Candidate Multilinear Maps from RLWE-variants

Andrew H. Russell* Nigel Smart†

August 6, 2015

Abstract

Multilinear maps are widely used cryptographic primitives in key exchange, functional encryption, obfuscation, and more. However, the only constructions we have are based on non-standard assumptions and have been successfully attacked in various ways. We propose a new multilinear map based on a ring variant of the GSW homomorphic encryption scheme that has its security based on variants of the standard ring learning with errors (RLWE) problem. Our construction exploits the properties that ciphertexts are small and decrypted elements are large if and only if the plaintext is zero. These features allow us to hide the decryption key and publish it as a zero-tester. Though the security of our construction does not fully reduce to the RLWE assumption, it avoids the attacks made on other constructions, so we present our candidate as a step forward in basing multilinear maps on more reasonable assumptions.

1 Introduction

Multilinear maps. Since the use of bilinear pairings to construct identity-based encryption by Boneh and Franklin [BF01] there has been interest in the natural extension to cryptographic maps with multilinearity $\kappa \geq 2$. Boneh and Silverberg [BS03] outlined numerous applications for multilinear maps, but were pessimistic about finding constructions in the field of algebraic geometry. Since then, there has been some progress in the field. Garg, Gentry, and Halevi [GGH13a] first proposed the idea of *graded encoding schemes*, which approximate true multilinear maps, and gave a candidate based on ideal lattices (referred to as the GGH construction). Subsequently, Coron, Lepoint, and Tibouchi gave a similar construction but over the integers (the CLT construction [CLT13, CLT15]), and Gentry, Gorbunov, and Halevi proposed a construction where the structure of the map is dependent on an underlying acyclic graph [GGH14a].

Multilinear maps have a range of applications, such as an N-way Diffie-Hellman key exchange, broadcast encryption with short keys [BS03] or functional encryption (without obfuscation) [BLR$^+$15]. Arguably the most important and prominent application that motivates the use of multilinear maps is general program obfuscation, originally outlined in [GGH$^+$13b] using the GGH scheme, due to its prolific use as a cryptographic primitive.

Cryptanalysis of current constructions. Unfortunately, the security of these graded encoding schemes is not a well-understood problem, and there has yet to be a candidate that is based on

*Pomona College, ahr12012@mymail.pomona.edu. Work performed while at the University of Bristol.
†nigel@cs.bris.ac.uk

more standard assumptions. The original CLT and GGH candidates have been successfully attacked [CHL+14, CGH+15, HJ15]. The original CLT scheme was broken completely [CHL+14, CGH+15]. The GGH construction has a weak-discrete log attack, and a total break in the particular use-case of key exchange [CGH+15, HJ15]. There have been multiple attempts to fix the CLT scheme [BWZ14, CLT14, CLT15]. However, only the updated CLT construction [CLT15] seems to avoid these attacks at the cost of some efficiency. The graph-induced construction in [GGH14a] is known to be insecure in some situations through the "approximate trapdoor" attacks outlined in the original paper, which makes it harder to reason about security in various applications, as you must prove that those situations do not arise in each specific use of the map.

Our contributions. We propose a new multilinear map from a variant of the GSW homomorphic encryption scheme [GSW13]. The primary advantage of our scheme is that sensitive parameters—the zero-tester, zero-tested encodings, encodings of 0—are either the products of (unknown) uniform matrices or instances of a RLWE-esque problem. In particular, our candidate completely avoids the zeroizing attacks used to attack the CLT and GGH schemes [CHL+14, CGH+15, HJ15]. While a full reduction to standard assumptions does not seem possible at this point, we present our construction both as a step forward in securing multilinear maps on standard problems and as an alternative post-zeroizing map to the new CLT construction.

Future work. It remains an open problem to construct multilinear maps based on standard hard problems, or to construct "true" multilinear maps rather than the graded encoding schemes presented here and in prior works. Further cryptanalysis of this new scheme could also bring about either greater confidence in its security or additional insight into the general strategy of basing graded encoding schemes on homomorphic encryption schemes as is done in all current constructions; essentially, what security problems arise when zero-testing functionality is added to a HE scheme. An additional issue with current multilinear map constructions is efficiency. None of the current candidates, including this one, have reasonable parameters for use in the real world. Any improvements would have a large impact on the practical viability of flagship applications like obfuscation that rely heavily on multilinear maps.

Overview of the GSW HE scheme. The encoding of elements and homomorphic operations are exactly as in the GSW HE scheme, except we work over a polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ instead of $\mathbb{Z}_q$. To be more specific, ciphertexts in GSW look like the following, where $\mu \in \mathbb{Z}_q$ is the plaintext message and A is the public key (a "uniform" matrix) hiding the plaintext with some noise R (a binary matrix):

$$C = \mu I_N + R \cdot A$$

In GSW the decryption key v (a vector) acts as an "approximate inverse" to the public key A, so that $A \cdot v = e$ where e is small. Thus when we multiply C by v we get the following:

$$C \cdot v = \mu v + R \cdot e \tag{1}$$

The plaintext μ can then be recovered because $R \cdot e$ is small.

Getting a zero-testing element. The main thing to note in (1) is that the GSW decryption key is itself "large," so that when we apply it to a ciphertext the result will be large if and only if μ is non-zero. Thus, if we can somehow securely publish v we can then zero-test encodings, which gives us a graded encoding scheme. A starting point for hiding the key is that ciphertexts are themselves small (binary matrices), as they are a result of the $\mathsf{Flatten}$ function, a helpful operation that allows noise growth to be bounded in the original GSW scheme. This way, we can add a "small" value ϵ to the decryption key and still get the desired property:

$$\begin{aligned} C \cdot (v + \epsilon) &= \mu v + R \cdot e + C \cdot \epsilon \\ &= \mu v + small \end{aligned}$$

Of course, this does not fully hide v nor does it make decryption any harder. So, we multiply v by a "small" value α, which brings us closer:

$$\begin{aligned} C \cdot (\alpha v + \epsilon) &= \mu(\alpha v) + \alpha R \cdot e + C \cdot \epsilon \\ &= \mu(\alpha v) + small \end{aligned}$$

This breaks decryption in the case that α is unknown, but still maintains the property that the result will be small if and only if μ is small, so zero-testing will still work.

