Graded Encoding, Variations on a Scheme

(Preliminary Report)

Shai Halevi
IBM

September 7, 2015

Abstract

In this note we provide a more-or-less unified framework to talk about the functionality and security of graded encoding schemes, describe some variations of recent schemes, and discuss their security. In particular we describe schemes that combine elements from both the GGH13 scheme of Garg, Gentry and Halevi (EUROCRYPT 2013) and the GGH15 scheme of Gentry, Gorbunov and Halevi (TCC 2015). On one hand, we show how to use techniques from GGH13 in the GGH15 construction to enable encoding of arbitrary plaintext elements (as opposed to only small ones) and to introduce "levels/subsets" (e.g., as needed to implement straddling sets). On the other hand, we show how to modify the GGH13 scheme to support graph-induced constraints (either instead of, or in addition to, the levels from GGH13).

Turning to security, we describe zeroizing attacks on the GGH15 scheme, similar to those described by Cheon et al. (EUROCRYPT 2015) and Coron et al. (CRYPTO 2015) on the CLT13 and GGH13 constructions. As far as we know, however, these attacks to not break the GGH15 multi-partite key-agreement protocol. We also describe a new multi-partite key-agreement protocol using the GGH13 scheme, which also seems to resist known attacks. That protocol suggests a relatively simple hardness assumption for the GGH13 scheme, that we put forward as a target for cryptanalysis.

Keywords. Cryptography Multilinear Maps, Graded Encoding, Multi-partite Key-Agreement, Zeroizing Attacks.

This work was done in part while the author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467 and additional support by NSF grant #1017660. This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) and Army Research Office(ARO) under Contract No. W911NF-15-C-0236.

Contents

1 Introduction

Graded encoding schemes (aka "cryptographic multilinear maps") are a powerful tool, enabling many new applications — most notably to obfuscation and functional encryption, e.g., [GGH+13b, GGHZ14]. The first graded encoding scheme candidate was described by Garg, Gentry and Halevi [GGH13a], quickly followed by another candidate of Coron, Lepoint and Tibouchi [CLT13], and a little later a third candidate by Gentry, Gorbunov and Halevi [GGH15]. (A variant of the GGH13 scheme, aimed at improved efficiency, was suggested by Langlois, Stehlé, and Steinfeld [LSS14].) The functionality of these different constructions is similar, but not quite the same and not as simple as we would want. Indeed, so far we do not have a commonly-accepted syntax for describing the intended functionality (in fact it is not even clear in what sense these different schemes implement the same primitive).

The situation with respect to security is even more confusing. Building on the initial cryptanalysis in [GGH13a, CLT13], recent months saw a slew of attacks (cf. [CHL+15, CGH+15, HJ15, CL15, BGH+15]), breaking several applications of these schemes and many concrete hardness assumptions that were made about them, along with some attempts to protect against these attacks [CLT15]. At present there is very little clarity in the research community regarding the security of these schemes, with contradictory statement being made about them.

In this note we attempt to somewhat clarify this muddled field, by providing a more-or-less unified framework to talk about the functionality and security of graded encoding schemes. Specifically we put forward a common simple syntax that captures the functionality of nearly all current schemes, and a framework for describing attacks and hardness assumptions about them.[1]

Roughly speaking, a graded-encoding scheme has three parts: key-generation that outputs a public key and a secret key, encoding procedure that uses the secret key to encode plaintext values, and operations on the encoded values using the public key. The security properties of a scheme are expressed relative to what plaintext values are "safe" to encode. That is, with each scheme we have a simple target for the attack (i.e., when the adversary wins), and a particular distribution of plaintext values is "unsafe" if the attacker can win when seeing the public key and an encoding of plaintext values from that distribution. The security properties of a candidate scheme tell us what plaintext distributions are "safe," and the crypto-design challenge is to construct applications that only use such "safe" distributions.

With this framework in place, we describe some new variations of current schemes, combining elements from the GGH13 and GGH15 schemes to get a mix of their functionality. Specifically we show how to add element from GGH13 to the GGH15 construction to enable encoding of arbitrary plaintext elements (as opposed to only small ones) and to introduce "levels/subsets" (e.g., as needed to implement straddling sets [BGK+14]), and we show how to modify the GGH13 scheme to support GGH15-like graph-induced constraints (either instead of, or in addition to, the levels from GGH13).

Turning to security, we describe zeroizing attacks on the GGH15 scheme, similar to those described by Cheon et al. (EUROCRYPT 2015) and Coron et al. (CRYPTO 2015) on the CLT13 and GGH13 constructions. These attacks require that certain type of zero encoding is available to the attacker, and they apply to all the variations that we describe in this note. We comment, however, that such attacks do not seem to break the GGH15 multi-partite key-agreement protocol. We also adapt the GGH15 multi-partite key-agreement protocol to use the GGH13 graded-encoding scheme instead, and note that this protocol too seems to resist all the known attacks. This last

[1] This framework was sketched in an invited talk by the author in CRYPTO 2015.[Hal15]

protocol suggests a relatively simple hardness assumption for the GGH13 scheme, that we put forward as a target for cryptanalysis.

Organization. In Section 2 we describe our syntax for the functionality of graded-encoding schemes, and also discuss a language for talking about their security properties. In Section 3 we recall some facts and definitions regarding lattice and ideal lattices, and then describe the GGH13 and GGH15 schemes. In Section 4 we show how to add elements from the GGH13 construction to the GGH15 scheme in order to enhance its functionality, followed by a description in Section 5 of zeroizing attacks on GGH15-type constructions. Next in Section 6 we show how to add graph-induced constraints to the GGH13 scheme, which can be used to implement the multi-partite key-agreement protocol from [GGH15] using the GGH13 scheme as the underlying graded-encoding scheme. Finally in Section 7 we describe a slightly different key-agreement protocol based on the GGH13 scheme, and put forward a relatively simple hardness assumption that seems to capture its security.

Acknowledgments. This work benefited from discussions with very many people, a partial list includes Ran Canetti, Craig Gentry, Tancrède Lepoint, Daniele Micciancio, Chris Peikert, and Mariana Raykova.

2 Syntax of Graded Encoding

Unfortunately, the syntax of current graded-encoding candidates is more complex than we would like, and moreover different constructions expose somewhat different interfaces. Below we describe syntax that captures (almost) all the schemes in the literature.[2] A graded-encoding scheme has three parts: key-generation, secret-key encoding, and public-key operations.

- *Key-generation* KeyGen takes as input the security parameter λ and a "functionality specifier" that govern the operations that can be applied to encoded values. Examples of the latter include the multi-linearity parameter κ, the set-system used to define straddling-sets, the DAG used in graph-induced schemes, etc. Specifically we are given a set of tags that are associated with encoded values (e.g., levels, paths, etc.), and rules for what operations are permitted on encoding relative to what tags and what is the tag of the resulting encoding.

  The key-generation procedure outputs a public key and a secret key $(\mathsf{pp}, \mathsf{sp})$, and the secret key sp includes also a description of the plaintext space, which is either a ring or a subset of a ring (e.g., the small matrices that can be encoded in GGH15).

- *Encoding* takes the secret key, an element of the plaintext space, and a tag, and returns an encoding of the given plaintext element relative to the tag.

- *Operations.* There are at least three operations: addition, multiplication, and zero-test, and sometime we also have an extraction operation.

  The addition and multiplication operations take as input two tagged encodings, and if their tags are "compatible" then it returns a new encoding of the sum/product of the arguments relative to an output tag.

[2] The only exception that we know of is the scheme of Garg et al. [GGHZ14] with dynamic levels, see discussion later in this section.

The zero test takes a tagged encoding, and if the zero-test for that tag is allowed then it returns a bit indicating whether or not the encoded element equals to zero. The extraction operation can be applied to the same tagged encodings that permit zero-test, and it returns a bit string in $\{0,1\}^\lambda$.

The tags that we consider in this report are paths in a DAG, subsets of a given universe, or some combination thereof, all coupled with some size bounds. For example, for the GGH15 scheme as descried below, the tag space depends on a given DAG $G = (V, E)$ and the modulus q, where each tag consists of a path $u \rightsquigarrow v$ in G and bounds on the sizes of plaintext and noise:

$$TAG_{GGH15} = \big\{(u \rightsquigarrow v, \beta, \nu) : u \rightsquigarrow v \text{ is a path in } G, \nu < q/2^\lambda, \beta \leqslant q/2\big\}. \tag{1}$$

In this example, an operation is only permitted if the resulting noise estimate remains smaller than $q/2^\lambda$, and also we only allow adding encodings relative to the same path, only allow multiplication of consecutive paths, and only allow zero-testing (or extraction from) encoding relative to a source-to-sink path.

