combinations. Furthermore, we hope that this second limitation arises from our analysis and is not a fundamental problem, leaving room for subsequent work.

  We observe that if a sub-exponentially secure PRF can be computed by vector-input branching programs, it should be possible to prove our assumption based on the security of said PRF.

Even without a justification based on general hardness assumptions, the only efficient annihilating polynomials we could find for vector-input branching programs are determinant polynomials as described above. These are interestingly also the only annihilating polynomials we know of for plain branching programs. Therefore, it seems reasonable at this time to conjecture that determinants are the only annihilating polynomials. If this conjecture holds, then any annihilating polynomial will require circuits of size roughly w^{ℓ}, where w is the width of matrices in the branching program. By setting w^{ℓ} to be 2^{λ} for desired security parameter λ, this will block known attacks. In the full version of this work [1], we give some evidence for why this conjecture should hold in restricted settings.

- Our discussion above only allows for directly pairing to the top level. If we are willing to sacrifice polynomial degree for constant degree, we can define pairing operations for intermediate levels. Adjacent levels (say 1,2) can easily be paired by simply constructing ℓ matrices that are the pairwise products of the ℓ matrices in the two levels. Due to the different Kilian randomization matrices between each pair of levels, we cannot directly pair non-adjacent levels, such as 1 and 3. For non-adjacent levels, instead of matrix multiplications, we can *tensor* the encodings, generating all degree 2 monomials. This tensoring can also be extended to higher levels. Unfortunately, this greatly expands the size of encodings and thus can only be done when the degree is constant.

  Alternatively, we note that the ability to only multiply adjacent levels corresponds to the "graph induced" multilinear map notion [20] for the line graph. Hence, we obtain a multilinear map for general line graphs. Such maps are sufficient for most applications. Moreover, such graphs can easily be used to build symmetric multilinear maps, by simply encoding at all possible singleton levels.
- Finally, it is not possible to multiply an encoding by scalar. One could try repeated doubling, but our scheme inherits the noisiness of CLT13, and this repeated doubling will cause the noise to increase too much. One potential solution is that an encoding of x actually consists of encodings of $x, 2x, 4x, 8x$, etc. Now instead of repeated doubling, multiplying by a scalar is just a subset sum. Of course, this operation "eats up" the powers of 2, so it can only be done a few times before one runs out of encodings.

  Another potential option, depending on the application, is to introduce additional "dummy" levels. To multiply by a scalar, first encode it in the "dummy" level, and then pair with the element. This of course changes the level at which the element is encoded, but for some applications this is sufficient. Below, we show how to use this idea to give a multiparty NIKE protocol for a polynomial number of users.

Multiparty Non-interactive Key Exchange (NIKE). Here, we very briefly describe how to use our new multilinear maps to construct multiparty NIKE. The basic scheme shown by Boneh and Silverberg [2] will not work because (1) they need an symmetric multilinear map, and (2) they need to be able to multiply encodings by ring elements. We show how to tweak the scheme to work with an asymmetric map that does not allow multiplying encodings by ring elements. For d users, instantiate our scheme with d levels, one more than is needed by [2].

User i chooses a random ring element a_i, and then computes encodings $[a_i]_u$ of a_i at every singleton level u. User i publishes all the encodings (after re-randomization), *except* the encoding at level 1.

Upon receiving the encodings from all other users, user i arbitrarily assigns each of the other $d-1$ users to the levels $2, \ldots, d$. Let u_j be the level assigned to user j. Then it pairs its private elements $[a_i]_1$ together with $[a_j]_{u_j}$ for each $j \neq i$. The result is an encoding of $\prod_j a_j$ at the top level. Everyone computes the same encoding, which can be extracted to get the shared secret key.

Meanwhile, an adversary, who never sees an encoding of a_i at level 1, cannot possibly construct an encoding of $\prod_j a_j$ without using the same level twice. Using a variant of the multilinear Diffie-Hellman assumption, this scheme can be proven secure. This assumption can be justified in the generic multilinear map model, and hence our scheme can be proven secure in the weak CLT13 model.

Concurrent Work: A Weak Model for GGH15. In a concurrent work, Bartusek, Guan, Ma, and Zhandry [45] propose a weak multilinear map model for the GGH15 maps [20]. They demonstrate that all known zeroizing attacks on the GGH15 construction are captured by their weak model, and they construct an obfuscation scheme that is provably secure in their weak model. We compare and contrast our models and results below.

- The CLT13 and GGH15 schemes are quite different, and the respective zeroizing attacks exploit different vulnerabilities. The security of CLT13 crucially depends on the secrecy of the primes p_i, for which there is no analogue in the GGH15 scheme. Thus, our weak model captures the adversary's ability to perform a certain step that all known attacks on CLT13 go through in order to recover the p_i's. The Bartusek et al. [45] weak model uses a different condition that captures an adversary's ability to learn non-trivial information about an encoded plaintext.
- Bartusek et al. [46] prove security against a slightly larger class of "arithmetic adversaries," initially considered by Miles, Sahai, and Weiss [47]. To achieve obfuscation secure against such adversaries, Bartusek et al. [45] rely on an additional p-Bounded Speedup Hypothesis of Miles et al. [47] (a strengthening of the Exponential Time Hypothesis).
- Our work proposes a candidate "fix" for the CLT13 multilinear maps that enables a direct Non-Interactive Key Exchange (NIKE) construction secure against zeroizing attacks under a new VBPUA assumption. Bartusek et al. do not propose a corresponding fix for the GGH15 maps.

- All of the constructions in this paper are trivially broken by quantum attacks that factor the public CLT13 modulus N.[7] In contrast, all currently known quantum attacks on GGH15 [49] fall under the class of zeroizing attacks, and as a result the obfuscation construction of Bartusek et al. resists all known classical and quantum attacks.

2 Preliminaries

2.1 Multilinear Maps and the Generic Model

A multilinear map (also known as a graded encoding scheme) with universe set $\mathbb{U}$ and a plaintext ring $\mathcal{R}_{\mathsf{ptxt}}$ supports encodings of plaintext elements in $\mathcal{R}_{\mathsf{ptxt}}$ at levels corresponding to subsets of $\mathbb{U}$. A plaintext element a encoded at $S \subseteq \mathbb{U}$ is denoted as $[a]_S$. Multilinear maps support some subset of the following operations on these encodings:

- (Encoding) Given an element $a \in \mathcal{R}_{\mathsf{ptxt}}$ and level set $S \in \mathbb{U}$, output $[a]_S$.
- (Addition) Two encodings at the same level $S \subseteq \mathbb{U}$ can be added / subtracted. Informally, $[a_1]_S \pm [a_2]_S = [a_1 \pm a_2]_S$.
- (Multiplication) An encoding at level $S_1 \subseteq \mathbb{U}$ can be multiplied with an encoding at level $S_2 \subseteq \mathbb{U}$, provided $S_1 \cap S_2 = \emptyset$. The product is an encoding at level $S_1 \cup S_2$. Informally: $[a_1]_{S_1} \cdot [a_2]_{S_2} = [a_1 \cdot a_2]_{S_1 \cup S_2}$.
- (Re-randomization) We will allow for schemes with non-unique encodings. In this case, we may want a re-randomization procedure, which takes as input an encoding of a potentially unknown element a, and outputs a "fresh" encoding of a, distributed statistically close to a direct encoding of a.
- (Zero-Testing) An encoding $[a]_{\mathbb{U}}$ at level $\mathbb{U}$ can be tested for whether $a = 0$.
- (Extraction) An encoding $[a]_{\mathbb{U}}$ at level $\mathbb{U}$ can be extracted, obtaining a string r. Different encodings of the same a must yield the same r.