2 Preliminaries

First, let $\mathcal{R} = \mathbb{Z}[x]/\langle x^n + 1\rangle$ where n is a power of 2, and let $\mathcal{R}_q = \mathcal{R}/q\mathcal{R} \cong \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ be the ring that contains $\ell = \lceil \log q \rceil$ bit coefficients. For an element $f \in \mathcal{R}$, let $\|f\|_k$ denote the ℓ_k-norm of the vector representation of f in $\mathbb{Z}^n$ for all $k \in \mathbb{N}$, where $\|f\|_\infty$ is the ℓ_∞-norm (the largest coefficient of f). For a vector $\vec{v} \in \mathcal{R}^N$, let $\|\vec{v}\|_k$ denote the largest element of $\vec{v}$ under the ℓ_k-norm. For any $m \in \mathbb{N}$ we denote the set $\{1, 2, \ldots, m\}$ by $[m]$.

2.1 Bit operations

The GSW paper outlines a few operations on vectors that we extend to the ring setting, as in [KGV14]. For some k, let $\ell = \lceil \log q \rceil$ and $N = k\ell$, and let $\vec{a}$ be a k-dimensional vector of ring elements in $\mathcal{R}_q$. Then, the bit-decomposition function $\mathsf{BitDecomp} : \mathcal{R}_q^k \to \mathcal{R}_q^N$ is given by

$$\mathsf{BitDecomp}(\vec{a}) = (a_{1,0}, \ldots, a_{1,\ell-1}, \ldots, a_{k,0}, \ldots, a_{k,\ell-1})$$

where $a_{i,j}$ is the polynomial whose entries are the j-th bit each coefficient in the i-th entry of $\vec{a}$. That is, we decompose a vector $\vec{a}$ that has polynomials with ℓ-bit coefficients into an N-dimensional vector with polynomials of 0/1 coefficients. $\mathsf{BitDecomp}^{-1} : \mathcal{R}_q^N \to \mathcal{R}_q^k$ is the inverse of this operation. Specifically, for a vector $\vec{a} = (a_{1,0}, \ldots, a_{1,\ell-1}, \ldots, a_{k,0}, \ldots, a_{k,\ell-1}) \in \mathcal{R}_q^N$, we have

$$\mathsf{BitDecomp}^{-1}(\vec{a}) = \left(\sum 2^j a_{1,j}, \ldots, \sum 2^j a_{k,j}\right)$$

Note that we define this function for any vector $\vec{a} \in \mathcal{R}_q^N$, not just vectors with 0/1 polynomials (which is what $\mathsf{BitDecomp}$ produces). Next, for a vector $\vec{a} \in \mathcal{R}_q^N$, define $\mathsf{Flatten} : \mathcal{R}_q^N \to \mathcal{R}_q^N$ by composing these functions:

$$\mathsf{Flatten}(\vec{a}) = \mathsf{BitDecomp}(\mathsf{BitDecomp}^{-1}(\vec{a}))$$

Finally, define $\mathsf{Powersof2} : \mathcal{R}_q^k \to \mathcal{R}_q^N$ by

$$\mathsf{Powersof2}(\vec{a}) = (a_1, 2a_1, \ldots, 2^{\ell-1}a_1, \ldots, a_k, 2a_k, \ldots, 2^{\ell-1}a_k)$$

As in GSW, the important properties of these operations are:

- $\langle \mathsf{BitDecomp}(\vec{a}), \mathsf{Powersof2}(\vec{b}) \rangle = \langle \vec{a}, \vec{b} \rangle$
- $\langle \mathsf{Flatten}(\vec{a}), \mathsf{Powersof2}(\vec{b}) \rangle = \langle \vec{a}, \mathsf{Powersof2}(\vec{b}) \rangle$

We extend these operations to matrices by performing them on each row. Thus, for matrices $A \in \mathcal{R}_q^{N \times k}$, $C \in \mathcal{R}_q^{N \times N}$ and a vector $\vec{b} \in \mathcal{R}_q^k$ we have:

- $\mathsf{BitDecomp}(A) \cdot \mathsf{Powersof2}(\vec{b}) = A \cdot \vec{b}$
- $\mathsf{Flatten}(C) \cdot \mathsf{Powersof2}(\vec{b}) = C \cdot \mathsf{Powersof2}(\vec{b})$

The $\mathsf{Flatten}$ function keeps our encodings small while maintaining correctness with the zero-tester, which is a result of the $\mathsf{Powersof2}$ function. In our scheme we set $k = 2$ so that $N = 2\ell$.

2.2 GSW encryption

In this section we define a couple nondeterministic helper functions that correspond to encryption in GSW:

$(\vec{v}, A) \leftarrow \mathsf{generate}(1^\lambda, m, \chi)$: For parameters $\lambda, m \in \mathbb{N}$ and χ, an error distribution on $\mathcal{R}_q^m$, $\mathsf{generate}$ outputs a GSW key pair $(\vec{v}, A) \in \mathcal{R}_q^N \times \mathcal{R}_q^{m \times 2}$ by drawing $\vec{b} \leftarrow \mathcal{R}_q^m$, $t \leftarrow \mathcal{R}_q$, and $\vec{e} \in \chi^m$. Set $\vec{a} = \vec{b}t + \vec{e}$ and $\vec{s} = (1, -t)^T$. Finally, output $(\vec{v} = \mathsf{Powersof2}(s), A = \left(\vec{b} \mid \vec{a}\right))$. Note that $A\vec{s} = \vec{a} - \vec{b}t = \vec{e}$. We use this for decryption.

