Immunizing Multilinear Maps Against Zeroizing Attacks

Dan Boneh
Stanford University

David J. Wu
Stanford University

Joe Zimmerman
Stanford University

Abstract

In recent work Cheon, Han, Lee, Ryu, and Stehlé presented an attack on the multilinear map of Coron, Lepoint, and Tibouchi (CLT). They show that given many low-level encodings of zero, the CLT multilinear map can be completely broken, recovering the secret factorization of the CLT modulus. The attack is a generalization of the "zeroizing" attack of Garg, Gentry, and Halevi.

We first strengthen the attack of Cheon, Han, Lee, Ryu, and Stehlé by showing that CLT can be broken even *without* low-level encodings of zero. This strengthening is sufficient to show that the subgroup elimination assumption does not hold for the CLT multilinear map.

We then present a generic defense against this type of "zeroizing" attack. For an arbitrary asymmetric composite-order multilinear map (including CLT), we give a functionality-preserving transformation that ensures that no sequence of map operations will produce valid encodings (below the zero-testing level) whose product is zero. We prove security of our transformation in a generic model of composite-order multilinear maps. Our new transformation rules out "zeroizing" leaving no currently known attacks on the decision linear assumption, subgroup elimination assumption, and other related problems for the CLT multilinear map. Of course, in time, it is possible that different attacks on CLT will emerge.

1 Introduction

Multilinear maps [BS03] have found numerous applications in cryptography. Most notably they give the first viable approach to code obfuscation [GGH+13b].

The first candidate construction for multilinear maps (or more precisely, for graded encoding schemes) is due to Garg, Gentry, and Halevi (GGH) [GGH13a] and makes use of ideal lattices. Garg, Gentry, and Halevi observed that their construction does not satisfy some useful assumptions in cryptography such as decision linear (DLIN) and subgroup membership [GGH13a, §4.4]. They presented a specific attack on instances of these problems that is now called a *zeroizing* attack.

Another multilinear map was presented by Coron, Lepoint, and Tibouchi (CLT) [CLT13] and works over the integers. The CLT multilinear map uses the same general paradigm as GGH, but appeared to satisfy the decision linear (DLIN) and subgroup membership assumptions. Consequently CLT has been used in several cryptographic applications of multilinear maps (see [CHL+14] for a brief survey) and in some early implementations [AHKM14]. Recently, a third candidate construction of multilinear maps was proposed by Gentry, Gorbunov, and Halevi [GGH14], making use of integer lattices.

Both the GGH and CLT multilinear maps provide the following abstract operations (see the next section for a complete definition). Let N be a positive integer and we refer to $\mathbb{Z}_N$ as the domain of the multilinear map. For a finite set $\mathcal{U}$ (called the top-level index set), we use $[x]_{\mathcal{S}}$ to denote an encoding of an element $x \in \mathbb{Z}_N$ at some index set $\mathcal{S} \subseteq \mathcal{U}$. From two encodings $[x]_{\mathcal{S}}$ and $[y]_{\mathcal{S}}$ we can obtain an encoding $[x+y]_{\mathcal{S}}$. Similarly, from two encodings $[x]_{\mathcal{S}_1}$ and $[y]_{\mathcal{S}_2}$ we can obtain

an encoding $[x \cdot y]_{\mathcal{S}_1 \cup \mathcal{S}_2}$ (for now we assume that $\mathcal{S}_1$ and $\mathcal{S}_2$ are disjoint). We say that an encoding $[x]_{\mathcal{S}}$ is at level $|\mathcal{S}|$ where $|\mathcal{S}|$ is the cardinality of $\mathcal{S}$. An encoding $[x]_{\mathcal{U}}$ at the top level $\mathcal{U}$ can be efficiently tested to determine if $x = 0$. We say that the map is a κ-way multilinear map where $\kappa = |\mathcal{U}|$. Every encoding of $0 \in \mathbb{Z}_N$ at any level is called a 0-encoding. We say that a 0-encoding is non-trivial if it is not the result of computing $\hat{x} - \hat{x}$ for some given encoding $\hat{x}$, or some similar such tautological expression (see Remark 3.10 for a precise definition).

The security of CLT. In important recent work, Cheon, Han, Lee, Ryu, and Stehlé [CHL+14] presented an attack on the CLT multilinear map. The attack generalizes the zeroizing attack of Garg, Gentry, and Halevi and obtains the following result: consider an application of CLT where the attacker can obtain sufficiently many fresh non-trivial 0-encodings at index set $\mathcal{S} \subsetneq \mathcal{U}$ and arbitrary encodings at index set $\mathcal{U} \setminus \mathcal{S}$. Cheon et al. show that in this case the attacker can recover all parameters of CLT that are intended to be secret (e.g. the secret factors of the CLT modulus). In applications of CLT where anyone should be able to create encodings of elements in $\mathbb{Z}_N$ (e.g. as in the n-way Diffie-Hellman protocol where n parties generate a shared key non-interactively [BS03, GGH13a, CLT13]) the public parameters of the scheme include sufficiently many 0-encodings to mount the attack. Therefore all these applications cannot be instantiated with CLT. We note that the attack of [CHL+14] requires many 0-encodings at some level below the zero-testing level, and so applications where the attacker cannot create such encodings are not directly impacted by the attack of [CHL+14].

Our contribution. Let MM be a multilinear map. We construct a new multilinear map ZMM whose operations are defined in terms of the original multilinear map MM. This transformation has the following property: given the public parameters of ZMM along with ZMM encodings of the attacker's choice, the attacker cannot construct a non-trivial 0-encoding for MM at any index set below the zero-testing level of MM. We refer to a transformation from MM to ZMM that has this property as a *zero-immunizing* transformation.

We emphasize that the map ZMM provides almost the same functionality as the underlying MM and can be used as a drop-in replacement for MM (one minor difference in functionality is explained in Remark 2.8). Applying our ZMM transformation to the CLT construction immunizes it against zeroizing attacks and restores it as a candidate construction for which assumptions like DLIN, subgroup membership, and subgroup elimination may hold.

Strengthening the [CHL+14] attack on CLT. To further illustrate the importance of the new defense, we also strengthen the [CHL+14] attack on the basic CLT scheme. The resulting attack has implications for other cryptographic assumptions such as the subgroup elimination assumption [GLW14, GLSW14].

Recall that the [CHL+14] attack requires 0-encodings at some level $t < \kappa$. We show that a similar attack is possible even if one is given appropriate *non-zero* encodings at levels less than κ, as long as the product of these encodings encodes 0 at the top level. We refer to a collection of encodings that multiply to 0 at level $\mathcal{U}$ as a set of *orthogonal encodings*. We show that sufficiently many fresh orthogonal encodings suffice to factor the CLT modulus.

Because our strengthened version of the [CHL+14] attack does not require low-level 0-encodings, we are able to apply it even in "secret-key" settings such as in [GLSW14], in which low-level 0-encodings are not made public. In particular, the attack can now be used to break the subgroup elimination problem when instantiated using the basic CLT construction. The reason is that many orthogonal sets can be generated from an instance of subgroup elimination.

Of course, when the CLT construction is augmented with our new defense, this attack is no longer possible. In fact, no attack is currently known on the subgroup elimination problem when instantiated using our proposed augmentation to CLT.

1.1 Overview of the Transformation

We briefly describe the structure of our zero-immunizing transformation. In what follows let $\mathbb{Z}_N$ be the desired domain of the multilinear map where $N = N_1 \cdots N_k$ for some positive integers $N_1, \ldots, N_k$ (in the notation of CLT this $N = g_1 \cdots g_k$). It will be convenient to denote an encoding of an element $x \in \mathbb{Z}_N$ by $[x_1, \ldots, x_k]_{\mathcal{S}}$ where $x = x_i \bmod N_i$ for all $i = 1, \ldots, k$. Recall that $\mathcal{U}$ is the top index set where zero-testing is possible and the degree of multilinearity is $\kappa = |\mathcal{U}|$.

Our zero-immunizing transformation creates a new multilinear map ZMM with a desired domain $\mathbb{Z}_N$ and top index set $\mathcal{U}$. To do so, the transformation instantiates MM with a domain $\mathbb{Z}_{N'}$ and top index set $\mathcal{U}'$. The domain $\mathbb{Z}_{N'}$ satisfies $N' = N \cdot N_{k+1} \cdot N_{k+2}$ for some secret integers N_{k+1} and N_{k+2}. The top index set $\mathcal{U}'$ for MM becomes about twice as big as $\mathcal{U}$: if $\mathcal{U} = \{A_1, \ldots, A_\kappa\}$ then

$$\mathcal{U}' = \{A_1^{(\mathrm{L})}, \ldots, A_\kappa^{(\mathrm{L})}\} \cup \{A_1^{(\mathrm{R})}, \ldots, A_\kappa^{(\mathrm{R})}\} \cup \{T\}$$

so that the degree of multilinearity of MM is $2\kappa + 1$. We use $\mathcal{U}_{\mathrm{L}}$ to denote the κ indexes in $\mathcal{U}'$ with superscript (L) and use $\mathcal{U}_{\mathrm{R}}$ to denote the κ indexes with superscript (R). We now use MM to setup a multilinear map for domain $\mathbb{Z}_{N'}$ and top index set $\mathcal{U}'$.

In the new multilinear map ZMM, an encoding $[x_1, \ldots, x_k]_{\mathcal{S}}$ of an element $x \in \mathbb{Z}_N$ at index set $\mathcal{S} \subseteq \mathcal{U}$ is the following pair of MM encodings:

$$\left(\quad [x_1, \ldots, x_k, \zeta, \nu_{\mathrm{L}}]_{\mathcal{S}_{\mathrm{L}}}, \quad [\eta_1, \ldots, \eta_k, \zeta, \nu_{\mathrm{R}}]_{\mathcal{S}_{\mathrm{R}}} \quad \right)$$

where $\mathcal{S}_{\mathrm{L}}$ is the copy of $\mathcal{S}$ in $\mathcal{U}_{\mathrm{L}}$ and $\mathcal{S}_{\mathrm{R}}$ is the copy of $\mathcal{S}$ in $\mathcal{U}_{\mathrm{R}}$. The scalars $\zeta, \nu_{\mathrm{L}}, \nu_{\mathrm{R}}$ and $\eta_1, \ldots, \eta_k$ are chosen at random in the appropriate rings $\mathbb{Z}_{N_i}$ for some $i \in \{1, \ldots, k+2\}$. Given two such encodings we can add and multiply them component-wise using the multilinear operations of MM. Now, to zero-test a ZMM top-level encoding $[x]_{\mathcal{U}}$ we make public two additional MM encodings:

$$\hat{t}_{\mathrm{L}} = [1, \ldots, 1, 1, 0]_{\mathcal{U}_{\mathrm{R}} \cup \{T\}} \quad \text{and} \quad \hat{t}_{\mathrm{R}} = [0, \ldots, 0, 1, 0]_{\mathcal{U}_{\mathrm{L}} \cup \{T\}} \ .$$

Then to zero-test $[x]_{\mathcal{U}} = (\hat{x}_{\mathrm{L}}, \hat{x}_{\mathrm{R}})$ we use the zero-test procedure of MM to zero-test the encoding:

$$\hat{y} = \hat{x}_{\mathrm{L}} \cdot \hat{t}_{\mathrm{L}} - \hat{x}_{\mathrm{R}} \cdot \hat{t}_{\mathrm{R}} \ .$$

Clearly $\hat{y}$ is an encoding at index set $\mathcal{U}'$ and can therefore be zero-tested. Moreover, by construction, $\hat{y}$ is a 0-encoding if and only if $x = 0$, as required.

It remains to argue that the transformation is zero-immunizing. We show that if MM is modeled as a generic multilinear map for domain $\mathbb{Z}_{N'}$ and top index set $\mathcal{U}'$ then using the public parameters of ZMM and any ZMM encodings of the attacker's choice, the attacker cannot produce a non-trivial 0-encoding for MM at any level below $\mathcal{U}'$. The proof makes use of the two additional components N_{k+1} and N_{k+2} and argues, using the Schwartz-Zippel lemma, that an attacker cannot w.h.p cause these components to become zero below the zero-testing level $\mathcal{U}'$. To apply the Schwartz-Zippel lemma we require that multilinearity degree κ is such that κ/N_{k+1} and κ/N_{k+2} are negligible. Moreover, we show that the attacker cannot even produce a pair of MM encodings $[x_1]_{S_1}$ and $[x_2]_{S_2}$, where $x_1, x_2 \in \mathbb{Z}_{N'}$, such that both encodings are below $\mathcal{U}'$ and $x_1 \cdot x_2 = 0$ in $\mathbb{Z}_{N'}$. This proves that ZMM is also immunized against our strengthening of the zeroizing attack of [CHL+14].

Immunized CLT. When applying this transformation to the CLT construction we must first suppress the publication of the CLT 0-encodings used for re-randomization as well as the level-0 CLT encodings, since those can be used to break the scheme. Obfuscation constructions that use CLT already suppress these values [GGH13a, GLSW14, Zim14] since re-randomization is not needed for obfuscation. This is also the case for some obfuscation-like constructions, such as the witness encryption scheme of [GLW14]. In particular, the constructions of [GLW14, GLSW14] whose security depends on the subgroup elimination assumption can use the basic immunized CLT scheme without re-randomization.

The CLT 0-encodings needed for re-randomization are required for several other applications of multilinear maps such as multi-user Diffie-Hellman. In Section 4.3 we use our approach to adapt the CLT re-randomization algorithm to obtain a candidate re-randomization algorithm that does not expose the scheme to the zeroizing attacks of [CHL$^+$14]. Moreover, in Remark 2.8 we explain that level-0 CLT encodings can be easily replaced by immunized level-1 encodings.

2 Preliminaries

2.1 Conventions

For integers n, a, b, we denote by $[n]$ the set $\{1, \ldots, n\}$, and by $[a, b]$ the set $\{a, \ldots, b\}$. We also denote by (a, b) the set $\{a + 1, \ldots, b - 1\}$. For a finite set S, we write $\mathsf{Uniform}(S)$ to mean the probability distribution that is uniform over the elements of S. For integers a, b, we write $\mathsf{Primes}[a, b]$ to mean the set of all prime numbers in $[a, b]$, and we overload this notation to refer to the distribution $\mathsf{Uniform}(\mathsf{Primes}[a, b])$.

Fix a ring R and an integer n. For a vector $\mathbf{c} \in R^n$ we write $\mathrm{diag}(\mathbf{c})$ to denote the matrix whose diagonal entries are the elements of $\mathbf{c}$ and whose off-diagonal entries are all 0. For a matrix $\mathbf{M} \in R^{n \times n}$ we write $\overline{\mathbf{M}}_{i,j}$ to denote the (i, j) minor of $\mathbf{M}$, i.e., the matrix $\mathbf{M}$ with the i^{th} row and j^{th} column removed.

We also assume standard conventions of cryptography. Specifically, we define a variable λ, called the security parameter. We define a *negligible function* to be a function $\varepsilon(\lambda)$ that is $o(1/\lambda^c)$ for every $c > 0$, and we write $\mathrm{negl}(\lambda)$ to denote a negligible function of λ. We define an *efficient algorithm* to be a probabilistic polynomial-time Turing machine. We say that an event occurs with *negligible probability* if the probability of the event is $\mathrm{negl}(\lambda)$, and an event occurs with *overwhelming probability* if its complement occurs with negligible probability.

2.2 Multilinear Maps

Multilinear maps [BS03], also known as graded multilinear maps or graded encodings [GGH13a, CLT13, GGH14], are a generalization of bilinear maps such as pairings over elliptic curves [Mil04, MOV93]. Intuitively, a multilinear map lets us take scalars x, y and produce corresponding encodings $\hat{x}, \hat{y}$ at any level of a given hierarchy, so that we can still perform arithmetic operations (e.g., $x + y, xy$) on the encoded representations, and yet it is hard to recover the original scalars x, y from encodings $\hat{x}, \hat{y}$. For example, in a symmetric bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ (where g generates $\mathbb{G}$, and $e(g, g)$ generates $\mathbb{G}_T$), a scalar $x \in \mathbb{Z}$ can be encoded in $\mathbb{G}$ as g^x, or encoded in $\mathbb{G}_T$ as $e(g, g)^x$. The levels of the hierarchy here are $\mathbb{G}$ and $\mathbb{G}_T$, and the hierarchy's structure enforces constraints on the arithmetic operations that we can perform. For instance, via the group operation we can compute g^{x+y} (an encoding of $x + y$) from g^x and g^y (encodings of x and y), but to obtain an encoding of xy, we must increase the level in the hierarchy from $\mathbb{G}$ to $\mathbb{G}_T$, by computing the pairing $e(g^x, g^y) = e(g, g)^{xy}$.

In the case of symmetric bilinear maps, this hierarchical structure can be identified with the integers $0, 1, 2$ as indices, where the index 0 represents scalars, 1 represents elements of $\mathbb{G}$, and 2 represents elements of $\mathbb{G}_T$. Elements at the same index can be added together, while elements at arbitrary indices can be multiplied, but their indices add. For asymmetric bilinear maps, the more natural analogy is that of a subset lattice: specifically, a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is identified with the subset lattice $\emptyset \subseteq \{A\},\{B\} \subseteq \{A, B\}$, where $\emptyset$ corresponds to scalars, $\{A\}$ to $\mathbb{G}_1$, $\{B\}$ to $\mathbb{G}_2$, and $\{A, B\}$ to $\mathbb{G}_T$.

More generally, in the case of asymmetric multilinear maps (which permit more than two sequential multiplications of encoded elements), it is standard to work with general subset lattices, where the sets may contain elements with multiplicity. By convention, we will say that these sets are made up of *formal symbols*, denoted by capital letters (A, B, C), which serve the same role as formal variables in polynomials. Formally, we state the following definitions.

Definition 2.1 (Formal Symbol)**.** A *formal symbol* is a bit string in $\{0, 1\}^*$, and distinct variables denote distinct bit strings. A *fresh* formal symbol is any bit string in $\{0, 1\}^*$ that has not already been assigned to another formal symbol.

Definition 2.2 (Index Sets)**.** An *index set* is a multi-set of formal symbols called *indices*. The multiplicity of each index is written in binary, and so the degree of an index set may be exponential in the size of its representation. By convention, for index sets we use set notation and product notation interchangeably, so that A^3BC^2 represents the multi-set $\{A, A, A, B, C, C\}$, and $A^3BC^2 \cdot ABC = A^4B^2C^3$. We further generalize this notation for quotients of formal symbols as appropriate, so that $A^4B^2C^3/(ABC) = A^3BC^2$.

Definition 2.3 (Composite-Order Multilinear Map ([BS03, GGH13a, CLT13, GLW14, Zim14]))**.** A *composite-order multilinear map* supports the following operations. Each operation ($\mathsf{MM.Setup}$, $\mathsf{MM.Add}$, $\mathsf{MM.Mult}$, $\mathsf{MM.ZeroTest}$, $\mathsf{MM.Encode}$) is implemented by an efficient randomized algorithm.

- The setup procedure receives as input an index set $\mathcal{U}$ (Definition 2.2), which we refer to as the "top-level index set", as well as the security parameter λ (in unary), and an integer k indicating the number of factors to generate for the modulus. It produces public parameters pp, secret parameters sp, and integers $N_1, \ldots, N_k$ as follows:

$$\mathsf{MM.Setup}(\mathcal{U}, 1^\lambda, k) \;\to\; (\mathsf{pp},\ \mathsf{sp},\ N_1, \ldots, N_k)$$

  Each integer $N_1, \ldots, N_k$ is a product of $\mathrm{poly}(\lambda)$ primes, and each of these $k \cdot \mathrm{poly}(\lambda)$ primes is drawn independently from $\mathsf{Primes}[2^{s(\lambda)}, 2^{s(\lambda)+1}]$ for some $s(\lambda) = \mathrm{poly}(\lambda)$. We also define $N = \prod_{i \in [k]} N_i$, the overall modulus.[1]

- For each index set $\mathcal{S} \subseteq \mathcal{U}$, and each scalar $x \in \mathbb{Z}_N$, there is a set of strings $[x]_{\mathcal{S}} \subseteq \{0, 1\}^*$, i.e., the set of all valid encodings of x at index set $\mathcal{S}$.[2] From here on, we will abuse notation to write $[x]_{\mathcal{S}}$ to stand for any element of $[x]_{\mathcal{S}}$ (i.e., any valid encoding of x at the index set $\mathcal{S}$).

[1] We remark here that our construction does not rely on the individual moduli $N_1, \ldots, N_k$ being composite, but we present the model in this full generality since it may be required in the chosen concrete instantiation, such as in the CLT multilinear map [CLT13].

[2] To be precise, we define $[x]_{\mathcal{S}} = \{\chi \in \{0, 1\}^* : \mathsf{MM.IsEncoding}(\mathsf{pp}, \chi, x, \mathcal{S})\}$, where the predicate $\mathsf{MM.IsEncoding}$ is specified by the concrete instantiation of the multilinear map. The predicate $\mathsf{MM.IsEncoding}$ need not be efficiently decidable—and indeed, for the security of the multilinear map, it should not be.