For another example, in (the "asymmetric" variant of) the GGH13 scheme, each tag consists of a subset of some given universe $[M]$ and a bound on the noise:

$$TAG_{asymGGH13} = \big\{(L, \nu) : L \subseteq [M], \nu \leqslant \sqrt{q}/2^\lambda\big\}. \tag{2}$$

Here too an operation is only permitted if the resulting noise estimate remains smaller than $\sqrt{q}/2^\lambda$, and also we only allow adding encodings relative to the same subset, only allow multiplication relative to disjoint subsets, and only allow zero-testing (or extraction from) encoding relative to the top subset $[M]$. In this note we also consider some combinations of the above, for example adding subsets to GGH15 we get the tag space

$$TAG_{both} = \big\{(u \rightsquigarrow v, L, \beta, \nu) : u \rightsquigarrow v \text{ is a path in } G, L \subseteq [M], \nu < q/2^\lambda, \beta \leqslant q/2\big\},$$

where we impose the constraints from both TAG_{GGH15} and $TAG_{asymGGH13}$ on the allowed operations.

Dynamic Tags. One variation of graded-encoding schemes which is not captured by the syntax above is the "dynamic levels" as used by Garg, Gentry, Halevi and Zhandry [GGHZ14]. In that case the levels correspond to subsets of some universe (as in $TAG2_\kappa$ from above), but the scheme allows users who only know the public key key to add new tags to the tag universe, and modify the public key and the public encodings to include also these new tags. We ignore this variation for the rest of this report, but note that the scheme from Section 4.2 (as well as the one from Section 6 with levels) supports also this extended syntax.

2.1 Correctness

The correctness requirement that we use here is taken from [GGH15] and adapted to our syntax. The tag space and rules about allowed operations, in conjunction with the procedures for sampling, encoding, and the operations, implicitly define the set $\mathcal{E}$ of "valid encodings" and its partition into sets $\mathcal{E}^{(\alpha)}$ of "valid encoding of α" for any α in the plaintext space or further partition to $\mathcal{E}^{(\alpha,t)}$ for every plaintext α and tag t.

For zero-testing we require that for every tag t such that zero-test is allowed for t we have $\mathsf{ZeroTest}(\mathsf{pp}, u) = 1$ for every $u \in \mathcal{E}^{(0,t)}$ (with probability one), and for every α in the plaintext space, $\alpha \neq 0$, it holds with overwhelming probability over key-generation that $\mathsf{ZeroTest}(\mathsf{pp}, u) = 0$ *for every encoding* $u \in \mathcal{E}^{(\alpha,t)}$.

For extraction,we roughly require that $\mathsf{Extract}$ outputs the same string on all the encodings of the same α, different strings on encodings of different α's, and random strings on encodings of "random α's." Formally, we require the following for every tag t such that $\mathsf{Extract}$ is allowed for t:

- For any plaintext element α, with overwhelming probability over $(\mathsf{sp}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}$, there exists a single value $x \in \{0,1\}^\lambda$ such that $\mathsf{Extract}(\mathsf{pp}, u) = x$ holds for all $u \in \mathcal{E}^{(\alpha)}$.
- For any $\alpha \neq \alpha'$, it holds with overwhelming probability over $(\mathsf{sp}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}$ that for any $u \in \mathcal{E}^{(\alpha,t)}$ and $u' \in \mathcal{E}^{(\alpha',t)}$, $\mathsf{Extract}(\mathsf{pp}, u) \neq \mathsf{Extract}(\mathsf{pp}, u')$.
- For any distribution $\mathcal{D}$ over plaintext elements with min-entropy 3λ or more, it holds with overwhelming probability over the keys $(\mathsf{sp}, \mathsf{pp}) \leftarrow \mathsf{KeyGen}$ that the induced distribution $\{\mathsf{Extract}(\mathsf{pp}, u) : \alpha \leftarrow \mathcal{D}, u \in S_d^{(\alpha)}\}$ is nearly uniform over $\{0,1\}^\lambda$.

We concede that the last conditions above is not always well defined, since the plaintext space itself depends on the keys pp, sp. One way to deal with this annoying technicality is to require a "meta plaintext space" that depends only on the security parameter, and can be mapped to the "actual plaintext space" once the keys are generated. For example in GGH13 we can use the integers $[N]$ for sufficiently large N as our meta plaintext space, and after choosing g interpret them as elements of the "actual space" R_g by identifying each integer n with the coset $n + gR$.

2.2 Security

As mentioned in the introduction, the security characteristics of a candidate graded-encoding scheme are determined by the question of what plaintext distributions are "safe" to encode using this scheme. In this context, an input distribution is a distribution over vectors of pairs (plaintext-value, tag), and we want to know if encoding plaintext values from this distribution and giving the encoded values to the adversary opens the scheme for attacks.

As for the meaning of "safe to encode", we hope to identify a "core computational task" for each candidate, that captures our intuition of what it means for an attacker to break that scheme, and then "safe" would mean that the attacker cannot perform that task. Jumping ahead, the core computation task for the GGH13 scheme is to find any basis for the ideal lattice corresponding to the plaintext space R/gR, but for GGH15 scheme we are not able to identify such a core computational task.[3] The meaning of "safe to encode" under GGH15 would therefore need to depend on what we are trying to hide in each particular input distribution.

3 Preliminaries

3.1 Lattices, ideals, and and trapdoors

Below we denote for an integer n the set $[n] = \{1, 2, \ldots, n\}$. Also for a modulus q and a real number x we denote $[n]_q$ as the reduction of x modulo q to the interval $[-q/2, +q/2)$. The latter notation

[3] For the CLT13 scheme the core task is to factor the composite modulus x_0.

extends naturally to vectors and matrices element-wise, and to elements of an extension ring/filed using their representations as vectors over $\mathbb{R}$.

Lattices. A lattice $L \subset \mathbb{R}^n$ is an additive discrete sub-group of $\mathbb{R}^n$. Every (nontrivial) lattice has bases: a basis for a full-rank lattice is a set of n linearly independent points $\vec{b}_1, \ldots, \vec{b}_n \in L$ such that $L = \{\sum_{i=1}^n z_i \vec{b}_i : z_i \in \mathbb{Z}\ \forall i\}$. For any vector $\vec{v} \in \mathbb{R}^n$, the L-coset of $\vec{v}$ is the set $\vec{v} + L = \{\vec{v} + \vec{u} : \vec{u} \in L\}$. For a modulus $q \in \mathbb{Z}$ and $u \in \mathbb{Z}_q^n$, the coset $\{\vec{x} \in \mathbb{Z}^m : \mathbf{A}\vec{x} = u \pmod q\}$ is denoted $\Lambda_{\vec{u}}^{\perp}(\mathbf{A})$ (with q implicit).

Rings and Ideal Lattices. Let R be the ring of algebraic integers in a degree-n number field, represented using some fixed basis (so every $x \in R$ is represented as a vector $\vec{x} \in \mathbb{Z}^n$). An ideal $I \subseteq R$ is then associated with a lattice $L_I = \{\vec{x} \in \mathbb{Z}^n : x \in I\}$ (wrt to the same fixed basis).

If I is a principle ideal, $I = gR$ for some $g \in R$, then the "circulant g-basis" of L_I consists of the vectors corresponding to the ring elements $\{g \cdot X^i \in R : i = 0, 1, \ldots, n-1\}$. If R has "nice enough geometry" (as all cyclotomics do using appropriate bases [LPR10, LPR13]) then all the vectors in the circulant g-basis have norm more or less the same as $\vec{g}$.

Gaussians. For a real $\sigma > 0$, define the (spherical) Gaussian function on $\mathbb{R}^n$ with parameter σ as $\rho_\sigma(\vec{x}) = \exp(-\pi\|\vec{x}\|^2/\sigma^2)$ for all $\vec{x} \in \mathbb{R}^n$. The *discrete Gaussian distribution* with parameter σ over a lattice (or a coset) L is $\forall\ \vec{x} \in L, D[L,\sigma](\vec{x}) = \rho_\sigma(\vec{x})/\rho_\sigma(L)$, where $\rho_\sigma(L)$ denotes $\sum_{\vec{x}\in L} \rho_\sigma(\vec{x})$. In other words, the probability $D[L,\sigma](\vec{x})$ is simply proportional to $\rho_\sigma(\vec{x})$, the denominator being a normalization factor.