$C \leftarrow \mathsf{encrypt}(\mu, A, m)$: For a plaintext $\mu \in \mathcal{R}_q$, a public key $A \in \mathcal{R}_q^{m \times 2}$ as output by the $\mathsf{generate}$ function and dimension $m \in \mathbb{N}$, $\mathsf{encrypt}$ outputs $\mathsf{Flatten}(\mu I_N + \mathsf{BitDecomp}(R \cdot A))$ where $R \leftarrow \{0,1\}^{N \times m}$. When A and m are unambiguous from context we will write $E(\mu) = \mathsf{encrypt}(\mu, A, m)$.

$\mu \leftarrow \mathsf{decrypt}(C, \vec{v})$: Given a ciphertext C of a plaintext μ and a decryption key $\vec{v}$ as output by the $\mathsf{generate}$ function, $\mathsf{decrypt}$ recovers μ from the product $C\vec{v}$.

Note that decryption is possible because $\vec{v} = \mathsf{Powersof2}(\vec{s})$ and $A\vec{s} = \vec{a} - \vec{b}t = \vec{e}$, so by the properties of the bit operations above, $\mathsf{BitDecomp}(R \cdot A)\vec{v} = R\vec{e}$ for any $R \in \mathcal{R}_q^{N \times m}$. Thus, as we can write $C = \mathsf{Flatten}(\mu I_N + \mathsf{BitDecomp}(R \cdot A))$ for a random binary matrix $R \leftarrow \{0,1\}^{N \times m}$, we have:

$$\begin{aligned} C\vec{v} &= (\mu I_N + \mathsf{BitDecomp}(R \cdot A))\vec{v} \\ &= \mu\vec{v} + R\vec{e} \end{aligned}$$

Thus, as $\vec{e}$ is small and the first ℓ coefficients of $\vec{v}$ are of the form 2^i for $0 \le i \le \ell - 1$, we can recover μ.

From [GSW13] we have a couple important homomorphic properties, which we summarize in the following lemma:

Lemma 1 ([GSW13]). *Let* $(\vec{v}, A) \leftarrow \mathsf{generate}(1^\lambda, m, \chi)$ *for some* λ, m, χ. *Then addition and multiplication are homomorphic with respect to* decrypt *and* encrypt*:*

- $\mathsf{decrypt}(E(\mu_1 + \mu_2), \vec{v}) = \mathsf{decrypt}(E(\mu_1) + E(\mu_2), \vec{v})$
- $\mathsf{decrypt}(E(\mu_1\mu_2), \vec{v}) = \mathsf{decrypt}(E(\mu_1) \cdot E(\mu_2), \vec{v})$

It is obvious that the first property holds; we briefly recall the proof from [GSW13] that shows the second does as well:

$$\begin{aligned} C_1 \cdot C_2\vec{v} &= C_1(\mu_2\vec{v} + \vec{e_2}) \\ &= \mu_2(\mu_1\vec{v} + \vec{e_1}) + C_1\vec{e_2} \\ &= \mu_1\mu_2\vec{v} + \mu_2\vec{e_1} + C_1\vec{e_2} \end{aligned}$$

Clearly, we can also recover $\mu_1\mu_2$ from this as long as μ_2 and the errors $\vec{e_1}, \vec{e_2}$ are all small relative to q because C_1 is the result of the $\mathsf{Flatten}$ function.

3 Our construction

We are now ready to outline our map, which is an instantiation of a graded encoding scheme as in prior works [CLT13, GGH13a, CLT15]. Refer to Appendix A for an explicit definition of graded encoding schemes.

In general, a fresh encoding of a plaintext $\mu \in \mathcal{R}_q$ at level-i will look like the following, where z is a random ring element and T is a random matrix:

$$C = z^{-i}T \cdot E(\mu) \cdot T^{-1}$$

recalling that $E(\mu) = \mathsf{Flatten}\left(\mu I_N + \mathsf{BitDecomp}(R \cdot A)\right))$ is a GSW encryption of μ. Thus, encodings are simply GSW ciphertexts (which are small, as they are a result of the $\mathsf{Flatten}$ function) pre/post-multiplied by a random matrix T and its inverse and by the ring element z^{-i} to introduce the notion of levels. For an encoding C of a plaintext μ we will denote the corresponding GSW encryption by C_μ.

Setup: $(\mathsf{pp}, \boldsymbol{p}_{zt}) \leftarrow \mathsf{setup}(1^\lambda, 1^\kappa)$**.** Given the security parameter λ and the multilinearity level κ, we generate the public parameters and the zero-testing element. The parameters q and n define the ring $\mathcal{R}_q$. Set $m = O(\log q)$ and $N = 2\ell$ where $\ell = \lceil \log q \rceil$. Let χ be an error distribution on $\mathcal{R}_q$. σ is a parameter for sampling plaintext elements from a discrete Gaussian distribution. We generate the GSW key pair $(\vec{v}, A)\mathsf{generate}(1^\lambda, m, \chi)$. We generate the **zero-testing parameter** $\boldsymbol{p}_{zt}$ by first drawing uniform matrices $U \leftarrow \mathcal{R}_q^{N\times 2}$ and $T \leftarrow \mathcal{R}_q^{N\times N}$ and setting