Most multilinear map schemes, due to security vulnerabilities, only support addition, multiplication, and zero-testing/extraction, but do not support public re-randomization or encoding. Instead, encoding must be performed by a secret key holder.

2.2 Overview of the CLT13 Multilinear Maps

We give a brief overview of the CLT13 multilinear maps, adapted from text in [50]. For a full description of the scheme, see [19]. The CLT13 scheme relies on the Chinese Remainder Theorem (CRT) representation. For large secret primes p_k, let $N = \prod_{k=1}^{n} p_k$. Let $\mathsf{CRT}(s_1, s_2, \ldots, s_n)$ or $\mathsf{CRT}(s_k)_k$ denote the number $s \in \mathbb{Z}_N$ such that $s \equiv s_k \pmod{p_k}$ for all $k \in [n]$. The plaintext space of the CLT13 scheme is $\mathbb{Z}_{g_1} \times \mathbb{Z}_{g_2} \times \cdots \times \mathbb{Z}_{g_n}$ for small secret primes g_k. An encoding of a vector $m = (m_1, \ldots, m_n)$ at level set $S = \{i_0\}$ is an integer $\alpha \in \mathbb{Z}_N$ such

[7] Our work leaves open the problem of devising composite-order multilinear maps whose security does not rely on the hardness of factoring [48].

that $\alpha = \mathsf{CRT}(m_1 + g_1 r_1, \ldots, m_n + g_n r_n)/z_{i_0} \pmod N$ for small integers r_k, and where z_{i_0} is a secret mask in $\mathbb{Z}_N$ uniformly chosen during the parameters generation procedure of the multilinear map. To support κ-level multilinearity, κ distinct z_i's are used.

Additions between encodings in the same level set can be done by modular additions in $\mathbb{Z}_N$. Multiplication between encodings can be done by modular multiplication in $\mathbb{Z}_N$, only when those encodings are in disjoint level sets, and the resulting encoding level set is the union of the input level sets. At the top level set $[\kappa]$, an encoding can be tested for zero by multiplying it by the zero-test parameter $p_{zt} = \sum_{k=1}^{n} p_k^* h_k ((\prod_{i \in [\kappa]} z_i) g_k^{-1} \bmod p_k) \pmod N$ in $\mathbb{Z}_N$ where $p_k^* = N/p_k$, and comparing the result to N. An encoding α can be expressed as $\frac{1}{\prod_{i=1}^{\kappa} z_i} \mathsf{CRT}(s_k)_k$ where s_k denotes the numerator of its kth CRT component. When α is an encoding of zero, it can be shown that

$$p_{zt}\alpha \pmod N = \sum_{k=1}^{n} \gamma_k s_k,$$

where $\gamma_k = p_k^* h_k g_k^{-1}$ are "smallish" global secret parameters that depend on the other CLT13 parameters. For encodings of zero, each s_k is "small", so $p_{zt}\alpha \pmod N$ is small relative to N. If α does not encode 0, then one heuristically expects $p_{zt}\alpha \pmod N$ to be large relative to N. Thus, we can zero test by determining if this quantity is small.[8] We can also extract a unique representation of any encoded element by computing $p_{zt}\alpha \pmod N$, and rounding appropriately.

2.3 Vector-input Branching Programs

We generalize matrix branching programs to vector-input branching programs, a new notion we define (for the formal definition of matrix branching programs we consider, see the full version [1]). Single-input branching programs consist of a sequence of pairs of matrices, where an input bit is read for each pair to select one of them. Our vector input generalization allows for selecting a *linear combination* of matrices. In addition, we replace pairs of matrices with sets of k matrices. Thus, a program input is a d-tuple of vectors of dimension k, each consisting of non-negative integers. We can stack these d column vectors into a matrix $(\mathbb{Z})^{k \times d}$, so for any input $x \in (\mathbb{Z})^{k \times d}$, $x_{i,j} \in \mathbb{Z}$ denotes the jth component of the ith vector.

These vector-input branching programs arise in the security proof of our new multilinear map construction in Sect. 5. Thus, we will only consider a restricted

[8] In the full CLT13 scheme, there is a vector of zero-testing elements created in a way to prove that the result is large for non-zero encodings. However, in practice this is far less efficient, so most implementations only use a single zero test vector as described here (see e.g. [38]). We stress that giving out fewer zero-testing parameters can only make the scheme more secure, and parameters can be set so that correctness of zero-testing still holds with overwhelming probability.

class of vector-input branching programs tailored to fit the requirements of our analysis. Specifically, we use read-ℓ programs, meaning that each vector in the input is read exactly ℓ times. These programs are single-input, so there are $d\ell$ sets of matrices. Furthermore, the input selection is fixed so that the ith input vector is read for matrix sets $i, d+i, \ldots, (\ell-1)d+i$. In other words, the input selection function is simply $\mathsf{inp}(j) = j \pmod{d}$.

Definition 1. *A read-ℓ vector-input branching program over a ring $\mathcal{R}$ with input length n, vector dimension k, and matrix width w is given by a sequence*

$$VBP = \big(s, t, \{\boldsymbol{B}_{i,j}\}_{i\in[d\ell], j\in[k]}\big)$$

where each $\boldsymbol{B}_{i,j}$ is a $w \times w$ matrix, s is a w-dimensional row vector, and t is a w-dimensional column vector. All entries are elements in $\mathcal{R}$. Then $VBP : (\mathbb{Z})^{k\times d} \to \mathcal{R}$ is computed as

$$VBP(x) = s \cdot \left(\prod_{i=1}^{d\ell} \left(\sum_{j=1}^{k} x_{i(mod\ d),j} \boldsymbol{B}_{i,j} \right) \right) \cdot t.$$

For further intuition and examples of vector-input branching programs, refer to the full version [1].

We remark that vector-input branching programs are reminiscent of arithmetic branching programs [51,52], where an input string specifies a set of accepting s-t paths in a weighted directed acyclic graph, and the output is a sum of the products of all edge weights on each accepting path. With some care, we can re-express vector-input branching programs as a certain type of arithmetic branching program. However, we use the vector-input formulation as it intuitively captures the structure of our multilinear map construction in Sect. 4.

2.4 Kilian Randomization of Matrix Sequences

Consider a collection of n columns of matrices, where each column may contain an arbitrary polynomial number of matrices. Denote the jth matrix in column i as $A_{i,j}$. Suppose the matrices within each column have the same dimensions, and across columns have compatible dimensions so that matrices in adjacent columns can be multiplied together, and multiplying one matrix from each column results in a scalar. Kilian [43] describes a method to partially randomize such branching programs. Randomly sample invertible square matrices R_i. Then matrix $A_{i,j}$ is left-multiplied by R_i^{-1} and right-multiplied by R_{i+1}. When performing an iterated matrix product selecting one matrix from each column, the R_i and R_i^{-1} cancel out, so the product is unchanged by this randomization.

3 The Model

In this section, we define two models. The first is the weak CLT13 model, intended to capture all known classical attacks on the CLT13 multilinear maps.

The second is the CLT13 annihilation model, a modification of the weak CLT13 model with different winning conditions. We justify our first model by demonstrating that it captures known attacks from the literature. The main theorem of this section is that an adversary in the weak CLT13 model implies the existence of an adversary in the CLT13 annihilation model. Combining this theorem with the Branching Program Un-Annihilatability Assumption of [37], we immediately obtain virtual black box (VBB) security of the obfuscator of Badrinarayanan et al. [30]. Additionally, this shows that the order-revealing encryption scheme of Boneh et al. [53] is secure in our model.