Trapdoors and Samplers.

Theorem 3.1 *[MP12, Thm 5.1] There is an efficient randomized algorithm* $\mathsf{TrapGen}(1^n, 1^m, q)$ *that, given any integers $n \geqslant 1$, $q \geqslant 2$, and sufficiently large $m = \Omega(n \log q)$, outputs a parity-check matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a 'trapdoor' τ such that the distribution of $\mathbf{A}$ is negl(n)-far from uniform. Moreover, there are efficient algorithms* Invert *and* SampleD *that with overwhelming probability over all random choices, do the following:*

- *For $\vec{b}^t = \vec{s}^t\mathbf{A} + \vec{e}^t$, where $\vec{s} \in \mathbb{Z}_q^n$ is arbitrary and either $\|\vec{e}\| < O(q/\sqrt{n \log q})$ or $\vec{e} \leftarrow D[\mathbb{Z}^m, \alpha q]$ for $1/\alpha > \omega(\sqrt{n \log n \log q})$, the deterministic algorithm* $\mathsf{Invert}(\tau, \mathbf{A}, \vec{b})$ *outputs $\vec{s}$ and $\vec{e}$.*
- *For $\vec{u} \in \mathbb{Z}_q^n$ and large enough $s = \omega(\sqrt{n \log n \log q})$, the randomized algorithm* $\mathsf{SampleD}(\tau, \mathbf{A}, \vec{u}, s)$ *samples from a distribution within negl(n) statistical distance of $D[\Lambda_{\vec{u}}^{\perp}(\mathbf{A}), s]$.*

This theorem extends also to larger rings where we have $\mathbf{A} \in R^{n'\times m}$. We also extend SampleD to matrices in the natural way, denoting $\mathbf{C} \leftarrow \mathsf{SampleD}(\tau, \mathbf{A}, \mathbf{U}, s)$ the procedure that chooses the i'th column of $\mathbf{C}$ is by running $\mathsf{SampleD}(\tau, \mathbf{A}, \vec{u}_i, s)$ with $\vec{u}_i$ the i'th column of $\mathbf{U}$.

For any lattice L (not just $L = \Lambda_{\vec{u}}^{\perp}(\mathbf{A})$), one can use a "good basis" for L to sample from a discrete Gaussian on L (or its cosets), where the Gaussian parameter depends on the quality of the basis.

Theorem 3.2 *(GPV sampler:[GPV08, Thm 4.1]) There is a probabilistic polynomial-time algorithm that, given a basis $\mathbf{B}$ of an n-dimensional lattice $L = L(B)$, a parameter $s \geqslant |\tilde{B}| \cdot \omega(\sqrt{\log n})$, and a center $\vec{c} \in \mathbb{R}^n$, outputs a sample from a distribution that is statistically close to $D[L, s, \vec{c}]$.*

3.2 The GGH13 Scheme

Below is a brief description of the GGH13 scheme from [GGH13a]. That scheme works over the quotient ring $R_q = R/qR$ where $R = \mathbb{Z}[x]/\Phi_n(X)$ is the n-th cyclotomic polynomial ring and q is a large modulus, roughly $\log q \approx 2(\lambda + k \log n)$.

In (the "asymmetric" version of) this scheme, key-generation gets the security parameter λ, multi-linearity parameter κ, and the size of the tag universe M. The secret key consists of a small secret element $g \in R$ and M uniform random secrets $z_i \in R_q$ for all $i \in [M]$, and the public key contains the zero-test parameter $p_{zt} = [h \cdot \prod_{i \in [M]} / g]_q$ where h is a random somewhat small element, $\|h\| \approx \sqrt{q}$.

The plaintext space of this scheme is the quotient ring $R_g = R/gR$, and the tag space is the set $TAG_{asymGGH13}$ from Eqn. (2). An encoding of $m \in R_g$ relative to a subset $L \subseteq [M]$ and size bound ν is $u = [c/z^i]_q$ where $c \in m + gR$ is of size $\|c\| \leqslant \nu$. For $\nu > n \log n$, such a numerator c can be found using the "circulant g-basis" and Theorem 3.2. ≪Sergey: Check params≫

Encodings at the same levels can be added (and the encoded values get added modulo R_g), and encodings can be multiplied when their subsets are disjoint (and the size does not exceed $\sqrt{q}/2^\lambda$). The zero-testing procedure of GGH13 consists in multiplying a level-$[M]$ encoding $u = [c/z^\kappa]_q$ by p_{zt} and checking that the result $w = [u \cdot p_{zt}]_q = [h \cdot (c/g)]_q$ is small (say, $\|w\| \leqslant q/2^\lambda$). It is easy to see that the test always pass when u is an encoding of zero (i.e., c is a small element in the ideal[4]). To argue that encoding of non-zero fail the zero-test, Garg et al used the following lemma:

Lemma 3.3 *(Zero-test lemma: [GGH13a, Lemma 3]) Let R be a ring, $g \in R$ be an element such that the principle ideal gR is a prime ideal in R, let q be an integer such that g has an inverse in $R_q = R/qR$, let $x, e \in R$ be two ring elements, and denote $w = [x \cdot e/g]_q$. If both $\|x \cdot e\| < q/2$ and $\|g \cdot w\| < q/2$ then at least one of x, e must belong to the ideal gR.*

3.3 The GGH15 Scheme with "Safeguards"

Below we describe the GGH15 scheme from [GGH15] with the "safeguards" that are mentioned there.

Key generation. In addition to the security parameter λ, the key-generation routine gets a functional specifier, which in this case consist of a directed acyclic graph (DAG) denoted $G = (V, E)$, and a bound β on the size of the plaintext elements. (E.g., $\beta = 1$ if the application only needs to encode 0-1 matrices.) Below we denote by d the diameter of the graph, and assume for simplicity that G has a single source and sink and that it is transitively closed.

Depending on the above, we choose "LWE parameters" n, m, q such that $(\beta m)^d < q/2^\lambda < 2^{n/\lambda}$ (say) and $m = \Theta(n \log q)$ and $\sigma = \omega(\sqrt{n \log n \log q})$ as needed for Theorem 3.1. Then we proceed as follows:

- For every vertex $u \in V$, use the trapdoor sampling procedure from Theorem 3.1 to choose $(\mathbf{A}_u, \tau_u) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$, where $\mathbf{A}_u$ is nearly uniform in $\mathbb{Z}_q^{n \times m}$ and τ_u is the corresponding trapdoor.
- For every vertex $u \in V$, choose at random also an invertible matrix $\mathbf{P}_u \in \mathbb{Z}_q^{m \times m}$.

[4]The proof relies on the technical condition that the norm of $1/g$ in the field of fractions is small.

- For zero-testing, choose two random small vectors $\vec{v} \leftarrow D[\mathbb{Z},\sigma]^n$ and $\vec{w} \leftarrow D[\mathbb{Z},\sigma]^m$.

The public key consists of the parameters n, m, q, σ and the two m-vectors

$$\tilde{v} = [\vec{v} \times \mathbf{A}_s \times \mathbf{P}_s^{-1}]_q \text{ and } \tilde{w} = [\mathbf{P}_t \times \vec{w}]_q$$

(where s, t are the source and sink in the graph). The secret key consists of all the matrices $\mathbf{P}_u$ and $\mathbf{A}_u$'s and the corresponding trapdoors τ_u's.

Encoding. The encoding procedure gets the secret key, a plaintext matrix $S \in \mathbb{Z}^{n\times n}$ with entries bounded by β in absolute value, and a path $e = (u_1 \rightsquigarrow u_2)$ (which is used as the "tag" for the encoding).

It chooses a small error matrix $\mathbf{E} \leftarrow D[\mathbb{Z},\sigma]^{n\times m}$ and set $\mathbf{B} = [\mathbf{S} \times \mathbf{A}_{u_2} + \mathbf{E}]_q$. Then is samples $\mathbf{C} \leftarrow \mathsf{SampleD}(\tau_{u_1}, \mathbf{A}_{u_1}, B, \sigma)$ using the trapdoor, solving the SIS condition $\mathbf{A}_{u_1} \times \mathbf{C} = \mathbf{B}$.