$$\vec{w} = (1, 1, \ldots, 1) \cdot \mathsf{BitDecomp}(U) \cdot T^{-1}$$

Then draw a uniform element $z \leftarrow \mathcal{R}_q$ and then a few "somewhat small" elements $\alpha \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma_\alpha}$, $\vec{\epsilon} \leftarrow (\mathcal{D}_{\mathbb{Z}^n,\sigma_\delta})^N$, and $\vec{\delta} \leftarrow (\mathcal{D}_{\mathbb{Z}^n,\sigma_\epsilon})^N$ for some parameters $\sigma_\alpha, \sigma_\delta$, and σ_ϵ. We then set

$$\vec{u} = z^\kappa T \cdot (\alpha(\vec{v} + \vec{\delta}) + \vec{\epsilon})$$

Intuitively, $\vec{u}$ is a (R)LWE "encryption" of $\vec{v}$, the original decryption key in GSW (plus some other noise that we need). We generate a level-1 encoding of 1

$$Y = z^{-1}T \cdot E(1) \cdot T^{-1}$$

We generate the rerandomization parameters

$$X_i = z^{-1}T \cdot E(0) \cdot T^{-1} \quad \forall i \in [\tau]$$

Note that these are level-1 encodings of 0. We output $\mathsf{pp} = (\kappa, q, n, N, \sigma, Y, \{X_i\}_{i=1}^{\tau})$, P, and $\boldsymbol{p}_{zt} = (\vec{w}, \vec{u})$.

Zero testing: $\mathsf{isZero}(C, \boldsymbol{p}_{zt}) \stackrel{?}{=} 0/1$. To zero-test an encoding $C = z^{-\kappa}TC_\mu T^{-1}$ of μ at level-κ, output 1 if $\|\langle \vec{w}, C\vec{u}\rangle\|_\infty < P$ and 0 otherwise. To see briefly why this is correct, recall the properties of the bit operations and GSW decryption from Section 2. Then we have:

$$\begin{aligned}
C\vec{u} &= z^{-\kappa}TC_\mu T^{-1} \cdot \vec{u} \\
&= z^{-\kappa}TC_\mu T^{-1} \cdot z^\kappa T \cdot ((\alpha(z^\kappa \vec{v} + \vec{\delta}) + \vec{\epsilon}) \\
&= (z^{-\kappa}z^\kappa)T \cdot C_\mu(\alpha(\vec{v} + \vec{\delta}) + \vec{\epsilon}) \\
&= T \cdot (\alpha(C_\mu \vec{v} + C_\mu \vec{\delta}) + C_\mu \vec{\epsilon}) \\
&= T \cdot (\alpha\left(\mathsf{Flatten}(\mu I_N + \mathsf{BitDecomp}(R \cdot A))\vec{v} + C_\mu \vec{\delta}\right) + C_\mu \vec{\epsilon}) \\
&= T \cdot (\alpha(\mu \vec{v} + \vec{e}) + C_\mu \vec{\delta}) + C_\mu \vec{\epsilon})
\end{aligned}$$

which gives us

$$\begin{aligned}
\langle \vec{w}, C\vec{u}\rangle &= (1, 1, \ldots, 1) \cdot \mathsf{BitDecomp}(U) \cdot T^{-1} \cdot T \cdot (\alpha((\mu \vec{v} + \vec{e}) + C_\mu \vec{\delta}) + C_\mu \vec{\epsilon}) \\
&= (1, 1, \ldots, 1) \cdot \mathsf{BitDecomp}(U) \cdot (\alpha((\mu \vec{v} + \vec{e}) + C_\mu \vec{\delta}) + C_\mu \vec{\epsilon}) \\
&= (1, 1, \ldots, 1) \cdot (\mu(\alpha U \vec{s}) + \mathsf{BitDecomp}(U) \cdot (\alpha \vec{e} + C_\mu(\alpha \vec{\delta} + \vec{\epsilon}))) \\
&= (1, 1, \ldots, 1) \cdot (\mu(big) + small)
\end{aligned}$$

Intuitively, note that this element is large if and only if $\mu \neq 0$ because all elements are small except for $\vec{v}$ and U. Formally, we prove the following lemma in the appendix.

Lemma 2. *This is the lemma that formally outlines the size difference between a zero and nonzero encoding (informally stated for now). Let $C = z^{-\kappa}TC_\mu T^{-1}$ where C_μ is a GSW encryption of $\mu \in \mathcal{R}_q$ (i.e., $C_\mu \vec{v}$ decrypts to μ). Then $\|\langle \vec{w}, C\vec{u}\rangle\|_\infty < P$ if and only if $\mu = 0$.*

Addition & subtraction. For two encodings C_1 and C_2 at the same level-i, addition and subtraction are the corresponding matrix operations. As $C_{\mu_1} + C_{\mu_2}$ is a valid GSW ciphertext of $\mu_1 + \mu_2$ by Lemma 1, the correctness with respect to zero-testing for $C_1 + C_2 = z^{-i}T(C_{\mu_1} + C_{\mu_2})T^{-1}$ follows from Lemma 2. Subtraction follows similarly.

Multiplication. Multiplication in our scheme is also matrix multiplication. Suppose that C_1 and C_2 are encodings at level-i and level-j, respectively, with $i + j \leq \kappa$ and corresponding GSW ciphertexts C_{μ_1} and C_{μ_2}. First note that

$$\begin{aligned} C_1 \cdot C_2 &= (z^{-i} T C_{\mu_1} T^{-1}) \cdot (z^{-j} T C_{\mu_2} T^{-1}) \\ &= z^{-i-j} T (C_{\mu_1} \cdot C_{\mu_2}) T^{-1} \end{aligned}$$

which is a level-$(i+j)$ encoding of $\mu_1\mu_2$ by Lemma 2, because $C_{\mu_1} \cdot C_{\mu_2}$ correctly decrypts to $\mu_1\mu_2$, by Lemma 1, so long as μ_2 is small—for this reason our message space is restricted, as in CLT and GGH (to control noise growth).

