Candidate Multilinear Maps from Ideal Lattices

Sanjam Garg* UCLA
Craig Gentry IBM
Shai Halevi IBM

March 17, 2013

Abstract

We describe plausible lattice-based constructions with properties that approximate the sought-after multilinear maps in hard-discrete-logarithm groups, and show an example application of such multi-linear maps that can be realized using our approximation. The security of our constructions relies on seemingly hard problems in ideal lattices, which can be viewed as extensions of the assumed hardness of the NTRU function.

This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D11PC20202. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

*Research conducted while at the IBM Research, T.J. Watson funded by NSF Grant No.1017660.

Contents

1 Introduction

Bilinear maps are extremely useful tools in cryptography. After being used to construct non-interactive key agreement [SOK00], tripartite Diffie-Hellman [Jou00], and identity-based encryption [BF01], the applications of bilinear maps have become too numerous to name. Boneh and Silverberg [BS03] argued that multilinear maps would have even more interesting applications, including multipartite Diffie-Hellman and very efficient broadcast encryption. They attempted to construct multilinear maps from abelian varieties (extending known techniques for constructing bilinear maps), but they identified serious obstacles, and concluded that "such maps might have to either come from outside the realm of algebraic geometry, or occur as 'unnatural' computable maps arising from geometry".

Since then, the persistent absence of cryptographically useful multilinear maps as not stopped researchers from proposing applications of them. For example, Rückert and Schröder [RS09] use multilinear maps to construct efficient aggregate and verifiably encrypted signatures without random oracles. Papamanthou et al. [PTT10] show that compact multilinear maps give very efficient authenticated data structures. Rothblum [Rot13] uses multilinear maps to construct a counterexample to the conjecture that all bit-encryption schemes are KDM-secure (secure when given bit-encryptions of the secret key).

Here, we construct multilinear maps from ideal lattices. Our multilinear maps are "noisy" and bounded to polynomial degree. For very high degree, the "noisiness" overwhelms the signal, somewhat like for ciphertexts in somewhat homomorphic encryption schemes. In light of their noisiness, one could say that our multilinear maps are indeed "unnatural" computable maps arising from geometry. Our candidate multilinear maps differ quite substantially from the "ideal" multilinear maps envisioned by Boneh and Silverberg, in particular some problems that are hard relative to contemporary bilinear maps are easy with our construction (see Section 4.4). Nonetheless, the multilinear analog of the decision Diffie-Hellman problem appears hard in our construction, which gives cause for optimism about its applications in cryptography. In this paper we only demonstrate the applicability of our candidate to the "obvious" application of multipartite Diffie-Hellman key exchange, but other applications are surly possible.

The boundedness of our encodings has interesting consequences, both positive and negative. On the positive side, it hinders an attack based on Boneh and Lipton's subexponential algorithm for solving the discrete logarithm in black box fields [BL96]. This attack cannot be used to solve the "discrete log" problem in our setting, since their algorithm requires exponentiations with exponential degree. On the negative size, the dependence between the degree and parameter-size prevents us from realizing applications such that Papamanthou et al. [PTT10] that needs "compact" maps. Similarly, so far we were not able to use our maps to realize Rothblum's counterexample to the KDM-security of bit encryption conjecture [Rot13]: That counterexample requires degree that is polynomial, but a polynomial that is always just out of our reach of our parameters.

The security of the multilinear-DDH problem in our constructions relies on new hardness assumptions, and we provide an extensive cryptanalysis to validate these assumptions. To make sure that our constructions are not "trivially" insecure, we prove that our constructions are secure against adversaries that merely run a straight-line program. We also analyze our constructions with respect to the best known averaging, algebraic and lattice attacks. Many of these attacks have been published before [CS97, HKL$^+$00, Gen01, GS02, Szy03, HGS04, NR06] in cryptanalysis of the NTRU [HPS01, HHGP$^+$03] and GGH [GGH97] signature schemes. We also present new attacks on *principal* ideal lattices, which arise in our constructions, that are more efficient than

(known) attacks on general ideal lattices. Our constructions remain secure against all of the attacks that we present, both old and new. However, we feel that more cryptanalysis needs to be done, and this is partly why we have tried to write our cryptanalysis sections as a thorough survey that will serve as a useful starting point for cryptanalysts.

A brief overview. Our constructions work in polynomial rings and use principal ideals in these rings (and their associated lattices). In a nutshell, an instance of our construction has a secret short ring element $\mathbf{g} \in R$, generating a principal ideal $\mathcal{I} = \langle \mathbf{g} \rangle \subset R$. In addition, it has an integer parameter q and another secret $\mathbf{z} \in R/qR$, which is chosen at random (and hence is not small).

We think of a term like g^x in a discrete-log system as an "encoding" of the "plaintext exponent" x. In our case the role of the "plaintext exponents" is played by the elements in $R/\mathcal{I}$ (i.e. cosets of $\mathcal{I}$), and we "encode" it via division by $\mathbf{z}$ in R_q. In a few more details, our system provides many levels of encoding, where a level-i encoding of the coset $\boldsymbol{e}_\mathcal{I} = \boldsymbol{e} + \mathcal{I}$ is an element of the form $\boldsymbol{c}/\mathbf{z}^i \bmod q$ where $\boldsymbol{c} \in \boldsymbol{e}_\mathcal{I}$ is short. It is easy to see that such encodings can be both added and multiplied, so long as the numerators remain short. More importantly, we show that it is possible to publish a "zero testing parameter" that enables to test if two elements encode the same coset at a given level, without violating security (e.g., it should still be hard to compute x from an encoding of x at higher levels). Namely, we add to the public parameters an element of the form $\mathbf{p}_{zt} = h \cdot z^\kappa / g \bmod q$ for a not-too-large h, and show that multiplying an encoding of 0 by $\mathbf{p}_{zt} \pmod q$ yields a small element, while multiplying an encoding of a non-zero by $\mathbf{p}_{zt} \pmod q$ yields a large element. Hence we can distinguish zero from non-zero, and by subtraction we can distinguish two encodings of the same element from encodings of two different elements.

Our schemes are somewhat analogous to graded algebras, hence we sometimes call them *graded encoding schemes*. Our schemes are quite flexible, and for example can be modified to support the analog of asymmetric maps by using several different z's. On the other hand, other variants such as composite-order groups turn out to be insecure with our encodings (at least when implemented in a straightforward manner).

Organization. We define the general notion of encoding that we use in Section 2, as well an abstract notion of our main hardness assumption (which is a multilinear analog of DDH). Then in Section 3 we provide some background on ideal lattices, and in Section 4 we describe our construction. Our cryptanalysis of the new construction is described in the Section 6.

Applications. In Section 5 we describe the application to multipartite key agreement. Using our multilinear maps [GGH+13] have provided a construction of an attribute based encryption (ABE) scheme for general circuits. Concurrently and independently Gorbunov, Vaikuntanathan, and Wee [GVW13] also achieve ABE for circuits. One nice feature of their result is that they reduce security to the Learning with Errors (LWE) problem. Goldwasser, Kalai, Popa, Vaikuntanathan, and Zeldovich [GKP+13] show how one can use such an ABE scheme along with fully homomorphic encryption to construct a succinct single use functional encryption scheme. This in turn implies results for reusable Yao garbled circuits and other applications. Subsequent to our work, using our multilinear maps, Garg, Gentry, Sahai, and Waters [GGSW13] constructed a witness encryption scheme where a user's decryption key need not be an actual "key" at all, but rather can be a witness for some arbitrary NP relation specified by the encrypter (the encrypter itself may not know a witness).

2 Multilinear Maps and Graded Encoding Systems

Below we define formally our notion of a "approximate" multilinear maps, which we call *graded encoding schemes* (termed after the notion of graded algebra). To make the analogy and differences from multilinear maps more explicit, we begin by recalling the notion of *cryptographic multilinear maps* of Boneh and Silverberg [BS03] (using a slightly different syntax).

2.1 Cryptographic Multilinear Maps

Definition 1 (Multilinear Map [BS03])**.** *For $\kappa+1$ cyclic groups $G_1, \ldots, G_\kappa, G_T$ (written additively) of the same order p, an* κ-multilinear map $e : G_1 \times \cdots \times G_\kappa \to G_T$ *has the following properties:*

1. *For elements $\{g_i \in G_i\}_{i=1,\ldots,\kappa}$, index $i \in [\kappa]$ and integer $\alpha \in \mathbb{Z}_p$, it holds that*

$$e(g_1, \ldots, \alpha \cdot g_i, \ldots, g_\kappa) = \alpha \cdot e(g_1, \ldots, g_\kappa).$$

2. *The map e is non-degenerate in the following sense: if the elements $\{g_i \in G_i\}_{i=1,\ldots,\kappa}$, are all generators of their respective groups, then $e(g_1, \ldots, g_\kappa)$ is a generator of G_T.*

Remark 1. *Boneh and Silverberg considered in [BS03] only the* symmetric *case $G_1 = \cdots = G_\kappa$, the asymmetric case with different G_i's was considered, e.g., by Rothblum in [Rot13].*

2.1.1 Efficient Procedures

To be useful for cryptographic applications, we need to be able to manipulate (representations of) elements in these groups efficiently, and at the same time we need some other manipulations to be computationally hard. Specifically, a cryptographic multilinear map scheme consists of efficient procedures for instance-generation, element-encoding validation, group-operation and negation, and multilinear map, $\mathcal{MMP} = (\mathsf{InstGen}, \mathsf{EncTest}, \mathsf{add}, \mathsf{neg}, \mathsf{map})$. These procedures are described below.