- Elements at the same index set $\mathcal{S} \subseteq \mathcal{U}$ can be added,[3] with the result also encoded at $\mathcal{S}$:

$$\mathsf{MM.Add}(\mathsf{pp},\ [x]_{\mathcal{S}},\ [y]_{\mathcal{S}}) \ \rightarrow\ [x+y]_{\mathcal{S}}$$

- Elements at two index sets $\mathcal{S}_1, \mathcal{S}_2$ can be multiplied, with the result encoded at the union of the two sets, as long as their union is still contained in $\mathcal{U}$:

$$\mathsf{MM.Mult}(\mathsf{pp},\ [x]_{\mathcal{S}_1},\ [y]_{\mathcal{S}_2}) \ \rightarrow\ \begin{cases} [xy]_{\mathcal{S}_1\mathcal{S}_2} & \text{if } \mathcal{S}_1\mathcal{S}_2 \subseteq \mathcal{U} \\ \perp & \text{otherwise} \end{cases}$$

- Elements at the top level $\mathcal{U}$ can be *zero-tested*:

$$\mathsf{MM.ZeroTest}(\mathsf{pp},\ [x]_{\mathcal{S}}) \ \rightarrow\ \begin{cases} \text{“zero”} & \text{if } \mathcal{S} = \mathcal{U} \text{ and } x = 0 \in \mathbb{Z}_N \\ \text{“nonzero”} & \text{otherwise} \end{cases}$$

- Using the secret parameters, one can generate a representation of a given scalar $x \in \mathbb{Z}$ at any index set $\mathcal{S} \subseteq \mathcal{U}$:

$$\mathsf{MM.Encode}(\mathsf{sp},\ x,\ \mathcal{S}) \ \rightarrow\ [x]_{\mathcal{S}}$$

- For the trivial index set $\mathcal{S} = \emptyset$, we specify that the valid encodings $[x]_{\emptyset}$ are just the integers congruent to x modulo N. (So, for instance, we can perform subtraction via $\mathsf{MM.Add}$, by scalar multiplication with -1.)

When the context is clear, we also abuse notation to write, for encodings $\hat{a}, \hat{b}$, the expression $\hat{a}+\hat{b}$ to mean $\mathsf{MM.Add}(\mathsf{MM.pp}, \hat{a}, \hat{b})$; the expression $\hat{a}\hat{b}$ to mean $\mathsf{MM.Mult}(\mathsf{MM.pp}, \hat{a}, \hat{b})$; and likewise for other arithmetic expressions.

Definition 2.4 (Degree of Multilinearity)**.** *Let* MM *be a multilinear map (Definition 2.3). When* $\mathsf{MM.Setup}$ *is instantiated with a top-level index set* $\mathcal{U}$*, we refer to the total degree of* $\mathcal{U}$ *as the* degree of multilinearity *of the map (denoted by* κ*).*

We note that in all known multilinear map constructions (including CLT), the maximum degree of multilinearity is polynomial in λ, due to the noise growth. We formalize this distinction as follows (rephrasing the definition of “noisy” multilinear maps in [Zim14]):

Definition 2.5 (Poly-Degree Multilinear Map)**.** We define a *poly-degree* multilinear map as in Definition 2.3, except that the top-level index set $\mathcal{U}$ has its multiplicities represented in unary.

Remark 2.6 (Notation for Encodings of Direct Products [Zim14])**.** We write $[x_1, x_2, \ldots, x_k]_S$ to refer to an encoding, at index set S, of the value $x \in \mathbb{Z}_N$ such that $x \equiv x_i \pmod{N_i}$ for each $i \in [k]$ (as determined by the Chinese Remainder Theorem). When the context is clear, we also generalize this notation to (non-encoded) scalars, writing $[x_1, x_2, \ldots, x_k]$ for the value $x \in \mathbb{Z}_N$ such that $x \equiv x_i \pmod{N_i}$ for each $i \in [k]$.

[3] Strictly speaking, in known multilinear maps such as the CLT scheme, we also require that not too many additions take place, so that repeated-doubling cannot increase the noise and destroy the encodings. We similarly require that integer constants remain small in scalar multiplication, i.e., of bit length $r(\lambda)$ for some fixed polynomial r determined by the multilinear map construction. Since these constraints do not affect our results here, we omit the formalization from the abstract model.

Remark 2.7 (Public Encoding via Re-randomization)**.** The abstract operations of Definition 2.3 suffice for "secret-key" multilinear maps, as in obfuscation constructions [GGH13a, BR14, BGK+14, GLSW14, Zim14], in which the parameters required to encode elements need not be public. In general, however, multilinear maps such as GGH and CLT support additional extended operations for *public encoding*. These schemes publish encodings of 0 and 1, which can be added and scaled to encode other scalars, and provide an additional "re-randomization" operation which takes an encoded scalar and "washes it out" into some canonical distribution to hide its provenance. Indeed, the encodings of 0 and 1 required for public encoding are part of the reason that CLT is vulnerable to the [CHL+14] attack (though, as we will see in Section 5, they are not the only reason).

The CLT re-randomization procedure requires many CLT 0-encodings. Since these 0-encodings by themselves enable a total break of CLT via the [CHL+14] attack, it is not clear how to define a zero-immunizing transformation "ZMM.ReRand" in terms of the underlying MM operation, as we do for the basic operations MM.Setup, MM.Encode, MM.Add, MM.Mult, MM.ZeroTest (Definition 2.3). We cannot simply use zero-immunized versions of these 0-encodings for re-randomization, since CLT requires some of the encodings to have their randomizers $r_{i,j}$ drawn from a custom distribution (as the columns of a certain matrix $\mathbf{\Pi}$). Rather, in Section 4.3 we will describe candidate procedures, beyond the basic abstract operations of Definition 2.3, that implement re-randomization and public encoding specifically for our zero-immunized version of the CLT multilinear map.

Remark 2.8 (Sampling and Level-0 Encodings)**.** The CLT multilinear map also provides a way to generate "level-0" encodings, i.e., other values, besides scalars in $\mathbb{Z}$, that are valid encodings at the "level-0" index set, $\emptyset$. Concretely, this is done by constructing terms with no z values in the denominator (see Section 4.1). It is not clear how to zero-immunize level-0 encodings at all, since *any* level-0 encoding will yield the analog of a zero-product (Definition 3.11): assuming the multilinear map is nontrivial, the adversary must be able to produce some 0-encoding at the top level $\mathcal{U}$, which can then be multiplied by any level-0 encoding to form a zero-product at $\mathcal{U}$.

Instead, our zero-immunizing transformation omits these CLT level-0 encodings. To replace them, in applications that require level-0 encodings, we propose introducing additional formal indices in the top-level index set as needed (Definition 2.2). Thus, for example, in n-way Diffie-Hellman key agreement based on CLT [CLT13], we would replace the level-0 encodings of random scalars with the same values encoded at some fresh singleton index set C, that would become part of the new top level $\mathcal{U}' = \mathcal{U} \cdot C$. Sampling of random encoded scalars would work just as in CLT, with the subset sum over level-0 encodings replaced by an analogous subset sum of these new public encodings at index set C. Moreover, we can even enable public encoding of arbitrary scalars, by publishing the appropriate encodings of powers of two at index set C, as described in Section 4.3.3.

Strictly speaking, this is a slight departure from the operations as defined in [CLT13], since it requires more multilinearity for each given application: namely, a new index set whose multiplicity corresponds to the maximum number of multiplications that the original application required for level-0 encodings. However, we do not foresee any difficulties in applications arising from this small change.

3 The Zero-Immunizing Transformation

We now present our main zero-immunizing transformation for the case where re-randomization of encodings is not necessary. We show how to address re-randomization in the next section.

Construction 3.1 (Zero-Immunizing Multilinear Map Transformation)**.** Let MM = (MM.Setup, MM.Add, MM.Mult, MM.ZeroTest, MM.Encode) be a composite-order multilinear map. We define

the transformed composite-order multilinear map, ZMM, in terms of MM as follows.

ZMM.Setup$(\mathcal{U}, 1^\lambda, k)$:

1. Let the index set $\mathcal{U}$ be written as $A_1^{d_1} \cdots A_s^{d_s}$ for some formal symbols $A_1, \ldots, A_s$ and integers $d_1, \ldots, d_s$. For each formal symbol A_i for $i \in [s]$, define a pair of fresh formal symbols $A_{i,\mathrm{L}}, A_{i,\mathrm{R}}$. For an index set $S = A_1^{e_1} \cdots A_s^{e_s} \subseteq \mathcal{U}$, we define two derived index sets $S_\mathrm{L} = A_{1,\mathrm{L}}^{e_1} \cdots A_{s,\mathrm{L}}^{e_s}$ and $S_\mathrm{R} = A_{1,\mathrm{R}}^{e_1} \cdots A_{s,\mathrm{R}}^{e_s}$.
2. Construct a new top-level index set $\mathcal{U}_Z = \mathcal{U}_\mathrm{L}\mathcal{U}_\mathrm{R}T$, for a fresh formal symbol T.
3. Run $(\mathsf{MM.pp}, \mathsf{MM.sp}, N_1, \ldots, N_{k+2}) \leftarrow \mathsf{MM.Setup}(\mathcal{U}_Z, 1^\lambda, k+2)$.
4. Generate encodings as follows:
$$\hat{t}_\mathrm{L} \leftarrow \mathsf{MM.Encode}(\mathsf{MM.pp},\ [1, \ldots, 1, 1, 0],\ \mathcal{U}_\mathrm{R}T)$$
$$\hat{t}_\mathrm{R} \leftarrow \mathsf{MM.Encode}(\mathsf{MM.pp},\ [0, \ldots, 0, 1, 0],\ \mathcal{U}_\mathrm{L}T)$$
5. Output $(\mathsf{pp}, \mathsf{sp}, N_1, \ldots, N_k)$, where $\mathsf{sp} = (\mathsf{MM.sp}, N_{k+1}, N_{k+2})$ and $\mathsf{pp} = (\mathsf{MM.pp}, \hat{t}_\mathrm{L}, \hat{t}_\mathrm{R})$.

ZMM.Encode$(\mathsf{sp}, [x_1, \ldots, x_k], S)$:

1. Parse sp as $(\mathsf{MM.sp}, N_{k+1}, N_{k+2})$.
2. Choose $\eta_x = [\eta_{x,1}, \ldots, \eta_{x,k}] \leftarrow \mathsf{Uniform}(\mathbb{Z}^*_{N_1 \cdots N_k})$, $\zeta_x \leftarrow \mathsf{Uniform}(\mathbb{Z}^*_{N_{k+1}})$, and $\nu_{x,\mathrm{L}}, \nu_{x,\mathrm{R}} \leftarrow \mathsf{Uniform}(\mathbb{Z}^*_{N_{k+2}})$.
3. Generate encodings as follows:
$$\hat{x}_\mathrm{L} \leftarrow \mathsf{MM.Encode}(\mathsf{MM.sp}, [x_1, \ldots, x_k, \zeta_x, \nu_{x,\mathrm{L}}], S_\mathrm{L})$$
$$\hat{x}_\mathrm{R} \leftarrow \mathsf{MM.Encode}(\mathsf{MM.sp}, [\eta_{x,1}, \ldots, \eta_{x,k}, \zeta_x, \nu_{x,\mathrm{R}}], S_\mathrm{R})$$
4. Output $\hat{x} := (\hat{x}_\mathrm{L}, \hat{x}_\mathrm{R})$.

ZMM.Add$(\mathsf{MM.sp}, \hat{x}, \hat{y})$:

1. Parse $\hat{x}$ as a pair of encodings $(\hat{x}_\mathrm{L}, \hat{x}_\mathrm{R})$, and parse $\hat{y}$ as a pair of encodings $(\hat{y}_\mathrm{L}, \hat{y}_\mathrm{R})$.
2. Using the procedure MM.Add, compute the encodings $\hat{w}_\mathrm{L} = \hat{x}_\mathrm{L} + \hat{y}_\mathrm{L}$ and $\hat{w}_\mathrm{R} = \hat{x}_\mathrm{R} + \hat{y}_\mathrm{R}$. (If any of these operations outputs $\perp$, then immediately output $\perp$.)
3. Output the result $\hat{w} := (\hat{w}_\mathrm{L}, \hat{w}_\mathrm{R})$.

ZMM.Mult$(\mathsf{pp}, \hat{x}, \hat{y})$:

1. Parse $\hat{x}$ as a pair of encodings $(\hat{x}_\mathrm{L}, \hat{x}_\mathrm{R})$, and parse $\hat{y}$ as a pair of encodings $(\hat{y}_\mathrm{L}, \hat{y}_\mathrm{R})$.
2. Using the procedure MM.Mult, compute the encodings $\hat{w}_\mathrm{L} = \hat{x}_\mathrm{L} \cdot \hat{y}_\mathrm{L}$ and $\hat{w}_\mathrm{R} = \hat{x}_\mathrm{R} \cdot \hat{y}_\mathrm{R}$. (If any of these operations outputs $\perp$, then immediately output $\perp$.)
3. Output the result $\hat{w} := (\hat{w}_\mathrm{L}, \hat{w}_\mathrm{R})$.

ZMM.ZeroTest$(\mathsf{pp}, \hat{x})$:

1. Parse pp as $(\mathsf{MM.pp}, \hat{t}_{\mathrm{L}}, \hat{t}_{\mathrm{R}})$, and parse $\hat{x}$ as a pair of encodings $(\hat{x}_{\mathrm{L}}, \hat{x}_{\mathrm{R}})$.

2. Using the procedures $\mathsf{MM.Add}$, $\mathsf{MM.Mult}$, compute the encoding $\hat{z} = \hat{x}_{\mathrm{L}} \cdot \hat{t}_{\mathrm{L}} - \hat{x}_{\mathrm{R}} \cdot \hat{t}_{\mathrm{R}}$. (If any of these operations outputs $\perp$, then immediately output $\perp$.)

3. Output the result of $\mathsf{MM.ZeroTest}(\mathsf{MM.pp}, \hat{z})$.

We first show functional correctness of the zero-immunizing transformation (Construction 3.1), which is fairly straightforward from the definitions.

Theorem 3.2 (Functional Correctness of ZMM Transformation)**.** *Let* MM *be a multilinear map implementing the operations described in Definition 2.3. Then the transformed version* ZMM *(Construction 3.1) also implements the operations described in Definition 2.3.*

Proof. It suffices to show correctness for the $\mathsf{ZMM.ZeroTest}$ operation. Let m be the number of invocations of $\mathsf{ZMM.Encode}$, and for each $j \in [m]$ let $\hat{x}_j := (\hat{x}_{j,\mathrm{L}}, \hat{x}_{j,\mathrm{R}})$ be the output of the j^{th} invocation, on input $x_j = [x_{j,1}, \ldots, x_{j,k}]$. Fix a multivariate polynomial f, and suppose we evaluate $f(\hat{x}_1, \ldots, \hat{x}_m)$ using the operations $\mathsf{ZMM.Add}, \mathsf{ZMM.Mult}$, then run $\mathsf{ZMM.ZeroTest}$ on the resulting element $\hat{f} = (\hat{f}_{\mathrm{L}}, \hat{f}_{\mathrm{R}})$. By definition, the output will be the result of $\mathsf{MM.ZeroTest}(\mathsf{MM.pp}, \hat{z})$, where $\hat{z} = \hat{f}_{\mathrm{L}} \cdot \hat{t}_{\mathrm{L}} - \hat{f}_{\mathrm{R}} \cdot \hat{t}_{\mathrm{R}}$. The value of this encoding $\hat{z}$ in the i^{th} component, for $i \leq k$, will be $f(x_{1,i}, \ldots, x_{m,i}) \cdot 1 - f(\eta_{1,i}, \ldots, \eta_{m,i}) \cdot 0 = f(x_{1,i}, \ldots, x_{m,i})$ (for randomizers $\eta_{j,i}$ chosen by $\mathsf{ZMM.Encode}$); while in component $(k+1)$, it will be $f(\zeta_{x_1}, \ldots, \zeta_{x_m}) - f(\zeta_{x_1}, \ldots, \zeta_{x_m}) = 0$ (for randomizers ζ_{x_j} chosen by $\mathsf{ZMM.Encode}$); and in component $(k+2)$, it will be zero since $\hat{t}_{\mathrm{L}}, \hat{t}_{\mathrm{R}}$ take the value zero there. Thus, the zero-test will output "zero" precisely when $f(x_{1,i}, \ldots, x_{m,i}) = 0 \pmod{N_i}$ for all $i \in [k]$, as desired. □

Remark 3.3 (Additional Components)**.** Our zero-immunizing transformation uses two additional components (N_{k+1}, N_{k+2}). In fact, the first component alone, using the random ζ values, suffices to rule out the [CHL+14] attack (even our strengthened version). The second component, using the random ν values, is only needed for an additional attractive property: the adversary cannot produce any low-level 0-encodings (even ones that cannot be completed to the top level $\mathcal{U}$), as described in Theorem 3.14, below.

Remark 3.4 (Blinding Factors)**.** To simplify the presentation, we set all nonzero components to 1 in the encodings $\hat{t}_{\mathrm{L}}, \hat{t}_{\mathrm{R}}$ in our transformation (Construction 3.1). As shown below, this suffices to rule out the [CHL+14] attack (even our strengthened version). We know of no attacks on this version, and indeed we will prove that no attack "similar" to [CHL+14] can succeed, by formalizing such attacks in a generic model. However, it still seems safer to set each nonzero component of $\hat{t}_{\mathrm{L}}$ to an independently uniform unit (and set the corresponding component of $\hat{t}_{\mathrm{R}}$ accordingly). This does not impact correctness, and so we recommend that any application of our transformation should use these additional blinding factors.

3.1 The Generic Multilinear Map Model

To define security, we will use the formulation of the generic group model of composite-order multilinear maps as described by Zimmerman [Zim14]. This model is based on other generic multilinear map models for the prime-order case [GGH+13b, BR14, BGK+14] and builds on the generic group model of Shoup [Sho97]. Intuitively, in the generic model, the only thing an adversary can do with encoded ring elements is arithmetic (addition, subtraction, multiplication) and apply the operations of the multilinear map.

More precisely, we say a scheme that uses multilinear maps is "secure in the generic model" if, for any concrete adversary breaking the real scheme, there is a generic adversary breaking a modified scheme in which the encoded ring elements are replaced by "handles" (concretely, fresh nonces), which the generic-model adversary can supply to a stateful oracle $\mathcal{M}$ (which performs the corresponding ring operations internally). We define the oracle $\mathcal{M}$ formally as follows.

Definition 3.5 (Generic Multilinear Map Oracle ([GGH+13b, BR14, BGK+14, Zim14]))**.** A *generic multilinear map oracle* is a stateful oracle $\mathcal{M}$ that responds to queries as follows.

- On a query $\mathsf{MM.Setup}(\mathcal{U}, 1^\lambda, k)$, the oracle will generate integers $N_1, \ldots, N_k$ as in the real setup procedure (Definition 2.3), generate pp, sp as fresh nonces (i.e., distinct from any previous choices) uniformly at random from $\{0,1\}^\lambda$, and return $(\mathsf{pp}, \mathsf{sp}, N_1, \ldots, N_k)$. It will also store the inputs and the values generated, initialize an internal table $T \leftarrow \{\}$ (to store "handles", as described below), and set internal state so that subsequent $\mathsf{MM.Setup}$ queries fail.

- On a query $\mathsf{MM.Encode}(k,\ x,\ \mathcal{S})$, where $k \in \{0,1\}^\lambda$ and $x \in \mathbb{Z}$, the oracle will check that $k = \mathsf{sp}$ and $\mathcal{S} \subseteq \mathcal{U}$ (returning $\perp$ if the check fails). If the check passes, the oracle will generate a fresh nonce ("handle") $h \leftarrow \mathsf{Uniform}(\{0,1\}^\lambda)$, add the entry $h \mapsto (x, \mathcal{S})$ to the table T, and return h.

- On a query $\mathsf{MM.Add}(k, h_1, h_2)$, where $k, h_1, h_2 \in \{0,1\}^\lambda$, the oracle will check that $k = \mathsf{pp}$, and that the handles h_1, h_2 are present in its internal table T, and are mapped to values, resp., $(x_1, \mathcal{S}_1)$ and $(x_2, \mathcal{S}_2)$ such that $\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S} \subseteq \mathcal{U}$ (returning $\perp$ if the check fails). If the check passes, the oracle will generate a fresh handle $h \leftarrow \mathsf{Uniform}(\{0,1\}^\lambda)$, add the entry $h \mapsto (x_1 + x_2, \mathcal{S})$ to the table T, and return h.

- On a query $\mathsf{MM.Mult}(k, h_1, h_2)$, where $k, h_1, h_2 \in \{0,1\}^\lambda$, the oracle will check that $k = \mathsf{pp}$, and that the handles h_1, h_2 are present in its internal table T, and are mapped to values, resp., $(x_1, \mathcal{S}_1)$ and $(x_2, \mathcal{S}_2)$ such that $\mathcal{S}_1\mathcal{S}_2 \subseteq \mathcal{U}$ (returning $\perp$ if the check fails). If the check passes, the oracle will generate a fresh handle $h \leftarrow \mathsf{Uniform}(\{0,1\}^\lambda)$, add the entry $h \mapsto (x_1 x_2, \mathcal{S}_1\mathcal{S}_2)$ to the table T, and return h.

- On a query $\mathsf{MM.ZeroTest}(k, h)$, where $k, h \in \{0,1\}^\lambda$, the oracle will check that $k = \mathsf{pp}$, and that the table T contains an entry $h \mapsto (x, \mathcal{U})$ (immediately returning $\perp$ if the check fails). If the check passes, the oracle will return "zero" if $x \equiv 0 \pmod{N = N_1 \cdots N_k}$, and "nonzero" otherwise.