Sampling: $\mu \leftarrow \mathsf{samp}(\mathsf{pp})$. To sample a level-0 encoding—that is, a plaintext element—simply draw $\mu \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$.

Rerandomization: $C' \leftarrow \mathsf{rerand}(\mathsf{pp}, C)$. To rerandomize a level-1 encoding C, draw $r_j \leftarrow \{0,1\}$ for $0 \leq j \leq \tau$ and compute

$$C' = C + \sum_{j=1}^{\tau} r_j X_j$$

using the addition operator of the scheme.

Encoding: $C \leftarrow \mathsf{enc}(\mathsf{pp}, U, i)$. A user can encode a plaintext μ to level-i by first computing $Y' \leftarrow \mathsf{rerand}(\mathsf{pp}, Y)$, which will be a rerandomized level-1 encoding of 1, and then setting

$$C = \mu \cdot \left(Y'\right)^i$$

using the multiplication operator of the scheme, which will be a level-i encoding of μ.

Extraction: $sk \leftarrow \mathsf{ext}(\mathsf{pp}, \boldsymbol{p}_{zt}, C^{(\kappa)})$. As in [GGH13a, CLT15] we apply the zero-tester to a level-κ encoding $C^{(\kappa)}$ and collect the most significant bits to extract a random function of the underlying plaintext μ.

Lemma 3. *This is the lemma that states we can extract a random function of λ bits from a level-κ encoding.*

3.1 Asymmetric variants

As in CLT and GGH we can instead compute z_i for $i \in [\kappa]$ and encode plaintexts to specific index sets $S \subseteq [\kappa]$ rather than levels. This is required for some applications like obfuscation. We can also break the commutativity of the scheme and enforce a particular order of multiplication by generating many matrices T_i rather than a single matrix T.

3.2 Setting parameters

As in homomorphic encryption schemes and multilinear maps, the size of the noise at level-κ is dependent on the size of the initial error B_e, the multilinearity level κ, and the initial size of plaintexts B_μ. Specifically, we have the noise bounded by:

$$B_\kappa = (nN \cdot B_\mu)^\kappa \cdot B_e \tag{2}$$

Then, when we apply the zero-testing element at level-κ we have:

$$C\boldsymbol{p}_{zt} = \alpha(\mu\vec{v} + \vec{e_\circ} + C\vec{\delta}) + C\vec{\epsilon} \tag{3}$$

where the size of $e_\circ$ is bounded by B_κ, as in (2). In the case of $\mu = 0$ we then have the size of (3) bounded by $B_\alpha(B_\kappa + nNB_\delta) + nNB_\epsilon$.

Parameters that need to be set.

- n, the degree of polynomials. Dependent on RLWE security.
- q, the size of the underlying field. Dependent on RLWE security and (more importantly) the size of zero-tested elements, which depends on $n, N, \chi, \kappa, \sigma, \sigma_\alpha, \sigma_\delta, \sigma_\epsilon$.
- σ, the parameter for the Gaussian that samples plaintexts. GGH bounds the size of their ciphertexts (not their plaintexts by $q^{1/8}$).
- $\sigma_\alpha, \sigma_\delta, \sigma_\epsilon$, the parameters for the Gaussians that draw noise elements for the zero-tester. GGH sets σ_α to be $\sqrt{q}$. Should be "as large as possible" but obviously cannot yield a uniform distribution.
- χ, the error distribution for the RLWE instance A.
- τ, the number of encodings of 0 we publish for rerandomization.

4 Security

4.1 Graded DDH

In this section we instantiate the Graded Decisional Diffie-Hellman assumption for our scheme, introduced in [GGH13a] as the graded encoding scheme analogue of the multilinear decisional Diffie-Hellman assumption in [BS03]. For a full definition of graded encoding schemes, refer to Appendix A.

Definition 1 (κ-GDDH)**.** *For a security parameter $\lambda \in \mathbb{N}$ the κ-Graded Decisional Diffie-Hellman assumption is to distinguish between the distributions $\mathcal{D}_0[(\mathsf{pp}, \boldsymbol{p}_{zt})]$ and $D_1[(\mathsf{pp}, \boldsymbol{p}_{zt})]$ for $(\mathsf{pp}, \boldsymbol{p}_{zt}) \leftarrow \mathsf{setup}(1^\lambda, 1^\kappa)$ where:*

$$\mathcal{D}_0[(\mathsf{pp}, \boldsymbol{p}_{zt})] = \left\{ \left(\{C'_i\}_{i\in[\kappa+1]}, \gamma\right) \;\middle|\; \begin{array}{ll} \forall i \in [\kappa+1], & U_i \leftarrow \mathsf{samp}(\mathsf{pp}), \\ & C_i \leftarrow \mathsf{enc}(\mathsf{pp}, 1, U_i) \\ & C'_i \leftarrow \mathsf{rerand}(\mathsf{pp}, C_i) \\ \gamma \leftarrow \mathsf{ext}(\mathsf{pp}, \boldsymbol{p}_{zt}, U_{\kappa+1} \cdot \prod_{i\in[\kappa]} C_i) & \end{array} \right\}$$

and

$$\mathcal{D}_1[(\mathsf{pp}, \boldsymbol{p}_{zt})] = \left\{ \left(\{C'_i\}_{i\in[\kappa+1]}, \gamma \right) \;\middle|\; \begin{array}{ll} \forall i \in [\kappa+1], & U_i \leftarrow \mathsf{samp}(\mathsf{pp}), \\ & C_i \leftarrow \mathsf{enc}(\mathsf{pp}, 1, U_i) \\ & C'_i \leftarrow \mathsf{rerand}(\mathsf{pp}, C_i) \\ \gamma \leftarrow \{0,1\}^\lambda & \end{array} \right\}$$

We conjecture that this problem is hard for our scheme.