Instance Generation. A randomized algorithm $\mathsf{InstGen}$ that takes the security parameter λ and the multi-linearity parameter κ (both in unary), and outputs $(G_1, \ldots, G_T, p, e, g_1, \ldots, g_\kappa)$. Here the G_i's and G_T describe the groups, $p \in \mathbb{Z}$ is their order, $e : G_1 \times \cdots \times G_\kappa \to G_T$ describes an κ-multilinear map as above, and $g_i \in \{0,1\}^*$ for $i = 1, \ldots, \kappa$ encode generators in these groups. To shorten some of the notations below, we denote $\mathsf{params} = (G_1, \ldots, G_T, p, e)$.

Element Encoding. Given the instance params from above, an index $i \in [\kappa]$, and a string $x \in \{0,1\}^*$, $\mathsf{EncTest}(\mathsf{params}, i, x)$ decides if x encodes an element in G_i (and of course the g_i's output by the instance-generator are all valid encodings). Similarly $\mathsf{EncTest}(\mathsf{params}, \kappa+1, x)$ efficiently recognizes description of elements in G_T.

It is usually assumed that elements have unique representation, namely for every i there are only p different strings representing elements in G_i. Below we therefore identify elements with their description, e.g. referring to "$x \in G_i$" rather than "x is a description of an element in G_i".

Group Operation. Given $x, y \in G_i$, $\mathsf{add}(\mathsf{params}, i, x, y)$ computes $x+y \in G_i$ and $\mathsf{neg}(\mathsf{params}, i, x)$ computes $-x \in G_i$. This implies also that for any $\alpha \in \mathbb{Z}_p$ we can efficiently compute $\alpha \cdot x \in G_i$.

Multilinear Map. For $\{x_i \in G_i\}_{i=1,\ldots,\kappa}$, $\mathsf{map}(\mathsf{params}, x_1, \ldots, x_\kappa)$ computes $e(x_1, \ldots, x_n) \in G_T$.

Another property, which was used by Papamanthou et al. [PTT10], is *compactness*, which means that the size of elements in the groups (as output by the instance generator) is independent of κ. We note that our multilinear maps will not satisfy this requirement, and are therefore unsuitable for the application in [PTT10].

2.1.2 Hardness Assumptions

For the multilinear map to be cryptographically useful, at least the discrete logarithm must be hard in the respective groups, and we usually also need the multilinear-DDH to be hard.

Multilinear Discrete-log (MDL). The Multilinear Discrete-Log problem is hard for a scheme $\mathcal{MMP}$, if for all $\kappa > 1$, all $i \in [\kappa]$, and all probabilistic polynomial time algorithms, the discrete-logarithm advantage of $\mathcal{A}$,

$$\mathsf{AdvDlog}_{\mathcal{MMP},\mathcal{A},\kappa}(\lambda) \stackrel{\text{def}}{=} \Pr\left[\mathcal{A}(\mathsf{params}, i, g_i, \alpha \cdot g_i) = \alpha \ : \ (\mathsf{params}, g_1, \ldots, g_l) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa), \alpha \leftarrow \mathbb{Z}_p\right],$$

is negligible in λ

Multilinear DDH (MDDH). For a symmetric scheme $\mathcal{MMP}$ (with $G_1 = G_2 = \cdots$), the Multilinear Decision-Diffie-Hellman problem is hard for $\mathcal{MMP}$ if for any κ and every probabilistic polynomial time algorithms $\mathcal{A}$, the advantage of $\mathcal{A}$ in distinguishing between the following two distributions is negligible in λ:

$$(\mathsf{params}, g, \alpha_0 g, \alpha_1 g, \ldots, \alpha_\kappa g, \ (\prod_{i=0}^{\kappa} \alpha_i) \cdot e(g \ldots, g))$$
$$\text{and} \quad (\mathsf{params}, g, \alpha_0 g, \alpha_1 g, \ldots, \alpha_\kappa g, \ \alpha \cdot e(g, \ldots, g))$$

where $(\mathsf{params}, g) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa)$ and $\alpha, \alpha_0, \alpha_1, \ldots, \alpha_\kappa$ are uniformly random in $\mathbb{Z}_p$.

2.2 Graded Encoding Schemes

The starting point for our new notion is viewing group elements in multilinear-map schemes as just a convenient mechanism of encoding the exponent: Typical applications of bilinear (or multilinear) maps use $\alpha \cdot g_i$ as an "obfuscated encoding" of the "plaintext integer" $\alpha \in \mathbb{Z}_p$. This encoding supports limited homomorphism (i.e., linear operations and a limited number of multiplications) but no more. In our setting we retain this concept of a somewhat homomorphic encoding, and have an algebraic ring (or field) R playing the role of the exponent space $\mathbb{Z}_p$. However we dispose of the multitude of algebraic groups, replacing them with "unstructured" sets of encodings of ring elements.

Perhaps the biggest difference between our setting and the setting of cryptographic multilinear maps, is that our encoding is randomized, which means that the same ring-element can be encoded in many different ways. (We do not even insist that the "plaintext version" of a ring element has a unique representation.) This means that checking if two strings encode the same element may not be trivial, indeed our constructions rely heavily on this check being feasible for some encodings and not feasible for others.

Another important difference is that our system lets us multiply not only batches of κ encodings at the time, but in fact any subset of encodings. This stands in stark contrast to the sharp threshold in multi-linear maps, where you can multiply exactly κ encodings, no more and no less.

A consequence of the ability to multiply any number of encodings is that we no longer have a single target group, instead we have a different "target group" for any number of multiplicands. This yields a richer structure, roughly analogous to *graded algebra*. In its simplest form (analogous to symmetric maps with a single source group), we have levels of encodings: At level zero we have the "plaintext" ring elements $\alpha \in R$ themselves, level one corresponds to $\alpha \cdot g$ in the source group, and level-i corresponds to a product of i level-1 encodings (so level-κ corresponds to the target group from multilinear maps).

Definition 2 (κ-Graded Encoding System)**.** *A κ-Graded Encoding System consists of a ring R and a system of sets $\mathcal{S} = \{S_i^{(\alpha)} \subset \{0,1\}^* : \alpha \in R,\ 0 \le i \le \kappa, \}$, with the following properties:*

1. *For every fixed index i, the sets $\{S_i^{(\alpha)} : \alpha \in R\}$ are disjoint (hence they form a partition of $S_i \stackrel{\text{def}}{=} \bigcup_\alpha S_{\boldsymbol{v}}^{(\alpha)}$).*

2. *There is an associative binary operation '$+$' and a self-inverse unary operation '$-$' (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every index $i \le \kappa$, and every $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, it holds that*
$$u_1 + u_2 \in S_i^{(\alpha_1+\alpha_2)} \quad \text{and} \quad -u_1 \in S_i^{(-\alpha_1)}$$
where $\alpha_1 + \alpha_2$ and $-\alpha_1$ are addition and negation in R.

3. *There is an associative binary operation '$\times$' (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every i_1, i_2 with $i_1 + i_2 \le \kappa$, and every $u_1 \in S_{i_1}^{(\alpha_1)}$ and $u_2 \in S_{i_2}^{(\alpha_2)}$, it holds that $u_1 \times u_2 \in S_{i_1+i_2}^{(\alpha_1 \cdot \alpha_2)}$. Here $\alpha_1 \cdot \alpha_2$ is multiplication in R, and $i_1 + i_2$ is integer addition.*

Clearly, Definition 2 implies that if we have a collection of n encodings $u_j \in S_{i_j}^{(\alpha_j)}$, $j = 1, 2 \ldots, n$, then as long as $\sum_j i_j \le \kappa$ we get $u_1 \times \cdots \times u_n \in S_{i_1+\cdots+i_n}^{(\prod_j \alpha_j)}$.

2.2.1 Efficient Procedures, the Dream Version

To be useful, we need efficient procedures for manipulating encodings well as as hard computational tasks. To ease the exposition, we first describe a "dream version" of the efficient procedures (which we do not know how to realize), and then explain how to modify them to deal with technicalities that arise from our use of lattices in the realization.