3.2 The Transformed Generic Model

We now adapt the generic model of Definition 3.5 to address the security of our zero-immunizing transformation ZMM (Construction 3.1). Intuitively, the transformation should be used as follows. Suppose we have some system that currently uses a composite-order multilinear map MM (such as CLT). Then we replace that system's invocations of the map operations (MM.Setup, MM.Encode, etc.) with the transformed versions (ZMM.Setup, ZMM.Encode, etc.). Now, we aim to show that an adversary $\mathcal{A}$, given the encodings produced by the ZMM-transformed system, cannot produce any set of encodings that multiply to 0 at the top level (or trigger some other specified failure event)—even when the adversary $\mathcal{A}$ is allowed to perform the operations of the *original* map MM, evaluated on any MM encodings that the transformation ZMM exposes. The following generic-model definitions capture this interaction.

Definition 3.6 (ZMM-Transformed Generic Multilinear Map Oracle)**.** A ZMM-transformed generic multilinear map oracle is a stateful oracle $\mathcal{M}_Z$ that operates as follows. The oracle maintains an internal (stateful) copy of a generic multilinear map oracle $\mathcal{M}_C$, and answers queries as follows.

- On a query $\mathsf{ZMM.Setup}(\mathcal{U}, 1^\lambda, k)$, the oracle constructs a new top-level index set $\mathcal{U}_Z = \mathcal{U}_{\mathrm{L}}\mathcal{U}_{\mathrm{R}}T$ as in the real scheme. The oracle issues the query $\mathsf{MM.Setup}(\mathcal{U}_Z, 1^\lambda, k+2)$ to $\mathcal{M}_C$ and obtains $(\mathsf{MM.pp}, \mathsf{MM.sp}, N_1, \ldots, N_{k+2})$. It issues queries $\mathsf{MM.Encode}(\mathsf{MM.pp}, [1, \ldots, 1, 1, 0], \mathcal{U}_{\mathrm{R}}T)$ and $\mathsf{MM.Encode}(\mathsf{MM.pp}, [0, \ldots, 0, 1, 0], \mathcal{U}_{\mathrm{L}}T)$ to $\mathcal{M}_C$ to obtain $\hat{t}_{\mathrm{L}}, \hat{t}_{\mathrm{R}}$, respectively. The oracle sets $\mathsf{pp} \leftarrow (\mathsf{MM.pp}, \hat{t}_{\mathrm{L}}, \hat{t}_{\mathrm{R}})$ and generates sp as a fresh nonce uniformly at random from $\{0,1\}^\lambda$. The oracle returns $(\mathsf{pp}, \mathsf{sp}, N_1, \ldots, N_{k+2})$. It also stores the inputs and all values generated, and sets its internal state so that subsequent $\mathsf{ZMM.Setup}$ queries fail.
- On a query $\mathsf{ZMM.Encode}(k, x, S)$, where $x \in \mathbb{Z}$, the oracle first checks that $k = \mathsf{sp}$ and $S \subseteq \mathcal{U}$. If either check fails, then the oracle returns $\perp$. Otherwise, the oracle chooses values $\eta_x = [\eta_{x,1}, \ldots, \eta_{x,k}] \leftarrow \mathsf{Uniform}(\mathbb{Z}^*_{N_1 \cdots N_k})$, $\zeta_x \leftarrow \mathsf{Uniform}(\mathbb{Z}^*_{N_{k+1}})$, and $\nu_{x,\mathrm{L}}, \nu_{x,\mathrm{R}} \leftarrow \mathsf{Uniform}(\mathbb{Z}^*_{N_{k+2}})$, and issues queries $\mathsf{MM.Encode}(\mathsf{MM.sp}, [x_1, \ldots, x_k, \zeta_x, \nu_{x,\mathrm{L}}], S_{\mathrm{L}})$ and $\mathsf{MM.Encode}(\mathsf{MM.sp}, [\eta_{x,1}, \ldots, \eta_{x,k}, \zeta_x, \nu_{x,\mathrm{R}}], S_{\mathrm{R}})$ to $\mathcal{M}_C$, obtaining handles $\hat{x}_{\mathrm{L}}$ and $\hat{x}_{\mathrm{R}}$, respectively. The oracle returns $h := (\hat{x}_{\mathrm{L}}, \hat{x}_{\mathrm{R}})$.
- The oracle also answers $\mathsf{MM.Add}, \mathsf{MM.Mult}, \mathsf{MM.ZeroTest}$ queries, by forwarding each such query to its internal copy of $\mathcal{M}_C$ and answering with the value that $\mathcal{M}_C$ outputs.

When we refer to $\mathcal{M}_Z$'s internal table, we implicitly refer to that of its internal copy of $\mathcal{M}_C$.

Definition 3.7 (ZMM-Transformed Generic Model)**.** Let MM be a composite-order multilinear map (Definition 2.3), and let ZMM be the corresponding transformed construction (Construction 3.1). Fix an adversary (a stateful oracle machine) $\mathcal{A}$, and a security parameter $\lambda \in \mathbb{N}$. A security game in the ZMM-transformed generic model takes the following form:

1. The adversary sends a description of a top-level index set $\mathcal{U}$, and an integer $k \in \mathbb{N}$.
2. Let $\mathcal{M}$ be the stateful oracle implementing the transformed multilinear map operations of ZMM (Definition 3.6). The oracle $\mathcal{M}$ is initialized by running $(\mathsf{ZMM.pp}, \mathsf{ZMM.sp}, N_1, \ldots, N_k) \leftarrow \mathsf{ZMM.Setup}(\mathcal{U}, 1^\lambda, k)$, and the adversary receives the public parameters $\mathsf{ZMM.pp}$.
3. The adversary interacts with $\mathcal{M}$ via the operations $\mathsf{MM.Add}, \mathsf{MM.Mult}, \mathsf{MM.ZeroTest}$. In addition, the adversary is given oracle access to the operation $\mathsf{ZMM.Encode}(\mathsf{ZMM.sp}, \cdot, \cdot)$.
4. Finally, the adversary outputs a string $\hat{v} \in \{0,1\}^*$.

Remark 3.8 (Oracle Queries Referring to Formal Polynomials)**.** Although the generic multilinear map oracle is defined formally in terms of "handles" (Definition 3.5), it is usually more intuitive to regard each oracle query as *referring* to a formal query polynomial. The formal variables are specified by the expressions initially supplied to the MM.Encode procedure (as determined by the details of the construction), and the adversary can construct terms that refer to new polynomials by making oracle queries for the generic-model ring operations MM.Add, MM.Mult. Rather than operating on a "handle", then, each valid MM.ZeroTest query refers to a formal query polynomial[4] encoded at the top-level index set $\mathcal{U}$. The result of the query is "zero" precisely if the polynomial evaluates to zero, when its variables are instantiated with the joint distribution over their values in $\mathbb{Z}_N$, generated as in the real security game. For the full formal description, we refer the reader to [Zim14, Appendix B].

[4]To represent a query polynomial concretely, we can use an arithmetic circuit—and thus, for instance, we can still perform efficient manipulations on query polynomials that have been subjected to repeated squaring.

Definition 3.9 (Formal Variables in the ZMM-Transformed Generic Model)**.** Fix an adversary $\mathcal{A}$ in the ZMM-transformed generic model (Definition 3.7), and let Q be the number of ZMM.Encode queries made by $\mathcal{A}$ to the oracle $\mathcal{M}_Z$. We say the *formal variables* of the model for this adversary are the variables $\hat{t}_{\mathrm{L}}, \hat{t}_{\mathrm{R}}$ (referring to the values generated by MM.Encode during ZMM.Setup), and, for each query index $i \in [Q]$, the variables $\hat{x}_{i,\mathrm{L}}, \hat{x}_{i,\mathrm{R}}$ (referring to the pair of values generated by MM.Encode during the adversary's i^{th} query to ZMM.Encode).

Remark 3.10 (Nontrivial Encodings)**.** In general, our security theorems below are only concerned with *nontrivial* 0-encodings, i.e., encodings whose real values are 0 but whose formal polynomials are not identically zero. We cannot rule out trivial 0-encodings, since the adversary can always compute the encoding $\hat{v} - \hat{v}$ for any encoding $\hat{v}$, obtaining a (trivial) 0-encoding. Concretely, these trivial 0-encodings are useless to the adversary, since an encoding whose formal polynomial is identically zero will be the integer $0 \in \mathbb{Z}_{N_{\text{outer}}}$ in the CLT construction (and will be similarly useless in other known multilinear maps).

3.3 Main Security Theorems

We now present our two main security theorems, capturing the desired properties of our zero-immunizing transformation (Construction 3.1). The first theorem says that in our transformed scheme, the adversary cannot construct a pair of orthogonal encodings below the top level.

Definition 3.11 (Zero Product Security Game)**.** Let MM be a poly-degree multilinear map (Definition 2.5), and let ZMM be the corresponding zero-immunized map (Construction 3.1). Let $\lambda \in \mathbb{N}$ be a security parameter, and fix an adversary $\mathcal{A}$ in the transformed generic model for ZMM with oracle $\mathcal{M}$. Let $\hat{v}$ be the adversary's output at the end of the security game. We say that $\mathcal{A}$ wins the *zero product security game* for $(\mathsf{MM}, \mathsf{ZMM})$ if $\hat{v}$ is a pair (h_1, h_2) such that h_1, h_2 refer to formal polynomials in $\mathcal{M}$'s table, resp., v_1, v_2, each not identically zero, at index sets $S_1, S_2 \subsetneq \mathcal{U}_Z$, resp.; the real value of $v_1 \cdot v_2$ is 0; and $S_1 S_2 = \mathcal{U}_Z$.

Theorem 3.12 (Zero Product Security)**.** *Let* MM *be a poly-degree multilinear map (Definition 2.5), and let* ZMM *be the corresponding zero-immunized map (Construction 3.1). For all adversaries $\mathcal{A}$, the probability that $\mathcal{A}$ wins the zero product security game for* $(\mathsf{MM}, \mathsf{ZMM})$ *(Definition 3.11) with security parameter $\lambda \in \mathbb{N}$ is* $\mathrm{negl}(\lambda)$.

Proof. Deferred to Section 3.5. □

Since even our strengthened version of the [CHL+14] attack (Section 5.1) requires orthogonal encodings, Theorem 3.12 implies that the attack does not apply to CLT if it is augmented with our new zero-immunizing transformation (assuming the public 0-encodings of CLT are suppressed).

Our second security theorem says that the adversary cannot construct a 0-encoding at any index set that is a proper subset of the top-level index set.[5]

Definition 3.13 (Low-Level Zero Encoding Security Game)**.** Let MM be a poly-degree multilinear map (Definition 2.5), and let ZMM be the corresponding zero-immunized map (Construction 3.1). Let $\lambda \in \mathbb{N}$ be a security parameter and fix an adversary $\mathcal{A}$ in the transformed generic model with oracle $\mathcal{M}$. Let h be $\mathcal{A}$'s output at the end of the experiment. We say that $\mathcal{A}$ wins the *low level zero encoding security game* for $(\mathsf{MM}, \mathsf{ZMM})$ if h is a handle that refers to a formal polynomial v at index set $S \subsetneq \mathcal{U}_Z$ in $\mathcal{M}$'s table, such that v is not identically zero but its real value is 0.

[5] We also note that Definition 3.13 is incomparable to Definition 3.11, since an adversary might be able to produce an encoding of zero at an index set $\mathcal{S} \subsetneq \mathcal{U}$, yet still be unable to construct an encoding at $\mathcal{U}/\mathcal{S}$ in order to complete that encoding to $\mathcal{U}$. Conversely, even if the adversary cannot construct 0-encodings below $\mathcal{U}$, it might be able to construct orthogonal encodings below the top-level whose product yields a 0-encoding at the top level.

Theorem 3.14 (Low-Level Zero Encoding Security)**.** *Let* MM *be a poly-degree multilinear map (Definition 2.5), and let* ZMM *be the corresponding zero-immunized map (Construction 3.1). For all adversaries* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *wins the low level zero encoding security game for* $(\mathsf{MM}, \mathsf{ZMM})$ *(Definition 3.13) with security parameter* $\lambda \in \mathbb{N}$ *is* $\mathrm{negl}(\lambda)$*.*

Proof. Deferred to Section 3.6. □

While we do not know of an attack that uses low-level encodings of zero without also needing a zero-product (Definition 3.11), in establishing Theorem 3.14 we err on the side of caution. Moreover, if new multilinear maps are discovered in which encodings are deterministic—for example, as in the setting of bilinear groups—then Theorem 3.14 shows that with our transformation, the adversary gains no benefit from the ability to equality-test at arbitrary index sets below the top level $\mathcal{U}$, since he cannot produce any encodings for which such a test would pass.

Remark 3.15 (Security for Maps with Arbitrary Degree)**.** The two main security theorems above pertain only to *poly-degree* multilinear maps (Definition 2.5), in which the maximum degree of the top-level index set $\mathcal{U}$ is $\mathrm{poly}(\lambda)$. This is the case for all known multilinear map constructions, including CLT, due to the noise growth. However, if in the future new multilinear maps are discovered that are "clean", i.e., permit operations of arbitrary degree (even exponential), then we would need to modify these results slightly. Zimmerman [Zim14] explores this extension in a similar context, and shows that many generic-model results extend to the setting of "clean" multilinear maps if we also assume the hardness of factoring, by means of a *computational* variant of the Schwartz-Zippel lemma [BL97, Zim14]. Applying the techniques of [Zim14] it is straightforward to generalize the results of this section to the "clean" setting as well.

3.4 Structure Lemmas

Before presenting the proofs of Theorems 3.12 and 3.14, we require a few basic "structure lemmas", showing how our design of the index sets in the ZMM-transformation has restricted the formal polynomials that an adversary can produce in the transformed generic model.

Lemma 3.16 (Characterization of Low-Level Formal Polynomials)**.** *Fix an adversary* $\mathcal{A}$ *in the transformed generic model, and consider a formal polynomial* w*, not identically zero, produced by* $\mathcal{A}$ *at some index set* $S \subsetneq \mathcal{U}_Z$ *over the formal variables* $\hat{t}_{\mathrm{L}}, \hat{t}_{\mathrm{R}}, \hat{x}_{1,\mathrm{L}}, \ldots, \hat{x}_{Q,\mathrm{L}}, \hat{x}_{1,\mathrm{R}}, \ldots, \hat{x}_{Q,\mathrm{R}}$ *(Definition 3.9). Here,* $Q \in \mathbb{N}$ *is the number of* $\mathsf{ZMM.Encode}$ *queries made by* $\mathcal{A}$*. Then one of the following conditions holds:*

1. *We have* $S \subseteq \mathcal{U}_{\mathrm{L}}$*, and for some multivariate polynomial function* p_{L}*, the following holds:*

$$w \equiv p_{\mathrm{L}}(\hat{x}_{1,\mathrm{L}}, \ldots, \hat{x}_{Q,\mathrm{L}})$$

2. *We have* $S \subseteq \mathcal{U}_{\mathrm{R}}$*, and for some multivariate polynomial function* p_{R}*, the following holds:*

$$w \equiv p_{\mathrm{R}}(\hat{x}_{1,\mathrm{R}}, \ldots, \hat{x}_{Q,\mathrm{R}})$$

3. *We have* $S \supseteq \mathcal{U}_{\mathrm{R}}T$*, and for some multivariate polynomial function* p_{L}*, the following holds:*

$$w \equiv \hat{t}_{\mathrm{L}} \cdot p_{\mathrm{L}}(\hat{x}_{1,\mathrm{L}}, \ldots, \hat{x}_{Q,\mathrm{L}})$$

4. *We have* $S \supseteq \mathcal{U}_{\mathrm{L}}T$*, and for some multivariate polynomial function* p_{R}*, the following holds:*

$$w \equiv \hat{t}_{\mathrm{R}} \cdot p_{\mathrm{R}}(\hat{x}_{1,\mathrm{R}}, \ldots, \hat{x}_{Q,\mathrm{R}})$$

5. *We have* $S \subseteq \mathcal{U}_{\mathrm{L}}\mathcal{U}_{\mathrm{R}}$, $S \not\subseteq \mathcal{U}_{\mathrm{L}}$, *and* $S \not\subseteq \mathcal{U}_{\mathrm{R}}$; *and for some multivariate polynomial function* p, *the following holds:*

$$w \equiv p(\hat{x}_{1,\mathrm{L}}, \ldots, \hat{x}_{Q,\mathrm{L}}, \hat{x}_{1,\mathrm{R}}, \ldots, \hat{x}_{Q,\mathrm{R}})$$

Proof. We first show the analogous claim for formal monomials v; the claim will then follow by considering all monomials v in the expansion of w.

Let v be a formal monomial at some index set $S \subsetneq \mathcal{U}_Z$ over the formal variables $\hat{t}_{\mathrm{L}}, \hat{t}_{\mathrm{R}}, \hat{x}_{1,\mathrm{L}}, \ldots, \hat{x}_{Q,\mathrm{L}}$, $\hat{x}_{1,\mathrm{R}}, \ldots, \hat{x}_{Q,\mathrm{R}}$. We proceed by case analysis:

- Suppose v contains the variable $\hat{t}_{\mathrm{L}}$. Since the index set of $\hat{t}_{\mathrm{L}}$ is $\mathcal{U}_{\mathrm{R}}T$, we have $S \supseteq \mathcal{U}_{\mathrm{R}}T$. We note that v cannot contain the variable $\hat{t}_{\mathrm{R}}$, since the top level $\mathcal{U}_Z$ contains only one copy of T. So the only remaining formal variables are $\hat{x}_{1,\mathrm{L}}, \ldots, \hat{x}_{Q,\mathrm{L}}, \hat{x}_{1,\mathrm{R}}, \ldots, \hat{x}_{Q,\mathrm{R}}$. Since each of the variables $\hat{x}_{1,\mathrm{R}}, \ldots, \hat{x}_{Q,\mathrm{R}}$ has an index set that is a nonempty subset of $\mathcal{U}_{\mathrm{R}}$, it follows that none of these variables can appear in v either, since $\mathcal{U}$ contains $\mathcal{U}_{\mathrm{R}}$ only once (and the index set of $\hat{t}_{\mathrm{L}}$ already contains $\mathcal{U}_{\mathrm{R}}$). So v satisfies case 3 of the lemma.

- Suppose v contains the variable $\hat{t}_{\mathrm{R}}$. Then by the same argument as in the first case, v satisfies case 4 of the lemma.

- Suppose v contains neither of the variables $\hat{t}_{\mathrm{L}}, \hat{t}_{\mathrm{R}}$. Then the only remaining formal variables are $\hat{x}_{1,\mathrm{L}}, \ldots, \hat{x}_{Q,\mathrm{L}}, \hat{x}_{1,\mathrm{R}}, \ldots, \hat{x}_{Q,\mathrm{R}}$, and by definition v satisfies either case 1, case 2, or case 5 of the lemma.

Now, to complete the proof, we consider monomials v in the expansion of w, using the fact that only terms at the same index set can be added. We again proceed by case analysis:

- Suppose some monomial in the expansion of w is of the form $v = h(\hat{x}_{1,\mathrm{L}}, \ldots, \hat{x}_{Q,\mathrm{L}})$, at an index set $\mathcal{S} \subseteq \mathcal{U}_{\mathrm{L}}$ (case 1). Then all monomials in the expansion of w must have this form (no other monomials are encoded at a subset of $\mathcal{U}_{\mathrm{L}}$), and we conclude that w satisfies case 1 of the lemma.

- Suppose some monomial in the expansion of w is of the form $v = h(\hat{x}_{1,\mathrm{R}}, \ldots, \hat{x}_{Q,\mathrm{R}})$ (case 2). Then by the same argument as in the previous case, w satisfies case 2 of the lemma.

- Suppose some monomial in the expansion of w is of the form $v = \hat{t}_{\mathrm{L}} \cdot h(\hat{x}_{1,\mathrm{L}}, \ldots, \hat{x}_{Q,\mathrm{L}})$, so that $\mathcal{S} \supseteq \mathcal{U}_{\mathrm{L}}T$ (case 3). The only other monomials encoded at a superset of $\mathcal{U}_{\mathrm{L}}T$ satisfy either case 3 or case 4 of the lemma; it suffices to rule out case 4. We note that the only index set that is a superset of both $\mathcal{U}_{\mathrm{L}}T$ and $\mathcal{U}_{\mathrm{R}}T$ is $\mathcal{U}_Z$, the top level. By assumption, $S \neq \mathcal{U}_Z$ (since w is assumed to be a low-level formal polynomial), so it follows that w satisfies case 3 of the lemma.

- Suppose some monomial in the expansion of w is of the form $v = \hat{t}_{\mathrm{R}} \cdot h(\hat{x}_{1,\mathrm{R}}, \ldots, \hat{x}_{Q,\mathrm{R}})$ (case 4). Then by the same argument as in the previous case, w satisfies case 4 of the lemma.