The encoding of S relative to the path $u_1 \rightsquigarrow u_2$ is the matrix $\tilde{\mathbf{C}} = [\mathbf{P}_{u_1} \times \mathbf{C} \times \mathbf{P}_{u_2}^{-1}]_q$, coupled with the plaintext size bound β, and noise and encoding size bound $\nu = \sqrt{\lambda}$.

Operations. The invariant that we keep in the system is that an encoding $(\tilde{\mathbf{C}}, \beta, \nu)$ relative to the path $u_1 \rightsquigarrow u_2$ satisfies $\mathbf{A}_{u_1} \times \mathbf{C} = \mathbf{S} \times \mathbf{A}_{u_2} + E$, where $\|\mathbf{S}\|_\infty \leqslant \beta$, $\|\mathbf{E}\|_\infty \leqslant \nu$, and $\mathbf{C} = [\mathbf{P}_{u_1}^{-1} \times \tilde{\mathbf{C}} \times \mathbf{P}_{u_2}]_q$ satisfies $\|\mathbf{C}\|_\infty < \nu$

Addition of two encodings $(\tilde{\mathbf{C}}_1, \beta_1, \nu_1)$ and $(\tilde{\mathbf{C}}_2, \beta_2, \nu_2)$ relative to the same path is just the sum $([\tilde{\mathbf{C}}_1 + \tilde{\mathbf{C}}_2]_q, \beta_1 + \beta_2, \nu_1 + \nu_2)$, and this encoding is also relative to the same path. It is clear that this operation maintains the invariant from above.

Multiplication of the encodings $(\tilde{\mathbf{C}}_1, \beta_1, \nu_1)$ relative to a path $u_1 \rightsquigarrow u_2$ by the encodings $(\tilde{\mathbf{C}}_2, \beta_2, \nu_2)$ relative to $u_2 \rightsquigarrow u_3$ is an encoding relative to $u_1 \rightsquigarrow u_3$ which is computed as

$$(\tilde{C}^* = [\tilde{\mathbf{C}}_1 \times \tilde{\mathbf{C}}_2]_q, \quad \beta^* = \beta_1 \cdot \beta_2 \cdot n, \quad \nu^* = (\nu_1 m + \beta n) \cdot \nu_2).$$

To see that this preserves the invariant, denote $\mathbf{C}_1 = [\mathbf{P}_{u_1}^{-1} \times \tilde{\mathbf{C}}_1 \times \mathbf{P}_{u_2}]_q$, $\mathbf{C}_2 = [\mathbf{P}_{u_2}^{-1} \times \tilde{\mathbf{C}}_2 \times \mathbf{P}_{u_3}]_q$, and $\mathbf{C}^* = [\mathbf{P}_{u_1}^{-1} \times (\tilde{\mathbf{C}}_1 \times \tilde{\mathbf{C}}_2) \times \mathbf{P}_{u_3}]_q$ (so $\mathbf{C}^* = \mathbf{C}_1 \times \mathbf{C}_2 \pmod q$). Then we have

$$\begin{aligned}
\mathbf{A}_{u_1} \times \mathbf{C}^* &= (\mathbf{A}_{u_1} \times \mathbf{C}_1) \times \mathbf{C}_2 &&= (\mathbf{S}_1 \times \mathbf{A}_{u_2} + \mathbf{E}_1) \times \mathbf{C}_2 \\
&= \mathbf{S}_1 \times (\mathbf{A}_{u_2} \times \mathbf{C}_2) + \mathbf{E}_1 \times \mathbf{C}_2 &&= \mathbf{S}_1 \times (\mathbf{S}_2 \times \mathbf{A}_{u_3} + \mathbf{E}_2) + \mathbf{E}_1 \times \mathbf{C}_2 \\
&= \underbrace{\mathbf{S}_1 \times \mathbf{S}_2}_{\mathbf{S}^*} \times \mathbf{A}_{u_3} + \underbrace{\mathbf{S}_1 \times \mathbf{E}_2 + \mathbf{E}_1 \times \mathbf{C}_2}_{\mathbf{E}^*} \pmod q,
\end{aligned}$$

and the size bounds are easy to verify.

The zero-test procedure only works for encodings $(\tilde{\mathbf{C}}, \beta, \nu)$ relative to the source-to-sink path $s \rightsquigarrow t$ with noise bound $\nu \leqslant q/2^\lambda$. It works by computing the scalar $y = [\tilde{v} \times \tilde{\mathbf{C}} \times \tilde{w}]_q$ and checking if $|y| \leqslant m\sigma^2\lambda \cdot q/2^\lambda$.

By the invariant above, if $(\tilde{\mathbf{C}}, \beta, \nu \leqslant q/2^\lambda)$ is an encoding of the zero matrix relative to the path $s \rightsquigarrow t$, then $\mathbf{A}_s \times \mathbf{P}_s^{-1} \times \tilde{\mathbf{C}} \times \mathbf{P}_t = 0 \times \mathbf{A}_t + \mathbf{E} = \mathbf{E} \pmod q$ with $\|\mathbf{E}\|_\infty < q/2^\lambda$. Hence we have

$$y \;=\; \tilde{v} \times \tilde{\mathbf{C}} \times \tilde{w} \;=\; \vec{v} \times \mathbf{A}_s \times \mathbf{P}_s^{-1} \times \tilde{\mathbf{C}} \times \mathbf{P}_t \times \vec{w} \;=\; \vec{v} \times \mathbf{E} \times \vec{w} \pmod q$$

and correctness follows since $\|\mathbf{E}\|_\infty \leqslant q/2^\lambda$ and whp $\|\vec{v}\|_\infty, \|\vec{w}\|_\infty \leqslant \sigma\sqrt{\lambda}$.

If $(\tilde{\mathbf{C}}, \beta, \nu \leqslant q/2^\lambda)$ is an encoding of a non-zero matrix $\mathbf{S}$ relative to the path $s \rightsquigarrow t$, then similarly we have $y = \vec{v}(\mathbf{S} \times \mathbf{A}_t + \mathbf{E})\vec{w} \pmod q$. Since $\mathbf{S} \neq 0$ and $\mathbf{A}_t$ is nearly uniform over $\mathbb{Z}_q$ then $\vec{v}\mathbf{S}\mathbf{A}\vec{w}$ (and therefore also y) is nearly uniform in $\mathbb{Z}_q$, and so $\Pr_{\mathbf{A}_t}[|y| < m\sigma^2\lambda \cdot q/2^\lambda] < \text{poly}(\lambda)/2^\lambda$.

4 Variants of the GGH15 Scheme

4.1 Encoding the Entire Plaintext Space

The scheme from [GGH15] can only encode small matrices over $\mathbb{Z}_q$, i.e. onces whose entries are all small, namely with bound $\beta \ll q$. In some application it may be convenient to be able to encode the entire plaintext space, not just the small elements in it. To do that, we would use the technique from GGH13 [GGH13a] to make the plaintext space R/gR for a degree-n ring R and some small $g \in R$, which are chosen so that $R/gR \cong \mathbb{Z}_p$ for some prime number p.

Note that for a ring $R = \mathbb{Z}[X]/F(X)$, an element $x \in R$ can be encoded as a matrix $\mathbf{S}(x) \in \mathbb{Z}^{n\times n}$ (i.e., the multiply-by-x matrix), so that adding and multiplying these matrices over $\mathbb{Z}$ corresponds to addition and multiplication of elements in R (and the same holds for $R_q = R/qR = \mathbb{Z}_q[X]/F(X)$ over $\mathbb{Z}_q$).[5]

Moreover if R has "nice enough geometry" (as do all the cyclotomic rings using appropriate bases [LPR10, LPR13]) then the size of the multiply-by-x matrix is roughly the same as the size of the representation of the element x. In that case, given a small $g \in R$ and an arbitrary $x \in R$ we can find a small $x' \in R$ in the same g-coset as x, i.e., $x' + gR = x + gR$, and then also the matrix $\mathbf{S}(x')$ that represents x' will be small. Hence for any g-coset $x \in R/gR$ we can find a small matrix $\mathbf{S}$ representing this coset, and can use the GGH15 scheme from above to encode it.