Instance Generation. The randomized $\mathsf{InstGen}(1^\lambda, 1^\kappa)$ takes as inputs the parameters λ, κ, and outputs $(\mathsf{params}, \mathbf{p}_{zt})$, where params is a description of a κ-Graded Encoding System as above, and $\mathbf{p}_{zt}$ is a zero-test parameter for level κ (see below).

Ring Sampler. The randomized $\mathsf{samp}(\mathsf{params})$ outputs a "level-zero encoding" $a \in S_0^{(\alpha)}$ for a nearly uniform element $\alpha \in_R R$. (Note that we require that the "plaintext" $\alpha \in R$ is nearly uniform, but not that the encoding a is uniform in $S_0^{(\alpha)}$.)

Encoding. The (possibly randomized) $\mathsf{enc}(\mathsf{params}, i, a)$ takes a "level-zero" encoding $a \in S_0^{(\alpha)}$ for some $\alpha \in R$ and index $i \le \kappa$, and outputs the "level-i" encoding $u \in S_i^{(\alpha)}$ for the same α.

Addition and negation. Given params and two encodings relative to the same index, $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, we have $\mathsf{add}(\mathsf{params}, i, u_1, u_2) = u_1 + u_2 \in S_i^{(\alpha_1+\alpha_2)}$, and $\mathsf{neg}(\mathsf{params}, i, u_1) = -u_1 \in S_i^{(-\alpha_1)}$.

Multiplication. For $u_1 \in S_{i_1}^{(\alpha_1)}$, $u_2 \in S_{i_2}^{(\alpha_2)}$ such that $i_1+i_2 \le \kappa$, we have $\mathsf{mul}(\mathsf{params}, i_1, u_1, i_2, u_2) = u_1 \times u_2 \in S_{i_1+i_2}^{(\alpha_1 \cdot \alpha_2)}$.

Zero-test. The procedure $\mathsf{isZero}(\mathsf{params}, u)$ output 1 if $u \in S_\kappa^{(0)}$ and 0 otherwise. Note that in conjunction with the subtraction procedure, this lets us test if $u_1, u_2 \in S_\kappa$ encode the same element $\alpha \in R$.

Extraction. This procedure extracts a "canonical" and "random" representation of ring elements from their level-κ encoding. Namely $\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u)$ outputs (say) $s \in \{0,1\}^\lambda$, such that:

(a) For any $\alpha \in R$ and two $u_1, u_2 \in S_\kappa^{(\alpha)}$, $\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_1) = \mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_2)$,

(b) The distribution $\{\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u) \ : \ \alpha \in_R R, u \in S_\kappa^{(\alpha)}\}$ is nearly uniform over $\{0,1\}^\lambda$.

2.2.2 Efficient Procedures, the Real-Life Version

Our realization of the procedures above over ideal lattices uses noisy encodings, where the noise increases with every operation and correctness is only ensured as long as it does not increase too much. We therefore modify the procedures above, letting them take as input (and produce as output) also a bound on the noise magnitude of the encoding in question. The procedures are allowed to abort if the bound is too high (relative to some maximum value which is part of the instance description params). Also, they provide no correctness guarantees if the bound on their input is "invalid." (When B is a noise-bound for some encoding u, we say that it is "valid" if it is at least as large as the bound produced by the procedure that produced u itself, and moreover any encoding that was used by that procedure (if any) also came with a valid noise bound.) Of course we also require that these procedure do not always abort, i.e. they should support whatever set of operations that the application calls for, before the noise becomes too large. Finally, we also relax the requirements on the zero-test and the extraction routines. Some more details are described next:

Zero-test. We sometime allow false positives for this procedure, but not false negatives. Namely, $\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, u) = 1$ for every $u \in S_\kappa^{(0)}$, but we may have $\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, u) = 1$ also for some $u \notin S_\kappa^{(0)}$. The weakest functionality requirement that we make is that for a uniform random choice of $\alpha \in_R R$, we have

$$\Pr_{\alpha \in_R R}\left[\exists\ u \in S_\kappa^{(\alpha)} \text{ s.t } \mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, u) = 1\right] = \text{negligible}(\lambda). \tag{1}$$

Additional requirements are considered security features (that a scheme may or may not possess), and are discussed later in this section.

Extraction. Our construction from Section 4 does not support full canonicalization. Instead, we settle for $\mathsf{ext}(\Lambda, \mathbf{p}_{zt}, u)$ that has a good chance of producing the same output when applied to different encoding of the same elements. Specifically, we replace properties (a)-(b) from above by the weaker requirements:

(a′) For a randomly chosen $a \leftarrow \mathsf{samp}(\mathsf{params})$, if we run the encoding algorithm twice to encode a at level κ and then extract from both copies then we get:

$$\Pr\left[\begin{array}{l}\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_1) \\ = \mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_2)\end{array} : \begin{array}{l} a \leftarrow \mathsf{samp}(\mathsf{params}) \\ u_1 \leftarrow \mathsf{enc}(\mathsf{params}, \kappa, a) \\ u_2 \leftarrow \mathsf{enc}(\mathsf{params}, \kappa, a)\end{array}\right] \geq 1 - \text{negligible}(\lambda).$$

(b′) The distribution $\{\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u) : a \leftarrow \mathsf{samp}(\mathsf{params}), u \leftarrow \mathsf{enc}(\mathsf{params}, \kappa, a)\}$ is nearly uniform over $\{0,1\}^\lambda$.

We typically need these two conditions to hold even if the noise bound that the encoding routine takes as input is larger than the one output by samp (upto some maximum value).

2.2.3 Hardness Assumptions

Our hardness assumptions are modeled after the discrete-logarithm and DDH assumptions in multilinear groups. For example, the most direct analog of the discrete-logarithm problem is trying to obtain a level-zero encoding $a \in S_0^{(\alpha)}$ for $\alpha \in R$ from an encoding relative to some other index $i > 0$.

The analog of DDH in our case roughly says that it is hard to recognize encoding of products, except relative to indexes upto κ. In other words, given $\kappa+1$ level-one encoding of random elements it should be infeasible to generate a level-κ encoding of their product, or even to distinguish it from random. To formalize the assumption we should specify how to generate level-κ encodings of the "the right product" and of a random element. The definition below just uses the encoding routine for that purpose, but see the appendix for a more general treatment. One way to formalize it is by the following process. (Below we suppress the noise bounds for readability):

1. $(\mathsf{params}, \mathbf{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa)$
2. For $i = 1, \ldots, \kappa + 1$:
3. Choose $a_i \leftarrow \mathsf{samp}(\mathsf{params})$ // level-0 encoding of random $\alpha_i \in_R R$
4. Set $u_i \leftarrow \mathsf{enc}(\mathsf{params}, 1, a_i)$ // level-1 encoding of the α_i's
5. Set $\tilde{a} = \prod_{i=1}^{\kappa+1} a_i$ // level-0 encoding of the product
6. Choose $\hat{a} \leftarrow \mathsf{samp}(\mathsf{params})$ // level-0 encoding of a random element
7. Set $\tilde{u} \leftarrow \mathsf{enc}(\mathsf{params}, \kappa, \tilde{a})$ // level-κ encoding of the product
8. Set $\hat{u} \leftarrow \mathsf{enc}(\mathsf{params}, \kappa, \hat{a})$ // level-κ encoding of random

(We note that with the noise bound, it may be important that the encoding routines for both $\tilde{a}$ and $\hat{a}$ get as input the same bound, i.e., the largest of the bounds for $\tilde{a}$ and $\hat{a}$.) The GDDH distinguisher gets all the level-one u_i's and either $\tilde{u}$ (encoding the right product) or $\hat{u}$ (encoding a random element), and it needs to decide which is the case. In other words, the GDDH assumption says that for any setting of the parameters, the following two distributions, defined over the experiment above, are computationally indistinguishable:

$$\mathcal{D}_{GDDH} = \{(\mathsf{params}, \mathbf{p}_{zt}, \{u_i\}_i, \tilde{u})\} \text{ and } \mathcal{D}_{RAND} = \{(\mathsf{params}, \mathbf{p}_{zt}, \{u_i\}_i, \hat{u})\}.$$

Zero-test security. In some settings (such as NIZK) we may be concerned with adversaries that can generate encodings in a malicious way and submit them to the zero-test procedure. In such settings, the statistical property from Equation (1) is not sufficient, instead we would like the zero-test to accept *only* encoding of zero at the right level. This can be statistical (i.e. no false positive exist) or computational (i.e. it is hard to find them).