- Suppose some monomial in the expansion of w is of the form $h(\hat{x}_{1,\mathrm{L}}, \ldots, \hat{x}_{Q,\mathrm{L}}, \hat{x}_{1,\mathrm{R}}, \ldots, \hat{x}_{Q,\mathrm{R}})$. By the index set constraint, all monomials in the expansion of w must have this form, and so w satisfies case 5 of the lemma. □

Corollary 3.17 (Characterization of Completable Formal Polynomials)**.** *Fix an adversary* $\mathcal{A}$ *in the transformed generic model, and let* w, w' *be formal polynomials produced by* $\mathcal{A}$, *each not identically zero, at nonempty index sets* $S, S' \subset \mathcal{U}_Z$, *resp., over the formal variables* $\hat{t}_{\mathrm{L}}, \hat{t}_{\mathrm{R}}, \hat{x}_{1,\mathrm{L}}, \ldots, \hat{x}_{Q,\mathrm{L}}, \hat{x}_{1,\mathrm{R}}, \ldots,$ $\hat{x}_{Q,\mathrm{R}}$ *(Definition 3.9). As above, let* $Q \in \mathbb{N}$ *denote the number of* MM.Encode *queries made by* $\mathcal{A}$. *If* $SS' = \mathcal{U}_Z$, *then the formal polynomial* w *and its index set* S *satisfy one of cases 1-4 of Lemma 3.16.*

Proof. Since S, S' are proper subsets of $\mathcal{U}_Z$, each of w, w' is a low-level encoding, satisfying the conditions of Lemma 3.16. Thus w must satisfy one of the cases 1-5 of Lemma 3.16, and it suffices to rule out case 5. Suppose that w satisfies case 5 of Lemma 3.16, so that $S \subseteq \mathcal{U}_{\text{L}}\mathcal{U}_{\text{R}}$, but $S \not\subseteq \mathcal{U}_{\text{L}}$ and $S \not\subseteq \mathcal{U}_{\text{R}}$. Now, since $SS' = \mathcal{U}_Z = \mathcal{U}_{\text{L}}\mathcal{U}_{\text{R}}T$ and $T \notin S$, it follows that w' satisfies either case 3 or case 4 of Lemma 3.16. Without loss of generality suppose w' satisfies case 3, so that $S' \supseteq \mathcal{U}_{\text{R}}T$. Then $S \subseteq \mathcal{U}_Z/(\mathcal{U}_{\text{R}}T) = \mathcal{U}_{\text{L}}$, contradicting the assumption that $S \not\subseteq \mathcal{U}_{\text{L}}$. We conclude that w cannot satisfy case 5 of Lemma 3.16, as desired. □

We are now equipped to prove the main security theorems.

3.5 Proof of Theorem 3.12

Suppose $\mathcal{A}$ wins the zero-product game with non-negligible probability. Then, by definition, at the end of the game $\mathcal{A}$ outputs a value h of the form (h_1, h_2), where h_1 refers to a formal polynomial v_1 at an index set $S_1 \subsetneq \mathcal{U}_Z$ and h_2 refers to a formal polynomial v_2 at an index set S_2, satisfying $S_1S_2 = \mathcal{U}_Z$ (and each of v_1, v_2 is not identically zero). Furthermore, letting x_1 (resp., x_2) be the real value of v_1 (resp., v_2), we are given $x_1 \cdot x_2 = 0$, and thus $v := v_1 \cdot v_2$ is a formal polynomial whose real value is 0 and whose index set is $\mathcal{U}_Z$. We proceed by case analysis on the form of v_1, as determined by Corollary 3.17:

1. Suppose $v_1 = p_{\text{L},1}(\hat{x}_{1,\text{L}}, \ldots, \hat{x}_{Q,\text{L}})$ for some polynomial $p_{\text{L},1}$. Then $S_1 \subseteq \mathcal{U}_{\text{L}}$, so $S_2 \supseteq \mathcal{U}_{\text{R}}T$, and thus Corollary 3.17 gives $v_2 = \hat{t}_{\text{L}} \cdot p_{\text{L},2}(\hat{x}_{1,\text{L}}, \ldots, \hat{x}_{Q,\text{L}})$ for some polynomial $p_{\text{L},2}$. Defining $p_{\text{L}} = p_{\text{L},1}p_{\text{L},2}$, we conclude that $v = \hat{t}_{\text{L}} \cdot p_{\text{L}}(\hat{x}_{1,\text{L}}, \ldots, \hat{x}_{Q,\text{L}})$. The real value of v in the component modulo N_{k+1} is thus $1 \cdot p_{\text{L}}(\zeta_{x_1}, \ldots, \zeta_{x_Q})$, and by the Schwartz-Zippel lemma, its value is nonzero with overwhelming probability (since v_1, v_2, and hence $p_{\text{L},1}, p_{\text{L},2}$, are not identically zero; and the total degree of each is at most $\deg(\mathcal{U}) = \text{poly}(\lambda)$). This contradicts our assumption that the real value of v is 0.

2. Suppose $v_1 = p_{r,1}(\hat{x}_{1,\text{R}}, \ldots, \hat{x}_{Q,\text{R}})$ for some polynomial $p_{r,1}$. Then $S_1 \subseteq \mathcal{U}_{\text{R}}$, so $S_2 \supseteq \mathcal{U}_{\text{L}}T$, and the claim follows by the argument of the first case.

3. Suppose $v_1 = \hat{t}_{\text{L}} \cdot p_{\text{L},1}(\hat{x}_{1,\text{L}}, \ldots, \hat{x}_{Q,\text{L}})$ for some polynomial $p_{\text{L},1}$. Then $S_1 \supseteq \mathcal{U}_{\text{R}}T$, so $S_2 \subseteq \mathcal{U}_{\text{L}}$, and the claim follows by the argument of the first case.

4. Suppose $v_1 = \hat{t}_{\text{R}} \cdot p_{r,1}(\hat{x}_{1,\text{R}}, \ldots, \hat{x}_{Q,\text{R}})$ for some polynomial $p_{r,1}$. Then $S_1 \supseteq \mathcal{U}_{\text{L}}T$, so $S_2 \subseteq \mathcal{U}_{\text{R}}$, and the claim follows by the argument of the first case.

3.6 Proof of Theorem 3.14

Suppose $\mathcal{A}$ wins the low-level zero game with non-negligible probability. Then, by definition, at the end of the game, $\mathcal{A}$ outputs a value h, where h is a handle referring to a formal polynomial v at an index set $S \subsetneq \mathcal{U}_Z$, such that the real value of v is 0 (and v is not identically zero). We proceed by case analysis on the form of v, as determined by Lemma 3.16:

1. Suppose $v = p_{\text{L}}(\hat{x}_{1,\text{L}}, \ldots, \hat{x}_{Q,\text{L}})$ for some polynomial p_{L}. The real value of $\hat{v}$ in the component modulo N_{k+1} is thus $p_{\text{L}}(\zeta_{x_1}, \ldots, \zeta_{x_Q})$, and by the Schwartz-Zippel lemma, its value is nonzero with overwhelming probability (since v, and hence p_{L}, is not identically zero; and its total degree is at most $\deg(\mathcal{U}) = \text{poly}(\lambda)$). This contradicts our assumption that the real value of v is 0.

2. Suppose $v = p_{\mathrm{R}}(\hat{x}_{1,\mathrm{R}}, \ldots, \hat{x}_{Q,\mathrm{R}})$ for some polynomial p_{R}. The claim follows by the Schwartz-Zippel lemma as in the first case.

3. Suppose $v = \hat{t}_{\mathrm{L}} \cdot p_{\mathrm{L}}(\hat{x}_{1,\mathrm{L}}, \ldots, \hat{x}_{Q,\mathrm{L}})$ for some polynomial p_{L}. The claim follows by the Schwartz-Zippel lemma as in the first case.

4. Suppose $v = \hat{t}_{\mathrm{R}} \cdot p_{\mathrm{R}}(\hat{x}_{1,\mathrm{R}}, \ldots, \hat{x}_{Q,\mathrm{R}})$ for some polynomial p_{R}. The claim follows by the Schwartz-Zippel lemma as in the first case.

5. Suppose $v = p(\hat{x}_{1,\mathrm{L}}, \ldots, \hat{x}_{Q,\mathrm{L}}, \hat{x}_{1,\mathrm{R}}, \ldots, \hat{x}_{Q,\mathrm{R}})$ for some polynomial p. The real value of $\hat{v}$ in the component modulo N_{k+2} is then $p(\nu_{x_1,\mathrm{L}}, \ldots, \nu_{x_Q,\mathrm{L}}, \nu_{x_1,\mathrm{R}}, \ldots, \nu_{x_Q,\mathrm{R}})$, and the claim again follows by the Schwartz-Zippel lemma.

4 Zero-Immunizing CLT

In this section, we describe how to apply our zero-immunizing transformation to the CLT multilinear map [CLT13], first implementing the basic abstract operations (Definition 2.3), then presenting candidate constructions for extending our zero-immunized version to recover re-randomization and public-encoding functionality.

4.1 The CLT Multilinear Map

We now briefly describe the operation of the CLT multilinear map [CLT13]. Our presentation is similar to that of [CHL+14], except that in order to use CLT as a composite-order multilinear map (making full use of the message space $\mathbb{Z}_N$ as a direct product $\mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_k}$, as in [GLW14, GLSW14, GGHZ14, Zim14]), we require some additional modifications. Specifically, we use the modification described by Gentry et al. [GLW14, Appendix B], in which each modulus $N_1, \ldots, N_k$ itself is a product of Θ primes $g_1, \ldots, g_\Theta$ for some $\Theta = \mathrm{poly}(\lambda)$, in order to rule out certain lattice reduction attacks based on encodings within subrings. For simplicity, we will also describe the CLT multilinear map only in the symmetric setting, in which the top-level index set $\mathcal{U}$ is just Z^κ for a fresh formal symbol Z. Our description generalizes immediately to the asymmetric case with many distinct indices (Definition 2.3), as detailed in [GLW14, Appendix B].

The CLT multilinear map operates as follows. First, we introduce the following integer parameters (our notation here largely follows that of [CLT13, CHL+14]):

- λ: the security parameter
- κ: the degree of multilinearity (for a instantiation with top-level index set Z^κ)
- k: the number of components in the message space (the message space is a direct product $\mathbb{Z}_N = \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_k}$)
- Θ: the number of primes that comprise each of the moduli $N_1, \ldots, N_k$ [GLW14]
- n: the number of primes that comprise the modulus for the message space ($n = k\Theta$)
- α: the bit-length of the message slots
- ρ: the bit-length of each prime g_i
- η: the bit-length of each prime p_i
- τ: the number of level-1 encodings of zero in the public parameters
- t: the number of level-0 encodings in the public parameters
- δ: the bit-length of the entries in the zero-test matrix H
- μ: the bit-length of the coefficients in the subset sum for the re-randomization procedure

– ξ: the number of bits to use for the extract operation

Coron et al. [CLT13] (and Gentry et al. [GLW14], in the case of Θ) suggest setting the above parameters as follows:

– $\Theta = (\rho \cdot \eta)^{1+\varepsilon}$ for constant $\varepsilon > 0$.
– $n = \omega(\eta \log \lambda)$ to prevent lattice-based attacks on the encodings. (Since we already require $n = k\Theta = \Omega(\eta^{1+\varepsilon})$, this constraint is satisfied by default.)
– $\alpha = \lambda$ so the order of the ring $\mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_k}$ does not contain small prime factors.
– $\rho = \Omega(\lambda)$ to prevent brute force attacks.
– $\eta \geq \rho_\kappa + \alpha + 2\delta + \lambda + 8$, where ρ_κ is the maximum bit size of the randomizers r_i in a level-κ encoding.
– $\tau \geq n \cdot (\rho + \log(2n)) + 2\lambda$ in order to apply the lattice leftover hash lemma from [CLT13, §4].
– $t \geq n \cdot \alpha + 2\lambda$ in order to apply the lattice leftover hash lemma from [CLT13, Lemma 1].
– $\delta = \Omega(\lambda)$ to avoid the GCD attack of [CLT13, §5.2]
– $\mu \geq \alpha + \rho + \lambda$ to apply the lattice leftover hash lemma from [CLT13, §4]
– $\xi = \eta - \delta - \rho_\kappa - \lambda - 3$ to ensure correctness of the zero-test.

Instance generation. The instance generation algorithm $\mathsf{CLT.InstGen}(\lambda, \kappa, k)$ takes as arguments a security parameter λ, the multilinearity degree κ, and the number of components in the message space k. For each $i \in [n]$, it generates an α-bit prime g_i, and an η-bit prime p_i. For $i \in [k]$, it sets

$$N_i = \prod_{i=(k-1)\Theta+1}^{k\Theta} g_i.$$

Finally, it sets $N = \prod_{i\in[k]} N_i$ and $N_{\text{outer}} = \prod_{i\in[n]} p_i$. CLT also generates a single value $z \in \mathbb{Z}_{N_{\text{outer}}}$ as well as an n-dimensional vector of zero test elements $\mathbf{p}_{\text{zt}}$ where the j^{th} zero test element $[\mathbf{p}_{\text{zt}}]_j$ is given by

$$[\mathbf{p}_{\text{zt}}]_j = \sum_{i\in[n]} h_{ij} \cdot \left(z^\kappa \cdot g_i^{-1} \bmod p_i\right) \cdot \prod_{i'\neq i} p_i \pmod{N_{\text{outer}}},$$

and the matrix $H = (h_{ij})$ is as defined in [CLT13, §3].

Next, CLT generates a set of t level-0 encodings, $x'_1, \ldots, x'_t$, of random values $a_i \leftarrow \mathsf{Uniform}(\mathbb{Z}_N)$ where

$$x'_{ij} \equiv (a_i \bmod g_j) + r'_{ij} \cdot g_j \pmod{p_j},$$

and each $r'_{ij} \leftarrow \mathsf{Uniform}([-2^\rho + 1, 2^\rho - 1])$. In addition, CLT generates a level-1 encoding y of 1, where for all $i \in [n]$, it is the case that

$$y \equiv (r_i \cdot g_i + 1)/z \pmod{p_i}$$

and $r_i \leftarrow \mathsf{Uniform}([-2^\rho + 1, 2^\rho - 1])$.

The CLT scheme then generates additional encodings needed for re-randomization that are also included in the public parameters. These additional elements are described later in this section. Finally, $\mathsf{CLT.InstGen}$ outputs the public parameters $\mathsf{CLT.pp} \leftarrow (\text{params}, \mathbf{p}_{\text{zt}})$, where params includes the level-0 encodings $x'_1, \ldots, x'_t$, the level-1 encoding y of 1, the encodings necessary for re-randomization, and the parameters $\lambda, \kappa, k, \Theta, n, \rho, \eta, \tau, t, \delta, \mu, \xi$, and N_{outer}. The secret parameters $\mathsf{CLT.sp}$ consist of the value z and the prime factors of N and N_{outer}. Note that CLT only need to

publish the level-0 encodings $x'_1, \ldots, x'_t$, the level-1 encoding y, and the re-randomization encodings if the instance of CLT is required to support *public encoding* (see Remark 2.7). For constructions based on "secret-key" multilinear maps, these elements do not need to be included in the public parameters.

Encodings in CLT. In the CLT scheme, an integer $c \in \mathbb{Z}$ is a valid encoding of a message $a \in \mathbb{Z}_N$ at level d if c satisfies the following condition for all $i \in [n]$:

$$c \equiv \frac{r_i g_i + (a \bmod g_i)}{z^d} \pmod{p_i},$$

where the r_i are integers bounded in a specified range. We refer the reader to [CLT13, GLW14] for further details.

Adding and multiplying encodings. The $\mathsf{CLT.Add}(\mathsf{CLT.pp}, c_1, c_2)$ function takes as input two encodings c_1 and c_2 at the same index set $\mathcal{S}$ and outputs an encoding $c' \leftarrow c_1 + c_2 \pmod{\mathbb{Z}_{N_{\text{outer}}}}$ at index set $\mathcal{S}$. The $\mathsf{CLT.Mult}(\mathsf{CLT.pp}, c_1, c_2)$ takes as input two encodings c_1, c_2 at index sets $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively, such that $\mathcal{S}_1\mathcal{S}_2 \subseteq \mathcal{U}$ and outputs an encoding $c' \leftarrow c_1 c_2 \pmod{\mathbb{Z}_{N_{\text{outer}}}}$ at index set $\mathcal{S}_1\mathcal{S}_2$.

Zero-testing. The $\mathsf{CLT.ZeroTest}(\mathsf{CLT.pp}, [c]_{\mathcal{U}})$ takes as input an encoding c at the top-level set $\mathcal{U}$ and outputs 1 if for all $j \in [n]$, $c \cdot [\mathbf{p}_{\text{zt}}]_j \pmod{N_{\text{outer}}} < N_{\text{outer}} \cdot 2^{-\xi}$. Otherwise, it outputs 0.

Extraction. The $\mathsf{CLT.Extract}(\mathsf{CLT.pp}, c)$ extract a random function of the value $m \in \mathbb{Z}_N$ from a top-level encoding c of m. The algorithm first collects the ξ most significant bits of $c \cdot [p_{\text{zt}}]_j$ for each $j \in [n]$, and then applies a strong randomness extractor to the resulting bits.

Re-randomization in CLT and the canonical noise distribution. The re-randomization algorithm $\mathsf{CLT.ReRand}(\mathsf{CLT.pp}, c)$ takes an encoding c of a value x at level d and produces a new encoding c' of the same value x at level d. The requirement is that for any two encodings c_1, c_2 of the same value x, the output distributions of $\mathsf{CLT.ReRand}(\mathsf{CLT.pp}, c_1)$ is statistically close to the output distribution of $\mathsf{CLT.ReRand}(\mathsf{CLT.pp}, c_2)$. For simplicity, we just describe the case where $d = 1$; the same principles generalize naturally to higher-level encodings.

A level-1 encoding c of x in the CLT scheme satisfies the following relation: for all $i \in [n]$, $c \equiv (r_i g_i + x)/z \pmod{p_i}$, where r_i is an integer drawn from some specified range. Furthermore, when CLT encodings are given to the adversary in constructions or assumptions, they are typically first run through CLT's re-randomization procedure, which takes an encoded scalar and "washes it out", so that the randomizers r_i are (statistically close to) some canonical noise distribution, independent of how the encoding originated.

Before describing the canonical distribution of the r_i's in the CLT scheme, we present some definitions similar to those given in [CLT13, §4]. Let L be a lattice of rank n with basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$. Then, every vector $\mathbf{x} \in \mathbb{Z}^n$ can be written as a linear combination $\sum_{i\in[n]} \alpha_i \mathbf{b}_i$ for some $\alpha_i \in \mathbb{R}$. Moreover, for every $\mathbf{x} \in \mathbb{Z}^n$, there exists a unique vector $\mathbf{a} \in L$ such that $\mathbf{y} = \mathbf{x} - \mathbf{a} = \sum_{i\in[n]} \alpha'_i \mathbf{b}_i$, where $0 \le \alpha'_i < 1$. In this case, we say $\mathbf{y} = \mathbf{x} \bmod L$. Thus, every element in $\mathbb{Z}^n/L$ has a unique representative in the half-open parallelepiped spanned by $\mathbf{b}_1, \ldots, \mathbf{b}_n$. We write $\mathcal{D}_{\mathbf{B}}$ to denote the distribution obtained by sampling a uniformly random element in $\mathbb{Z}^n/L$ and taking its unique representative in the half-open parallelepiped generated by the basis $\mathbf{B}$. Lastly, given a basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, we write $\mu\mathbf{B}$ to denote the basis $(\mu\mathbf{b}_1, \ldots, \mu\mathbf{b}_n)$.

In the CLT scheme, the canonical noise distribution for encodings is $\mathcal{D}_{2^\mu \mathbf{\Pi}}$, where $\mathbf{\Pi} \in \mathbb{Z}^{n \times n}$ is a matrix constructed as follows. The diagonal entries $\mathbf{\Pi}_{ii}$ are drawn from $\mathsf{Uniform}([n \cdot 2^\rho + 1, (n+1) \cdot 2^\rho - 1])$ while the off-diagonal entries $\mathbf{\Pi}_{ij}$ $(j \neq i)$ are drawn from $\mathsf{Uniform}([-2^\rho + 1, 2^\rho - 1])$. We note that $\mathbf{\Pi}$ is a diagonally-dominant matrix, according the following definition:

Definition 4.1 (Diagonally-Dominant Matrices)**.** *Let $\mathbf{M} \in \mathbb{R}^{n \times n}$ be a matrix, and write $\Lambda_i(\mathbf{M})$ to denote the sum $\sum_{j \neq i} |\mathbf{M}_{ij}|$. We say that $\mathbf{M}$ is diagonally dominant if $|\mathbf{M}_{ii}| > \Lambda_i(\mathbf{M})$ for all $i \in [n]$.*

In the following exposition, we will make use of the following two properties on diagonally dominant matrices. These are taken from [CLT13, Var75, Pri51]:

Fact 4.2. *Let $\mathbf{M} \in \mathbb{R}^{n \times n}$ be a diagonally dominant matrix. Then $\mathbf{M}$ is invertible and $\left\|\mathbf{M}^{-1}\right\|_\infty \leq \max_{i \in [n]}(|\mathbf{M}_{ii}| - \Lambda_i(\mathbf{M}))^{-1}$. Here, $\|\cdot\|_\infty$ denotes the operator norm on $n \times n$ matrices with respect to the ℓ_∞ norm in $\mathbb{R}^n$, that is, for all $\mathbf{M} \in \mathbb{R}^{n \times n}$, $\|\mathbf{M}\|_\infty = \max_i \sum_{j \in [n]} |\mathbf{M}_{ij}|$.*

Fact 4.3. *Let $\mathbf{M} \in \mathbb{R}^{n \times n}$ be a diagonally dominant matrix. Then,*

$$\prod_{i=1}^{n}(|\mathbf{M}_{ii}| - \Lambda_i(\mathbf{M})) \leq |\det \mathbf{M}| \leq \prod_{i=1}^{n}(|\mathbf{M}_{ii}| + \Lambda_i(\mathbf{M})).$$

To allow public re-randomization of encodings in the CLT scheme, CLT augments the public parameters $\mathsf{CLT.pp}$ with n level-1 encodings $\Pi_1, \ldots, \Pi_n$ of 0 where for all $i, j \in [n]$, $\Pi_j \equiv (\mathbf{\Pi}_{ij} \cdot g_i)/z \pmod{p_i}$. In addition, the public parameters contain a set of τ encodings $x_1, \ldots, x_\tau$ of zero, where for all $i \in [n], j \in [\tau]$, $x_j \equiv (r_{ij} \cdot g_i)/z \pmod{p_i}$, and each vector $\mathbf{r}_j = (r_{1j}, \ldots, r_{nj}) \leftarrow \mathcal{D}_{\mathbf{\Pi}}$.