The main challenge is to modify the zero-test parameter, so that we can identify encoding of matrices $\mathbf{S}(x)$ which are not the zero matrix but belong to the zero coset, i.e., they represent $x \in gR$. To do that, we only need to change the way we choose the matrix $\mathbf{A}_t$ which is associated with the sink node t in the graph. Instead of choosing it at random, we will choose a small matrix $\mathbf{E}_t \leftarrow D[\mathbb{Z}, \sigma]^{n\times m}$ and set $\mathbf{A}_t = [\mathbf{G}^{-1} \times \mathbf{E}_t]_q$, where $\mathbf{G}$ is the matrix representation of g and $\mathbf{G}^{-1}$ is its inverse modulo q. We note that $\mathbf{A}_t$ is the only matrix for which *we never need a trapdoor* in the construction from above. This way, if we zero-test an encoding of a matrix $\mathbf{S}(rg) = \mathbf{S}(r) \times G$, then we have

$$\mathbf{S} \times \mathbf{A}_t + \mathbf{E} \;=\; (\mathbf{S}(r) \times G) \times (G^{-1} \times \mathbf{E}_t) + \mathbf{E} \;=\; \mathbf{S}(r) \times \mathbf{E}_t + \mathbf{E}.$$

Assuming that $\mathbf{S}(r)$ is small (which we can ensure), then $\mathbf{S} \times \mathbf{A}_t + \mathbf{E}$ will also be small, and we get our zero-test procedure.

Key generation. The key-generation routine gets a the security parameter and a DAG (and a bound β) as above. As before, we assume that G has a single source and sink and it transitively closed, and denote the diameter by d and the source and sink by s, t, respectively.

Depending on the above, we choose the LWE parameters n, m, q such that $(\beta n)^d < q < 2^{n/\lambda}$ (say) and $m = \Theta(n \log q)$ and $\sigma = \omega(\sqrt{n \log n \log q})$ as needed for Theorem 3.1. We also choose a degree-n ring with "good geometry" (e.g., a cyclotomic ring), which we denote by R. Then we proceed as follows:

- Repeatedly choose elements $g \leftarrow D[\mathbb{Z}^n, \sigma]$ until you find one such that R/gR has prime order, g is invertible in R_q, and moreover in the field of fractions we have $\|1/g\| < \tau = \mathrm{poly}(\lambda)$. This is the same procedure as in GGH13 [GGH13a]. Let $\mathbf{G}$ be the multiply-by-g integer matrix.

[5] This means in particular that these matrices commute under multiplications, since R itself is commutative.

- For every vertex $u \in V$ except the sink t, use the trapdoor sampling procedure from Theorem 3.1 to choose $(\mathbf{A}_u, \tau_u) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$, where $\mathbf{A}_u$ is nearly uniform in $\mathbb{Z}_q^{n\times m}$ and τ_u is the corresponding trapdoor. The only exception is for the sink vertex t where instead we choose $\mathbf{E}_t \leftarrow D[\mathbb{Z},\sigma]^{n\times m}$ and set $\mathbf{A}_t = [G^{-1} \times \mathbf{E}_t]_q$.
- For every vertex $u \in V$, choose at random also an invertible matrix $\mathbf{P}_u \in \mathbb{Z}_q^{m\times m}$.
- For zero-testing, choose two random small vectors $\vec{v} \leftarrow D[\mathbb{Z},\sigma]^n$ and $\vec{w} \leftarrow D[\mathbb{Z},\sigma]^m$.

The public key consists of the parameters n, m, q, σ, and the two m-vectors

$$\tilde{v} = [\vec{v} \times \mathbf{A}_s \times \mathbf{P}_s^{-1}]_q \text{ and } \tilde{w} = [\mathbf{P}_t \times \vec{w}]_q$$

(where s, t are the source and sink in the graph). The secret key consists of the element $g \in R$, all the matrices $\mathbf{P}_u$ and $\mathbf{A}_u$'s and the corresponding trapdoors τ_u's.

Encoding. The encoding procedure gets the secret key, a scalar $x \in \mathbb{Z}_p$, and a path $e = (u_1 \rightsquigarrow u_2)$ (which is used as the "tag" for the encoding). It begins by using g to find a short representative x' of the coset $x + gR$, and let $\mathbf{S}(x')$ be the integer matrix representation of that element, and note that $\|\mathbf{S}(x')\|_\infty \leqslant \sigma\sqrt{\lambda}$.

The procedure then proceeds to encode $\mathbf{S}(x')$ as above. It chooses a small error matrix $\mathbf{E} \leftarrow D[\mathbb{Z},\sigma]^{n\times m}$ and set $\mathbf{B} = [\mathbf{S}(x') \times \mathbf{A}_{u_2} + \mathbf{E}]_q$. Then is samples $\mathbf{C} \leftarrow \mathsf{SampleD}(\tau_{u_1}, \mathbf{A}_{u_1}, B, \sigma)$ using the trapdoor, solving the SIS condition $\mathbf{A}_{u_1} \times \mathbf{C} = \mathbf{B}$.

The encoding of S relative to the path $u_1 \rightsquigarrow u_2$ is the matrix $\tilde{\mathbf{C}} = [\mathbf{P}_{u_1} \times \mathbf{C} \times \mathbf{P}_{u_2}^{-1}]_q$, coupled with the size bound $\beta = \sigma\sqrt{\lambda}$ and noise bound $\nu = \sigma\sqrt{\lambda}$.

Operations. The invariant that we keep in the system is as before, and so are the addition and multiplication operations.

The zero-test procedure is essentially the same as before, except it has a slightly different threshold. Specifically, given the encodings $(\tilde{\mathbf{C}}, \beta, \nu)$ relative to the source-to-sink path $s \rightsquigarrow t$ with size bound $\beta < q/2^\lambda$ and noise bound $\nu \leqslant q/2^\lambda$, it computes the scalar $y = [\tilde{v} \times \tilde{\mathbf{C}} \times \tilde{w}]_q$ and checks if $|y| \leqslant (1 + n\tau\sigma)m\sigma^2\lambda q/2^\lambda$.

The correctness proof is a little different, and it essentially follows the lines of the GGH13 proof from [GGH13a]. If $(\tilde{\mathbf{C}}, \beta \leqslant q/2^\lambda, \nu \leqslant q/2^\lambda)$ is an encoding of a matrix representing $\mathbf{S}(x)$ then by our invariant we have $\mathbf{A}_s \times (\mathbf{P}_s^{-1} \times \tilde{\mathbf{C}} \times \mathbf{P}_t) = \mathbf{S}(x) \times (\mathbf{G}^{-1} \times \mathbf{E}_t) + E \pmod q$.

If $x = x' \cdot g \in R$ then $\mathbf{S}(x) = \mathbf{S}(x') \times \mathbf{G}$ and so $\mathbf{S}(x) \times (\mathbf{G}^{-1} \times \mathbf{E}_t) = \mathbf{S}(x') \times \mathbf{E}_t \pmod q$. Moreover if $\|x\| \leqslant \beta$ then $\|x'\| \leqslant \beta \cdot \mathrm{poly}(\lambda)\tau$ and therefore also $\|\mathbf{S}(x')\|_\infty \leqslant\approx \beta \cdot \tau$. Hence we have

$$y = \tilde{v} \times \tilde{C} \times \tilde{w} = \vec{v} \times (\mathbf{S}(x') \times \mathbf{E}_t + \mathbf{E}) \times \vec{w}$$

and therefore $|y| \leqslant m \cdot \|\vec{v}\|_\infty \cdot \|\vec{w}\|_\infty (\|\mathbf{S}(x') \times \mathbf{E}_t\|_\infty + \|\mathbf{E}\|_\infty) \leqslant (1 + n\tau\sigma)m\sigma^2\lambda q/2^\lambda$.

On the other hand if $x \notin gR$ then consider one columns $\vec{e}_i$ in $\mathbf{E}_t$ and let $e_i \in R$ be the ring element that it represents. Then $[\mathbf{S}(x) \times G^{-1} \times \vec{e}_i]_q$ represents the element $w_i = [x \cdot e_i/g]_q \in R$. With high probability over the choice of $\mathbf{E}_t$ we have $e_i \notin gR$, and we know that $\|x \cdot e_i\| < \beta \cdot \sigma \cdot m \ll q/2$. By Lemma 3.3, this implies that for $x \notin gR$ we necessarily have $\|w_i\| \geqslant q/2$. This means that every column of the matrix $[\mathbf{S}(x) \times G^{-1} \times \mathbf{E}_t]_q$ has norm larger than $q/2$ whp, and thus whp over $\vec{v}, \vec{w}$ we have $|[\vec{v} \times \mathbf{S}(x) \times G^{-1} \times \mathbf{E}_t \times \vec{w}]_q| \approx q$, and therefore also $|y| \approx q$. «Shai: Make more precise»

Encoding matrices. The variant above uses native GGH15 encoding of matrices in order to encode scalars $x \in R_g = R/gR$, but it is easy to modify it to instead encode matrices over R_g. Given a matrix $M \in R_g^{a \times a}$, we just represent each element in M by an $n \times n$ matrix as above (hence obtaining a $na \times na$ matrix) and then use native GGH15 encodings to encode these higher-dimension matrices.