3.1 CLT13 Weak Multilinear Map Model

Notation. We will let uppercase letters such as M, S, Γ denote formal variables, and lower case letters such as m, s, γ denote actual values. Bold letters will be used to distinguish vectors from scalars. Let m_{ji} be a set of elements indexed by j and i. We introduce $\langle \mathbf{m} \rangle_i$ as shorthand for the set $\{m_{ji}\}_j$ of all elements with index i, and $\langle \mathbf{m} \rangle$ to denote the set $\{\langle \mathbf{m} \rangle_i\}_i$. For a set $M_{j,i}$ of formal variables indexed by j and i, define $\langle \mathbf{M} \rangle_i$ and $\langle \mathbf{M} \rangle$ analogously.

We now define our weak CLT13 model, with the following interfaces:

Initialize Parameters. At the beginning of the interaction with the model $\mathcal{M}$, $\mathcal{M}$ is initialized with the security parameter λ and the multilinearity parameter $\kappa \leq \mathsf{poly}(\lambda)$. We generate the necessary parameters of the CLT13 scheme (including the vector dimension n, the primes g_i, p_i for $i \in [n]$) according to the distributions suggested by Coron et al. [19]. Let $\mathcal{R}_{\mathsf{ptxt}} = \mathbb{Z}_{\prod_i g_i} = \otimes_i \mathbb{Z}_{g_i}$ be the plaintext ring. Let $\mathcal{R}_{\mathsf{ctxt}} = \mathbb{Z}_{\prod_i p_i} = \otimes_i \mathbb{Z}_{p_i}$. We will usually interpret elements in $\mathcal{R}_{\mathsf{ptxt}}$ and $\mathcal{R}_{\mathsf{ctxt}}$ as vectors of their Chinese Remaindering components.

Initialize Elements. Next, $\mathcal{M}$ is given a number of plaintext vectors $\mathbf{m}_j \in R$ as well as an encoding level S_j for each plaintext. $\mathcal{M}$ generates the CLT13 numerators $\mathbf{s}_j$ where $s_{ji} = m_{ji} + g_i r_{ji}$ as in the CLT13 encoding procedure. For each j, $\mathcal{M}$ stores the tuple $(\mathbf{m}_j, \mathbf{s}_j, S_j)$ in the a pre-zero test table.

Zero-testing. The adversary submits a polynomial p_u to $\mathcal{M}$, represented as a polynomial-size *level-respecting* algebraic circuit. Here, level-respecting means that all wires are associated with a level S, input wires are associated to the sets S_j, add gates must add wires with the same level and output a wire with the same level, multiply gates must multiply wires with sets $S_0 \cap S_1 = \emptyset$ and output a wire with the level $S_0 \cup S_1$, and the final output wire must have set $\{1, \ldots, \kappa\}$.

Next, $\mathcal{M}$ checks whether $p_u(\langle \mathbf{m} \rangle_i) = 0$ for all i. If the check fails for any i, $\mathcal{M}$ returns "fail". If the check passes for all i, $\mathcal{M}$ returns "success". We assume without loss of generality that the set $\{p_u\}$ of *successful* zero tests are linearly independent as polynomials (since otherwise a zero-test on one p_u can be derived from the result of a zero-test on several other p_u).

If we stop here, we recover the plain generic multilinear map model [44]. However, in our model, a successful zero test does more. If zero testing is successful, p_u corresponds to a valid construction of a top-level zero encoding. $\mathcal{M}$ then additionally returns a handle T_u to the value $t_u(\langle\boldsymbol{\gamma}\rangle, \langle\mathbf{s}\rangle) = \sum_i \gamma_i p_u(\langle\mathbf{s}\rangle_i)$, the result of the zero-test computation. Each handle T_u along with the corresponding the zero-test result is stored in a *zero-test table.*

Post-zero-test. Finally, the adversary submits a polynomial Q on the handles $\{T_u\}_u$ and the formal variables $\langle\mathbf{S}\rangle_i$ for some $i \in [n]$ (that $\mathcal{A}$ picks). This Q must be represented by a polynomial-sized algebraic circuit, and the degree must be at most $2^{o(\lambda)}$.[9] The model looks up each handle T_u in the zero-test table and plugs in the corresponding values t_u. The model outputs "WIN" if the following two conditions are satisfied.

1. $Q(\{t_u(\langle\boldsymbol{\gamma}\rangle, \langle\mathbf{s}\rangle)\}_u, \langle\mathbf{S}\rangle_i) \not\equiv 0$ as a polynomial over the formal variables $\langle\mathbf{S}\rangle_i$.
2. $Q(\{t_u(\langle\boldsymbol{\gamma}\rangle, \langle\mathbf{s}\rangle)\}_u, \langle\mathbf{s}\rangle_i) = 0$.

Intuitively, these conditions imply that Q is a polynomial with non-zero degree over the $\langle\mathbf{S}\rangle_i$ formal variables that is "solved" when the correct values $\langle\mathbf{s}\rangle_i$ are plugged in.

Plain Annihilation Model. We define a modification of the above CLT13 weak multilinear map model, which is identical except for post-zero-test queries:

Post-zero-test. $\mathcal{A}$ submits a polynomial Q' on a set of formal variables $\{P_u\}_u$, where P_u represents the successful zero test polynomial p_u. Again, this Q' must be represented by a polynomial-sized algebraic circuit, and the degree must be at most $2^{o(\lambda)}$. The model outputs "WIN" if the following conditions are satisfied.

1. $Q'(\{P_u\}_u)$ *is not* identically zero over the $\{P_u\}_u$ formal variables.
2. $Q'(\{p_u(\langle\mathbf{S}\rangle_i)\}_u)$ *is* identically zero over the $\langle\mathbf{S}\rangle_i$ formal variables.

In other words, $\mathcal{A}$ wins if it submits a Q' that annihilates the $\{p_u\}_u$ polynomials.

3.2 Classical Attacks in the Weak CLT13 Model

We first show that the original attack on the CLT13 multilinear maps by Cheon et al. fits into this framework [21].

Mounting this attack requires that the set of plaintext vectors $\{\mathbf{m}_j\}$ given to M can be divided into three distinct sets of vectors, A, B, C that satisfy certain properties. We can discard/ignore any other plaintext vectors. For ease of exposition, we relabel the vectors in these sets as:

$$A = \{\mathbf{m}_1^A, \ldots, \mathbf{m}_n^A\} \qquad B = \{\mathbf{m}_1^B, \mathbf{m}_2^B\} \qquad C = \{\mathbf{m}_1^C, \ldots, \mathbf{m}_n^C\}$$

These vectors can be encoded at arbitrary levels, as long as for any j, σ, k, $\mathbf{m}_j^A \cdot \mathbf{m}_\sigma^B \cdot \mathbf{m}_k^C$ is a plaintext of zeros at the top level. Accordingly, $\mathcal{A}$ submits polynomials $p_{j,\sigma,k}$ for all $j, k \in [n], \sigma \in [2]$ for zero-testing where

$$p_{j,\sigma,k}(m_1^A, \ldots, m_n^A, m_1^B, m_2^B, m_1^C, \ldots, m_n^C) = m_j^A \cdot m_\sigma^B \cdot m_k^C$$

[9] We note that these restrictions are analogous to restrictions made for annihilation attacks on GGH13 [37].