We can now describe the operation of the $\mathsf{CLT.ReRand}(\mathsf{CLT.pp}, c)$ algorithm. To re-randomize an encoding c, CLT computes the following subset-sum of the x_j and linear combination of the Π_j:

$$c' = c + \sum_{j \in [\tau]} (b_j \cdot x_j) + \sum_{j \in [n]} (b'_j \cdot \Pi_j) \pmod{N_{\text{outer}}},$$

where $b_j \leftarrow \mathsf{Uniform}(\{0,1\})$, $b'_j \leftarrow \mathsf{Uniform}([0, 2^\mu - 1])$ for all i. Coron et al. show that this procedure induces the correct distribution on the re-randomized encodings by appealing to a variant of the leftover hash lemma over lattices [CLT13, §4].

4.2 Implementing the Basic Operations using Zero-Immunized CLT

We now show how the CLT multilinear map can be used to implement the basic abstract operations of Definition 2.3. We proceed in two steps.

- First, we show how to implement the abstract operations of Definition 2.3 via the CLT multilinear map. The adaptation is standard and is roughly the same as in [GLW14]. (We note that this step, by itself, does *not* produce a secure multilinear map MM, since the CLT scheme as-is permits the [CHL+14] attack on 0-encodings.)

  - $\underline{\mathsf{MM.Setup}(\mathcal{U} = Z^\kappa, \lambda, k)}$: Run the procedure $\mathsf{CLT.InstGen}(\lambda, \kappa, k)$ to obtain the public parameters $\mathsf{CLT.pp}$ and the secret parameters $\mathsf{CLT.sp}$. To implement the basic functionality of Definition 2.3, it is unnecessary to include the level-0 encodings, or the level-1 encodings of 0 and 1. Thus, let $\mathsf{CLT.pp'}$ be the version of $\mathsf{CLT.pp}$ where we have suppressed these additional encodings. Set $\mathsf{MM.pp} \leftarrow \mathsf{CLT.pp'}$ and $\mathsf{MM.sp} \leftarrow (\mathsf{CLT.pp}, \mathsf{CLT.sp})$. Note

that the secret parameters $\mathsf{MM.sp}$ includes the *non-suppressed* CLT parameters, which in particular, contain the encodings necessary for re-randomization. Finally, output the tuple $(\mathsf{MM.pp}, \mathsf{MM.sp}, N_1, \ldots, N_k)$ (note that the individual ring moduli $N_1, \ldots, N_k$ are included as part of the CLT secret parameters $\mathsf{CLT.sp}$).

- $\underline{\mathsf{MM.Encode}(\mathsf{MM.sp},\ x,\ \mathcal{S})}$: To encode an element $x \in \mathbb{Z}_N$ at an index set $\mathcal{S} = Z^k$ for some integer $k < \kappa$, we use the CLT secret parameters $\mathsf{CLT.sp}$ to compute an encoding $\hat{x}$ such that for all $i \in [n]$, $\hat{x} = (r_i g_i + (x \bmod g_i))/z^k \pmod{p_i}$, and $r_i \leftarrow \mathsf{Uniform}([-2^\rho + 1, 2^\rho - 1])$. Then, we output $\mathsf{CLT.ReRand}(\mathsf{CLT.pp}, \hat{x})$. Note that $\mathsf{MM.sp}$ contains a copy of $\mathsf{CLT.pp}$.
- $\underline{\mathsf{MM.Add}(\mathsf{MM.pp}, [x]_{\mathcal{S}}, [y]_{\mathcal{S}})}$: Output $\mathsf{CLT.Add}(\mathsf{MM.pp}, [x]_{\mathcal{S}}, [y]_{\mathcal{S}})$.
- $\underline{\mathsf{MM.Mult}(\mathsf{MM.pp}, [x]_{\mathcal{S}_1}, [y]_{\mathcal{S}_2})}$: Output $\mathsf{CLT.Mult}(\mathsf{MM.pp}, [x]_{\mathcal{S}_1}, [x]_{\mathcal{S}_2})$.
- $\underline{\mathsf{MM.ZeroTest}(\mathsf{MM.pp}, [x]_{\mathcal{S}})}$: Output $\mathsf{CLT.ZeroTest}(\mathsf{MM.pp}, [x]_{\mathcal{S}})$.

- Second, we apply our new zero-immunizing transformation (Construction 3.1) to the map MM we have just defined. This yields a modified multilinear map ($\mathsf{ZMM.Setup}, \mathsf{ZMM.Add}$, $\mathsf{ZMM.Mult}$, $\mathsf{ZMM.Encode}$, $\mathsf{ZMM.ZeroTest}$) based on CLT that is robust to zeroizing attacks and implements the basic abstract operations of Definition 2.3.

4.3 Extending Zero-Immunized CLT: Re-randomization and Public Encoding

While the basic abstract operations of Definition 2.3 suffice for "secret-key" multilinear maps, as in obfuscation constructions [GGH13a, BR14, BGK+14, GLSW14, Zim14], in general we would like to support the extended set of operations needed for public encoding via re-randomization (Remark 2.7).

More precisely, let $\mathsf{ZCLT}^{(\mathrm{sk})} = (\mathsf{ZMM.Setup}, \mathsf{ZMM.Add}, \mathsf{ZMM.Mult}, \mathsf{ZMM.Encode}, \mathsf{ZMM.ZeroTest})$ be the result of applying our zero-immunizing transformation to the abstract multilinear map based on CLT (Section 4.2). For clarity, we will define $\mathsf{ZCLT.PublicEncode}(\mathsf{ZCLT.pp}, \cdot, \cdot)$ to be the *public* analog of the $\mathsf{ZMM.Encode}$ function in $\mathsf{ZCLT}^{(\mathrm{sk})}$. In this section, we will describe the new candidate procedures $\mathsf{ZCLT.ReRand}$, $\mathsf{ZCLT.PublicEncode}$, and $\mathsf{ZCLT.Extract}$, along with the modifications to $\mathsf{ZCLT.Setup}$ required to support these extended operations.

4.3.1 ZCLT.Setup: extended setup operations.

In this section, we will show the modifications we must make to the $\mathsf{ZMM.Setup}$ function in $\mathsf{ZCLT}^{(\mathrm{sk})}$ in order to support $\mathsf{ZCLT.PublicEncode}$, $\mathsf{ZCLT.ReRand}$, and $\mathsf{ZCLT.Extract}$. To support these operations, we need to publish additional public parameters beyond what is included in the basic scheme $\mathsf{ZCLT}^{(\mathrm{sk})}$. As noted in Remark 2.8, it is not secure to include level-0 encodings in the public parameters. In lieu of these level-0 elements, we instead provide encodings of elements at a fresh singleton index set C. We can support sampling encodings of uniformly random values by including encodings of random values at index set C in the public parameters and taking subset-sums, exactly as CLT does with the level-0 encodings. Here, however, we will describe an even more general method that allows public encoding of arbitrary scalars in $\mathbb{Z}_N$; instead of providing encodings of random values at index set C, we include encodings of the powers of two: $2^0, 2^1, \ldots, 2^{\lfloor \log N \rfloor}$ at index set C. This way, by taking sums of the appropriate encodings, it is possible to construct encodings of arbitrary values in $\mathbb{Z}_N$.

To simplify the presentation in this section, we will describe the $\mathsf{ZCLT.Setup}$ procedure when applied to a symmetric CLT map with top-level set $\mathcal{U} = Z^\kappa$ and a k-dimensional message space.

We will specify the additional public parameters needed to publically encode elements at the index set Z. We note, however, that these results naturally generalize to the case of asymmetric maps where we have a separate index set C for the scalars (the case described above and in Remark 2.8).

We now describe the modified $\mathsf{ZCLT.Setup}$ procedure that includes the generation of the additional public parameters necessary for the extended functionality. First, $\mathsf{ZCLT.Setup}$ will instantiate a CLT map with a $(k+2)$-dimensional message space and new top-level index set $\mathcal{U}_Z = Z_{\mathrm{L}}^{\kappa} Z_{\mathrm{R}}^{\kappa} T$, for some fresh formal symbol T. Let $n_z = \Theta \cdot (k+2)$ be the number of primes that comprise the modulus for the message space in the underlying CLT map for the ZCLT scheme. We say that c is an encoding of a at the index set $Z_{\mathrm{L}}^{d_{\mathrm{L}}}$ for some $d_{\mathrm{L}} < \kappa$ if for all $i \in [n_z]$, we have that $c \equiv (r_i g_i + a)/(z_{\mathrm{L}}^{d_{\mathrm{L}}}) \pmod{p_i}$. Here, z_{L} is a (secret) parameter of the underlying CLT scheme. Respectively, c is an encoding of a at the index set $Z_{\mathrm{R}}^{d_{\mathrm{R}}}$ for $d_{\mathrm{R}} < \kappa$ if for all $i \in [n_z]$, we hat $c \equiv (r_i g_i + a)/(z_{\mathrm{R}}^{d_{\mathrm{R}}}) \pmod{p_i}$. Again, z_{R} is a secret parameter for the underlying CLT scheme.

We now describe the additional encodings that are included in $\mathsf{ZCLT.pp}$ to support the public encode operation $\mathsf{ZCLT.PublicEncode}(\mathsf{ZCLT.pp}, \cdot, Z)$. As noted above, we publish ZCLT encodings of $2^0, 2^1, \ldots, 2^{\lfloor \log N \rfloor}$, where $N = N_1 \cdots N_k$ is the modulus for the message space. More specifically, for each $j = 0, \ldots, \lfloor \log N \rfloor$, we apply the following procedure during $\mathsf{ZCLT.Setup}$ (we note that these operations are just the result of applying Construction 3.1 to CLT):

1. Choose $\zeta_j \leftarrow \mathsf{Uniform}(\mathbb{Z}^*_{N_{k+1}})$ and $\nu_{j,\mathrm{L}}, \nu_{j,\mathrm{R}} \leftarrow \mathsf{Uniform}(\mathbb{Z}^*_{N_{k+2}})$. Additionally, choose $\eta_j = [\eta_{j,1}, \ldots, \eta_{j,k}] \leftarrow \mathsf{Uniform}(\mathbb{Z}^*_{N_1 \cdots N_k})$ Then, define $y_{j,\mathrm{L}} = [(2^j \bmod N_1), \ldots, (2^j \bmod N_k), \zeta_j, \nu_{j,\mathrm{L}}]$ and $y_{j,\mathrm{R}} = [\eta_{j,1}, \ldots, \eta_{j,k}, \zeta_j, \nu_{j,\mathrm{R}}]$.

2. For $i \in [n_z]$, choose $r_{i,j,\mathrm{L}}, r_{i,j,\mathrm{R}} \leftarrow \mathsf{Uniform}([-2^\rho + 1, 2^\rho - 1])$. Then, construct $\hat{y}_{j,\mathrm{L}}$ and $y_{j,\mathrm{R}}$ where for all $i \in [n_z]$,

$$\hat{y}_{j,\mathrm{L}} \equiv \frac{(r_{i,j,\mathrm{L}} g_i + (y_{j,\mathrm{L}} \bmod g_i))}{z_{\mathrm{L}}} \pmod{p_i} \qquad \hat{y}_{j,\mathrm{R}} \equiv \frac{(r_{i,j,\mathrm{R}} g_i + (y_{j,\mathrm{R}} \bmod g_i))}{z_{\mathrm{R}}} \pmod{p_i}$$

3. Included the values $\hat{y}_j = (\hat{y}_{j,\mathrm{L}}, \hat{y}_{j,\mathrm{R}})$ with $\mathsf{ZCLT.pp}$.

Thus, to support public encoding of arbitrary ring elements, we augment the public parameters $\mathsf{ZCLT.pp}$ with $1 + \lfloor \log N \rfloor$ encodings of the powers of two. In addition, our candidate re-randomization procedure (described in Section 4.3.2) also requires additional encodings to be included in $\mathsf{ZCLT.pp}$. Here, we just present the additional steps in $\mathsf{ZCLT.Setup}$ that are needed to support $\mathsf{ZCLT.ReRand}$; we defer the full discussion of the candidate re-randomization procedure to Section 4.3.2.

1. Construct two matrices $\mathbf{\Pi}_{\mathrm{L}}, \mathbf{\Pi}_{\mathrm{R}} \in \mathbb{Z}^{n_z \times n_z}$, where the diagonal entries $(\mathbf{\Pi}_{\mathrm{L}})_{ii}$ and $(\mathbf{\Pi}_{\mathrm{R}})_{ii}$ are drawn independently from $\mathsf{Uniform}([n_z \cdot 2^\rho + 1, (n_z + 1) \cdot 2^\rho - 1)])$, and the off-diagonal components, $(\mathbf{\Pi}_{\mathrm{L}})_{ij}$ and $(\mathbf{\Pi}_{\mathrm{R}})_{ij}$ for $j \neq i$, are drawn independently from $\mathsf{Uniform}([-2^\rho + 1, 2^\rho - 1])$.

2. For each $j \in [n_z], \beta \in [\mu]$, choose $\zeta^{(\beta)}_{\pi_j} \leftarrow \mathsf{Uniform}(\mathbb{Z}^*_{N_{k+1}})$ and $\nu^{(\beta)}_{\pi_j,\mathrm{L}}, \nu^{(\beta)}_{\pi_j,\mathrm{R}} \leftarrow \mathsf{Uniform}(\mathbb{Z}^*_{N_{k+2}})$. Define the quantities

$$\pi^{(\beta)}_{j,\mathrm{L}} = \left[0, \ldots, 0, \zeta^{(\beta)}_{\pi_j}, \nu^{(\beta)}_{\pi_j,\mathrm{L}}\right]$$
$$\pi^{(\beta)}_{j,\mathrm{R}} = \left[0, \ldots, 0, \zeta^{(\beta)}_{\pi_j}, \nu^{(\beta)}_{\pi_j,\mathrm{R}}\right].$$

3. For each $i, j \in [n_z]$, and $\beta \in [\mu]$, choose $\chi_{i,j,\mathrm{L}}^{(\beta)}$ and $\chi_{i,j,\mathrm{R}}^{(\beta)}$ from $\mathsf{Uniform}([-2^\rho + 1, 2^\rho - 1])$. Then, for each $j \in [n_z], \beta \in [\mu]$, construct $\Pi_{j,\mathrm{L}}^{(\beta)}$ and $\Pi_{j,\mathrm{R}}^{(\beta)}$, such that for all $i \in [n_z]$,

$$\Pi_{j,\mathrm{L}}^{(\beta)} \equiv \frac{\left(2^\beta \cdot (\mathbf{\Pi}_{\mathrm{L}})_{ij} + \chi_{i,j,\mathrm{L}}^{(\beta)}\right) \cdot g_i + \left(\pi_{j,\mathrm{L}}^{(\beta)} \bmod g_i\right)}{z_{\mathrm{L}}} \pmod{p_i}$$
$$\Pi_{j,\mathrm{R}}^{(\beta)} \equiv \frac{\left(2^\beta \cdot (\mathbf{\Pi}_{\mathrm{R}})_{ij} + \chi_{i,j,\mathrm{R}}^{(\beta)}\right) \cdot g_i + \left(\pi_{j,\mathrm{R}}^{(\beta)} \bmod g_i\right)}{z_{\mathrm{R}}} \pmod{p_i}.$$

4. For $j \in [\tau]$, choose $\zeta_{\gamma_j} \leftarrow \mathsf{Uniform}(\mathbb{Z}^*_{N_{k+1}})$ and $\nu_{\gamma_j,\mathrm{L}}, \nu_{\gamma_j,\mathrm{R}} \leftarrow \mathsf{Uniform}(\mathbb{Z}^*_{N_{k+2}})$. Define the quantities

$$\gamma_{j,\mathrm{L}} = \left[0, \ldots, 0, \zeta_{\gamma_j}, \nu_{\gamma_j,\mathrm{L}}\right]$$
$$\gamma_{j,\mathrm{R}} = \left[0, \ldots, 0, \zeta_{\gamma_j}, \nu_{\gamma_j,\mathrm{R}}\right],$$

5. Let $\mathcal{D}_{\mathbf{\Pi}_{\mathrm{L}}}$ (resp., $\mathcal{D}_{\mathbf{\Pi}_{\mathrm{R}}}$) denote the uniform distribution over $\mathbb{Z}^{n_z}/\mathbf{\Pi}_{\mathrm{L}}$ (resp., $\mathbb{Z}^{n_z}/\mathbf{\Pi}_{\mathrm{R}}$), as described in Section 4.1. For $j \in [\tau]$, sample a vector $\mathbf{s}_{j,\mathrm{L}} = (s_{1,j,\mathrm{L}}, \ldots, s_{n_z,j,\mathrm{L}})$ from $\mathcal{D}_{\mathbf{\Pi}_{\mathrm{L}}}$ and a vector $\mathbf{s}_{j,\mathrm{R}} = (s_{1,j,\mathrm{R}}, \ldots, s_{n_z,j,\mathrm{R}})$ from $\mathcal{D}_{\mathbf{\Pi}_{\mathrm{R}}}$. Then, for each $j \in [\tau]$, construct the encodings $x_{j,\mathrm{L}}, x_{j,\mathrm{R}}$, such that for all $i \in [n_z]$,

$$x_{j,\mathrm{L}} \equiv \frac{s_{i,j,\mathrm{L}} \cdot g_i + (\gamma_{j,\mathrm{L}} \bmod g_i)}{z_{\mathrm{L}}} \pmod{p_i} \quad x_{j,\mathrm{R}} \equiv \frac{s_{i,j,\mathrm{R}} \cdot g_i + (\gamma_{j,\mathrm{R}} \bmod g_i)}{z_{\mathrm{R}}} \pmod{p_i},$$

6. Include the encodings $\Pi_{j,\mathrm{L}}^{(\beta)}$ and $\Pi_{j,\mathrm{R}}^{(\beta)}$ for all $j \in [n_z]$ and $\beta \in [\mu]$ as well as the encodings $x_{j,\mathrm{L}}, x_{j,\mathrm{R}}$ for all $j \in [\tau]$ in $\mathsf{ZCLT.pp}$.

Thus, to support public encoding of values at index set Z, we must publish $1 + \lfloor \log N \rfloor$ additional encodings to allow encoding of arbitrary scalars in $\mathbb{Z}_N$ and another $\mu \cdot n_z + \tau$ encodings for the candidate re-randomization procedure.

4.3.2 ZCLT.ReRand: candidate re-randomization procedure.

We now describe our candidate procedure for implementing the $\mathsf{ZCLT.ReRand}$ functionality. For simplicity, we again only describe our methods for the case of re-randomizing level-1 encodings, but the procedure naturally generalizes to arbitrary index sets.

We begin by giving some intuition for our candidate re-randomization procedure. First, we observe that it *almost* suffices to use the ordinary CLT re-randomization procedure, with ZCLT encodings of 0 in lieu of the ordinary CLT encodings of 0. The problem is that since our zero-immunized 0-encodings include nonzero components (the ζ, ν values), we cannot scale these encodings by the desired random factors in $[0, 2^\mu - 1]$ without blowing up the noise by the product of ζ or ν with 2^μ: this is much greater than the primes g_i, and would destroy the randomizer distribution r needed for the "leftover hash lemma over lattices" [CLT13].

To avoid this problem, we will instead give out a list of μ "pre-multiplied" re-randomization encodings, with randomizers roughly equal to the columns of $\mathbf{\Pi}$ scaled up by each of the powers of two, 2^β for each $\beta \in [\mu]$. To re-randomize, then, instead of multiplying these re-randomization encodings by the desired random factors in $[0, 2^\mu - 1]$, we add together the corresponding subset-sum of the "pre-multiplied" encodings, corresponding to the bit-decomposition of the random factor in $[0, 2^\mu - 1]$. This application of bit-decomposition is similar to its application in fully-homomorphic

encryption (e.g. [BV11]). We note that each randomizer in our procedure is only *roughly* equal to a scaled column of $\mathbf{\Pi}$ (deviating by additional noise terms $\chi^{(\beta)}_{i,j,\mathrm{L}}, \chi^{(\beta)}_{i,j,\mathrm{R}}$), so that the adversary cannot trivially cancel out g_i terms by forming linear combinations of our re-randomization encodings, thereby obtaining values $\equiv m_i/z \pmod{p_i}$ for $m_i \approx g_i$. Heuristically, these "noisy bit-decompositions" seem to improve the security of the candidate procedure, since now such linear combinations produce values that resemble ordinary encodings (including the noise induced by the g_i terms). We will also show that noisy bit-decomposition does not impact correctness, since the noise from the public re-randomization encodings simply adds with the noise from the encoding being re-randomized, which had noisy randomizers r to begin with.

We now give a concrete algorithm for $\mathsf{ZCLT.ReRand}(\mathsf{ZCLT.pp}, \hat{a}, Z)$. Then, in Theorem 4.4, we precisely characterize the noise distribution of the re-randomized encodings in the ZCLT scheme.