Hiding the ring. A curious property of the scheme above is that the operations are all done over $\mathbb{Z}_q$, and in particular they do not depend on the structure of the ring R. Hence it may be possible to use this scheme while hiding the ring R itself from the adversary. Note that although R is not explicitly used while carrying out the operations, we still rely on it to have "nice geometry" so as to ensure the the noise does not grow to fast. It is not clear if there are very many different rings that have such "nice geometry", and in particular it is not clear if hiding the ring is possible.

4.2 Introducing Subsets

The level structure of the GGH13 scheme (especially in its "asymmetric" setting) was quite useful in devising many schemes (for example it was crucial in constructing "straddling sets" that make many obfuscation constructions possible). It is therefore desirable to replicate this structure also in the context of the graph-induced constructions.

Adding subsets to the constructions above is fairly straightforward, simply by adapting the denominators from the GGH13 construction. We first need to switch to working over a larger ring R (which must be explicit in the construction), then choose many denominators $z_i \in R_q$ uniformly at random, divide an encoding $\tilde{C}$ relative to level $L \subseteq [\kappa]$ by the product $\prod_{i \in L} z_i$, and multiply the zero-test parameter by $\prod_{i=1}^{\kappa} z_i$ (modulo q). Below we describe a variant where only small elements can be encoded, but of course it is possible to incorporate also the modifications from above to be able to encode the entire plaintext space.

Key generation. In addition to the security parameter λ, the key-generation routine gets a DAG $G = (V, E)$, the "top level" κ and a bound β on the size of the plaintext elements. We assume that G has a single source and sink (denoted s, t, respectively), and is transitively closed, and denote by d its diameter.

Depending on the above, we choose RLWE parameters n, m, q and a degree-n ring R with "good geometry" (e.g., a cyclotomic ring), such that $(\beta n)^d < q < 2^{n/\lambda}$ (say) and $m = \Theta(\log q)$ and $\sigma = \omega(\sqrt{n \log n \log q})$ as needed for Theorem 3.1. Then we proceed as follows:

- For every vertex $u \in V$, use the trapdoor sampling procedure from Theorem 3.1 to choose $(\mathbf{A}_u, \tau_u) \leftarrow \mathsf{TrapGen}(R, 1^m, q)$, where $\mathbf{A}_u$ is nearly uniform in $\mathbb{R}_q^{1 \times m}$ and τ_u is the corresponding trapdoor.
- For every vertex $u \in V$, choose at random also an invertible matrix $\mathbf{P}_u \in \mathbb{R}_q^{m \times m}$.
- For every $i \in [\kappa]$, choose a uniformly random $z_i \in R_q$.
- For zero-testing, choose a random small vector $\vec{w} \leftarrow D[R, \sigma]^m$.

The public key consists of the parameters n, m, q, σ and the two m-vectors

$$\tilde{v} = [\mathbf{A}_s \times \mathbf{P}_s^{-1} \cdot \prod_{i=1}^{\kappa} z_i]_q \text{ and } \tilde{w} = [\mathbf{P}_t \times \vec{w}]_q$$

(where s, t are the source and sink in the graph). The secret key consists of all the matrices $\mathbf{P}_u$ and $\mathbf{A}_u$'s and the corresponding trapdoors τ_u's.

Encoding. The encoding procedure gets the secret key, a plaintext element $s \in R$ with $\|s\| \leqslant \beta$, a path $e = (u_1 \rightsquigarrow u_2)$ and a level $L \subseteq [\kappa]$ (which are used as the "tag" for the encoding).

It chooses a small error matrix $\mathbf{E} \leftarrow D[R, \sigma]^{1 \times m}$ and set $\mathbf{B} = [\mathbf{S} \times \mathbf{A}_{u_2} + \mathbf{E}]_q$. Then is samples $\mathbf{C} \leftarrow \mathsf{SampleD}(\tau_{u_1}, \mathbf{A}_{u_1}, B, \sigma)$ using the trapdoor, solving the RSIS condition $\mathbf{A}_{u_1} \times \mathbf{C} = \mathbf{B}$.

The encoding of s relative to the path $u_1 \rightsquigarrow u_2$ and level L is the matrix $\tilde{\mathbf{C}} = [\mathbf{P}_{u_1} \times \mathbf{C} \times \mathbf{P}_{u_2}^{-1} / \prod_{i \in L} z_i]_q$, coupled with the size bound β and noise bound $\nu = \sigma\sqrt{\lambda}$.

Operations. The invariant and operations are similar to those from Section 3.3, except that everything is over the ring R (or R_q). We note that when zero-testing an encoding relative to a source-to-sink edge and level $[\kappa]$, all the z_i's cancel out and we are left with the original zero-test (except over R). Correctness is exactly as before.

5 Zeroizing Attacks on GGH15

Gentry, Gorbunov and Halevi described in [GGH15] some attacks that uses encoding of zeros as "approximate trapdoors" for the $\mathbf{A}_u$ matrices, but the "safeguards" provided by the $\mathbf{P}_u$'s (and the fact that the schemes that we describe above do not publish the $\mathbf{A}_u$ matrices) seem to thwart these attacks. Unfortunately, it does not prevent zeroizing attacks similar to those of Cheon et al. [CHL+15] and Coron et al. [CGH+15], as we now describe.

5.1 Bird-eye View of the Attack

On a high-level, the attacks of Cheon et al. [CHL+15] and Coron et al. [CGH+15] consist of setting up a 3-linear set of equations of the form $y_{ijk} = \vec{v}_i \times M_j \times \vec{w}_k$, where $\vec{v}_i, M_j, \vec{w}_k$ are related to encoded values that the attacker knows. In these attacks the M_j's are non-singular square matrices that depend on some secrets of the scheme, and the goal of the attacker is to recover these secrets. Fixing one such M_j and using many $\vec{v}_i$'s and many $\vec{w}_k$'s we get a matrix

$$Y_j = [y_{ijk}]_{i,k} = V \times M_j \times W$$

where the rows of V are the $\vec{v}_i$'s and the columns of W are the $\vec{w}_k$'s, and V, W are non-singular square matrices (whp). Setting $Y := Y_1 \times Y_2^{-1}$ we have $Y = V \times (M_1 \times M_2^{-1}) \times W^{-1}$, hence Y share the same eigenvalues (and more generally the same characteristic polynomial) as $(M_1 \times M_2^{-1})$. Analyzing the eigenvalues of Y therefore yields information on M_1 and M_2, which is used to break the scheme.

In the attacks described in [CHL+15, CGH+15], it was important that all these relations hold over the integers (without mod-q reduction), since the eigenvalues of $M_1 \times M_2^{-1}$ were then used in GCD computations over the integers. Obtaining these relations over the integers is the technical reason why these attacks need encoding of zeros, and it makes these attacks hard to mount in cases where the zero-encodings are very constrained (as in most obfuscation schemes). For the GGH15 scheme we also need the relations to hold over the integers, but for a different reason. Here the issue is that the modular relations that we get are inherently non-full-rank, hence the matrices Y_j that we obtain in the attack cannot be inverted. But when we have encoding of zeros then these relations hold also over the integers, where they have full rank.