Definition 3. *A graded-encoding scheme $\mathcal{GE}$ enjoys* statistical *zero-test security if the only strings that pass the zero-test are encodings of zero, except with a negligible probability over the instance generation. That is, for every κ:*

$$\Pr_{\mathsf{params},\mathbf{p}_{zt}}[\exists\ u \notin S_\kappa^{(0)}\ s.t.\ \mathsf{isZero}(\mathsf{params},\mathbf{p}_{zt},u)=1] \leq \mathit{negligible}(\lambda),$$

where the probability is taken over $(\mathsf{params},\mathbf{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda,1^\kappa)$. *$\mathcal{GE}$ enjoys* computational *zero-test security if for every adversary $\mathcal{A}$ and parameters as above:*

$$\Pr_{\substack{(\mathsf{params},\mathbf{p}_{zt})\leftarrow\mathsf{InstGen}(1^\lambda,1^\kappa)\\ u\leftarrow\mathcal{A}(\mathsf{params},\mathbf{p}_{zt})}}\Big[u \notin S_\kappa^{(0)}\ \textit{but}\ \ \mathsf{isZero}(\mathsf{params},\mathbf{p}_{zt},u)=1\Big] \leq \mathit{negligible}(\lambda).$$

3 Preliminaries

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 $\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n \in L$ such that $L = \{\sum_{i=1}^n z_i\boldsymbol{b}_i : z_i \in \mathbb{Z}\ \forall i\}$. If we arrange the vectors $\boldsymbol{b}_i$ as the columns of a matrix $B \in \mathbb{R}^{n\times n}$ then we can write $L = \{B\boldsymbol{z} : \boldsymbol{z} \in \mathbb{Z}^n\}$. If B is a basis for L then we say that B spans L. For a lattice $L \subset \mathbb{R}^n$, its *dual lattice* consists of all the points in $\mathrm{span}(L)$ that are orthogonal to L modulo one, namely $L^* = \{\boldsymbol{y} \in \mathrm{span}(L) : \forall \boldsymbol{x} \in L, \langle \boldsymbol{x},\boldsymbol{y}\rangle \in \mathbb{Z}\}$

Gaussians. For a real $\sigma > 0$, define the (spherical) Gaussian function on $\mathbb{R}^n$ with parameter σ as $\rho_\sigma(\boldsymbol{x}) = \exp(-\pi\|\boldsymbol{x}\|^2/\sigma^2)$ for all $\boldsymbol{x} \in \mathbb{R}^n$. This generalizes to ellipsoid Gaussians, where the different coordinates are jointly Gaussian but not independent, where we replace the parameter $\sigma \in \mathbb{R}$ by the (square root of the) covariance matrix $\Sigma \in \mathbb{R}^{n\times n}$. For a rank-$n$ matrix $S \in \mathbb{R}^{m\times n}$, the ellipsoid Gaussian function on $\mathbb{R}^n$ with parameter S is defined by $\rho_S(\boldsymbol{x}) = \exp\left(-\pi\boldsymbol{x}^T(S^TS)^{-1}\boldsymbol{x}\right)\ \forall \boldsymbol{x} \in \mathbb{R}^n$. Obviously this function only depends on $\Sigma = S^TS$ and not on the particular choice of S. It is also clear that the spherical case can be obtained by setting $S = \sigma I_n$, with I_n the n-by-n identity matrix.

The *ellipsoid discrete Gaussian distribution* over lattice L with parameter S is $\forall\, \boldsymbol{x} \in L, D_{L,S}(\boldsymbol{x}) = \rho_S(\boldsymbol{x})/\rho_S(L)$, where $\rho_S(L)$ denotes $\sum_{\boldsymbol{x}\in L}\rho_S(\boldsymbol{x})$. In other words, the probability $D_{L,S}(\boldsymbol{x})$ is simply proportional to $\rho_S(\boldsymbol{x})$, the denominator being a normalization factor. The same definitions apply to the spherical case, $D_{L,\sigma}(\cdot)$.

Smoothing parameter. Micciancio and Regev defined in [MR07] the *smoothing parameter* for a lattice L and real $\epsilon > 0$, denoted $\eta_\epsilon(L)$, as the smallest s such that $\rho_{1/s}(L^* \setminus \{\mathbf{0}\}) \leq \epsilon$. Intuitively, for a small enough ϵ, the number $\eta_\epsilon(L)$ is sufficiently larger than L's fundamental parallelepiped so that sampling from the corresponding Gaussian "wipes out the internal structure" of L. It is easy to show that the size of vectors drawn from $D_{L,S}$ is roughly bounded by the largest singular value of S. (Recall that the largest and least singular values of a full rank matrix $X \in \mathbb{R}^{m\times n}$ are defined as $\sigma_1(X) = \sup(U_X)$ and $\sigma_n(X) = \inf(U_X)$, respectively, where $U_X = \{\|X\boldsymbol{u}\| : \boldsymbol{u} \in \mathbb{R}^n, \|\boldsymbol{u}\| = 1\}$.)

Lemma 1. *For a rank-n lattice L, constant $0 < \epsilon < 1$ and matrix S s.t. $\sigma_n(S) \geq \eta_\epsilon(L)$, we have* $\Pr_{\boldsymbol{v}\leftarrow\mathcal{D}_{L,S}}\left(\|\boldsymbol{v}\| \geq \sigma_1(S)\sqrt{n}\right) \leq \frac{1+\epsilon}{1-\epsilon}\cdot 2^{-n}$.

Sum of Discrete Gaussians. A recent work [AGHS12] considered the process that begins by choosing "once and for all" m points in a lattice L, drawn independently from a "wide discrete

Gaussian" $\boldsymbol{x}_i \leftarrow D_{L,S}$. Once the $\boldsymbol{x}_i$'s are fixed, they are arranged as the rows of an m-by-n matrix $X = (\boldsymbol{x}_1|\boldsymbol{x}_2|\ldots|\boldsymbol{x}_m)^T$, and we consider the distribution $\mathcal{D}_{X,\sigma}$, induced by choosing an integer vector $\boldsymbol{v}$ from a discrete spherical Gaussian over $\mathbb{Z}^m$ with parameter σ and outputting $\boldsymbol{y} = X^T\boldsymbol{v}$, $\mathcal{E}_{X,\sigma} \stackrel{\text{def}}{=} \{X^T\boldsymbol{v} : \boldsymbol{v} \leftarrow D_{\mathbb{Z}^m,\sigma}\}$. [AGHS12] proved that with high probability over the choice of X, the distribution $\mathcal{D}_{X,\sigma}$ is statistically close to the ellipsoid Gaussian $D_{L,\sigma X}$, and moreover the singular values of X are of size roughly $\sigma\sqrt{m}$:

Theorem 1 ([AGHS12])**.** *Let L be a full-rank lattice $L \subset \mathbb{R}^n$ and B a matrix whose rows form a basis of L, and denote $\chi = \sigma_1(B)/\sigma_n(B)$. Also let ϵ be negligible in n, and let m, s, s' be parameters such that $s \geq \eta_\epsilon(\mathbb{Z}^n)$, $m \geq 10n\log(8(mn)^{1.5}s\chi)$ and $s' \geq 4mn\chi\ln(1/\epsilon)$.*

Then, when choosing the rows of an m-by-n matrix X from the spherical Gaussian over L, $X \leftarrow (\mathcal{D}_{L,s})^m$, we have with all but probability $2^{-O(m)}$ over the choice of X, that the statistical distance between $\mathcal{E}_{X,s'}$ and the ellipsoid Gaussian $\mathcal{D}_{L,s'X}$ is bounded by 2ϵ.

Lemma 2 ([AGHS12])**.** *There exists a universal constant $K > 1$ such that for all $m \geq 2n$, $\epsilon > 0$ and every n-dimensional real lattice $L \subset \mathbb{R}^n$, the following holds: Choosing the rows of an m-by-n matrix X independently at random from a spherical discrete Gaussian on L with parameter $\rho > 2K\eta_\epsilon(L)$, $X \leftarrow (D_{L,\rho})^m$, we have*

$$\Pr\left[s\sqrt{2\pi m}/K < \sigma_n(X) \leq \sigma_1(X) < \rho K\sqrt{2\pi m}\right] > 1 - (4m\epsilon + O(\exp(-m/K))).$$

Ideal lattices. For n a power of two, we consider the $2n$'th cyclotomic polynomial ring $R = \mathbb{Z}[X]/(X^n+1)$, and identify an element $\boldsymbol{u} \in R$ with the coefficient vector[1] of the degree-$(n-1)$ integer polynomial that represents $\boldsymbol{u}$. In this way, R is identified with the integer lattice $\mathbb{Z}^n$. Additionally we sometimes consider also the ring $R_q = R/qR = \mathbb{Z}_q[X]/(X^n+1)$ for a (large enough) integer q. Obviously, addition in these rings is done component-wise in their coefficients, and multiplication is polynomial multiplication modulo the ring polynomial X^n+1. In some cases we also consider the corresponding number field $\mathbb{K} = \mathbb{Q}[X]/(X^n+1)$, which is likewise associated with the linear space $\mathbb{Q}^n$.