1. For $j \in [\tau]$, choose $b'_j \leftarrow \mathsf{Uniform}(\{0,1\})$. Next, for $\beta \in [\mu], j \in [n_z]$, choose $b^{(\beta)}_j \leftarrow \mathsf{Uniform}(\{0,1\})$.

2. Compute $\hat{a}'_{\mathrm{L}}$ and $\hat{a}'_{\mathrm{R}}$ as follows:

$$\hat{a}'_{\mathrm{L}} = \hat{a}_{\mathrm{L}} + \sum_{j\in[\tau]} (b'_j \cdot x_{j,\mathrm{L}}) + \sum_{j\in[n_z]} \sum_{\beta\in[\mu]} \left(b^{(\beta)}_j \cdot \Pi^{(\beta)}_{j,\mathrm{L}} \right) \pmod{N_{\mathrm{outer}}}$$
$$\hat{a}'_{\mathrm{R}} = \hat{a}_{\mathrm{R}} + \sum_{j\in[\tau]} (b'_j \cdot x_{j,\mathrm{R}}) + \sum_{j\in[n_z]} \sum_{\beta\in[\mu]} \left(b^{(\beta)}_j \cdot \Pi^{(\beta)}_{j,\mathrm{R}} \right) \pmod{N_{\mathrm{outer}}},$$

   Note that $x_{j,\mathrm{L}}$ and $x_{j,\mathrm{R}}$ for all $j \in [\tau]$ as well as $\Pi^{(\beta)}_{j,\mathrm{L}}$ and $\Pi^{(\beta)}_{j,\mathrm{R}}$ for all $j \in [n_z]$ and $\beta \in [\mu]$ are included as part of ZCLT.pp (Section 4.3.1).

3. Output the re-randomized encoding $\hat{a}' \leftarrow (\hat{a}'_{\mathrm{L}}, \hat{a}'_{\mathrm{R}})$.

We now show that this procedure induces the correct distribution on the re-randomized encodings.

Theorem 4.4 (ZCLT.ReRand Induces the Correct Distribution)**.** *Let $\lambda, \Theta, \rho, \alpha, \tau, \mu, n$, as well as, g_i, p_i for all $i \in [(k+2)\Theta]$, be parameters generated by the* ZCLT.Setup *procedure. Set $n_z = (k+2)\Theta$. Let $\hat{a} = (\hat{a}_{\mathrm{L}}, \hat{a}_{\mathrm{R}})$ be a* ZCLT *encoding with noise at most $2^{\rho+\log(n\alpha)+2}$, that is, for all $i \in [n_z]$, $\hat{a}_{\mathrm{L}} = (r_{i,\mathrm{L}} g_i + m_{i,\mathrm{L}})/z_{\mathrm{L}} \pmod{p_i}$ and $\hat{a}_{\mathrm{R}} = (r_{i,\mathrm{R}} g_i + m_{i,\mathrm{R}})/z_{\mathrm{R}} \pmod{p_i}$ for some $r_{i,\mathrm{L}}, r_{i,\mathrm{R}} \in (-2^{\alpha+\rho+1}, 2^{\alpha+\rho+1})$. Let $(\hat{a}'_{\mathrm{L}}, \hat{a}'_{\mathrm{R}}) \leftarrow \mathsf{ZCLT.ReRand}(\mathsf{ZCLT.pp}, Z, \hat{a})$. For all $i \in [n_z]$, write $\hat{a}'_{i,\mathrm{L}}$ as $(r'_{i,\mathrm{L}} g_i + m'_{i,\mathrm{L}})/z_{\mathrm{L}} \pmod{p_i}$ and $a'_{i,\mathrm{R}}$ as $(r'_{i,\mathrm{R}} g_i + m'_{i,\mathrm{R}})/z_{\mathrm{R}} \pmod{p_i}$. Suppose $\tau \geq \max\{\alpha, n_z \cdot (\rho + \log n_z + 1)\} + 2\lambda$ and $\mu \geq \lambda$. Then, the following holds:*

1. *Let $\mathbf{r}'_{\mathrm{L}} = (r'_{1,\mathrm{L}}, \ldots, r'_{n_z,\mathrm{L}})$ and $\mathbf{r}'_{\mathrm{R}} = (r'_{1,\mathrm{R}}, \ldots, r'_{n_z,\mathrm{R}})$. The distribution of $(\mathsf{ZCLT.pp}, \mathbf{r}'_{\mathrm{L}})$ is statistically close to that of $(\mathsf{ZCLT.pp}, \mathbf{r}''_{\mathrm{L}})$ where $\mathbf{r}''_{\mathrm{L}}$ is drawn from $\mathcal{D}_{2^\mu \mathbf{\Pi}_{\mathrm{L}}}$ (as defined in Section 4.1). Similarly, the distribution of $(\mathsf{ZCLT.pp}, \mathbf{r}'_{\mathrm{R}})$ is statistically close to that of $(\mathsf{ZCLT.pp}, \mathbf{r}''_{\mathrm{R}})$ where $\mathbf{r}'_{\mathrm{R}}$ is drawn from $\mathcal{D}_{2^\mu \mathbf{\Pi}_{\mathrm{R}}}$ (as defined in Section 4.1).*

2. *Let a_{L} be the message encoded by $\hat{a}_{\mathrm{L}}$ and let a'_{L} be the message encoded by $\hat{a}'_{\mathrm{L}}$. Define a_{R} and a'_{R} analogously. Then, a'_{L} and a'_{R} are each individually uniform, but not necessarily independently uniform, over $\mathbb{Z}_{N_{k+1}}$. Moreover, if $a_{\mathrm{L}} = a_{\mathrm{R}} \pmod{N_{k+1}}$, then $a'_{\mathrm{L}} = a'_{\mathrm{R}} \pmod{N_{k+1}}$.*

3. *Let a'_{L} be the message encoded by $\hat{a}'_{\mathrm{L}}$ and let a'_{R} be the message encoded by $\hat{a}'_{\mathrm{R}}$. Then, $(a'_{\mathrm{L}} \bmod N_{k+2})$ and $(a'_{\mathrm{R}} \bmod N_{k+2})$ are independently uniform over $\mathbb{Z}_{N_{k+2}}$.*

Proof. To prove claim 1, we follow the proof of [CLT13, Lemma 2]. We demonstrate the claim for $\mathbf{r}'_{\mathrm{L}}$; the claim for $\mathbf{r}'_{\mathrm{R}}$ follows by symmetry. First, define a vector $\tilde{\mathbf{r}}_{\mathrm{L}} \in \mathbb{Z}^{n_z}$, where the i^{th} component $\tilde{\mathbf{r}}_{i,\mathrm{L}}$ is given by

$$\tilde{\mathbf{r}}_{i,\mathrm{L}} = r_{i,\mathrm{L}} + \sum_{j\in[n_z]} \sum_{\beta\in[\mu]} \chi^{(\beta)}_{i,j,\mathrm{L}} + \left\lfloor \frac{\sum_{j\in[\tau]} b'_j \cdot (\gamma_{j,\mathrm{L}} \bmod g_i) + \sum_{j\in[n_z]} \sum_{\beta\in[\mu]} b^{(\beta)}_j \cdot \left(\pi^{(\beta)}_{j,\mathrm{L}} \bmod g_i\right)}{g_i} \right\rfloor$$

In addition, define the matrix $(\mathbf{X}_{\mathrm{L}})_{ij} = (s_{i,j,\mathrm{L}}) \in \mathbb{Z}^{n_z \times \tau}$. Then, the re-randomization relation can be stated as

$$\begin{aligned}\mathbf{r}'_{\mathrm{L}} &= \tilde{\mathbf{r}}_{\mathrm{L}} + \mathbf{X}_{\mathrm{L}} \cdot \mathbf{b}' + \sum_{\beta\in[\mu]} (2^\beta \cdot \mathbf{\Pi}_{\mathrm{L}}) \cdot \mathbf{b}^{(\beta)} \\ &= \tilde{\mathbf{r}}_{\mathrm{L}} + \mathbf{X}_{\mathrm{L}} \cdot \mathbf{b}' + \mathbf{\Pi}_{\mathrm{L}} \cdot \sum_{\beta\in[\mu]} 2^\beta \mathbf{b}^{(\beta)}\end{aligned}$$

where $\mathbf{b}' \leftarrow \mathsf{Uniform}(\{0,1\}^\tau)$ and $\mathbf{b}^{(\beta)} \leftarrow \mathsf{Uniform}(\{0,1\}^{n_z})$. Since $\mathbf{b}^{(\beta)}$ is uniform over $\{0,1\}^{n_z}$, it follows that

$$\sum_{\beta\in[\mu]} 2^\beta \mathbf{b}^{(\beta)} \sim \mathsf{Uniform}\left([0, 2^\mu - 1]^{n_z}\right)$$

Thus, we can invoke the leftover hash lemma for lattices [CLT13, Lemma 6] to conclude that the distribution of $(\mathsf{ZCLT.pp}, \mathbf{r}'_{\mathrm{L}})$ is ε_1-close to the distribution $(\mathsf{ZCLT.pp}, \tilde{\mathbf{r}}_{\mathrm{L}} + \mathcal{D}_{2^\mu \mathbf{\Pi}_{\mathrm{L}}})$, where

$$\varepsilon_1 = \tau \cdot n_z \cdot 2^{-\mu} + \frac{1}{2}\sqrt{\left|\det \mathbf{\Pi}_{\mathrm{L}}\right| / 2^\tau}.$$

Since $\mathbf{\Pi}_{\mathrm{L}}$ is diagonally dominant, we have from Fact 4.3 that

$$\left|\det \mathbf{\Pi}_{\mathrm{L}}\right| \le \prod_{i=1}^{n_z} \left(\left|(\mathbf{\Pi}_{\mathrm{L}})_{ii}\right| + \Lambda_i(\mathbf{\Pi}_{\mathrm{L}})\right) \le (2 \cdot n_z \cdot 2^\rho)^{n_z} \le 2^{n_z(\rho + \log(n_z)+1)}.$$

Thus, $\varepsilon_1 \le \tau \cdot n_z \cdot 2^{-\mu} + (1/2) \cdot 2^{(n_z(\rho+\log n_z+1)-\tau)/2}$. Since $\tau \ge n_z \cdot (\rho + \log n_z + 1) + 2\lambda$ and $\mu \ge \lambda$, we have that $\varepsilon_1 = \mathrm{negl}(\lambda)$.

To complete the proof, we show that the distribution of $(\mathsf{ZCLT.pp}, \mathbf{r}'_{\mathrm{L}})$ is $(\varepsilon_1 + \varepsilon_2)$-close to $(\mathsf{ZCLT.pp}, \mathcal{D}_{2^\mu \mathbf{\Pi}_{\mathrm{L}}})$. From [CLT13, Lemma 7], we have that this claim holds for

$$\varepsilon_2 = 2^{-\mu}\left(\left\|\tilde{\mathbf{r}}_{\mathrm{L}}\right\|_\infty \cdot \left\|\mathbf{\Pi}_{\mathrm{L}}^{-1}\right\|_\infty + 1\right).$$

Thus, we bound $\left\|\tilde{\mathbf{r}}_{\mathrm{L}}\right\|_\infty$. We do so by bounding each component $|\tilde{\mathbf{r}}_{i,\mathrm{L}}|$ for all $i \in [n_z]$:

- Suppose $i \le n\Theta$. We use the bound that $|r_{i,\mathrm{L}}| \le 2^{\rho+\log(n\alpha)+2}$, and $\gamma_{i,\mathrm{L}} = 0 = \pi^{(\beta)}_{i,\mathrm{L}}$ for all $\beta \in [\mu]$ to conclude that

$$|\tilde{\mathbf{r}}_{i,\mathrm{L}}| \le |r_{i,\mathrm{L}}| + \sum_{j\in[n_z]} \sum_{\beta\in[\mu]} \left|\chi^{(\beta)}_{i,j,\mathrm{L}}\right| \le 2^{\rho+\log(n\alpha)+2} + n_z\mu \cdot 2^\rho.$$

- Suppose $n\Theta < i \le (n+1)\Theta$. These indices correspond to the message in the $(k+1)^{\text{st}}$ component (the ζ component). Then, we have

$$\begin{aligned}|\tilde{\mathbf{r}}_{i,\mathrm{L}}| &\le |r_{i,\mathrm{L}}| + \sum_{j\in[n_z]} \sum_{\beta\in[\mu]} \left|\chi^{(\beta)}_{i,j,\mathrm{L}}\right| + \left\lfloor \frac{\sum_{j\in[\tau]} b'_j(\zeta_{\gamma_j} \bmod g_i) + \sum_{j\in[n_z],\beta\in[\mu]} b^{(\beta)}_j \left(\zeta^{(\beta)}_{\pi_j} \bmod g_i\right)}{g_i} \right\rfloor \\ &\le 2^{\rho+\log(n\alpha)+2} + n_z\mu \cdot 2^\rho + \tau + n_z\mu\end{aligned}$$

- Suppose $(n+1)\Theta < i \leq (n+2)\Theta$. An analogous calculation as the one for the previous case shows that

$$|\tilde{\mathbf{r}}_{i,\mathrm{L}}| \leq 2^{\rho+\log(n\alpha)+2} + n_z\mu \cdot 2^{\rho} + \tau + n_z\mu.$$

Thus, we conclude that

$$\|\tilde{\mathbf{r}}_{\mathrm{L}}\|_{\infty} \leq 2^{\rho+\log(n\alpha)+2} + n_z\mu \cdot 2^{\rho} + \tau + n_z\mu.$$

Next, using Fact 4.2, we have that

$$\left\|\mathbf{\Pi}_{\mathrm{L}}^{-1}\right\|_{\infty} \leq \frac{1}{\min_{i\in[n_z]}((\mathbf{\Pi}_{\mathrm{L}})_{ii} - \Lambda_i(\mathbf{\Pi}_{\mathrm{L}}))} \leq \frac{1}{(n_z)2^{\rho} - (n_z-1)2^{\rho}} \leq 2^{-\rho}.$$

This gives

$$\begin{aligned}\varepsilon_2 &\leq 2^{-\mu} + (2^{\rho+\log(n\alpha)+2} + n_z\mu \cdot 2^{\rho} + \tau + n_z\mu) \cdot 2^{-\rho} \cdot 2^{-\mu} \\ &\leq 2^{-\mu} + (4n\alpha + n_z\mu + (\tau + n_z\mu) \cdot 2^{-\rho}) \cdot 2^{-\mu}.\end{aligned}$$

Since $\mu \geq \lambda$, and $\tau, \mu, n, n_z, \alpha$ are $\mathrm{poly}(\lambda)$, we have $\varepsilon_2 = \mathrm{negl}(\lambda)$, the claim follows.

It remains to show that claims 2 and 3 hold regarding the re-randomization. For claim 2, we use the fact that $N_{k+1} = g_{n\Theta+1} \cdots g_{(n+1)\Theta}$, and demonstrate that for each $i = n\Theta+1, \ldots, (n+1)\Theta$, the claim holds with g_i in place of N_{k+1}; the overall claim then follows by CRT. First, we have

$$a'_{\mathrm{L}} = a_{\mathrm{L}} + \sum_{j\in[\tau]} b'_j(\zeta_{\gamma_j} \bmod g_i) + \sum_{j\in[n_z]} \sum_{\beta\in[\mu]} b_j^{(\beta)} \left(\zeta_{\pi_j}^{(\beta)} \bmod g_i\right) \pmod{g_i}.$$

To show that a'_{L} is uniform random over $\mathbb{Z}_{g_i}$, we appeal to the leftover hash lemma applied to subset sums of abelian group elements [CLT13, Lemma 5]. First, we have that for all $j \in [\tau]$, ζ_{γ_j} is chosen uniformly and independently from $\mathbb{Z}_{N_{k+1}}$, so each $(\zeta_{\gamma_j} \bmod g_i)$ is independently uniform in $\mathbb{Z}_{g_i}$. In addition, the b'_j are chosen uniformly and independently from $\{0,1\}$. By [CLT13, Lemma 5], we have that the statistical distance ε between $\left((\zeta_{\gamma_1} \bmod g_i), \ldots, (\zeta_{\gamma_\tau} \bmod g_i), \sum_{j\in[\tau]} b'_j(\zeta_{\gamma_j} \bmod g_j)\right)$ and $\mathsf{Uniform}(\mathbb{Z}_{g_i}^{\tau+1})$ is bounded as follows:

$$\varepsilon \leq \frac{1}{2}\sqrt{|\mathbb{Z}_{g_i}| / 2^{\tau}} \leq 2^{(\alpha-\tau)/2+1}.$$

Since $\tau \geq \alpha + 2\lambda$, $\varepsilon = \mathrm{negl}(\lambda)$. Because $\sum_{j\in[\tau]} b'_j(\zeta_{\gamma_j} \bmod g_i)$ is statistically close to uniform over $\mathbb{Z}_{g_i}$, and independent of the values $\zeta_{\pi_j}^{(\beta)}$, we conclude that $(a'_{\mathrm{L}} \bmod g_i)$ is statistically close to uniform in $\mathbb{Z}_{g_i}$. The same argument shows that the analogous expression holds for a'_{R}, so we conclude that a'_{R} is statistically close to uniform over $\mathbb{Z}_{g_i}$, and furthermore, when $a_{\mathrm{L}} = a_{\mathrm{R}} \pmod{g_i}$, then we have $a'_{\mathrm{L}} = a'_{\mathrm{R}} \pmod{g_i}$, as required.

To prove claim 3, we use the same argument as in claim 2, except we substitute $\nu_{\gamma_j,\mathrm{L}}$ (resp., $\nu_{\gamma_j,\mathrm{R}}$) and $\nu_{\pi_j,\mathrm{L}}^{(\beta)}$ (resp., $\nu_{\pi_j,\mathrm{R}}^{(\beta)}$) for the ζ_{γ_j} and $\zeta_{\pi_j}^{(\beta)}$. Since $\nu_{\gamma_j,\mathrm{L}}$ and $\nu_{\gamma_j,\mathrm{R}}$ are independently uniform, we additionally conclude that $(a'_{\mathrm{L}} \bmod N_{k+2})$ and $(a'_{\mathrm{R}} \bmod N_{k+2})$, are independently uniform in $\mathbb{Z}_{N_{k+2}}$, as required. □

4.3.3 ZCLT.PublicEncode: public encoding via re-randomization.

We now describe how to implement the public variant of the encode operation (ZCLT.PublicEncode) in our zero-immunized CLT scheme. As before, we will only describe the procedure for encoding values at level 1 (i.e., at some singleton index set Z). Our method naturally generalizes to higher-level encodings.

For an integer $a < \lfloor \lg N \rfloor$, the public encoding procedure $\mathsf{ZCLT.PublicEncode}(\mathsf{ZCLT.pp}, a, Z)$ operates as follows:

1. Let $a_{\lfloor \log N \rfloor} \cdots a_1 a_0$ be the binary representation of a over the integers. Then, compute the following:
$$\hat{a}_{\mathrm{L}} = \sum_{j=0}^{\lfloor \log N \rfloor} a_j \hat{y}_{j,\mathrm{L}} \qquad\qquad \hat{a}_{\mathrm{R}} = \sum_{j=0}^{\lfloor \log N \rfloor} a_j \hat{y}_{j,\mathrm{R}}$$
where $\hat{y}_{j,\mathrm{L}}$ and $\hat{y}_{j,\mathrm{R}}$ are the ZCLT encodings of 2^j included in ZCLT.pp. Set $\hat{a} = (\hat{a}_{\mathrm{L}}, \hat{a}_{\mathrm{R}})$.
2. Output $\mathsf{ZCLT.ReRand}(\mathsf{ZCLT.pp}, \hat{a})$

We now bound the noise in the encodings $\hat{a}$ formed in step 1 of the above procedure. This is necessary to satisfy the hypothesis in Theorem 4.4, and show that the re-randomization procedure induces the correct distribution on the noise in the re-randomized encodings.

Lemma 4.5 (Noise in Level-1 Encodings before Re-randomization)**.** *Let $k, \Theta, n, N, \rho, \alpha$, as well as g_i, p_i for all $i \in [(k+2)\Theta]$ be parameters output by* ZCLT.Setup*. Set $n_z = (k+2)\Theta$. Fix a value $a \in \mathbb{Z}_N$, and let $a_{\lfloor \log N \rfloor} \cdots a_1 a_0$ be its binary representation over the integers. Let $\hat{y}_0, \ldots, \hat{y}_{\lfloor \log N \rfloor}$ be the* ZCLT *encodings of the powers of two in* ZCLT.pp*. Let*
$$\hat{a}_{\mathrm{L}} = \sum_{j=0}^{\lfloor \log N \rfloor} a_j \hat{y}_{j,\mathrm{L}} \qquad\qquad \hat{a}_{\mathrm{R}} = \sum_{j=0}^{\lfloor \log N \rfloor} a_j \hat{y}_{j,\mathrm{R}}$$
and write $\hat{a}_{\mathrm{L}} \equiv (r_{i,\mathrm{L}} g_i + m_{i,\mathrm{L}})/z \pmod{p_i}$ and $\hat{a}_{\mathrm{R}} \equiv (r_{i,\mathrm{R}} g_i + m_{i,\mathrm{R}})/z \pmod{p_i}$ for all $i \in [n_z]$. Then, for all $i \in [n_z]$, the noise $|r_{i,\mathrm{L}}|, |r_{i,\mathrm{R}}| \leq 2^{\rho + \log(n\alpha) + 2}$.