Each of these polynomials clearly gives a successful zero-test. In response to each query, $\mathcal{M}$ returns a handle $T_{j,\sigma,k}$ to the value

$$t_{j,\sigma,k} = \sum_{i=1}^{n} \gamma_i s^A_{j,i} \cdot s^B_{\sigma,i} \cdot s^C_{k,i}.$$

For $\sigma \in \{1,2\}$, define W_σ to be the $n \times n$ matrix whose (j,k)th entry is $T_{j,\sigma,k}$. In the real attack, the adversary computes the matrix $W_1 W_2^{-1}$, which Cheon et al. [21] show has eigenvalues $\frac{s^B_{1,i}}{s^B_{2,i}}$. The adversary solves the characteristic polynomial of $W_1 W_2^{-1}$ for these eigenvalues. In our model $\mathcal{A}$ cannot immediately submit this characteristic polynomial, as it involves rational functions of the handles T, and can only be solved for ratios of the s_{ji} values. However, we observe that the characteristic polynomial

$$\det\left(W_1 W_2^{-1} - \lambda I\right) = \det\left(W_1 W_2^{-1} - \left(\frac{S^B_{1,i}}{S^B_{2,i}}\right) I\right) = 0$$

can be re-written by substituting $W_2^{-1} = \frac{W_2^{adj}}{\det(W_2)}$. (where W_2^{adj} denotes the adjoint matrix of W_2). Applying properties of the determinant then gives

$$\det(W_1 W_2^{adj} S^B_{2,i} - S^B_{1,i} \det(W_2) I) = 0$$

$\mathcal{A}$ submits the left-hand side expression above as its polynomial Q in a post-zero-test query. Since the Cheon et al. attack is successful, we know Q is nonzero over the formal variables $\langle \mathbf{S} \rangle_i$ after the values associated with the handles T are plugged in. Additionally, plugging in the appropriate solutions $\langle \mathbf{s} \rangle_i$ satisfies the above expression, so both win conditions are satisfied. Thus, $\mathcal{A}$ wins in our model.

In the full version of this work [1], we show how the general attack framework of Coron et al. [27] can be expressed in our model.

3.3 Model Conversion Theorem

Theorem 1. *If there exists an adversary $\mathcal{A}$ that wins with non-negligible probability in the weak CLT13 multilinear map model, there exists an adversary $\mathcal{A}'$ that wins with non-negligible probability in the CLT13 annihilation model. $\mathcal{A}'$ is the same as $\mathcal{A}$ up to and including the zero-test queries, and only differs on the post-zero test queries.*

We give a brief outline of the proof strategy (for the whole proof, refer to the full version of this paper [1]). An adversary that wins in the weak CLT13 model produces a non-trivial polynomial Q that evaluates to 0 on the actual CLT13 parameters and the numerators of the encodings. Since the CLT13 parameters and encodings are sampled using randomness hidden from the adversary, we can use a generalization of the Schwartz-Zippel lemma to conclude that the polynomial must be identically zero over its formal variables. We can view this polynomial as being over the formal variables corresponding to the CLT13 parameters, where the remaining formal variables constitute "coefficients". Since the overall polynomial is identically zero, all coefficients must also be identically zero. To conclude, we use the fact that the polynomial is non-trivial to show that there must exist a coefficient which acts as an annihilating polynomial for the zero-test polynomials.

3.4 Secure Obfuscation and Order-revealing Encryption in the Weak CLT13 Model

Security in our weak CLT13 model means security in the plain generic multilinear map model, plus the inability to construct an annihilating polynomial. We observe that Badrinarayanan et al. [30] show for their obfuscator (which is a tweak of the obfuscator of Barak et al. [41]), the only successful zero tests an adversary can perform are linear combinations of honest obfuscation evaluations on some inputs. Moreover, the linear combinations can only have polynomial support. Recall that in [30], evaluation is just a branching program evaluation over the encoded values. Therefore, any annihilating polynomial in the plain annihilation model is actually an annihilating polynomial for branching programs. Therefore, using the branching program un-annihilatability assumption of Garg et al. [37], we immediately conclude that no such annihilating polynomial is possible. Thus, there is no weak CLT13 attack on this obfuscator.[10]

We similarly observe that in the order-revealing encryption (ORE) scheme of Boneh et al. [53], any successful zero is also a linear combination of polynomially-many branching program evaluations. Therefore, by a similar argument, we immediately obtain that Boneh et al.'s scheme is secure in our weak CLT13 model.

4 A New Multilinear Map Candidate

In this section, we give a candidate polynomial-degree multilinear map scheme. We show, given an assumption about annihilating vector-input branching programs, that this multilinear map is secure in the weak CLT13 model. Here, we discuss our basic scheme; in the full version of this work, we show how to leverage the "slotted" structure of CLT13 encodings to obtain efficiency improvements [1].

4.1 Construction Overview

The levels will be non-empty subsets of $[d]$ for some polynomial d. For simplicity, here we describe how to build a multilinear map that only allows a pairing

[10] The Garg et al. obfuscator is defined as a dual-input obfuscator, which is the version we consider. The dual-input requirement is crucial; a single-input variant of this obfuscator is insecure and was attacked by Coron et al. [54]. The key point is that the branching program un-annihilatability assumption only holds for branching programs with significant interleaving of input bits, which can be ensured by a dual-input requirement.

operation that takes d elements, one from each singleton set, directly to a top-level encoding. This is the style of multilinear map envisioned by Boneh and Silverberg [2].

Our construction is logically organized into $d\ell$ columns, numbered from 1 to $d\ell$. The columns are further partitioned into d groups numbered 1 through d of ℓ columns, where the columns in each group are interleaved: group u consists of columns $u, u+d, u+2d, \ldots, u+(\ell-1)d$. Each column will correspond to one level of the underlying CLT13 maps, and each group of columns will correspond to one meta-level of our scheme. We set the plaintext space of both the underlying CLT13 scheme and our scheme to be $\mathcal{R}_{\mathsf{ptxt}} = \mathbb{Z}_M$ where $M = \prod_i g_i$. Recall that in the CLT13 scheme, M is not public.

We first describe the format of a meta-encoding in our scheme. An encoding at singleton level u will consist of ℓ matrices of CLT13 encodings, one in each of the columns corresponding to column group u. We will denote by $A_i^{(u)}$ the ith matrix in the encoding for level u. To construct $A_i^{(u)}$, we first define the diagonal matrix $\widetilde{A_i^{(u)}}$, of the form

$$\mathrm{diag}(m_i, v_i, w_i, \xi_1 I, \ldots, \xi_{u-1} I, E_i^{(u)}, \xi_{u+1} I, \ldots, \xi_d I).$$

These components of the diagonal matrix work as follows:

- m_i is a plaintext element used for the actual plaintext encoding.
- v_i and w_i are freshly sampled uniformly random elements from the plaintext space $\mathbb{Z}_M$. Their sole purpose is to enforce a requirement called non-shortcutting, which will arise in the security proof. They are canceled out in valid products by 0's in the bookend vectors, defined later.
- The remainder of the diagonal consists of d block matrices, where $d-1$ blocks are essentially unused and set to random multiples of the identity, while the uth matrix is set to be an "enforcing matrix" $E_i^{(u)}$. Note that i corresponds to this being the ith matrix for encoding u. The purpose of these matrices is roughly to prevent an adversary from arbitrarily mixing and matching the matrices from different encodings. We defer the details of these matrices to Sect. 4.2.