For an element $\mathbf{g} \in R$, let $\langle\mathbf{g}\rangle$ be the principal ideal in R generated by $\mathbf{g}$ (alternatively, the sub-lattice of $\mathbb{Z}^n$ corresponding to this ideal), namely $\langle\mathbf{g}\rangle = \{\mathbf{g}\cdot\boldsymbol{u} : \boldsymbol{u} \in R\}$. We call $\langle\mathbf{g}\rangle$ an *ideal lattice* to stress its dual interpretation as both an ideal and a lattice. Let $B(\mathbf{g})$ denote the basis of the lattice $\langle\mathbf{g}\rangle$ consisting of the vectors $\{\mathbf{g}, X\mathbf{g}, X^2\mathbf{g}, \ldots, X^{n-1}\mathbf{g}\}$.

For an arbitrary element $\boldsymbol{u} \in R$, denote by $[\boldsymbol{u}]_\mathbf{g}$ the reduction of $\boldsymbol{u}$ modulo the fundamental cell of $B(\mathbf{g})$, which is *symmetric around the origin*. To wit, $[\boldsymbol{u}]_\mathbf{g}$ is the unique element $\boldsymbol{u}' \in R$ such that $\boldsymbol{u}-\boldsymbol{u}' \in \langle\mathbf{g}\rangle$ and $\boldsymbol{u}' = \sum_{i=0}^{n-1}\alpha_i X^i\mathbf{g}$ where all the α_i's are in the interval $[-\frac{1}{2}, \frac{1}{2})$. We use the similar notation $[t]_p$ for integers t, p to denote the reduction of t modulo p into the interval $[-p/2, p/2)$.

4 The New Encoding Schemes

An instance of our basic construction is parametrized by the security parameter λ and the required multi-linearity level $\kappa \leq$poly(λ). Based on these parameters, we choose a cyclotomic ring $R =$

[1] Other representations of polynomials are also possible, for example representing a polynomial by its canonical embedding is sometimes preferable to the coefficient representation. Here we stick to coefficient representation for simplicity.

$\mathbb{Z}[X]/(X^n+1)$ (where n is large enough to ensure security), a modulus q that defines $R_q = R/qR$ (with q large enough to support functionality), and another parameter m (chosen so that we can apply Theorem 1). The specific constraints that these parameters must satisfy are discussed at the end of this section, an approximate setting to keep in mind is $n = \tilde{O}(\kappa\lambda^2)$, $q = 2^{n/\lambda}$ and $m = O(n^2)$.

4.1 The Basic Graded Encoding Scheme

An instance of our scheme relative to the parameters above encodes elements of a quotient ring $QR = R/\mathcal{I}$, where $\mathcal{I}$ is a principal ideal $\mathcal{I} = \langle \mathbf{g} \rangle \subset R$, generated by a short vector $\mathbf{g}$. Namely, the "ring elements" that are encoded in our scheme are cosets of the form $\boldsymbol{e} + \mathcal{I}$ for some vector $\boldsymbol{e}$. The short generator $\mathbf{g}$ itself is kept secret, and no "good" description of $\mathcal{I}$ is made public in our scheme. In addition, our system depends on another secret element $\mathbf{z}$, which is chosen at random in R_q (and hence is not short).

A level-zero ("plaintext") encoding of a coset $\boldsymbol{e} + \mathcal{I} \in R/\mathcal{I}$ is just a short vector in that coset (which must exist, since the generator $\mathbf{g}$ is short and therefore the basic cell of $\mathcal{I}$ is quite small). For higher-level encodings, a level-i encoding of the same coset is a vector of the form $\boldsymbol{c}/\mathbf{z}^i \in R_q$ with $\boldsymbol{c} \in \boldsymbol{e} + \mathcal{I}$ short. Specifically, for $i \in \{0, 1, \ldots, \kappa\}$ the set of all level-i encodings is $S_i = \{\boldsymbol{c}/\mathbf{z}^i \in R_q : \|\boldsymbol{c}\| < q^{1/8}\}$, and the set of levle-$i$ encodings of the "plaintext element" $\boldsymbol{e} + \mathcal{I}$ is $S_i^{(\boldsymbol{e}+\mathcal{I})} = \{\boldsymbol{c}/\mathbf{z}^i \in R_q : \boldsymbol{c} \in \boldsymbol{e} + \mathcal{I},\ \|\boldsymbol{c}\| < q^{1/8}\ \}$. Throughout the construction we use the size of the numerator as the "noise level" in the encoding. Namely, with each level-i encoding $\boldsymbol{c}/\mathbf{z}^i$ we produce also an upper bound on $\|\boldsymbol{c}\|$.

Instance generation: $(\mathsf{params}, \mathbf{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa)$**.** Our instance-generation procedure chooses at random the ideal-generator $\mathbf{g}$ and denominator $\mathbf{z}$, as well as several other vectors that are used in the other procedures and are described later in the section. The denominator $\mathbf{z}$ is chosen uniformly at random in R_q. For technical reasons, the generator $\mathbf{g} \in R$ should be chosen so that both $\mathbf{g}$ and $\mathbf{g}^{-1} \in \mathbb{K}$ are short. (Recall that we denote $\mathbb{K} = \mathbb{Q}[X]/(X^n+1)$. The reason that we need $\mathbf{g}^{-1} \in \mathbb{K}$ to be short is explained when we describe the zero-testing procedure.) We simply draw $\mathbf{g}$ from a discrete Gaussian over $\mathbb{Z}^n$, say $\mathbf{g} \leftarrow D_{\mathbb{Z}^n,\sigma}$ with $\sigma = \tilde{O}(\sqrt{n})$. Clearly $\mathbf{g}$ itself is short (of size less than $\sigma\sqrt{n}$), and we claim that with good probability its inverse in the field of fractions is also rather short. To see this, notice that with probability $1 - o(1/n)$, evaluating $\mathbf{g}$ at any complex n'th root of unity $\zeta \in \mathbb{C}$ yields $\mathbf{g}(\zeta)$ which is not too tiny, say larger than $1/n$. Hence with probability $1 - o(1)$ we have $\mathbf{g}^{-1}(\zeta) = 1/\mathbf{g}(\zeta) < n$ for all the primitive $2n$'th roots of unity ζ, which means that $\mathbf{g}^{-1}$ itself is not too large, say $\|1/\mathbf{g}\| < n^2$. We can draw repeatedly until we get this condition to hold.

Once we have $\mathbf{g}, \mathbf{z}$, we choose and publish some other elements in R_q that will be used for the various procedures below. Specifically we have $m+1$ elements $rand_1, \ldots, \mathbf{x}_m, \mathbf{y}$ that are used for encoding, and an element $\mathbf{p}_{zt}$ that is used as a zero-testing parameter. These elements are described later. finally we also choose a random seed s for a strong randomness extractor. The instance-generation procedure outputs $\mathsf{params} = (n, q, \mathbf{y}, \{\mathbf{x}_i\}_i, s)$ and $\mathbf{p}_{zt}$.

Sampling level-zero encodings: $\boldsymbol{d} \leftarrow \mathsf{samp}(\mathsf{params})$**.** To sample a level-zero encoding of a random coset, we just draw a random short element in R, $\boldsymbol{d} \leftarrow D_{\mathbb{Z}^n,\sigma'}$, where $\sigma' = \sigma n$ (for σ that was used to sample $\mathbf{g}$). Since whp $\sigma' \geq \eta_{2^{-\lambda}}(\mathcal{I})$, then the induced distribution over the cosets of $\mathcal{I}$ is close to uniform, upto a negligible distance. Also the size of this level-zero encoding is bounded by $\sigma'\sqrt{n}$ (and we use this as our noise-bound for this encoding).

Encodings at higher levels: $u_i \leftarrow \mathsf{enc}(\mathsf{params}, i, \boldsymbol{d})$. To allow encoding of cosets at higher levels, we publish as part of our instance-generation a level-one encoding of $1+\mathcal{I}$, namely an element $\mathbf{y} = [\mathbf{a}/\mathbf{z}]_q$ where $\mathbf{a} \in 1+\mathcal{I}$ is short. A simplistic method of doing that is drawing $\mathbf{a} \leftarrow D_{1+\mathcal{I},\sigma'}$, then computing $\mathbf{y}$ from $\mathbf{a}$. (Later we describe a somewhat more involved procedure, which we believe is more secure.) Given a level-zero encoding $\boldsymbol{d}$ as above, we can multiply it by $\mathbf{y}$ over R_q to get $\boldsymbol{u}_1 := [\mathbf{y}\boldsymbol{d}]_q$. Note that $\boldsymbol{u}_1 = [\boldsymbol{d}\mathbf{a}/\mathbf{z}]_q$, where $\boldsymbol{d}\mathbf{a} \in \boldsymbol{d}+\mathcal{I}$ as needed, and the size of the numerator is bounded by $\|\boldsymbol{d}\| \cdot \|\mathbf{a}\| \cdot \sqrt{n} = \mathrm{poly}(n)$. More generally we can generate a level-i encoding as $\boldsymbol{u}_i := [\boldsymbol{d}\mathbf{y}^i]_q = [\boldsymbol{d}\mathbf{a}^i/\mathbf{z}^i]_q$. The numerator $\boldsymbol{d}\mathbf{a}^i$ is obviously in $\boldsymbol{d}+\mathcal{I}$, and its size is at most $\|\boldsymbol{d}\| \cdot \|\mathbf{a}\|^i \cdot n^{i/2}$.