Proof. For each $i \in [n_z]$, we can write $\hat{y}_{j,\mathrm{L}}$ as $\left(r_{i,\mathrm{L}}^{(y_j)} g_i + m_{i,\mathrm{L}}^{(y_j)}\right)/z \pmod{p_i}$, where $r_{i,\mathrm{L}}^{(y_j)} \in (-2^\rho, 2^\rho)$ and $m_{i,\mathrm{L}}^{(y_j)} \in \mathbb{Z}_{g_i}$. Then, we can write $r_{i,\mathrm{L}}$ as
$$r_{i,\mathrm{L}} = \sum_{j=0}^{\lfloor \log N \rfloor} a_j \hat{y}_{j,\mathrm{L}} = \left(\sum_{j=0}^{\lfloor \log N \rfloor} a_j r_{i,\mathrm{L}}^{(y_j)} \right) + \left\lfloor \frac{\sum_{j=0}^{\lfloor \log N \rfloor} a_j m_{i,\mathrm{L}}^{(y_j)}}{g_i} \right\rfloor$$
Since $a_j \in \{0,1\}$, we can bound $|r_{i,\mathrm{L}}|$ as follows:
$$|r_{i,\mathrm{L}}| \leq (\lfloor \log N \rfloor + 1) \cdot \left| r_{i,\mathrm{L}}^{(y_j)} \right| + \left\lfloor \frac{(\lfloor \log N \rfloor + 1) \cdot m_{i,\mathrm{L}}^{(y_j)}}{g_i} \right\rfloor \leq (\lfloor \log N \rfloor + 1) \cdot (2^\rho + 1).$$
Finally, $N = N_1 \cdots N_k$ and each N_i is a product of Θ α-bit primes. Thus, $\log N \leq k\Theta\alpha = n\alpha$. We conclude that
$$|r_{i,\mathrm{L}}| \leq (n\alpha + 1) \cdot (2^\rho + 1) \leq 2^{\rho + \log(n\alpha) + 2}.$$
An analogous argument shows the claim for $|r_{i,\mathrm{R}}|$. □

Remark 4.6 (Updating the Secret-Encoding Operation)**.** Technically, the public-encoding extension we have just described also requires a minor change to the original secret-encoding procedure, for the following reason. The new re-randomization procedure invoked by ZCLT.PublicEncode induces a slightly different distribution from the procedure ZCLT.Encode specified in Section 4.2, just because in the new setup procedure for public-encoding (Section 4.3.1), we choose new randomizer lattices $\mathbf{\Pi}$. Thus, to match the original intended functionality of CLT precisely, in our extended version with public-encoding we would also update the secret-encoding operation to use our new re-randomization procedure ZCLT.ReRand, instead of the original CLT.ReRand. Of course, this modification is not required if we are only implementing the basic multilinear map operations (Definition 2.3, Section 4.2).

4.3.4 ZCLT.Extract: extraction for top-level encodings.

The zero-immunized extraction operation ZCLT.Extract is an immediate generalization of the corresponding operation CLT.Extract in the CLT scheme. Specifically, given an encoding $\hat{a} = (\hat{a}_{\mathrm{L}}, \hat{a}_{\mathrm{R}})$ at the top level set ($\hat{a}_{\mathrm{L}}$ is encoded at Z_{L}^{κ} and $\hat{a}_{\mathrm{R}}$ is encoded at Z_{R}^{κ}), the ZCLT.Extract(ZCLT.pp, $\hat{a}$) function works as follows:

1. First, form the encoding $\hat{z} = \hat{t}_{\mathrm{L}} \cdot \hat{a}_{\mathrm{L}} - \hat{t}_{\mathrm{R}} \cdot a_{\mathrm{R}}$. Note that $\hat{t}_{\mathrm{L}}$ and $\hat{t}_{\mathrm{R}}$ are both included in the public parameters ZCLT.pp.
2. Output the result of CLT.Extract(CLT.pp, $\hat{z}$). Note that the necessary parameters in CLT.pp needed for zero-testing are included in ZCLT.pp.

The correctness of the zero-immunized extraction operation is immediate by the same arguments as in [CLT13].

5 Attacks

In this section we strengthen the attack of [CHL+14], so that the attack does not require low-level 0-encodings, but rather applies whenever the adversary can produce many encodings that multiply to 0 at the zero-testing index set $\mathcal{U}$. We call such encodings *orthogonal encodings*. Using the techniques of [CHL+14] we show that orthogonal encodings lead to a total break of the CLT multilinear map.

Definition 5.1 (Orthogonal Encodings)**.** Consider a composite-order multilinear map whose order is a product of n primes $p_1, \ldots, p_n$. For sets $I_x, I_{x'}, I_c \subseteq [n]$ we say that the encodings $(x_1, \ldots, x_n,\ x'_1, \ldots, x'_n,\ c, c')$ form a tuple of *orthogonal encodings* if:

- All the encodings $x_1, \ldots, x_n$ are at some index set $S_x \subset \mathcal{U}$. Their values $m_1^{(x)}, \ldots, m_n^{(x)}$ satisfy $m_r^{(x)} = 0 \pmod{p_r}$ for all $r \in I_x$.
- All the encodings $x'_1, \ldots, x'_n$ are at some index set $S_{x'} \subset \mathcal{U}$. Their values $m_1^{(x')}, \ldots, m_n^{(x')}$ satisfy $m_r^{(x')} = 0 \pmod{p_r}$ for all $r \in I_{x'}$.
- The encodings c, c' are at the index set $S_c = \mathcal{U}/(S_x S_{x'})$ where $S_c \neq 1$. Their values $m^{(c)}, m^{(c')}$ are congruent to 0 modulo p_r for all $r \in I_c$.

Furthermore, we require that $I_x \cup I_{x'} \cup I_c = [n]$ (i.e., every orthogonal component is zeroed out by some encoding).

5.1 Description of the Attack

We next describe the attack on CLT given a set of orthogonal encodings. The attack follows the "zeroizing" attack of [CHL+14]. We form the $n \times n$ matrix $\mathbf{W}_c$:

$$\begin{aligned}(\mathbf{W}_c)_{jk} &= c \cdot x'_j \cdot x_k \cdot (\mathbf{p}_{\text{zt}})_1 \pmod{N_{\text{outer}}} \\ &= \sum_{i \in [n]} c_i x'_{ji} x_{ki} \cdot (g_i^{-1} \bmod p_i) \cdot h_i \cdot \frac{N_{\text{outer}}}{p_i} \pmod{N_{\text{outer}}} \\ &= \sum_{i \in I_c} r_i^{(c)} x'_{ji} x_{ki} h_i \frac{N_{\text{outer}}}{p_i} + \sum_{i \in I_{x'}} c_i r_{ji}^{(x')} x_{ki} h_i \frac{N_{\text{outer}}}{p_i} + \sum_{i \in I_x} c_i x'_{ji} r_{ki}^{(x)} h_i \frac{N_{\text{outer}}}{p_i} \pmod{N_{\text{outer}}}\end{aligned}$$

Now, just as in [CHL+14], we note that since each $c \cdot x'_j \cdot x_k$ is an encoding of zero, all of the terms in the sum for each $(\mathbf{W}_c)_{jk}$ must be so small that no reduction modulo N_{outer} takes place. Thus the equation above also holds over $\mathbb{Z}$. Re-expressing the equation as a matrix product, we have:

$$\mathbf{W}_c = \mathbf{X}' \cdot \text{diag}(\tilde{\mathbf{c}}) \cdot \text{diag}(h'_1, \ldots, h'_n) \cdot \mathbf{X}$$

where we define $h'_i = h_i N_{\text{outer}}/p_i$, and:

$$\mathbf{X}'_{ji} = \begin{cases} r_{ji}^{(x')} & \text{if } i \in I_{x'} \\ x'_{ji} & \text{otherwise} \end{cases} \qquad \tilde{\mathbf{c}}_i = \begin{cases} r_i^{(c)} & \text{if } i \in I_c \\ c_i & \text{if } i \notin I_c \end{cases} \qquad \mathbf{X}_{ik} = \begin{cases} r_{ki}^{(x)} & \text{if } i \in I_x \\ x_{ki} & \text{otherwise} \end{cases}$$

We also compute the analogous relation, with c' in place of c:

$$\mathbf{W}_{c'} = \mathbf{X}' \cdot \text{diag}(\tilde{\mathbf{c}}') \cdot \text{diag}(h'_1, \ldots, h'_n) \cdot \mathbf{X}$$

(where $\tilde{\mathbf{c}}'$ is defined exactly as $\tilde{\mathbf{c}}$, with c' in place of c), and we note that the other matrix factors $\mathbf{X}', \text{diag}(h'_1, \ldots, h'_n), \mathbf{X}$ do not change between the cases of c and c'.

Suppose that $\mathbf{X}, \mathbf{X}'$ are nonsingular, and hence so is $\mathbf{W}_{c'}$ (we will show below that this holds with overwhelming probability). Then we compute the following matrix over $\mathbb{Q}$:

$$\mathbf{W}_c \cdot \mathbf{W}_{c'}^{-1} = \mathbf{X}' \cdot \text{diag}\left(\frac{\tilde{\mathbf{c}}_1}{\tilde{\mathbf{c}}'_1}, \ldots, \frac{\tilde{\mathbf{c}}_n}{\tilde{\mathbf{c}}'_n}\right) \cdot (\mathbf{X}')^{-1}$$

Finally, again following [CHL+14], we compute the eigenvalues of this matrix (e.g. by factoring the characteristic polynomial); these are precisely the values $\tilde{\mathbf{c}}_1/\tilde{\mathbf{c}}'_1, \ldots, \tilde{\mathbf{c}}_n/\tilde{\mathbf{c}}'_n \in \mathbb{Q}$. We reduce these fractions to lowest terms, $s_1/s'_1, \ldots, s_n/s'_n$, and compute the following quantity for each $i \in [n]$:

$$\gcd(s_i \cdot c' - s'_i \cdot c, N_{\text{outer}})$$

We will show below that with overwhelming probability this common factor is p_i, which yields a total break of CLT as described in [CHL+14].

5.2 Correctness of the Attack

Procedurally, the attack described in Section 5.1 is very similar to that of [CHL+14]. However, there are two crucial differences. First, we do not need 0-encoding, but only orthogonal encodings. Second, we do not need the encodings to be fresh (i.e., the output of MM.Encode). Instead, the encodings can be the result of a series of arithmetic operations satisfying some independence

conditions. Showing that the attack works in these more general settings requires some work. The difficulty is in showing that the matrices $\mathbf{X}$ and $\mathbf{X}'$ are nonsingular with high probability.

To ensure that the matrices $\mathbf{X}, \mathbf{X}'$ are nonsingular in the attack, we will require the encodings x'_j, x_k to be linearly independent polynomials over fresh encodings. We require a similar condition over the encodings c, c'. These conditions are much weaker than requiring all encodings to be fresh and this makes the attack work with far fewer encodings than needed in [CHL+14].

Correctness of the attack algorithm is captured in the following theorem.

Theorem 5.2 (Correctness of the Attack)**.** *Suppose that the attack of Section 5.1 is carried out on orthogonal encodings* $(x_1, \ldots, x_n,\ x'_1, \ldots, x'_n,\ c, c')$ *where:*

- *The encodings* $x_1, \ldots, x_n$ *are computed by linearly independent polynomials evaluated on fresh encodings (i.e., are each output by* MM.Encode*).*
- *The encodings* $x'_1, \ldots, x'_n$ *are computed by linearly independent polynomials evaluated on fresh encodings (i.e., are each output by* MM.Encode*).*
- *The encodings* c, c' *are computed by linearly independent polynomials evaluated on fresh encodings (i.e., are each output by* MM.Encode*).*

Then the attack recovers the secret primes p_i *with overwhelming probability.*

Before we can prove Theorem 5.2, we will need to introduce a number of definitions and lemmas. To begin with, we will describe the matrices $\mathbf{X}, \mathbf{X}'$ in terms of the corresponding formal polynomials over encodings.

Definition 5.3 (Matrix Expansion)**.** Fix integers m, n, formal variables $\hat{t}_1, \ldots, \hat{t}_m$, and a tuple of n formal polynomials $f_1, \ldots, f_n \in \mathbb{Z}[\hat{t}_1, \ldots, \hat{t}_m]$. We define the *matrix expansion* of $(f_1, \ldots, f_n)$ to be the following matrix in $\mathbb{Z}[t_{1,1}, \ldots, t_{1,n}, \ldots, t_{m,1}, \ldots, t_{m,n}]^{n \times n}$:

$$\begin{pmatrix} f_1(t_{1,1}, \ldots, t_{m,1}) & \cdots & f_1(t_{1,n}, \ldots, t_{m,n}) \\ \vdots & \ddots & \vdots \\ f_n(t_{1,1}, \ldots, t_{m,1}) & \cdots & f_n(t_{1,n}, \ldots, t_{m,n}) \end{pmatrix}$$

Lemma 5.4 (Expansions of Independent Polynomials are Nonsingular)**.** *Fix integers* m, n *and formal variables* $\hat{t}_1, \ldots, \hat{t}_m$*. Let* $f_1, \ldots, f_n \in \mathbb{Z}[\hat{t}_1, \ldots, \hat{t}_m]$ *be linearly independent polynomials (regarding* $\mathbb{Z}[\hat{t}_1, \ldots, \hat{t}_m]$ *as a module over the base ring* $\mathbb{Z}$*). Then the matrix expansion* $\mathbf{M}$ *of* $(f_1, \ldots, f_n)$ *is nonsingular.*

Proof. By induction on n. The claim holds by definition for $n = 1$, so suppose $n > 1$, and consider the determinant $\det \mathbf{M} \in \mathbb{Z}[t_{1,1}, \ldots, t_{1,n}, \ldots, t_{m,1}, \ldots, t_{m,n}]$. We write:

$$\det \mathbf{M} = \sum_{i \in [n]} (-1)^{i+1} f_i(t_{1,1}, \ldots, t_{m,1}) \cdot \det \overline{\mathbf{M}}_{i,1}$$

(recall that $\overline{\mathbf{M}}_{i,1}$ denotes the $(i,1)$ minor of $\mathbf{M}$, i.e., the matrix $\mathbf{M}$ with the first column and the i^{th} row removed). We now claim that $\det \mathbf{M}$ is not identically zero. Suppose otherwise: then for all integers $(v_{i,j})_{i \in [n], j \in [2,n]}$, when we substitute every $v_{i,j}$ for the corresponding variable $t_{i,j}$ in $\det \mathbf{M}$, the result must be identically zero in $\mathbb{Z}[t_{1,1}, \ldots, t_{m,1}]$. On the other hand, by the inductive hypothesis, for each $i \in [n]$ we have $\det \overline{\mathbf{M}}_{i,1} \not\equiv 0 \in \mathbb{Z}[t_{1,1}, \ldots, t_{1,n}, \ldots, t_{m,1}, \ldots, t_{m,n}]$, and thus there exist integers $(v_{i,j})_{i \in [n], j \in [2,n]}$ which, when substituted for the corresponding $t_{i,j}$ in $\det \overline{\mathbf{M}}_{1,1}$,

produce a nonzero value in $\mathbb{Z}$. For each $i \in [n]$, let a_i be the result of substituting these $(v_{i,j})$ for the corresponding variables $t_{i,j}$ in $\det \overline{\mathbf{M}}_{i,1}$, so that in particular, we have $a_1 \neq 0 \in \mathbb{Z}$. Then we have:

$$0 = \sum_{i \in [n]} (-1)^{i+1} f_i(t_{1,1}, \ldots, t_{m,1}) \cdot a_i \ \in \mathbb{Z}[t_{1,1}, \ldots, t_{m,1}]$$

and, renaming variables, we find

$$0 = \sum_{i \in [n]} (-1)^{i+1} f_i(\hat{t}_1, \ldots, \hat{t}_m) \cdot a_i \ \in \mathbb{Z}[\hat{t}_1, \ldots, \hat{t}_m]$$

which is a nontrivial linear combination of $f_1, \ldots, f_n$, contradicting our assumption. □

We also require the following technical lemmas. The first lemma, due to Ostrowski [Ost52], states that for diagonally dominant matrices $\mathbf{\Pi}$ (such as the randomization matrices in CLT), removing a dimension to form a minor $\overline{\mathbf{\Pi}}_{i,i}$ decreases the determinant by an appropriate factor.

Lemma 5.5 (Determinants of Diagonally Dominant Matrices [Ost52])**.** *Let $\mathbf{M} \in \mathbb{Z}^{n \times n}$ be diagonally dominant (Definition 4.1), and define:*

$$\Lambda_i(\mathbf{M}) = \sum_{j \neq i} |\mathbf{M}_{ij}| \ , \qquad \sigma_i(\mathbf{M}) = \frac{\Lambda_i(\mathbf{M})}{|\mathbf{M}_{ii}|} \ , \qquad t_i(\mathbf{M}) = \max_{j \neq i}(\sigma_j(\mathbf{M})) \ .$$

Suppose that for all $i \in [n]$, we have $t_i(\mathbf{M}) \leq 1$ and $t_i(\mathbf{M})\sigma_i(\mathbf{M}) < 1$. Then for all $i \in [n]$, we have $|\det \mathbf{M}| \geq (|\mathbf{M}_{i,i}| - t_i(\mathbf{M})\Lambda_i(\mathbf{M})) \cdot |\det \overline{\mathbf{M}}_{i,i}|$.

Corollary 5.6 (Determinants of Randomization Matrices)**.** *Let n, q, ρ, r be integers such that $1 < n \leq r$ and $r > 2$. Let $\mathbf{\Pi} \in \mathbb{Z}^{n \times n}$ be a matrix such that for $i \in [n]$ we have $r \cdot 2^\rho \leq \mathbf{\Pi}_{i,i} \leq (r+1) \cdot 2^\rho$ and for $i \neq j \in [n]$ we have $|\mathbf{\Pi}_{i,j}| \leq 2^\rho$. Then for all $i \in [n]$, we have $|\det \mathbf{\Pi}| \geq 2^\rho \cdot |\det \overline{\mathbf{\Pi}}_{i,i}|$.*

Proof. Since $\mathbf{\Pi}$ is diagonally dominant, we can invoke Lemma 5.5. By definition, we have $\Lambda_i(\mathbf{\Pi}) \leq 2^\rho(n-1)$, and hence $\sigma_i(\mathbf{\Pi}) \leq (n-1)/r < 1$ and $t_i(\mathbf{\Pi}) \leq (n-1)/r < 1$. Lemma 5.5 now implies

$$|\det \mathbf{\Pi}| \ \geq \ \left(2^\rho r - 2^\rho(n-1) \cdot \frac{n-1}{r}\right) \cdot |\det \overline{\mathbf{\Pi}}_{i,i}| \ \geq \ 2^\rho \cdot |\det \overline{\mathbf{\Pi}}_{i,i}|$$

as desired. □

Our next technical lemma concerns the distribution $\mathcal{D}$ of CLT randomizers $r_{j,i}$. Intuitively, the lemma shows that since $\mathcal{D}$ is uniform over $\mathbb{Z}^n/L$, where L is the lattice spanned by the columns of the (well-conditioned) CLT re-randomizing matrix $\mathbf{\Pi}$, we can conclude that $\mathcal{D}$ is nearly uniform when regarded over $\mathbb{Z}_q^n$ for a somewhat smaller prime q.

Lemma 5.7. *Let n, q, ρ, r be integers such that $1 < n \leq r$, $r > 2$, and $16qr < 2^\rho$, and let $\mathbf{\Pi} \in \mathbb{Z}^{n \times n}$ be a matrix such that for $i \in [n]$ we have $r \cdot 2^\rho \leq \mathbf{\Pi}_{i,i} \leq (r+1) \cdot 2^\rho$ and for $i \neq j \in [n]$ we have $|\mathbf{\Pi}_{i,j}| \leq 2^\rho$. Let $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{Z}^n$ be the columns of $\mathbf{\Pi}$, and let L be the integer lattice spanned by $\mathbf{v}_1, \ldots, \mathbf{v}_n$. Let $\mathcal{D} = \mathsf{Uniform}(\mathbb{Z}^n/L)$, where each element of $\mathbb{Z}^n/L$ is canonically identified with its representative in the parallelepiped spanned by $\mathbf{v}_1, \ldots, \mathbf{v}_n$. Let $\mathcal{D}_q$ be the distribution over $\mathbb{Z}_q^n$ defined by reducing $\mathcal{D}$ modulo q. Then the statistical distance between $\mathcal{D}_q$ and $\mathsf{Uniform}(\mathbb{Z}_q^n)$ is at most $16qr/2^\rho$.*

Proof. The volume of the parallelepiped $\mathbb{R}^n/L$ is bounded as follows:

$$2^{\rho n} = (r \cdot 2^\rho - (n-1) \cdot 2^\rho)^n \leq |\det \mathbf{\Pi}| \leq ((r+1) \cdot 2^\rho + (n-1) \cdot 2^\rho)^n \leq (2r)^n \cdot 2^{\rho n}$$

since $\mathbf{\Pi}$ is diagonally dominant (Fact 4.3). We consider the tiling of $\mathbb{R}^n$ by n-dimensional (axis-aligned) boxes of side length q, and we say that a box $[x_1 q, (x_1+1)q] \times \ldots \times [x_n, (x_n+1)q]$ is a *boundary box* if it intersects, but is not contained in, $\mathbb{R}^n/L$. Denoting by $\mathrm{bd}(n)$ the number of boundary boxes, we will first show the following bound:

$$\mathrm{bd}(n) \leq \frac{8qr}{2^\rho} \cdot \frac{|\det \mathbf{\Pi}|}{q^n} \tag{1}$$

Fix an axis $i \in [n]$ (without loss of generality $i = n$), and consider the plane $\mathbb{R}^{n-1} \times \{0\}$. Every boundary box, when projected to this plane, must intersect the projection of $\mathbb{R}^n/L$ to the plane, and thus the number of boxes in $\mathbb{R}^{n-1}$ that can be the projection of a boundary box is bounded above as follows. For $n = 2$, it is at most $((r+1) \cdot 2^\rho + (n-1) \cdot 2^\rho)/q$, which satisfies the bound of equation (1). For $n > 2$, we reason as follows. For $n > 2$, we note that the projection of $\mathbb{R}^n/L$ is itself an $(n-1)$-dimensional parallelepiped, defined by the (n,n) minor $\overline{\mathbf{\Pi}}_{n,n}$. Thus the number of possible projected boundary boxes is at most $|\det \overline{\mathbf{\Pi}}_{n,n}|/q^{n-1} + \mathrm{bd}(n-1)$ (accounting for both the projected boundary and the volume of the projected interior).