Next, $d\ell+1$ Kilian randomization matrices R_i are generated [43] to left and right multiply each of the $d\ell$ columns. All encodings will share the same Kilian matrices. Each $\widetilde{A_i^{(u)}}$ matrix at meta-level u is left- and right- multiplied by the appropriate Kilian matrices, giving

$$R_{u+(i-1)d-1}^{-1} \widetilde{A_i^{(u)}} R_{u+(i-1)d}.$$

Each element of this Kilian-randomized matrix is encoded in an asymmetric CLT13 multilinear map at level $\{u+(i-1)d\}_{CLT13}$ (we differentiate levels of the underlying CLT13 map with this subscript, to avoid confusion with the levels of our multilinear map) corresponding to the column it belongs to. The resulting matrix of CLT13 encodings is taken to be $A_i^{(u)}$.

For a matrix $A_i^{(u)}$, we refer to the underlying ring element m_i as the *matrix plaintext*. This is the only component of the matrix used for encoding actual plaintext elements. Therefore, as described so far, every meta-encoding in our scheme encodes a length-ℓ vector of ring elements.

We also give out "bookends" s, t, which are CLT13 encodings of the vectors

$$\hat{s} = (1, 1, 0, F_1, \ldots, F_d) \cdot R_0 \ , \ \hat{t} = R_{d\ell}^{-1} \cdot (1, 0, 1, G_1, \ldots, G_d)^T \ .$$

For the sake of clarity, we defer discussing the F, G vectors until Sect. 4.2. The 1 in the first position is used to extract the matrix plaintexts. The 0 in the second position of $\hat{t}$ will zero-out the v_i terms, while the 0 in the third position of $\hat{s}$ will zero-out the w_i terms. s is encoded at CLT13 level 0, while t is encoded at level $d\ell + 1$.

A meta-encoding of $x \in \mathcal{R}_{\mathsf{ptxt}} = \mathbb{Z}_M$ at singleton level $\{u\}$ is simply a sequence of matrices $(A_i^{(u)})_{i \in [\ell]}$ whose corresponding sequence of matrix plaintexts is $(x, 0, 0, \ldots, 0)$.

At instance generation, we generate and publish a set of initial public encodings.

- For each singleton level $\{u\} \subseteq [d]$, we publish encodings of $1, 2, 4, \ldots, 2^{\rho-1}$, where ρ is specified later.
- For each singleton level $\{u\} \subseteq [d]$, we publish τ encodings of zero, where τ is specified later.
- For the top level $[d]$, we publish a special *pre-zero-test encoding* that will have most of the structure of a valid top level encoding, except that it will not correctly encode an actual plaintext element. Its sequence of matrix plaintexts will be $(0, 1, 1, \ldots, 1)$, which differs from a normal encoding where the matrix plaintexts are all 0 after the first slot. The sole purpose of this encoding is to be added to any top level encoding we seek to zero test. Roughly, the element submitted for zero testing is the product of an encoding's matrix plaintexts, and without this step the product would always be zero.

To add/subtract two meta-encodings at the same singleton level $\{u\}$, which are two sequences of ℓ matrices, we line up the sequences of matrices and add/subtract the corresponding matrices component-wise. The resulting sequence of ℓ matrices is taken as the encoding of the sum. Intuitively, this works because adding these matrices also adds the sequence of matrix plaintexts. As we show in Sect. 4.2, the structure of the enforcing matrices is also preserved. If the input encodings have matrix plaintexts $(x_1, 0, \ldots, 0)$ and $(x_2, 0, \ldots, 0)$, the result of addition/subtraction has matrix plaintexts $(x_1 \pm x_2, 0, \ldots, 0)$.

To pair d meta-encodings, one from each singleton level, we do the following. For each $i \in [\ell]$, we line up the ith matrix from each encoding, in the order specified by the columns of our scheme, and multiply the matrices together. The resulting i matrices are then added to the corresponding i matrices from the pre-zero-test encoding. Based on the structure of our encoding scheme, the resulting i matrices have the matrix plaintext sequence $(\prod_u x_u, 1, \ldots, 1)$, where x_u was the value encoded at level $\{u\}$. Finally, we multiply all of these matrices together. The resulting matrix will have $\prod_u x_u$ in the upper-left corner. Finally, we multiply by the bookends s, t to obtain a single top-level CLT13 encoding. We set up the enforcing matrices in Sect. 4.2 to guarantee that this product becomes a CLT13 encoding of $\prod_u x_u$.

The remaining procedures work as follows.

Encode. To encode a plaintext $x \in \mathbb{Z}_{2^\rho}$ at a singleton level $\{u\}$, write the plaintext in base 2, and then sum the appropriate public encodings of powers of 2. We note that we do not publish a description of the plaintext ring $\mathcal{R}_{\mathsf{ptxt}} = \mathbb{Z}_M$, where $M = \prod_i g_i$. Therefore, the input to the encoding procedure is some integer $x \in \mathbb{Z}_{2^\rho}$, and the output is an encoding of $x \bmod M$, where $\mathcal{R}_{\mathsf{ptxt}} = \mathbb{Z}_M$. We set $\rho = M \times 2^\lambda$ for a security parameter λ so that a random $x \in \mathbb{Z}_{2^\rho}$ yields an element $x \bmod M$ that is statistically close to random in $\mathcal{R}_{\mathsf{ptxt}}$.

Re-randomize. To re-randomize this encoding, add a random subset sum of the public encodings of zero available for the level. We choose the parameter τ, roughly, to be large enough so that the result is statistically close to a fresh random encoding. For further discussion, see the full version [1].

Zero-test and Extract. Zero testing and extraction on top-level encodings (which are just top-level CLT13 encodings) are performed exactly as in CLT13.

4.2 Enforcing Matrix Structure

We now describe the enforcing matrix structure used in the matrices of our scheme. Consider the ℓ matrices associated to an encoding at any singleton level $\{u\}$, which all have a block diagonal form. For each matrix, all the diagonal entries except the top left three entries are responsible for providing the enforcing structure. As described in Sect. 4.1, the rest of the diagonal entries are divided into d equally-sized diagonal matrices. The uth block is set to $E_i^{(u)}$, which provides the enforcing structure for level $\{u\}$, while the other $d-1$ blocks are set to random multiples of the identity to avoid interfering with the enforcing structure of the other singleton levels.

To construct $E_i^{(u)}$ for a new encoding, we sample a random vector α of dimension ℓ. Denote the ith component as α_i. The matrix $E_i^{(u)}$ is set to be the following diagonal matrix of width $2(\ell-1)$:

$$E_i^{(u)} = \mathrm{diag}(\alpha_i, \alpha_{\sigma_{(12)}(i)}, \alpha_i, \alpha_{\sigma_{(23)}(i)}, \dots, \alpha_i, \alpha_{\sigma_{(\ell-1,\ell)}(i)})$$

Here, $\sigma_{(ab)}$ denotes the transposition swapping a and b. We additionally have two bookend vectors F, G, used by all enforcing matrices for a particular singleton level. The left bookend vector F is simply the all 1's row vector of dimension $2(\ell-1)$. The right bookend vector G is a column vector of dimension $2(\ell-1)$. To set the entries of G, we sample $\ell-1$ random values $\{\eta_i\}_{i\in[\ell-1]}$, and set the entry in position $2i-1$ to η_i, and the entry in position $2i$ to $-\eta_i$ for all $i \in [\ell-1]$.

Restrictions on Matrix Products. The sole purpose of the enforcing matrices is to ensure that the adversary respects the meta-encoding structure. For example, since each meta-encoding consists of ℓ separate matrices, an adversary may try to swap some of these matrices for matrices from other meta-encodings. We show that any attempt to do so will inevitably lead to a useless top-level encoding of a random plaintext.