5.2 More Details

Consider the three-edge graph $u \to u' \to u'' \to t$, with encodings $\tilde{C}_i$ of plaintext matrices $\mathbf{S}_i$ on the edge $u \to u'$, encodings $\tilde{C'}_j$ of plaintext matrices $\mathbf{S}'_j$ on the edge $u' \to u''$, and encodings $\tilde{C''}_k$ of plaintext matrices $\mathbf{S}''_k$ on the edge $u'' \to t$, and with zero-test vectors $\tilde{v}, \tilde{w}$. Recall that the encodings are set as

$$\tilde{C}_i = \mathbf{P}_u \times \mathbf{C}_i \times \mathbf{P}_{u'}^{-1},\ \tilde{C'}_j = \mathbf{P}_{u'} \times \mathbf{C}'_j \times \mathbf{P}_{u''}^{-1},\ \tilde{C''}_k = \mathbf{P}_{u''} \times \mathbf{C}''_k \times \mathbf{P}_t^{-1},$$

and we have $\tilde{w} = \vec{w} \times \mathbf{A}_u \times \mathbf{P}_u^{-1}$ and $\tilde{w} = \mathbf{P}_t \times \vec{w}$. Denote by $\mathbf{E}_i, \mathbf{E}'_j, \mathbf{E}''_k$ the error matrices that were used to generate the intermediate $\mathbf{C}_i, \mathbf{C}'_j, \mathbf{C}''_k$, respectively. Then for all i, j, k we have

$$\begin{aligned} y_{i,j,k} = \tilde{v} \times \tilde{C}_i \times \tilde{C}'_j \times \tilde{C}''_k \times \tilde{w} &= \vec{v}\big(\mathbf{S}_i\mathbf{S}'_j\mathbf{S}''_k\mathbf{A}_t + \mathbf{S}_i\mathbf{S}'_j\mathbf{E}''_k + \mathbf{S}_i\mathbf{E}'_j\mathbf{C}''_k + \mathbf{E}_i\mathbf{C}'_j\mathbf{C}''_k\big)\vec{w} \\ &= \underbrace{\vec{v} \times (\mathbf{S}_i \mid \mathbf{E}_i)}_{\vec{v}_i} \times \underbrace{\begin{pmatrix} \mathbf{S}'_j & \mathbf{E}'_j \\ 0 & \mathbf{C}'_j \end{pmatrix}}_{M_j} \times \underbrace{\begin{pmatrix} \mathbf{S}''_k\mathbf{A}_t + \mathbf{E}''_k \\ \mathbf{C}''_k \end{pmatrix} \times \vec{w}}_{\vec{w}_k} \end{aligned} \tag{3}$$

Assume that we are given many $\mathbf{C}_i$'s that encode random $\mathbf{S}_i$'s, two $\mathbf{C}'_j$'s that encode $\mathbf{S}'_j$'s ($j = 1, 2$), and many $\mathbf{C}''_k$'s that encode (arbitrary) $\mathbf{S}''_k$'s. For $j = 1, 2$ we can then construct the matrices

$$Y_j = [y_{i,j,k}]_{i,k} = V \times M_j \times W \pmod q \tag{4}$$

where the rows of V are the $\vec{v}_j$'s and the columns of W are the $\vec{w}_k$'s. We would like to compute $Y = Y_1 Y_2^{-1} \bmod q$, but we note that the matrix W is inherently singular modulo q (and hence so are the Y_j's), since we have $\mathbf{A}_{u''}\mathbf{C}''_k = \mathbf{S}''_k\mathbf{A}_t + \mathbf{E}''_k \pmod q$ for every k.

However, when $\mathbf{S}''_k = 0$ for all k then the same relations from Eqn. (3) holds not only modulo q but also over the integers (since all the quantities involved are much smaller than q). With high probability, the matrix W has full rank over the integers, so we can use it in attacks on the scheme.

Remarks. We note that this attack can be adapted easily to the schemes from Sections 4.1 and 4.2. We also note that we can get relations over the integers also in other cases, but not all of them seem useful for an attack. For example if we have $\mathbf{S}'_j = 0$ or $\mathbf{S}_i = 0$ then we get the relations

$$\begin{aligned} \mathbf{S}_i = 0 \quad &: \quad y_{ijk} = \vec{v} \times \mathbf{E}_i \times \mathbf{C}'_j \times \mathbf{C}''_k \times \vec{w} \\ \mathbf{S}'_j = 0 \quad &: \quad y_{ijk} = \vec{v} \times (\mathbf{S}_i \mid \mathbf{E}_i) \times \begin{pmatrix} \mathbf{E}'_j \\ \mathbf{C}'_j \end{pmatrix} \times \mathbf{C}''_k \times \vec{w}. \end{aligned}$$

But these relations do not seem to be useful, since the middle matrices in these relations do not reveal the "secret quantities" that we care about.

6 Graph Constraints for GGH13

In many applications of graded encoding schemes, the encodings are multiplied in a fixed order which is known a-priory, and so it makes sense to add to the scheme constraints that would only allow to multiply encoding in the given order. Namely we want to add to the "functionality specifier" also a graph structure similar to GGH15, so that each encoding is tagged by a path in

the graph (either in addition to or instead of its level), and to only allow adding encoding relative to the same path and multiplying encoding relative to subsequent paths.

To that end, we can use transformation matrices P_v for vertexes $v \in V$ as in the "safeguard" of the GGH15 scheme described above. If we are encoding matrices over R_g (rather than scalars) then these matrices P_v can be over the ring R_q,[6] and if we are only encoding scalars in R_g then we can identify a scalar $x \in R_q$ by its multiply-by-x matrix over $\mathbb{Z}_q$ and use integer transformation matrices $R_v \in \mathbb{Z}^{n \times n}$. Below we describe in some detail the variant that encodes scalars in R_g and has only graph-based constraints but no levels. The other variants are similarly defined.

Key generation. Key generation takes the security parameter λ and a DAG (V, E), which is transitively closed, has a single source s and a single sink t and diameter d. It computes the parameters n, q that satisfy $n^d \cdot 2^\lambda < q < 2^{n/\lambda}$, and consider the rings $R = \mathbb{Z}[X]/\Phi_n(X)$ and $R_q = R/qR$.

Next it chooses a random small $g \in R$ (say by drawing its representation vector $\vec{g}$ from $D[\mathbb{Z}^n, \sqrt{n \log n}]$), thus setting the plaintext space as $R_g = R/gR$.[7] It also chooses two small vector $\vec{v}, \vec{w}$ from the same distribution and for each vertex $v \in V$ it chooses a random invertible matrix $P_v \in \mathbb{Z}_q^{n \times n}$. The secret key includes the P_v's and g and the public key consists of the two vectors $\tilde{v} = [\vec{v} \times P_s]_q$ and $\tilde{w} = [P_t^{-1} \times G^{-1} \times \vec{w}]_q$. where $G^{-1} \in \mathbb{Z}_q^{n \times n}$ is the divide-by-g matrix.

Encoding. To encode an element $\alpha \in R_g$ relative to the path $u \rightsquigarrow v$ in the graph, we use the circular g-basis and Theorem 3.2 to sample a small element $c \in \alpha + gR$, then compute the multiply-by-c matrix, $C \in \mathbb{Z}_q^{n \times n}$, and output the encoding $\tilde{C} = [P_u^{-1} \times C \times P_v]_q$.

Operations. Addition and multiplication are just matrix addition and multiplication over $\mathbb{Z}_q$. To test for zero we compute $y = [\tilde{v} \times \tilde{C} \times \tilde{w}]_q$ and check that $|y| \ll q$. If $\tilde{C}$ is an encoding relative to the source-to-sin path $s \rightsquigarrow t$ then $\tilde{C} = [P_s^{-1} \times C \times P_t]_q$ where C is the multiply-by-c matrix for some small element $c \in R_q$. The matrices P_s, P_t cancel out and we get

$$\tilde{v} \times \tilde{C} \times \tilde{w} = \vec{v} \times C \times G^{-1} \times \vec{w} \pmod{q}.$$

Recalling that $\vec{u} = [C \times G^{-1} \times \vec{w}]_q$ is the representation of the ring element $c \cdot g^{-1} \cdot w \in R_q$, the zero-test lemma tells us that (assuming w, g are co-prime) $\vec{u}$ is small if and only if c belongs to the ideal gR. Zero-test correctness now follows since $\vec{v}$ is small and random and $y = [\langle \vec{v}, \vec{u} \rangle]_q$.

6.1 Remarks

The size of $\vec{v}, \vec{w}$. In the original GGH13 scheme from [GGH13a] the element h was size $\sqrt{q}$ to ensure that squaring the zero-test parameter does not yield a working zero-test for level 2κ. This attack does not seem relevant here so the vectors $\vec{v}, \vec{w}$ can be as small as $\vec{g}$.

Hiding the ring. As in the scheme from Section 4.1, here too the structure of the ring R need not be made explicit, so we could perhaps hope to hide it (but again it is not clear how realistic this hope is).

[6] A similar variant was mentioned in [CGH+15, Sec. 4]

[7] Technically we also need to ensure that $1/g$ in the field of fractions is small, and may want to ensure that gR is a prime ideal, but we ignore these details here.