The above encoding is insufficient, however, since from $\boldsymbol{u}_1$ and $\mathbf{y}$ it is easy to get back $\boldsymbol{d}$ by simple division in R_q. We therefore include in the public parameters also the "randomizers" $\mathbf{x}_i$, these are just random encodings of zero, namely $\mathbf{x}_i = [\mathbf{b}_i/\mathbf{z}]_q$ where the $\mathbf{b}_i$'s are short elements in $\mathcal{I}$. A simplistic procedure for choosing these randomizers would be to draw short these elements as $\mathbf{b}_i \leftarrow D_{\mathcal{I},\sigma'}$ and publish $\mathbf{x}_i = [\mathbf{b}_i/\mathbf{z}]_q$. As we note in Section 6.3.2, we have reasons to suspect that this simplistic method is insecure so instead we use a somewhat more involved sampling procedure, see details in Section 6.3.2. Below we denote by $\mathbf{X}$ the matrix with the vectors $\mathbf{x}_i$ as rows, namely $\mathbf{X} = (\mathbf{x}_1|\ldots|\mathbf{x}_m)^T$. We also use $\mathbf{B}$ to denote the matrix with the numerators $\mathbf{b}_i$ as rows, i.e., $\mathbf{B} = (\mathbf{b}_1|\ldots|\mathbf{b}_m)^T$.

We use the $\mathbf{x}_i$'s to randomize level-one encodings: Given $\boldsymbol{u}' = [\boldsymbol{c}'/\mathbf{z}]_q$ with noise-bound $\|\boldsymbol{c}'\| < \gamma$, we draw an m-vector of integer coefficients $\boldsymbol{r} \leftarrow D_{\mathbb{Z}^m,\sigma^*}$ for large enough σ^* (e.g. $\sigma^* = 2^\lambda\gamma$), and output

$$\boldsymbol{u} \;:=\; [\boldsymbol{u}' + \mathbf{X}\boldsymbol{r}]_q \;=\; [\boldsymbol{u}' + \sum_{i=1}^{m} r_i\mathbf{x}_i]_q \;(=\; [\frac{\boldsymbol{c}' + \sum_i r_i\mathbf{b}_i}{\mathbf{z}}]_q).$$

We write $\mathbf{B}\boldsymbol{r}$ as a shorthand for $\sum_i r_i\mathbf{b}_i$ and similarly $\mathbf{X}\boldsymbol{r}$ as a shorthand for $\sum_i r_i\mathbf{x}_i$.

Since all the $\mathbf{b}_i$'s are in the ideal $\mathcal{I}$, then obviously $\boldsymbol{c}' + \sum_i r_i\mathbf{b}_i$ is in the same coset of $\mathcal{I}$ as $\boldsymbol{c}'$ itself. Moreover since $\|\mathbf{b}_i\| < \mathrm{poly}(n)$ then $\|\mathbf{B}\boldsymbol{r}\| < \sigma^*\mathrm{poly}(m,n)$. If indeed $\|\boldsymbol{c}'\| < \gamma$, then $\|\boldsymbol{c}' + \mathbf{B}\boldsymbol{r}\| < \gamma + \sigma^*\mathrm{poly}(m,n)$. We also claim that the distribution of $\boldsymbol{u}$ is nearly independent of original $\boldsymbol{u}'$ (except of course its coset). To see why, note that if the $\mathbf{b}_i$'s are chosen from a wide enough spherical distribution then we can use Theorem 1 to conclude that $\mathbf{B}\boldsymbol{r}$ is close to a wide ellipsoid Gaussian. With our choice of σ^* the "width" of that distribution is much larger than the original $\boldsymbol{c}'$, hence the distribution of $\boldsymbol{c}' + \mathbf{B}\boldsymbol{r}$ is nearly independent of $\boldsymbol{c}'$, except in the coset that it belongs to.

A different approach is to re-randomize $\mathbf{y}$, setting $\mathbf{y}' := \mathbf{y} + \mathbf{X}\boldsymbol{r}$ and then encode via $\boldsymbol{u}_1 := [\mathbf{y}'\boldsymbol{d}]_q$. This does not have the information-theoretic same-distribution guarantee as above (since the distributions $[\mathbf{y}'\boldsymbol{d}]_q$ and $[\mathbf{y}'\boldsymbol{d}']_q$ may differ, even if $\boldsymbol{d}, \boldsymbol{d}'$ are both short and in the same coset). But on the plus side, it is more convenient to use this re-randomization method for encoding at high levels $i > 1$: After computing the randomized $\mathbf{y}'$, we can use it by setting $\boldsymbol{u}_i := [\boldsymbol{d}(\mathbf{y}')^i]_q$.

Remark 2. *Note that in the above description we used the matrix $\mathbf{X}$ to randomize level-one encodings. Using similar pubic parameter $\mathbf{X}_i$ we can generalize the re-randomization procedure to work at any level $i \le \kappa$. In particular we abstract this procedure as $\mathsf{reRand}(\mathbf{y}, i, \boldsymbol{u}')$: Given $\boldsymbol{u}' = [\boldsymbol{c}'/\mathbf{z}^i]_q$ with noise-bound $\|\boldsymbol{c}'\| < \gamma$, we draw an m-vector of integer coefficients $\boldsymbol{r} \leftarrow D_{\mathbb{Z}^m,\sigma^*}$ for large enough σ^* (e.g. $\sigma^* = 2^\lambda\gamma$), and output $\boldsymbol{u} := [\boldsymbol{u}' + \mathbf{X}_i\boldsymbol{r}]_q$ as a re-randomized version of $\boldsymbol{u}$. Using the same argument as above we can conclude that the distribution generated in this way will be independent of $\boldsymbol{c}'$, except in the coset that it belongs to.*

Note that for some applications it might be useful to use the re-randomization operation multiple times. We consider the case in which a constant number of re-randomizations is needed. In this case, with the ℓ^{th} re-randomization (for any constant ℓ) we can generate an encoding by choosing $\boldsymbol{r}$ from $D_{\mathbb{Z}^m,\sigma^*}$ where $\sigma^* = 2^{\lambda^\ell}$ and re-randomizing as above. Since the addition and multiplication of constant number of terms increases noise by a small factor we can claim that each re-randomization wipes the structure that was present previously (even with multiple additions and multiplications).

We define a canonicalizing encoding algorithm $\mathsf{cenc}_\ell(\mathsf{params}, i, \boldsymbol{u}')$ which takes as input an encoding of $\boldsymbol{u}'$ and generates another encoding according with a noise factor of 2^{λ^ℓ}.

Adding and multiplying encodings. It is easy to see that the encoding as above is additively homomorphic, in the sense that adding encodings yields an encoding of the sum. This follows since if we have many short $\boldsymbol{c}_j$'s then their sum is still short, $\|\sum_j \boldsymbol{c}_j\| \ll q$, and therefore the sum $\boldsymbol{c} = \sum_j \boldsymbol{c}_j = [\sum_j \boldsymbol{c}_j]_q \in R_q$ belong to the coset $\sum_j(\boldsymbol{c}_j + \mathcal{I})$. Hence, if we denote $\boldsymbol{u}_j = \boldsymbol{c}_j/\mathbf{z} \in R_q$ then each $\boldsymbol{u}_j$ is an encoding of the coset $\boldsymbol{c}_j + \mathcal{I}$, and the sum $[\sum_j \boldsymbol{u}_j]_q$ is of the form $\boldsymbol{c}/\mathbf{z}$ where $\boldsymbol{c}$ is still a short element in the sum of the cosets.

Moreover, since $\mathcal{I}$ is an ideal then multiplying upto κ encodings can be interpreted as an encoding of the product, by raising the denominator to the appropriate power. Namely, for $\boldsymbol{u}_j = \boldsymbol{c}_j/\mathbf{z} \in R_q$ as above, we have

$$\boldsymbol{u} = \prod_{j=1}^{\kappa} \boldsymbol{u}_j = \frac{\prod_j \boldsymbol{c}_j}{\mathbf{z}^\kappa} \quad \text{(all the operations in } R_q\text{)}.$$

As long as the $\boldsymbol{c}_j$'s are small enough to begin with, we still have $\|\prod_j \boldsymbol{c}_j\| \ll q$, which means that $[\prod_j \boldsymbol{c}_j]_q = \prod_j \boldsymbol{c}_j$ (operations in R), hence $[\prod_j \boldsymbol{c}_j]_q$ belongs to the product coset $\prod_j(\boldsymbol{c}_j + \mathcal{I})$.