Consider a setting where the adversary has access to dk meta-encodings of various matrix plaintext vectors, k in each singleton level. These encodings form a $k \times d\ell$ grid, with k matrices in each of the $d\ell$ columns. Since we have k encodings per singleton level, we modify our notation slightly; now $A_{i,j}^{(u)}$ will denote the jth encoding of the ith matrix for meta-level u. Furthermore, we can ignore the first three rows and columns of each $A_{i,j}^{(u)}$ matrix, as they play no role in the enforcing structure. Let $C_{i,j}^{(u)}$ be the width $2d(\ell - 1)$ diagonal matrix that remains.

To recap, $d - 1$ of the blocks of $C_{i,j}^{(u)}$ are set to be width-$2(\ell - 1)$ identity matrices (randomly scaled) and the uth block is set to $E_{i,j}^{(u)}$. For any meta-encoding j at level u, a fresh random set of $\{E_{i,j}^{(u)}\}_{i \in [\ell]}$ is generated. The bookends are formed by concatenating d independently generated instances of our $2(\ell - 1)$ dimensional bookends. The arrangement of the $C_{i,j}^{(u)}$ matrices (without the bookends) in the $k \times d\ell$ grid is shown below:

$$\begin{array}{cccc|cccc|c|cccc}
C_{1,1}^{(1)} & C_{1,1}^{(2)} & \cdots & C_{1,1}^{(d)} & C_{2,1}^{(1)} & C_{2,1}^{(2)} & \cdots & C_{2,1}^{(d)} & & C_{\ell,1}^{(1)} & C_{\ell,1}^{(2)} & \cdots & C_{\ell,1}^{(d)} \\
C_{1,2}^{(1)} & C_{1,2}^{(2)} & \cdots & C_{1,2}^{(d)} & C_{2,2}^{(1)} & C_{2,2}^{(2)} & \cdots & C_{2,2}^{(d)} & \cdots & C_{\ell,2}^{(1)} & C_{\ell,2}^{(2)} & \cdots & C_{\ell,2}^{(d)} \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\
C_{1,k}^{(1)} & C_{1,k}^{(2)} & \cdots & C_{1,k}^{(d)} & C_{2,k}^{(1)} & C_{2,k}^{(2)} & \cdots & C_{2,k}^{(d)} & & C_{\ell,k}^{(1)} & C_{\ell,k}^{(2)} & \cdots & C_{\ell,k}^{(d)}
\end{array}$$

The matrices are divided into ℓ groups, each consisting of k rows and d columns of matrices. Picking the matrix in row j and column u for each group gives the ℓ matrices that comprise the enforcing component of the jth meta-encoding at level u.

Notice that adding point-wise $C_{1,j_0}^{(u)}, \ldots, C_{\ell,j_0}^{(u)}$ to $C_{1,j_1}^{(u)}, \ldots, C_{\ell,j_1}^{(u)}$ or scaling $C_{1,j}^{(u)}, \ldots, C_{\ell,j}^{(u)}$ preserves the form of the matrices (though now the α's are different). Notice also that multiplying all the matrices in $C_{1,j}^{(u)}, \ldots, C_{\ell,j}^{(u)}$ together along with our bookend vectors gives 0. Therefore, we can take arbitrary linear combinations of meta-encodings, multiply them together, and still get 0. We will show that this is essentially the only way to combine different $C_{i,j}^{(u)}$ to get zero.

Applied to our construction, the matrices are the various meta-encodings of 0 and powers of 2 that the adversary is given in the public parameters. The adversary also has access to a pre-zero-test encoding, which does not fit this pattern. However, we can think of the pre-zero-test encoding as arising from d meta-encodings with matrix plaintext sequence $(0, 1, \ldots, 1)$, one encoding per singleton level. The actual pre-zero-test encoding is obtained by multiplying these encodings together.

While the CLT13 level structure allows an adversary to multiply together any collection of matrices that picks one from each column, our enforcing matrix structure will restrict the adversary to taking products of linear combinations of meta-encodings (or linear combinations of such products).

To formalize this notion, we first introduce the following definition.

Definition 2. *We define a valid monomial to be a polynomial representing a product of the $C_{i,j}^{(u)}$ matrices, so that exactly one matrix is taken from each column, in column order, along with the bookends.*

Next, we re-cast the $C_{i,j}^{(u)}$ matrices as the matrices of a read-ℓ vector-input branching program (an extension of matrix branching programs defined in Sect. 2.3) that takes inputs $x \in (\mathbb{Z})^{k \times d}$. The point of adopting the vector-input branching program (VBP) view is that read-ℓ VBP evaluations correspond exactly to valid manipulations of meta-encodings.

If we expand out any linear combination of read-ℓ VBP evaluations, the resulting polynomial is a linear combination of valid monomials. However, given an arbitrary linear combination of valid monomials, it is not immediately clear if it can be expressed as a linear combination of read-ℓ VBP evaluations (and hence a valid combination of meta-encodings). The following lemma characterizes precisely when this occurs.

Lemma 1. *Let Q be a linear combination of valid monomials. If Q evaluates to 0 as a polynomial over the underlying randomness of the $C_{i,j}^{(u)}$ matrices, Q is a linear combination of read-ℓ VBP evaluations, where the VBP is the one defined by the same $C_{i,j}^{(u)}$ matrices.*

For a more precise statement of this lemma and its proof, see the full version [1].

5 Security of Our Multilinear Map

Strategy Overview. To prove the security of our multilinear map construction within the CLT13 weak model, we define "real" and "ideal" experiments. Recall that an encoding of a plaintext in our scheme consists of numerous CLT13 plaintexts at different CLT13 levels. The "real" experiment allows the adversary to perform operations on any of these individual CLT13 encodings, and win through any of the victory conditions in the weak model. The "ideal" experiment provides the same interface to the adversary, but is run by a simulator that only has access to the vanilla generic multilinear map model. Intuitively, if there exists a simulator for which the adversary cannot distinguish between the two experiments, then no extra information is leaked in the real world and our multilinear map achieves ideal security.

Real and Ideal Experiments. Suppose the multilinear map is used to encode a sequence $m_1, \ldots, m_v$ of plaintexts at levels $S_1, \ldots, S_v$.

In the "real" world experiment, denoted $\mathtt{EXP}_{real}$, the adversary interacts with our weak CLT13 model, whose plaintexts consist of all the elements of all the matrices output by our multilinear map encoding procedure as well as all the elements of the public parameter matrices. These plaintexts are encoded at the appropriate levels derived from $S_1, \ldots, S_v$. The adversary can submit level-respecting polynomials over these plaintexts as zero-test queries, and receives a handle to the result of the zero-test computation for successful queries. Then the adversary enters a post-zero test stage, and can win by submitting a polynomial Q that satisfies the win conditions of the CLT13 weak model.

In the "ideal" world experiment, denoted $\mathtt{EXP}_{ideal}$, the adversary interacts with a simulator $\mathcal{S}$ that can interact with a vanilla generic multilinear map model. In this world, the model only stores the actual plaintexts $m_1, \ldots, m_v$ and levels $S_1, \ldots, S_v$. The adversary submits level-respecting polynomials over the plaintexts of the real world. The simulator $\mathcal{S}$ answers these queries using only queries to its model over the actual plaintexts. As in the real world, the adversary enters a post-zero test stage, which the simulator must respond to.

For any adversary $\mathcal{A}$, let $\mathtt{EXP}_{real}(\mathcal{A})$ (respectively $\mathtt{EXP}_{ideal}(\mathcal{S}, \mathcal{A})$) denote the probability that $\mathcal{A}$ can win in the "real" (respectively "ideal") world.