Using both paths and subsets. To use both subsets and paths as the encoding tags, we compute an encoding of $\alpha \in R_g$ relative to the path $u \rightsquigarrow v$ and subset L by choosing a small element $c \in \alpha + gR$ as before, then outputting $\tilde{C} = [P_u^{-1} \times C' \times P_v]_q$ where C' is the matrix for multiply-by-$(c/\prod_{i \in L} z_i)$ in R_q. The zero-test parameters are modified accordingly by setting $\tilde{w} = [P_t^{-1} \times M \times \vec{w}]_q$ where M is the matrix for multiply-by-$(g^{-1} \cdot \prod_{i \in [M]} z_i)$ in R_q.

7 Multi-Partite Key-Agreement from GGH13

The GGH13 variant with graph constraints above can be used to implement the key-agreement protocol of Gentry et al. from [GGH15] as-is, and we do not know of any attack on this key-agreement protocol. Below, however, we describe a somewhat different key-agreement protocol using the standard "asymmetric" GGH13 (with the GGHlite optimizations of Langlois, Stehlé and Steinfeld [LSS14]). We also present a relatively simple related hardness assumption as target for cryptanalysis. This assumption seems to capture the security of the key-agreement protocol, even though it is technically neither sufficient nor necessary.

Setup. For a k-party key-agreement protocol, our tags/levels are subsets of a universe of size $M = k(k-1)$, for convenience below we describe the universe as pairs $(i,j) \in [k] \times [k-1]$. After running the key-generation procedure of GGH13 with these tags, the key-agreement protocol chooses $2k$ random elements of the plaintext space, and encode each of them wrt k different levels, as follows:

For $j = 1, 2, \ldots, k$:

1. Choose random $a_j, b_j \in R_g$, subject to the constraint that they are co-prime;
2. If $j < k$ then for $i = 1, 2, \ldots, k$ let $A_{i,j}, B_{i,j}$ be an encoding of a_j, b_j respectively, both relative to the singleton subset $\{(i,j)\}$;
3. If $j = k$ then for $i = 1, 2, \ldots, k$ let $A_{i,j}, B_{i,j}$ be an encoding of a_i, b_i respectively, both relative to the subset $\{(i',j') \in [k] \times [k-1] : i' \neq i\}$;

Note that with this encoding, fixing any i and multiplying $\prod_{j=1}^{k} A_{i,j}$ yields a top-level encoding relative to the entire universe $[k] \times [k-1]$, and similarly for multiplying the $B_{i,j}$'s or any mix-and-match of A's and B's. The public parameters of the key-agreement protocol include the zero-test parameter and all the encoded values $A_{i,j}, B_{i,j}$, $i, j \in [k]$.

Protocol. In the protocol, each party j chooses two random scalars $\alpha_j, \beta_j \in R$ from a Gaussian distribution as in [LSS14], then for $i = 1, 2, \ldots, k$ set $C_{i,j} = [\alpha_j \cdot A_{i,j} + \beta_j \cdot B_{i,j}]_q$. Party j then broadcasts all the $C_{i,j}$'s *except* $C_{j,j}$.

Finally, party j collects all the broadcast messages $C_{j,j'}$ for $j' \neq j$, together with its own $C_{j,j}$ (that wasn't broadcast), and computes $K_j = [\prod_{j'=1}^{k} C_{j,j'}]_q$. It is easy to see that each K_j is a top-level encoding of the scalar $\prod_{j=1}^{k} (\alpha_j \cdot a_j + \beta_j \cdot b_j)$. Each party j then uses the extraction procedure of GGH13 to get the shared key (i.e. multiplying by p_{zt} and taking the top bits, possibly followed by hashing).

Security. Formally, security of the above protocol means that the derived key should be pseudo-random given the public parameters and all the broadcast information. However intuitively it seems that the heart of the problem facing the cryptanalyst is to identify the plaintext space R_g, and that if R_g can be identified then it should be possible to use it in an attack. We therefore put forward as target for cryptanalysis the computational problem of finding (any basis of) the plaintext space R_g given the public parameters. To aid readability, below we recap this computational problem.

7.1 A concrete target for cryptanalysis of GGH13

Parameters and setup. We are given the security parameter λ and the multi-linearity parameter κ, and for concreteness we believe that it is sufficient to consider specifically $\lambda \in \{80, 128\}$ and $\kappa = 3$.

From λ, κ we compute the parameters n, q such that n is a power of two and $q > n^{4\kappa} \cdot 2^{\lambda}$ and $n > \log(q)(\lambda + 110)/7.2$ (the last equality is taken from [GHS12, Appendix C]). For example with $\kappa = 3, \lambda = 80$ we can use $n = 2^{13}$ and $\log q \approx 236$, while with $\kappa = 3, \lambda = 128$ we can use $n = 2^{14}$ and $\log q \approx 296$. These define for us the rings $R = \mathbb{Z}[X]/(X^n + 1)$ and $R_q = R/qR$.

Key generation. We choose a vector $\vec{g} \leftarrow D[\mathbb{Z}^n, \sqrt{n \log n}]$ and $\vec{h} \leftarrow D[\mathbb{Z}^n, \sqrt{q}]$ and view them as representing two elements $g, h \in R$. We also choose $M = \kappa(\kappa - 1)$ uniformly random elements $z_{i,j} \in R_q$, $i = 1, \ldots, \kappa$ and $j = 1, \ldots, \kappa - 1$. We compute the element

$$p_{zt} = \left[hg^{-1} \cdot \prod_{i,j \in [\kappa] \times [\kappa-1]} z_{i,j}\right]_q.$$

We extend the set of $z_{i,j}$'s by defining $z_{i,\kappa} = [\prod_{i' \neq i, j} z_{i',j}]_q$ for all i, and note that for all i the product $\prod_{j=1}^{\kappa} z_{i,j}$ is the same, specifically it equals the product of all the M $z_{i,j}$'s that were chosen at random.

Encoding. For $i = 1, 2, \ldots \kappa$ we choose at random $\vec{a}_i, \vec{b}_i \leftarrow D[\mathbb{Z}^n, \sqrt{n \log n}]$ and view them as representing two elements $a_i, b_i \in R$, and we re-sample until a_i and b_i are co-prime. Then for every $i, j \in [\kappa]$ we sample $a_{i,j} \leftarrow D[a_i + gR, n \log n]$ and $a_{i,j} \leftarrow D[b_i + gR, n \log n]$ and compute $A_{i,j} = [a_{i,j}/z_{i,j}]_q$ and $B_{i,j} = [b_{i,j}/z_{i,j}]_q$.

Computational task. The attacker is given p_{zt} and all the $A_{i,j}, B_{i,j}$, $i, j \in [\kappa]$, and its goal is to find any basis for the ideal lattice gR.

References

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

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 221–238. Springer, 2014.

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015*, volume 9215 of *Lecture Notes in Computer Science*, pages 247–266. Springer, 2015. Long version in `http://eprint.iacr.org/2015/596`.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 3–12. Springer, 2015. `http://eprint.iacr.org/2014/906`.

[CL15] Jung Hee Cheon and Changmin Lee. Cryptanalysis of the multilinear map on the ideal lattices. Cryptology ePrint Archive, Report 2015/461, 2015. `http://eprint.iacr.org/`.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 476–493. Springer, 2013.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In *CRYPTO 2015*, 2015. To appear.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Advances in Cryptology - EUROCRYPT'13*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 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 *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, Berkeley, CA, USA*, pages 40–49. IEEE Computer Society, 2013.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 498–527. Springer, 2015. `https://eprint.iacr.org/2014/645`.

[GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure functional encryption without obfuscation. Cryptology ePrint Archive, Report 2014/666, 2014. `http://eprint.iacr.org/`.

[GHS12] Craig Gentry, Shai Halevi, and Nigel Smart. Homomorphic evaluation of the AES circuit. In *"Advances in Cryptology - CRYPTO 2012"*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012. Full version at `http://eprint.iacr.org/2012/099`.

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

[Hal15] Shai Halevi. The state of cryptographic multilinear maps. Invited Talk of CRYPTO 2015, 2015.

[HJ15] Yupu Hu and Huiwen Jia. Cryptanalysis of ggh map. Cryptology ePrint Archive, Report 2015/301, 2015. `http://eprint.iacr.org/`.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT'10*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT'13*, volume 7881 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2013.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. Gghlite: More efficient multilinear maps from ideal lattices. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT*, volume 8441 of *Lecture Notes in Computer Science*, pages 239–256. Springer, 2014.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT'12*, pages 700–718, 2012.