Thus, if each $\boldsymbol{u}_j$ is a level-1 encoding of the coset $\boldsymbol{c}_j + \mathcal{I}$ with short-enough numerator, then their product is a level-κ encoding of the product coset. We note that just like level-1 encoding, level-κ encoding still offers additive homomorphism.

Zero testing: $\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{u}_\kappa) \stackrel{?}{=} 0/1$. Since the encoding is additively homomorphic, we can test equality between encodings by subtracting them and comparing to zero. To enable zero-testing, we generate the zero-testing parameter as follows: We draw a "somewhat small" ring element $\boldsymbol{h} \leftarrow D_{\mathbb{Z}^n,\sqrt{q}}$, and the zero-testing parameter is set as $\mathbf{p}_{zt} = [\boldsymbol{h}\mathbf{z}^\kappa/\mathbf{g}]_q$. To test if a level-$\kappa$ encoding $\boldsymbol{u} = [\boldsymbol{c}/\mathbf{z}^\kappa]_q$ is an encoding of zero, we just multiply it in R_q by $\mathbf{p}_{zt}$ and check whether the resulting element $\boldsymbol{w} = [\mathbf{p}_{zt} \cdot \boldsymbol{u}]_q$ is short (e.g., shorter than $q^{3/4}$). Namely, we use the test

$$\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{u}) \;=\; \begin{cases} 1 & \text{if } \|[\mathbf{p}_{zt}\boldsymbol{u}]_q\|_\infty < q^{3/4} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

To see why this works, note that

$$\boldsymbol{w} \;=\; \mathbf{p}_{zt} \cdot \boldsymbol{u} \;=\; \frac{\boldsymbol{h}\mathbf{z}^\kappa}{\mathbf{g}} \cdot \frac{\boldsymbol{c}}{\mathbf{z}^\kappa} \;=\; \boldsymbol{h} \cdot \boldsymbol{c}/\mathbf{g} \quad \text{(all the operations in } R_q\text{)}.$$

If $\boldsymbol{u}$ is an encoding of zero then $\boldsymbol{c}$ is a short vector in $\mathcal{I}$, which means that it is divisible by $\mathbf{g}$ in R. Hence the element $\boldsymbol{c}/\mathbf{g} \in R_q$ is the same as the element $\boldsymbol{c} \cdot \mathbf{g}^{-1} \in \mathbb{K}$, which means that it has size at most $\|\boldsymbol{c}\| \cdot \|\mathbf{g}^{-1}\| \cdot \sqrt{n} = \|\boldsymbol{c}\| \cdot \mathrm{poly}(n)$. This, in turn, implies that $\|\boldsymbol{w}\| \le \|\boldsymbol{h}\| \cdot \|\boldsymbol{c}\| \cdot \mathrm{poly}(n)$, which for our choice of parameter is $q^{1/2} \cdot q^{1/8} \cdot \mathrm{poly}(n) < q^{3/4}$.

If $\boldsymbol{u}$ is an encoding of a different coset, then $\boldsymbol{c}$ is a short vector in some coset of $\mathcal{I}$. In this case we have $\boldsymbol{w} = [\boldsymbol{c} \cdot \boldsymbol{h}/\mathbf{g}]_q$, where $\boldsymbol{c}, \mathbf{g}$ are small (and $\boldsymbol{h}$ is "somewhat small"). Intuitively, since $\boldsymbol{h}/\mathbf{g}$ is large whp then for a "random enough" $\boldsymbol{c}$ we expect the size of $\boldsymbol{w}$ to be large. More formally, we argue below that when choosing a uniformly random coset of $\mathcal{I} = \langle \mathbf{g} \rangle$, there are *no short elements* $\boldsymbol{c}$ in that coset such that $[\boldsymbol{c} \cdot \boldsymbol{h}/\mathbf{g}]_q$ is small.

Lemma 3. *Let $\boldsymbol{w} = [\boldsymbol{c} \cdot \boldsymbol{h}/\mathbf{g}]_q$ and suppose $\|\mathbf{g} \cdot \boldsymbol{w}\|$ and $\|\boldsymbol{c} \cdot \boldsymbol{h}\|$ are each at most $q/2$. Suppose $\langle \mathbf{g} \rangle$ is a prime ideal. Then, either $\mathbf{c}$ or $\boldsymbol{h}$ is in the ideal $\langle \mathbf{g} \rangle$.*

Proof. Since $\mathbf{g} \cdot \boldsymbol{w} = \boldsymbol{c} \cdot \boldsymbol{h} \bmod q$, and since $\|\mathbf{g} \cdot \boldsymbol{w}\|$ and $\|\boldsymbol{c} \cdot \boldsymbol{h}\|$ are each at most $q/2$, we have $\mathbf{g} \cdot \boldsymbol{w} = \boldsymbol{c} \cdot \boldsymbol{h}$ exactly. We also have an equality of ideals $\langle \mathbf{g} \rangle \cdot \langle \boldsymbol{w} \rangle = \langle \boldsymbol{c} \rangle \cdot \langle \boldsymbol{h} \rangle$, and, since $\langle \mathbf{g} \rangle$ is a prime ideal and our cyclotomic ring is a unique factorization domain, we have that $\langle \mathbf{g} \rangle$ divides either $\langle \boldsymbol{c} \rangle$ or $\langle \boldsymbol{h} \rangle$ (or both). The result follows. □

Lemma 4. *Let n, q, σ be as in our parameter setting, suppose $q = n^{\omega(1)}$, and consider drawing $\mathbf{g} \leftarrow D_{\mathbb{Z}^n, \sigma'}$ subject to $\langle \mathbf{g} \rangle$ being prime and $\boldsymbol{h} \leftarrow D_{\mathbb{Z}^n, \sqrt{q}}$ not being in $\langle \mathbf{g} \rangle$. Then, there is no $\epsilon > 0$ and $\boldsymbol{c}$ in a nonzero coset of $\mathcal{I}$ such that $\|\boldsymbol{c}\| < q^{1/8}$ and $\|[\boldsymbol{c} \cdot \boldsymbol{h}/\mathbf{g}]_q\| < q^{1-\epsilon}$.*

Proof. This follows directly from Lemma 3, our parameter setting (with $\|\mathbf{g}\| = \mathsf{poly}(n)$) and the fact that in the coefficient embedding $\|\mathbf{a} \cdot \mathbf{b}\| \leq n \cdot \|\mathbf{a}\| \cdot \|\mathbf{b}\|$. □

Extraction: $s \leftarrow \mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_\kappa)$. To extract a "canonical" and "random" representation of a coset from an encoding $\boldsymbol{u} = [\boldsymbol{c}/\mathbf{z}^\kappa]_q$, we just multiply by the zero-testing parameter $\mathbf{p}_{zt}$, collect the $(\log q)/4 - \lambda$ most-significant bits of each of the n coefficients of the result, and apply a strong randomness extractor to the collected bits (using the seed from the public parameters). Namely

$$\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{u}) = \textsc{Extract}_s(\mathsf{msbs}([\boldsymbol{u} \cdot \mathbf{p}_{zt}]_q)) \quad (\mathsf{msbs} \text{ of coefficient representation}).$$

This works because for any two encodings $\boldsymbol{u}, \boldsymbol{u'}$ of the same coset we have

$$\|\mathbf{p}_{zt}\boldsymbol{u} - \mathbf{p}_{zt}\boldsymbol{u'}\| = \|\mathbf{p}_{zt}(\boldsymbol{u} - \boldsymbol{u'})\| < q^{3/4},$$

so we expect $\mathbf{p}_{zt}\boldsymbol{u}$, $\mathbf{p}_{zt}\boldsymbol{u'}$ to agree on their $(\log q)/4 - \lambda$ most significant bits. (There is a negligible (in λ) chance that $\boldsymbol{u}$ and $\boldsymbol{u'}$ are such that $\mathbf{p}_{zt}\boldsymbol{u}$ and $\mathbf{p}_{zt}\boldsymbol{u'}$ are on opposite sides of a boundary, such that they have different MSBs.) On the other hand, by Lemma 4, we know that we cannot have $\|\mathbf{p}_{zt}(\boldsymbol{u} - \boldsymbol{u'})\| < q^{1-\epsilon}$ when $\boldsymbol{u} - \boldsymbol{u'}$ encodes something nonzero, and therefore (since $\lambda \ll \log q/4$) the values $\mathbf{p}_{zt}\boldsymbol{u}$ and $\mathbf{p}_{zt}\boldsymbol{u'}$ cannot agree on their $(\log q)/4 - \lambda$ MSBs.