4.2 Overview

Let n, q and χ be set such that the $\mathrm{RLWE}_{n,q,\chi}$ assumption holds. Let $\mathcal{D}_\alpha, \mathcal{D}_\delta, \mathcal{D}_\epsilon$ be discrete Gaussian distributions on $\mathbb{Z}^n$ with widths $\sigma_\alpha, \sigma_\delta, \sigma_\epsilon$ respectively.

The security of our scheme relies on hiding $\vec{v}$, z and T; with these parameters an adversary could decrypt an encoding at any level. We attempt to hide these values through creating (R)LWE instances. Thus, the security of the scheme relies on the fact that using the basic operations of the scheme (zero-testing, multiplication, addition), an adversary is only able to produce these RLWE-esque instances.

4.3 Zeroizing attacks

Attacks on previous schemes [CHL+14, CGH+15, HJ15] that make use of encodings of 0 fail against our scheme for a number of reasons, but the principal advantage that our scheme has over others is that zero-tested encodings of 0 are not exact multiples of noise elements, and instead result in RLWE-esque instances with some additive noise. To illustrate, a zero-tested encoding of 0 in [CLT13] or [GGH13a] is of the form hr where h is a "small-ish" noise element, like our α, and r is the noise present in the encoding. In our scheme, however, a zero-tested encoding of 0 is of the form

$$\langle \vec{w}, C\vec{u} \rangle = (1, 1, \ldots, 1) \cdot (\alpha(\mathsf{BitDecomp}(U)\vec{e_\circ} + \mathsf{BitDecomp}(U) \cdot C\vec{\delta}) + \mathsf{BitDecomp}(U) \cdot C\vec{\epsilon})$$

where $\vec{e_\circ}$ is the noise present in the encoding. So this result is essentially a ring element $hr + e$ in the notation of CLT and GGH, which breaks the previous attacks that rely on this multiplicative structure for things like computing GCDs and determinants.

The most recent CLT construction [CLT15] makes the zero-testing parameter so that zero-tested elements also have some additive term, due to the unknown modulus x_0, so that the attack also fails in a similar way. For that reason, we also conjecture that the subgroup membership and decision linear problems are hard in our scheme as the attacks that break those assumptions in the old CLT and GGH fail in our setting.

4.4 The homomorphism of Flatten

It is helpful to note that the Flatten function does not preserve all algebraic structure with respect to Powersof2 vectors. For example, matrix multiplication does not work in this sense:

$$\begin{aligned} \mathsf{Flatten}(AB) \cdot \mathsf{Powersof2}(\vec{v}) &= AB \cdot \mathsf{Powersof2}(\vec{v}) \\ &\neq \mathsf{Flatten}(A) \cdot \mathsf{Flatten}(B) \cdot \mathsf{Powersof2}(\vec{v}) \\ &= \mathsf{Flatten}(A) \cdot B \cdot \mathsf{Powersof2}(\vec{v}) \end{aligned}$$

Thus we would expect the use of this function to only reduce the number of possible algebraic attacks.

4.5 Attacks on the zero-testing element

We first want to rule out that we cannot get useful information out of the zero-testing elements themselves. Note that our zero-testing elements are the following vectors:

$$\vec{u} = z^\kappa T \cdot (\alpha(\vec{v} + \vec{\delta}) + \vec{\epsilon})$$
$$\vec{w} = (1, 1, \ldots, 1) \cdot \mathsf{BitDecomp}(U) \cdot T^{-1}$$

where z^κ, T, and U are uniformly random elements. Multiplying them together we get:

$$z^\kappa (1, 1, \ldots, 1) \cdot (\alpha(U\vec{s} + \mathsf{BitDecomp}(U) \cdot \vec{\delta}) + \mathsf{BitDecomp}(U) \cdot \vec{\epsilon})$$

4.6 Getting parameters from zero-tested encodings

We now draw our attention to using the zero-testing element in breaking the scheme, as is done in all attacks on previous candidates. A zero-tested encoding at level-i will look like the following, where $\vec{e_\circ}$ is the error after some number of operations:

$$\langle \vec{w}, C\vec{u} \rangle = z^{\kappa - i}(1, 1, \ldots, 1) \cdot (\mu(\alpha U \vec{s}) + \mathsf{BitDecomp}(U) \cdot (\alpha \vec{e_\circ} + C_\mu(\alpha\vec{\delta} + \vec{\epsilon})))$$

This is a standard RLWE instance in the case that $\mu \neq 0$ as U is a uniformly random matrix. However, in the case that $\mu = 0$ and $i = \kappa$ we have:

$$\langle \vec{w}, C\vec{u} \rangle = (1, 1, \ldots, 1) \cdot (\alpha(\mathsf{BitDecomp}(U) \cdot \vec{e_\circ} + C_\mu \vec{\delta}) + \mathsf{BitDecomp}(U) \cdot C_\mu \vec{\epsilon}) \tag{4}$$

This essentially reduces to this assumption:

Assumption 1 (RLWE-variant)**.** *Given $\alpha s + e \in \mathcal{R}_q$ where $(\alpha, b, c) \leftarrow \mathcal{D}_\alpha \times \mathcal{D}_\delta \times \mathcal{D}_\epsilon$, recover s or e.*

4.7 Using other parameters

Note that this assumption is a "first line of defense." For one thing, the parameters s and e are subset-sums of the coefficients of the actual noise $\vec{\delta}$ and $\vec{\epsilon}$ present in the zero-tester or the canonical RLWE instance A. Additionally, even in the case that these noise parameters are recovered it seems infeasible to recover the decryption key from the zero-testing parameters without knowing T or z.