5.1 The Vector-input Branching Program Un-annihilatability (VBPUA) Assumption

The security of our multilinear map rests on a new assumption about annihilating vector-input branching programs (VBPs), defined in Sect. 2.3. Define a *generic* vector-input branching program (VBP) to be a VBP whose matrix entries are all distinct formal variables, instead of fixed ring elements. A generic VBP is evaluated just like a regular VBP, but the program output is a polynomial over formal variables. We refer to these outputs as generic VBP evaluation polynomials.

Note to the reader: The following is the simplest formal statement of our assumption. As the assumption relies on a number of newly introduced terms, it may be helpful to refer to the full version of this work [1] where we give concrete illustrations of our assumption for small cases.

We now define the polynomial-size arithmetic circuits A_r that will be necessary for the assumption statement. A_r takes the matrices of the vector-input branching program as input. The output of A_r is restricted to be a linear combination of vector-input branching program evaluations (though note that a polynomial-size arithmetic circuit can compute exponentially large linear combinations).

Assumption 1 (The (ℓ, w, k)-VBPUA Assumption). *Let $\ell = \mathsf{poly}(d, \lambda)$, $w = \mathsf{poly}(d, \lambda), k = \mathsf{poly}(d, \lambda)$ be parameters, and let $f(x)$ for $(x_1, \ldots, x_d) \in (\mathbb{Z}^k)^d$ (where each input vector x_i is a k-dimensional integer vector) be a generic* *vector-input branching program that reads each input vector x_i ℓ times and consists of $w \times w$ matrices. For each $r = 1, \ldots, m$, let A_r be any arithmetic circuits satisfying the definition above, each with size $\mathsf{poly}(d, k, \lambda)$. The output of A_r is a polynomial over the formal variables of the generic vector-input branching program, and denote it by P_r. Suppose further that $P_1, \ldots, P_m$ are linearly independent as polynomials over the formal variables of the generic vector-input branching program. Then there does not exist any polynomial-sized circuit Q of degree at most $2^{o(\lambda)}$ such that $Q(\{P_r\}_r) \equiv 0$ as a polynomial over the formal variables of the generic VBP.*

Note that each pair of functions ℓ, w, k gives a distinct assumption. In general, increasing ℓ or w intuitively gives a harder problem (and therefore milder assumption), while increasing k gives a potentially easier problem (and therefore stronger assumption. Our construction is quite flexible, and can be tailored to work with multiple possible settings of ℓ, w, k. Importantly, however, we will usually need k to be substantially larger than $w^2\ell$. For more precise bounds on ℓ, w, k, refer to the full version [1].

We conjecture that the assumption is for any choices of ℓ, w, k provided w^ℓ is exponential in λ, d. We conjecture that determinant-style annihilating attacks (discussed in the full version [1]) give the lowest-complexity annihilating polynomials for VBPs, as this appears to be the case for matrix branching programs. w^ℓ lower bounds the size of such determinant-style annihilations. Verifying this would imply security for the choices of ℓ, w, k that we require.

This assumption is similar to the Branching Program Un-Annihilatability Assumption of Garg et al. [37], but we do not know how to base our assumption on PRFs. For a detailed comparison of the statements of these assumptions, refer to the full version [1].

Note that the assumption states that no polynomial-size circuits Q of degree at most $2^{o(\lambda)}$ can annihilate a non-trivial set of VBP evaluations (generated by polynomial-size circuits). As evidence that this assumption is not trivially false, we heuristically argue in the full version of this work [1] that there are no circuits Q that can annihilate VBP evaluations up to a certain polynomial degree, even if we allow Q to have exponential size.

5.2 Security Proof

Our multilinear map is secure as long as (1) the adversary can never create a successful polynomial Q in the real world, and (2) any information the adversary gets in the real world, the adversary can also obtain in the ideal world. Formally, this requires proving the existence of a simulator such that no PPT adversary can distinguish between the two worlds.

Let d be the desired asymmetric degree of the multilinear map. We instantiate the construction with the number of meta-encodings k released per level set large enough to support secure re-randomization of meta-encodings under the Leftover Hash Lemma, and the number of matrices per meta-encoding ℓ set large enough so that brute-force determinant attacks are blocked. The width w of each matrix is set to $3 + 2d(\ell - 1)$ to fit the construction. For further details on the recommended parameter choices, refer to the full version [1].

With this setting of ℓ, w, k, security of our construction follows from the (ℓ, w, k)-VBPUA assumption.

Theorem 2. *Under the (ℓ, w, k)-VBPUA Assumption, our construction with the above parameter choices is secure in the CLT13 weak model. That is, there exists a PPT simulator $\mathcal{S}$ such that for all PPT adversaries $\mathcal{A}$,*

$$\Pr[\mathit{EXP}_{real}(\mathcal{A}) = \mathit{EXP}_{ideal}(\mathcal{S}, \mathcal{A})] = 1 - negl(\lambda)$$

Moreover, S always responds to post-zero test queries with 0.

Due to space restrictions, we give a high level overview of the major proof techniques and defer the full proof to [1].

Proof Sketch. We start with an application of Theorem 1, which states that if an adversary $\mathcal{A}$ can break our scheme in the weak CLT13 model, there exists an adversary $\mathcal{A}'$ in the CLT13 annihilating model. Recall that in the annihilating model, $\mathcal{A}'$ wins if it can annihilate the formal polynomials that correspond to successful zero-tests. Therefore, our first task is to show that the only successful zero-tests the adversary can compute are those that correspond to valid manipulations of our multilinear map meta-encodings. Then if we view the matrices in our scheme as the matrices of a read-ℓ vector-input branching program (VBP), a valid top-level meta-encoding corresponds to a linear combination of honest VBP evaluations. Given the VBPUA assumption, an adversary cannot successfully annihilate these evaluations.

Recall that the meta-encodings are themselves sequences of individual matrices. We must first show that the adversary must respect the structure of these individual matrices. We rely on an extension of Lemma 5.2 in [30], which we state and prove in the full version of this paper [1]. At a high level, our lemma shows that Kilian randomization matrices force the adversary to respect the original matrix structure; if the adversary attempts to pluck individual scalar entries out of these matrices and obtain zero-tests that do not correspond to products of matrices, then it will be unable to obtain successful zero-tests with any non-negligible probability. The next step is to show, roughly speaking, that an adversary cannot manipulate (i.e. add/multiply) one matrix from a meta-encoding without simultaneously performing the same operation on all of them. This follows from Lemma 1 (proven in the full version [1]), which implies that if the adversary does not respect the meta-encoding structure, our "enforcing matrices" will guarantee it does not obtain a successful zero-test except with negligible probability.

Taken together, these steps show that the adversary can never be successful in the post-zero-test stage of the CLT13 annihilating model. Therefore it can only distinguish the real experiment from the ideal experiment by making ordinary zero-test queries. In the full proof [1], we conclude by showing how a simulator $\mathcal{S}$ with access to an ideal implementation of our multilinear map can correctly simulate the 0/1 response to any zero-test query.