Now, every boundary box must intersect some face of $\mathbb{R}^n/L$. The two faces of $\mathbb{R}^n/L$ not parallel to $\mathbf{v}_n$ are $F_{n,0} = \mathrm{span}(\mathbf{v}_1, \ldots, \mathbf{v}_{n-1})$ and $F_{n,1} = \mathbf{v}_n + \mathrm{span}(\mathbf{v}_1, \ldots, \mathbf{v}_{n-1})$. Without loss of generality, we consider the face $F_{n,0}$, and fix integers $x_1, \ldots, x_{n-1}$ that define the projection of a possible boundary box, $B_{n-1} = [x_1 q, (x_1+1)q] \times \ldots \times [x_{n-1}, (x_{n-1}+1)q]$. Fix two points in the face $F_{n,0}$, given by $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n) = \mathbf{a} + \delta_1 \mathbf{v}_1 + \ldots + \delta_{n-1}\mathbf{v}_{n-1}$, such that $(a_1, \ldots, a_{n-1}), (b_1, \ldots, b_{n-1}) \in B_{n-1}$. Then for each $i \in [n-1]$ we have:

$$q \;\geq\; |b_i - a_i| \;\geq\; \delta_i \cdot 2^\rho r - \sum_{i' \neq i \in [n]} (\delta_i \cdot 2^\rho) \;\geq\; \delta_i \cdot 2^\rho (r - (n-1)) \;\geq\; \delta_i \cdot 2^\rho$$

and thus:

$$|b_n - a_n| \;\leq\; \sum_{i \in [n-1]} |\delta_i \mathbf{\Pi}_{i,n}| \;\leq\; 2^{-\rho}(n-1)q$$

Since $2^\rho > 16qr > (n-1)$, we conclude that $|b_n - a_n| < q$, and hence at most two integer values of x_n can make $B_{n-1} \times [x_n, (x_n+1)q]$ a boundary box. Taking the union over all $2n$ faces, we conclude that the number of boundary boxes is bounded as follows:

$$\mathrm{bd}(n) \;\leq\; 4n \left(\frac{|\det \overline{\mathbf{\Pi}}_{n,n}|}{q^{n-1}} + \mathrm{bd}(n-1) \right)$$

Now we note that the minor $\overline{\mathbf{\Pi}}_{n,n}$ also satisfies the hypothesis, and thus inductively we conclude:

$$\begin{aligned}
\mathrm{bd}(n) \;&\leq\; 4n \left(\frac{|\det \overline{\mathbf{\Pi}}_{n,n}|}{q^{n-1}} + \frac{8qr}{2^\rho} \cdot \frac{|\det \overline{\mathbf{\Pi}}_{n,n}|}{q^{n-1}} \right) \\
&=\; 4n \cdot \frac{|\det \overline{\mathbf{\Pi}}_{n,n}|}{q^{n-1}} \left(1 + \frac{8qr}{2^\rho} \right) \\
&\leq\; 8n \cdot \frac{|\det \overline{\mathbf{\Pi}}_{n,n}|}{q^{n-1}} \\
&\leq\; \frac{8qr}{2^\rho} \cdot \frac{|\det \mathbf{\Pi}|}{q^n} \cdot \frac{2^\rho \cdot |\det \overline{\mathbf{\Pi}}_{n,n}|}{\det \mathbf{\Pi}}
\end{aligned}$$

at which point the bound of equation (1) follows from Corollary 5.6.

Finally, since the number of interior (non-boundary) boxes is at least $|\det \mathbf{\Pi}|/q^n - \mathrm{bd}(n)$, the probability that a random sample from $\mathbb{Z}^n/L$ falls in a boundary box is at most:

$$\begin{aligned}\frac{\mathrm{bd}(n)}{|\det \mathbf{\Pi}|/q^n - \mathrm{bd}(n)} &= \left(\frac{|\det \mathbf{\Pi}|}{\mathrm{bd}(n) \cdot q^n} - 1\right)^{-1} \\ &\leq \left(\frac{2^\rho}{8qr} - 1\right)^{-1} \\ &\leq \left(\frac{2^\rho}{8qr} - \frac{2^\rho}{16qr}\right)^{-1} \\ &= 16qr/2^\rho\end{aligned}$$

as desired. □

We are now equipped to prove the main correctness properties of the attack of Section 5.1: specifically, given appropriate conditions on the input encodings, the matrices $\mathbf{X}, \mathbf{X}'$ are nonsingular, and the final gcd operations succeed in factoring the modulus.

Lemma 5.8 (Nonsingular Matrices in the Attack)**.** *Suppose that the attack of Section 5.1 is carried out on orthogonal encodings $(x_1, \ldots, x_n,\ x'_1, \ldots, x'_n,\ c, c')$ such that for some fresh encodings $\hat{t}_1, \ldots, \hat{t}_m$ (i.e., each output by* MM.Encode*), the following conditions are satisfied:*

- *The encodings $x_1, \ldots, x_n$ are computed by linearly independent polynomials, resp., $f_1, \ldots, f_n \in \mathbb{Z}[\hat{t}_1, \ldots, \hat{t}_m]$, evaluated on the encodings $\hat{t}_1, \ldots, \hat{t}_m$.*
- *The encodings $x'_1, \ldots, x'_n$ are computed by linearly independent polynomials, resp., $f'_1, \ldots, f'_n \in \mathbb{Z}[\hat{t}_1, \ldots, \hat{t}_m]$, evaluated on the encodings $\hat{t}_1, \ldots, \hat{t}_m$.*

Then the matrices $\mathbf{X}, \mathbf{X}'$ defined in the attack are nonsingular with overwhelming probability.

Proof. Without loss of generality it suffices to establish the claim for the matrix $\mathbf{X}'$. We will show that $\det \mathbf{X}'$ is nonzero in $\mathbb{Q}$. For each $j \in [m], i \in [n]$, define $\hat{t}_j \equiv t_{j,i} \pmod{p_i}$. Now by definition, we have:

$$\mathbf{X}'_{ji} = \begin{cases} r_{ji}^{(x')} & \text{if } i \in I_{x'} \\ x'_{ji} & \text{otherwise} \end{cases}$$

Defining the modified matrix $\mathbf{X}''_{ji} = x'_{ji}$, we note that $\mathbf{X}''$ differs from $\mathbf{X}'$ only by a factor of g_i for columns $i \in I_{x'}$. Since g_i is nonzero in $\mathbb{Q}$ with overwhelming probability, $\mathbf{X}'$ will be nonsingular if $\mathbf{X}''$ is, and so it suffices to show that $\det \mathbf{X}'' \neq 0 \in \mathbb{Q}$.

We now observe that for each $j \in [n]$ we have $x'_{ji} = f_j(t_{1,i}, \ldots, t_{m,i})$, and by Lemma 5.4 we conclude that $\det \mathbf{X}'' \not\equiv 0 \in \mathbb{Z}[t_{1,1}, \ldots, t_{1,n}, \ldots, t_{m,1}, \ldots, t_{m,n}]$. Each value $t_{j,i}$ in the CLT scheme satisfies $t_{j,i} = r_{j,i} g_i + m_{j,i}$, where the values $(r_{j,1}, \ldots, r_{j,n})$ are statistically close to uniform over the parallelepiped spanned by the columns of $\mathbf{\Pi}$, where $\mathbf{\Pi}$ is some re-randomizing matrix corresponding to the index set of the fresh encoding $\hat{t}_j$. Letting q be a $(\rho/2)$-bit prime, we regard the equation $t_{j,i} = r_{j,i} g_i + m_{j,i}$ over $\mathbb{Z}_q$. By Lemma 5.7, the distribution of each $t_{j,i}$ is statistically close to independently uniform over $\mathbb{Z}_q$. Further, each polynomial f_j has degree at most $\kappa = \mathrm{poly}(\lambda)$ in the CLT scheme, and thus the determinant $\det \mathbf{X}''$ has degree at most $n \cdot \kappa = \mathrm{poly}(\lambda)$. The Schwartz-Zippel lemma now implies that $\det \mathbf{X}'' \neq 0 \in \mathbb{Z}_q$, and hence also over $\mathbb{Q}$, with overwhelming probability. □

Lemma 5.9 (Nontrivial Divisors in the Attack)**.** *Suppose that the attack of Section 5.1 is carried out on orthogonal encodings* $(x_1, \ldots, x_n,\ x'_1, \ldots, x'_n,\ c, c')$*, where the encodings* c, c' *are computed by linearly independent polynomials, resp.,* $f, f' \in \mathbb{Z}[\hat{t}_1, \ldots, \hat{t}_m]$*, evaluated on encodings* $\hat{t}_1, \ldots, \hat{t}_m$ *that are fresh (i.e., are each output by* MM.Encode*). Then in the final step, letting* $d_i = s_i \cdot c' - s'_i \cdot c$*, with overwhelming probability we have* $\gcd(d_i, N_{\text{outer}}) = p_i$ *for all* $i \in [n]$*.*

Proof. First we note that $s_i/s_{i'} = \tilde{\mathbf{c}}_i/\tilde{\mathbf{c}}'_i = c_i/c'_i$ over $\mathbb{Q}$, and hence also over $\mathbb{Z}_{p_i}$, so we have

$$d_i = s_i \cdot c' - s'_i \cdot c = s_i \cdot c'_i - s'_i \cdot c_i = 0 \pmod{p_i}$$

Thus it suffices to show that for all $i_1 \neq i_2 \in [n]$, with overwhelming probability the prime p_{i_2} does not divide d_{i_1}. In this case, we have:

$$s_{i_1} \cdot c' - s'_{i_1} \cdot c = s_{i_1} \cdot c'_{i_2} - s'_{i_1} \cdot c_{i_2} \pmod{p_{i_2}}$$

Now if this quantity is zero modulo p_{i_2}, then p_{i_2} also divides the following quantity:

$$\Delta \stackrel{\text{def}}{=} \gcd(\tilde{c}_{i_1}, \tilde{c}'_{i_1}) \cdot (s_{i_1} \cdot c'_{i_2} - s'_{i_1} \cdot c_{i_2}) = c_{i_1} \cdot c'_{i_2} - c'_{i_1} \cdot c_{i_2} = \det \begin{pmatrix} c_{i_1} & c_{i_2} \\ c'_{i_1} & c'_{i_2} \end{pmatrix}$$

It now follows that $\Delta \neq 0 \in \mathbb{Z}$ with overwhelming probability, by the same argument as in the proof of Lemma 5.8. Finally, since the matrix entries $c_{i_1}, c_{i_2}, c'_{i_1}, c'_{i_2}$ are the numerators of encodings that multiply to reach the top level, they are each so small that $\Delta < p_{i_2}$, by correctness of the CLT scheme. Thus with overwhelming probability, p_{i_2} does not divide $s_{i_1} \cdot c'_{i_2} - s'_{i_1} \cdot c_{i_2}$, as desired. □

We are now equipped to prove the main theorem stated above, which shows that our strengthened version of the [CHL+14] attack succeeds with overwhelming probability (given appropriate conditions).

Theorem 5.2 (Correctness of the Attack)**.** *Suppose that the attack of Section 5.1 is carried out on orthogonal encodings* $(x_1, \ldots, x_n,\ x'_1, \ldots, x'_n,\ c, c')$ *where:*

- *The encodings* $x_1, \ldots, x_n$ *are computed by linearly independent polynomials evaluated on fresh encodings (i.e., are each output by* MM.Encode*).*
- *The encodings* $x'_1, \ldots, x'_n$ *are computed by linearly independent polynomials evaluated on fresh encodings (i.e., are each output by* MM.Encode*).*
- *The encodings* c, c' *are computed by linearly independent polynomials evaluated on fresh encodings (i.e., are each output by* MM.Encode*).*

Then the attack recovers the secret primes p_i *with overwhelming probability.*

Proof. Immediate from Lemmas 5.8 and 5.9. □

5.3 Attacking Subgroup Elimination

We now present an explicit attack on the subgroup elimination assumption [GLW14, GLSW14], based on our strengthened version of the [CHL+14] attack (Section 5.1). Because our strengthened version does not require low-level encodings of zero, but rather only requires orthogonal encodings, we are able to apply it even in "secret-key" settings such as in [GLSW14] in which 0-encodings for re-randomization are not made public.

We emphasize that the attack here applies only to the original version of the CLT multilinear map [CLT13, GLW14], *without* our new zero-immunizing transformation. (With our new transformation, we do not know of any attacks on subgroup elimination or other related problems.)

Assumption 5.10 (Multilinear Subgroup Elimination Assumption ([GLW14, GLSW14], adapted))**.** Let MM be a composite-order multilinear map. For a security parameter λ, integer parameters $\kappa, \tau = \mathrm{poly}(\lambda)$, an adversary $\mathcal{A}$, and a bit $b \in \{0,1\}$, the security game proceeds as follows.

1. The top-level index set $\mathcal{U}$ is defined as Z^κ for a fresh formal symbol Z.
2. The multilinear map instance is initialized by running
$$(\mathsf{pp},\ \mathsf{sp},\ N_1, \ldots, N_{\kappa+\tau+1}) \leftarrow \mathsf{MM.Setup}(\mathcal{U}, 1^\lambda, \kappa+\tau+1)\ .$$
3. Using the operation $\mathsf{MM.Encode}(\mathsf{sp}, \cdot, \cdot)$, encodings of the following form are generated:
$$\begin{aligned}
e_1 &= [\$, 0, \ldots, 0,\ \ 0, \ldots, 0,\ \ 0]_Z \\
e_2 &= [0, \$, \ldots, 0,\ \ 0, \ldots, 0,\ \ 0]_Z \\
&\vdots \\
e_\kappa &= [0, 0, \ldots, \$,\ \ 0, \ldots, 0,\ \ 0]_Z \\
e_{\kappa+1} &= [0, 0, \ldots, 0,\ \ \$, \ldots, 0,\ \ 0]_Z \\
&\vdots \\
e_{\kappa+\tau} &= [0, 0, \ldots, 0,\ \ 0, \ldots, \$,\ \ 0]_Z
\end{aligned}
\qquad
\begin{aligned}
f_1 &= [0, \$, \$, \ldots, \$,\ \ 0, \ldots, 0,\ \ \$]_Z \\
f_2 &= [\$, 0, \$, \ldots, \$,\ \ 0, \ldots, 0,\ \ \$]_Z \\
&\vdots \\
f_\kappa &= [\$, \$, \$, \ldots, 0,\ \ 0, \ldots, 0,\ \ \$]_Z
\end{aligned}$$
Here we use the notation "\$" to denote a unit selected uniformly at random from the corresponding subring.[6]
4. Using the operation $\mathsf{MM.Encode}(\mathsf{sp}, \cdot, \cdot)$, the following challenge encodings are generated:
$$\begin{aligned}
y_0 &= [\$, \$, \$, \ldots, \$,\ \ 0, \ldots, 0,\ \ 0]_Z \\
y_1 &= [\$, \$, \$, \ldots, \$,\ \ 0, \ldots, 0,\ \ \$]_Z
\end{aligned}$$
5. The adversary is given the encodings $(e_1, e_2, \ldots, e_{\kappa+\tau}, f_1, f_2, \ldots, f_\kappa, y_b)$ along with the public parameters pp.
6. The adversary outputs a bit $b' \in \{0,1\}$.

For an adversary $\mathcal{A}$, let $W_0(\mathcal{A}, \lambda)$ be the probability that $\mathcal{A}$ outputs 1 when $b = 0$ (with security parameter $\lambda \in \mathbb{N}$), and $W_1(\mathcal{A}, \lambda)$ be the probability that $\mathcal{A}$ outputs 1 when $b = 1$. We say the *multilinear subgroup elimination assumption* holds for MM if for all efficient adversaries $\mathcal{A}$, we have $|W_0(\mathcal{A}, \lambda) - W_1(\mathcal{A}, \lambda)| < \mathrm{negl}(\lambda)$.

Theorem 5.11. *For some constant $s > 0$, the multilinear subgroup elimination assumption is false for the CLT multilinear map with $\kappa > s \log \lambda$ and any integer τ.*

Proof. We first recall that in the CLT construction (as modified for composite order in [GLW14]), the number of primes n is given by $n = k\Theta = (\kappa+\tau+1)\Theta$, where $\Theta > (\rho \cdot \eta)^{1+\varepsilon}$ determines the number of primes in $g_1, \ldots, g_n$ that compose each of the inner moduli $N_1, \ldots, N_k$.

[6] In [GLW14, GLSW14], it is not specified whether the generators selected for the encodings $e_1, e_2, \ldots, e_{\kappa+\tau}$ are encodings of 1 or are encodings of random units. We assume what seems to be the harder case to attack; our attack also applies for arbitrary choices of the generators.

Now, to attack the assumption, the adversary will first form n linearly independent encodings as follows. Assume without loss of generality that κ is even, and let $\omega = (\kappa - 2)/2$. Starting with the encodings $e_1, \ldots, e_\omega$, for every tuple $(a_1, \ldots, a_{\omega-1}) \in \{0,1\}^{\omega-1}$, we consider the product encoding $e_1^{a_1} \cdots e_{\omega-1}^{a_{\omega-1}} e_\omega^{\omega - \sum_i a_i}$. Since $n = \text{poly}(\lambda)$, we know that for large enough constant s, there are at least n such tuples; let the adversary compute the n first lexicographically, and call the corresponding product encodings $h_1, \ldots, h_n$. The adversary now executes the attack described in Section 5.1, with the following parameter settings:

$$\begin{aligned}
(x_1, \ldots, x_n) &= (h_1, \ldots, h_n) \\
(x'_1, \ldots, x'_n) &= (h_1, \ldots, h_n) \\
c &= e_\kappa e_{\kappa-1} \\
c' &= e_\kappa e_{\kappa-2} \\
I_c &= [\kappa + \tau + 1] \\
I_x &= \emptyset \\
I_{x'} &= \emptyset
\end{aligned}$$

By Theorem 5.2, the attack succeeds with overwhelming probability, recovering the secret primes $p_1, \ldots, p_{k\Theta}$ (and hence all secret quantities, including z and the primes $g_1, \ldots, g_{k\Theta}$, as described in [CHL+14]). The adversary can then compute the term $y_b \cdot z$ modulo $p_{(k-1)\Theta+1}, \cdots, p_{k\Theta}$, and test whether it is a multiple of $N_k = g_{(k-1)\Theta+1} \cdots g_{k\Theta}$, thereby achieving advantage $1 - \text{negl}(\lambda)$. □

6 Acknowledgments

This work was supported by an NSF Graduate Research Fellowship and the DARPA PROCEED program. Opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of DARPA.

References

[AHKM14] Daniel Apon, Yan Huang, Jonathan Katz, and Alex J. Malozemoff. Implementing cryptographic program obfuscation. Cryptology ePrint Archive, Report 2014/779, 2014. `http://eprint.iacr.org/`.

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *EUROCRYPT*, 2014.

[BL97] D. Boneh and R. J. Lipton. Effect of operators on straight line complexity. In *ISTCS*, 1997.

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *TCC*, 2014.

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

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, 2011.

[CHL+14] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. Cryptology ePrint Archive, Report 2014/906, 2014. http://eprint.iacr.org/.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *CRYPTO*, 2013.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, 2013.

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

[GGH14] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graded multilinear maps from lattices. Cryptology ePrint Archive, Report 2014/645, 2014. http://eprint.iacr.org/.

[GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure attribute based encryption from multilinear maps. Cryptology ePrint Archive, Report 2014/622, 2014. http://eprint.iacr.org/.

[GLSW14] Craig Gentry, Allison B. Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. Cryptology ePrint Archive, Report 2014/309, 2014. http://eprint.iacr.org/.

[GLW14] Craig Gentry, Allison B. Lewko, and Brent Waters. Witness encryption from instance independent assumptions. In *CRYPTO*, 2014.

[Mil04] Victor S Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4), 2004.

[MOV93] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5), 1993.

[Ost52] A. M. Ostrowski. Note on bounds for determinants with dominant principal diagonal. *Proceedings of the American Mathematical Society*, 3(1):pp. 26–30, 1952.

[Pri51] G. Baley Price. Bounds for determinants with dominant principal diagonal. *Proceedings of the American Mathematical Society*, 2(3):pp. 497–502, 1951.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, 1997.

[Var75] J.M. Varah. A lower bound for the smallest singular value of a matrix. *Linear Algebra and its Applications*, 11(1):3 – 5, 1975.

[Zim14] Joe Zimmerman. How to obfuscate programs directly. Cryptology ePrint Archive, Report 2014/776, 2014. http://eprint.iacr.org/.