We now consider some cases where other parameters are leaked: an adversary could do some malicious things by recovering T and z—for example, it would be possible to zero-test at any level, and move encodings arbitrarily between levels—but the decryption key in the zero-testing parameter is would still be hidden by RLWE instance with α as a "public key," so recovering plaintexts by getting the GSW decryption key seems unlikely using just T and z.

If an adversary had all of $\alpha, T, \vec{\epsilon}$, and z then the adversary could definitely recover the decryption key; however if they were missing even one of those it seems infeasible under similar assumptions to Assumption 1. We have not been able to attack the scheme in any way in the event that A or α were made public.

5 An application: N-way key exchange

We now present the canonical use-case for multilinear maps: N-way one round key exchange, which we outline for our construction in the following steps.

1. Let $\kappa = N - 1$. A trusted third party obtains the public parameters $(\mathsf{pp}, \boldsymbol{p}_{zt}) \leftarrow \mathsf{setup}(1^\lambda, 1^\kappa)$ and subsequently distributes the parameters to each player.
2. Each player i runs the $\mathsf{samp}(\mathsf{pp})$ algorithm to obtain a random level-0 encoding U_i. Then the player computes $C_i \leftarrow \mathsf{enc}(\mathsf{pp}, 1, U_i)$ and publishes this level-1 encoding.
3. The ith player now computes the level-κ encoding

$$C_i' = U_i \cdot \prod_{j \neq i \in [N]} C_j$$

   using the multiplication operator of the scheme.
4. Finally, each player computes $s \leftarrow \mathsf{ext}(\mathsf{pp}, \boldsymbol{p}_{zt}, C_i')$ to extract the shared secret s.

Correctness follows from Lemma 3, whereas security follows from the hardness of the GDDH assumption for our scheme.

References

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213-229, 2001.

[BLR+15] Dan Boneh and Kevin Lewi and Mariana Raykova and Amit Sahai and Mark Zhandry and Joe Zimmerman. Semantically Secure Order-Revealing Encryption: Multi-Input Functional Encryption Without Obfuscation. EUROCRYPT 2015.

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

[BWZ14] Dan Boneh, David J. Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. Cryptology ePrint Archive, Report 2014/930, 2014. http://eprint.iacr.org/.

[CGH+15] Jean-Sebastien Coron and Craig Gentry and Shai Halevi and Tancrede Lepoint and Hemanta K. Maji and Eric Miles and Mariana Raykova and Amit Sahai and Mehdi Tibouchi. Zeroizing Without Low-Level Zeroes: New MMAP Attacks and Their Limitations. Cryptology ePrint Archive, Report 2015/596, 2015. http://eprint.iacr.org/. To appear at CRYPTO 2015.

[CHL+14] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, Damien Stehl. Cryptanalysis of the multilinear maps over the integers. Cryptology ePrint Archive, Report 2014/906, 2014. http://eprint.iacr.org/. To appear at EUROCRYPT 2015.

[CLT13] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *CRYPTO*, pages 476-493, 2013.

[CLT14] Jean-Sbastien Coron, Tancrde Lepoint, and Mehdi Tibouchi. Cryptanalysis of two candidate fixes of multilinear maps over the integers. Cryptology ePrint Archive, Report 2014/975, 2014. http://eprint.iacr.org/.

[CLT15] Jean-Sbastien Coron, Tancrde Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In R. Gennaro and M. Robshaw, editors, CRYPTO 2015, Lecture Notes in Computer Science. Springer, 2015. To appear.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT 2013*. Springer Berlin Heidelberg, 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. Cryptology ePrint Archive, Report 2013/451, 2013. http://eprint.iacr.org/.

[GGH14a] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. Cryptology ePrint Archive, 2014:645, 2014. To appear at TCC 2015.

[GGH+14b] Craig Gentry, Shai Halevi, Hemanta K. Maji, and Amit Sahai. Zeroizing without zeroes: Cryptanalyzing multilinear maps without encodings of zero. Cryptology ePrint Archive, Report 2014/929, 2014. http://eprint.iacr.org/.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In Advances in Cryptology—CRYPTO 2013, R. Canetti and J. Garay, Eds., vol. 8042 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 75-92.

[HJ15] Y. Hu and H. Jia. Cryptanalysis of GGH map. Cryptology ePrint Archive, Report 2015/301, 2015. http://eprint.iacr.org/.

[KGV14] Alhassan Khedr, Glenn Gulak, and Vinod Vaikuntanathan, SHIELD: Scalable Homomorphic Implementation of Encrypted Data-Classifiers, Cryptology ePrint Archive, Report 2014/838, 2014. http://eprint.iacr.org/.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In EUROCRYPT. 2010.

A Graded Encoding Schemes

Graded encoding schemes are an approximation of multilinear maps. In particular, encodings in graded encoding schemes can be randomized rather than deterministic. We first recall the original definition of multilinear maps:

Definition 1 (Multilinear Maps [BS03])**.** *A map $e : G_1^\kappa \to G_2$ is a* κ-multilinear map *if it satisfies the following properties:*

1. *G_1 and G_2 are groups of the same prime order p;*
2. *If $a_1, \ldots, a_\kappa \in \mathbb{Z}$ and $g_1, \ldots, g_\kappa \in G_1$ then*
$$e(g_1^{a_1}, \ldots, g_\kappa^{a_\kappa}) = e(g_1, \ldots, g_\kappa)^{a_1 \cdots a_\kappa};$$
3. *If g is a generator of G_1 then $e(g, \ldots, g)$ is a generator of G_2.*

Graded encoding schemes are similar, except they give rise to many maps $G_i \times G_j \to G_{i+j}$ through the multiplication operation, and encodings can be randomized. We begin by recalling the definition of graded encoding systems:

Definition 2 (Graded Encoding Systems [GGH13a])**.** *A* κ-graded encoding system *for a ring R is a system of sets $\mathcal{S} = \{S_v^{(\alpha)} \in \{0,1\}^* \mid v \in \mathbb{N}, \alpha \in R\}$ that satisfies the following properties:*

1. *For every $v \in \mathbb{N}$, the sets $\{S_v^{(\alpha)} \mid \alpha \in R\}$ are disjoint;*
2. *There are binary operations $+$ and $-$ such that for every α_1, α_2, $v \in \mathbb{N}$, and every $u_1 \in S_v^{(\alpha_1)}$ and $u_2 \in S_v^{(\alpha_2)}$ we have*
$$u_1 + u_2 \in S_v^{(\alpha_1+\alpha_2)}$$
$$u_1 - u_2 \in S_v^{(\alpha_1-\alpha_2)}$$
where $\alpha_1 + \alpha_2$ and $\alpha_1 - \alpha_2$ are addition and subtraction in the ring R;

3. *There is an (associative) binary operation* $\cdot$ *such that for every* $\alpha_1, \alpha_2 \in R$ *and* $v_1, v_2 \in \mathbb{N}$ *such that* $v_1 + v_2 \leq \kappa$ *and every* $u_1 \in S_v^{(\alpha_1)}$ *and* $u_2 \in S_v^{(\alpha_2)}$ *we have*

$$u_1 \cdot u_2 \in S_{v_1+v_2}^{(\alpha_1 \cdot \alpha_2)}$$

where $\alpha_1 \cdot \alpha_2$ *is multiplication in* R.

The related definition of graded encoding schemes specifies the public sampling and rerandomization of elements, which we adapt from [CLT15]:

Definition 3 (Graded Encoding Schemes)**.** *A (symmetric)* κ-graded encoding scheme *for a ring* R *is a system of sets* $\mathcal{S} = \{S_v^{(\alpha)} \in \{0,1\}^* \mid v \in \mathbb{N}, \alpha \in R\}$ *consists of the following PPT procedures:*

$\mathsf{setup}(1^\lambda, 1^\kappa)$: *For a security parameter* λ *and multilinearity level* κ*, output* $(\mathsf{pp}, \boldsymbol{p}_{zt})$ *where* pp *is the description of a graded encoding system and* $\boldsymbol{p}_{zt}$ *is a zero-test parameter;*

$\mathsf{samp}(\mathsf{pp})$: *Outputs a random level-0 encoding* $u \in S_0^{(\alpha)}$ *for a random* $\alpha \in R$*;*

$\mathsf{enc}(\mathsf{pp}, i, u)$: *For a level* $i \leq k$ *and a level-0 encoding* $u \in S_0^{(\alpha)}$*, outputs a level-*i *encoding* $u' \in S_i^{(\alpha)}$*;*

$\mathsf{rerand}(\mathsf{pp}, i, u)$: *For a level-*i *encoding* $u \in S_i^{(\alpha)}$*, outputs another level-*i *encoding* $u' \in S_i^{(\alpha)}$ *such that for any two* $u_1, u_2 \in S_i^{(\alpha)}$ *the output distributions of* $\mathsf{rerand}(\mathsf{pp}, i, u_1)$ *and* $\mathsf{rerand}(\mathsf{pp}, i, u_2)$ *are nearly the same;*

$\mathsf{neg}(\mathsf{pp}, u)$: *For a level-*i *encoding* $u \in S_i^{(\alpha)}$*, outputs another level-*i *encoding* $u' \in S_i^{(-\alpha)}$*;*

$\mathsf{add}(\mathsf{pp}, u_1, u_2)$: *For two level-*i *encodings* $u_1 \in S_i^{(\alpha_1)}$ *and* $u_2 \in S_i^{(\alpha_2)}$*, outputs another level-*i *encoding* $u' \in S_i^{(\alpha_1+\alpha_2)}$*;*

$\mathsf{mult}(\mathsf{pp}, u_1, u_2)$: *For a level-*i *encoding* $u_1 \in S_i^{(\alpha_1)}$ *and a level-*j *encoding*$u_2 \in S_j^{(\alpha_2)}$ *where* $i+j \leq \kappa$*, outputs a level-*$(i+j)$ *encoding* $u' \in S_{i+j}^{(\alpha_1 \cdot \alpha_2)}$*;*

$\mathsf{isZero}(\mathsf{pp}, u, \boldsymbol{p}_{zt})$: *For a level-*κ *encoding* $u \in S_\kappa^{(\alpha)}$ *and zero-testing parameter* $\boldsymbol{p}_{zt}$*, outputs* 1 *if* $\alpha = 0$ *and* 0 *otherwise with negligible probability;*

$\mathsf{ext}(\mathsf{pp}, \boldsymbol{p}_{zt}, u)$: *For a level-*k *encoding* $u \in S_\kappa^{(\alpha)}$ *and zero-testing parameter* $\boldsymbol{p}_{zt}$*, outputs a* λ*-bit string* s *such that:*

1. *For any* $\alpha \in R$ *and* $u_1, u_2 \in S_\kappa^{(\alpha)}$*,* $\mathsf{ext}(\mathsf{pp}, \boldsymbol{p}_{zt}, u_1) = \mathsf{ext}(\mathsf{pp}, \boldsymbol{p}_{zt}, u_2)$*;*
2. *The distribution* $\{\mathsf{ext}(\mathsf{pp}, \boldsymbol{p}_{zt}, u) \mid \alpha \leftarrow R, u \leftarrow S_\kappa^{(\alpha)}\}$ *is nearly uniform over* $\{0,1\}^\lambda$.

B Proof of Lemma 2

C Proof of Lemma 3