Acknowledgements. We thank Amit Sahai, Tancrède Lepoint, James Bartusek, and Leon Zhang for helpful discussions. We also thank the anonymous reviewers for feedback on prior versions of this work. Research supported in part from a DARPA SAFEWARE award and NSF. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

References

1. Ma, F., Zhandry, M.: The MMAP strikes back: obfuscation and new multilinear maps immune to CLT13 zeroizing attacks. Cryptology ePrint Archive, Report 2017/946 (2017). https://eprint.iacr.org/2017/946
2. Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. Contemp. Math. **324**, 71–90 (2003)
3. Garg, S., Gentry, C., Halevi, S., Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 479–499. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_27
4. Boneh, D., Waters, B., Zhandry, M.: Low overhead broadcast encryption from multilinear maps. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 206–223. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_12
5. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications, pp. 467–476 (2013)
6. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits, pp. 40–49 (2013)
7. Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Functional encryption without obfuscation. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 480–511. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49099-0_18
8. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more, pp. 475–484 (2014)
9. Hohenberger, S., Sahai, A., Waters, B.: Replacing a random oracle: full domain hash from indistinguishability obfuscation. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 201–220. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_12
10. Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 480–499. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_27
11. Pandey, O., Prabhakaran, M., Sahai, A.: Obfuscation-based non-black-box simulation and four message concurrent zero knowledge for NP. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 638–667. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_25
12. Chung, K.-M., Lin, H., Pass, R.: Constant-round concurrent zero-knowledge from indistinguishability obfuscation. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 287–307. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_14
13. Hubacek, P., Wichs, D.: On the communication complexity of secure function evaluation with long output, pp. 163–172 (2015)

14. Ananth, P., Sahai, A.: Functional encryption for turing machines. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9562, pp. 125–153. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49096-9_6
15. Bitansky, N., Paneth, O., Wichs, D.: Perfect structure on the edge of chaos. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9562, pp. 474–502. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49096-9_20
16. Bun, M., Zhandry, M.: Order-revealing encryption and the hardness of private learning. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9562, pp. 176–206. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49096-9_8
17. Bitansky, N., Paneth, O., Rosen, A.: On the cryptographic hardness of finding a Nash equilibrium, pp. 1480–1498 (2015)
18. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_1
19. Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 476–493. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_26
20. Gentry, C., Gorbunov, S., Halevi, S.: Graph-induced multilinear maps from lattices. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 498–527. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_20
21. Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 3–12. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_1
22. Hu, Y., Jia, H.: Cryptanalysis of GGH map. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 537–565. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_21
23. Coron, J.-S., Lee, M.S., Lepoint, T., Tibouchi, M.: Cryptanalysis of GGH15 multilinear maps. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 607–628. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_21
24. Coron, J.-S., Lepoint, T., Tibouchi, M.: New multilinear maps over the integers. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 267–286. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_13
25. Boneh, D., Wu, D.J., Zimmerman, J.: Immunizing multilinear maps against zeroizing attacks. Cryptology ePrint Archive, Report 2014/930 (2014). http://eprint.iacr.org/2014/930
26. Halevi, S.: Graded encoding, variations on a scheme. Cryptology ePrint Archive, Report 2015/866 (2015). http://eprint.iacr.org/2015/866
27. Coron, J.S., et al.: Zeroizing without low-level zeroes: new MMAP attacks and their limitations. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 247–266. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_12
28. Brakerski, Z., Gentry, C., Halevi, S., Lepoint, T., Sahai, A., Tibouchi, M.: Cryptanalysis of the quadratic zero-testing of GGH. Cryptology ePrint Archive, Report 2015/845 (2015). http://eprint.iacr.org/2015/845

29. Cheon, J.H., Fouque, P.-A., Lee, C., Minaud, B., Ryu, H.: Cryptanalysis of the new CLT multilinear map over the integers. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 509–536. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_20
30. Badrinarayanan, S., Miles, E., Sahai, A., Zhandry, M.: Post-zeroizing obfuscation: new mathematical tools, and the case of evasive circuits. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 764–791. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_27
31. Cramer, R., Ducas, L., Peikert, C., Regev, O.: Recovering short generators of principal ideals in cyclotomic rings. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 559–585. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_20
32. Albrecht, M., Bai, S., Ducas, L.: A subfield lattice attack on overstretched NTRU assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 153–178. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_6
33. Cheon, J.H., Jeong, J., Lee, C.: An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without a low level encoding of zero. Cryptology ePrint Archive, Report 2016/139 (2016). http://eprint.iacr.org/2016/139
34. Cheon, J.H., Hhan, M., Kim, J., Lee, C.: Cryptanalyses of branching program obfuscations over GGH13 multilinear map from the NTRU problem. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10993, pp. 184–210. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96878-0_7
35. Pellet-Mary, A.: Quantum attacks against indistinguishablility obfuscators proved secure in the weak multilinear map model. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10993, pp. 153–183. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96878-0_6
36. Miles, E., Sahai, A., Zhandry, M.: Annihilation attacks for multilinear maps: cryptanalysis of indistinguishability obfuscation over GGH13. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 629–658. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_22
37. Garg, S., Miles, E., Mukherjee, P., Sahai, A., Srinivasan, A., Zhandry, M.: Secure obfuscation in a weak multilinear map model. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 241–268. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_10
38. Lewi, K., et al.: 5Gen: A framework for prototyping applications using multilinear maps and matrix branching programs, pp. 981–992 (2016)
39. Paneth, O., Sahai, A.: On the equivalence of obfuscation and multilinear maps. Cryptology ePrint Archive, Report 2015/791 (2015). http://eprint.iacr.org/2015/791
40. Albrecht, M.R., Farshim, P., Hofheinz, D., Larraia, E., Paterson, K.G.: Multilinear maps from obfuscation. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9562, pp. 446–473. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49096-9_19
41. Barak, B., Garg, S., Kalai, Y.T., Paneth, O., Sahai, A.: Protecting obfuscation against algebraic attacks. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 221–238. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_13

42. Boneh, D., Ishai, Y., Sahai, A., Wu, D.J.: Lattice-based SNARGs and their application to more efficient obfuscation. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10212, pp. 247–277. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56617-7_9
43. Kilian, J.: Founding cryptography on oblivious transfer, pp. 20–31 (1988)
44. Brakerski, Z., Rothblum, G.N.: Virtual black-box obfuscation for all circuits via generic graded encoding. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 1–25. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_1
45. Bartusek, J., Guan, J., Ma, F., Zhandry, M.: Preventing zeroizing attacks on GGH15. Cryptology ePrint Archive, Report 2018/511 (2018). https://eprint.iacr.org/2018/511
46. Bartusek, J., Guan, J., Ma, F., Zhandry, M.: Return of GGH15: provable security against zeroizing attacks. In: TCC 2018 (2018)
47. Miles, E., Sahai, A., Weiss, M.: Protecting obfuscation against arithmetic attacks. Cryptology ePrint Archive, Report 2014/878 (2014). http://eprint.iacr.org/2014/878
48. Zimmerman, J.: How to obfuscate programs directly. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 439–467. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_15
49. Chen, Y., Gentry, C., Halevi, S.: Cryptanalyses of candidate branching program obfuscators. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10212, pp. 278–307. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56617-7_10
50. Coron, J.S., Lee, M.S., Lepoint, T., Tibouchi, M.: Zeroizing attacks on indistinguishability obfuscation over CLT13. Cryptology ePrint Archive, Report 2016/1011 (2016). http://eprint.iacr.org/2016/1011
51. Beimel, A., Gál, A.: On arithmetic branching programs. J. Comput. Syst. Sci. **59**(2), 195–220 (1999)
52. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: Widmayer, P., Eidenbenz, S., Triguero, F., Morales, R., Conejo, R., Hennessy, M. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 244–256. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45465-9_22
53. Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., Zimmerman, J.: Semantically secure order-revealing encryption: multi-input functional encryption without obfuscation. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 563–594. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_19
54. Coron, J.-S., Lee, M.S., Lepoint, T., Tibouchi, M.: Zeroizing attacks on indistinguishability obfuscation over CLT13. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10174, pp. 41–58. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54365-8_3