This means, however, that no two points in the basic cell of $\mathcal{I}$ agree on their collected bits when multiplied by $\mathbf{p}_{zt}$, so the collected bits from an encoding of a random coset have min-entropy at least $\log |R/\mathcal{I}|$. We can therefore use a strong randomness extractor to extract a nearly uniform bit-string of length (say) $\lfloor \log |R/\mathcal{I}| \rfloor - \lambda$.

4.2 Setting the parameters

The parameters for the basic setting above should be set so as to meet the following requirements:

- The basic Gaussian parameter σ that we use to draw the ideal generator, $\mathbf{g} \leftarrow D_{\mathbb{Z}^n, \sigma}$, should satisfy $\sigma \geq \eta_{2^{-\lambda}}(\mathbb{Z}^n)$, which means that we have $\sigma = \sqrt{\lambda n}$. This means that the size of $\mathbf{g}$ is bounded with overwhelming probability by $\|\mathbf{g}\| \leq \sigma\sqrt{n} = n\sqrt{\lambda}$.

- Once we have the ideal lattice $\mathcal{I} = \langle \mathbf{g} \rangle$, the Gaussian parameter σ' that we use for sampling level-zero elements must satisfy $\sigma' \geq \eta_{2^{-\lambda}}(\mathcal{I})$, namely we should have $\sigma' \geq \|\mathbf{g}\|\sqrt{\lambda n}$. Given the bound from above bound on the size of $\mathbf{g}$, it is sufficient to set $\sigma' = \lambda n^{3/2}$, which means that the size of level-zero elements is bounded with overwhelming probability by λn^2.

- It is shown in Section 6.4 that the size of the numerators in $\mathbf{y}$ and the $\mathbf{x}_i$'s is bounded by σn^4 whp.

- The Gaussian parameter σ^* that we use to draw the coefficient vector $\boldsymbol{r}$ during re-randomization of newly generated level-1 encodings, must be large enough so that the resulting distribution on $\sum r_i \mathbf{x}'_i$ "drowns" the initial vector $\mathbf{a}\boldsymbol{d}/\mathbf{z}$. Setting $\sigma^* = 2^\lambda$ is certainly enough for that purpose. For this value of σ^*, a re-randomized level-one encoding is of the form $[\boldsymbol{c}/\mathbf{z}]_q$ with the size of $\boldsymbol{c}$ is bounded by $\|\boldsymbol{c}\| \leq 2^\lambda \cdot \mathrm{poly}(n)$.

- A level-κ encoding is obtained by multiplying κ level-one encodings (which will always be re-randomized). Hence it is of the form $[\boldsymbol{c}/\mathbf{z}^\kappa]_q$ with $\boldsymbol{c}$ of size bounded whp by $\|\boldsymbol{c}\| \leq (2^\lambda \cdot \mathrm{poly}(n))^\kappa = 2^{\kappa\lambda} \cdot n^{O(\kappa)}$. To use Lemma 4 for level-κ encodings, we need $\|\boldsymbol{c}\| \leq q^{1/8}$, so it is sufficient to set $q \geq 2^{8\kappa\lambda} \cdot n^{O(\kappa)}$.

- To get λ-level security against lattice attacks, we roughly need to set the dimension n large enough so that $q < 2^{n/\lambda}$, which means that $n > \tilde{O}(\kappa\lambda^2)$.

- Finally, to use Theorem 1 we need m to be sufficiently larger than $n \log q$, which we can do here by setting $m = O(n^2)$.

4.3 Variants

Some applications of multi-linear maps require various modifications to the basic encoding scheme from above, such as "assymetric maps" that have difference source groups. We briefly describe some of these variants below.

Asymmetric encoding. In this variant we still choose just one ideal generator $\mathbf{g}$, but several different denominators $\mathbf{z}_j \xleftarrow{r} R_q$, $j = 1, \ldots, \tau$. Then, a vector of the form $\boldsymbol{c}/\mathbf{z}_j \in R_q$ with $\boldsymbol{c}$ short is a level-one encoding of the coset $\boldsymbol{c} + \mathcal{I}$ relative to the "j'th dimension". In this case we use vectors rather than integers to represent the different levels, where for an index $\boldsymbol{w} = \langle w_1, \ldots, w_\tau \rangle \in \mathbb{N}^\tau$ and a coset $\boldsymbol{c}' + \mathcal{I}$, the encodings of $\boldsymbol{c}' + \mathcal{I}$ relative to the index $\boldsymbol{w}$ are

$$S_{\boldsymbol{w}}^{(\boldsymbol{c}'+\mathcal{I})} \;=\; \left\{ \boldsymbol{c}/\boldsymbol{z}^* \;:\; \boldsymbol{c} \in \boldsymbol{c}' + \mathcal{I},\ \|\boldsymbol{c}\| < q^{1/8},\ \boldsymbol{z}^* = \prod_{i=1}^{\tau} \mathbf{z}_i^{w_i} \right\}.$$

To enable encoding in this asymmetric variant, we provide the public parameters $\mathbf{y}_j = [\mathbf{a}_j/\mathbf{z}_j]_q$ and $\{\mathbf{x}_{i,j} = [\mathbf{b}_{i,j}/\mathbf{z}_j]_q\}_i$ for all $j = 1, 2, \ldots, \kappa$, with short $\mathbf{a}_i \in 1 + \mathcal{I}$ and $\mathbf{b}_{i,j} \in \mathcal{I}$. To enable zero-test relative to index $\langle v_1, \ldots, v_\tau \rangle \in \mathbb{N}^\tau$ we provide the zero-test parameter $\mathbf{p}_{zt} = (\boldsymbol{h} \cdot \prod_{i=1}^{\tau} \mathbf{z}_i^{v_i})/\mathbf{g} \in R_q$. (The parameters should be set so as to provide functionality upto $\sum_i v_i$ levels.)

Providing zero-test security. In applications that require resilience of the zero test even against invalid encodings, we augment the zero-test parameter by publishing many elements $\mathbf{p}_{\mathrm{zt},i} = [\boldsymbol{h}_i \mathbf{z}^\kappa/\mathbf{g}]_q$ for several different $\boldsymbol{h}_i$'s. A level-κ encoding must pass the zero-test relative to *all* the parameters $\mathbf{p}_{\mathrm{zt},i}$.

Consider a purported encoding $\boldsymbol{u} = \boldsymbol{c}/\mathbf{z}^\kappa$ where in this case we do not assume necessarily that $\|\boldsymbol{c}\| < q^{1/8}$ (as would be true for a valid encoding). Applying multiple zero-testers, we obtain

$$\mathbf{p}_{\mathrm{zt},1}\boldsymbol{u} = \boldsymbol{h}_i\boldsymbol{c}/\mathbf{g}, \quad \ldots, \quad \mathbf{p}_{\mathrm{zt},t}\boldsymbol{u} = \boldsymbol{h}_t\boldsymbol{c}/\mathbf{g} \ .$$

This t-dimensional vector lies in a lattice L generated by the vector $(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_t)$ modulo q, Note that since $\|\boldsymbol{h}_i\| \ll q$ for all i, the vector $(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_t)$ is quite short modulo q. Moreover, by making t large enough (but still polynomial), we can ensure that *all* of the vectors in L whose lengths are much less than q are *unreduced* (small) multiples of $(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_t)$. Therefore, if the encoding passes the multiple zero-test, $\boldsymbol{c}/\mathbf{g}$ must be small, and therefore $\boldsymbol{u}$ has the form of an encoding of zero. (We omit the details in this preliminary report.)

Avoiding Principal Ideals. In our construction above we use a principal ideal $\mathcal{I} = \langle \mathbf{g} \rangle$, where the norm $\mathsf{N}(\mathbf{g}) = |R/\mathcal{I}|$ is prime, and both $\|\mathbf{g}\|$ and $\|1/\mathbf{g}\|$ are small. Heuristically, restricting Landau's prime ideal theorem [Lan03] to principal ideals with small generators, one could expect that $\mathsf{N}(\mathbf{g})$ is prime (when $\|\mathbf{g}\|$ is small) with probability about $1/\ln \mathsf{N}(\mathbf{g})$, and therefore it is efficient to find an element $\mathbf{g}$ satisfying the primeness requirement. This is indeed the approach Smart and Vercauteren [SV10] took for key generation in their variation of Gentry's fully homomorphic encryption scheme [Gen09]. When $\mathbf{g}$ is generated according to a discrete Gaussian with $\sigma = \mathsf{poly}(n)$, heuristically the requirement that $\|\mathbf{g}^{-1}\|$ be small will also be satisfied – in particular, $\|\mathbf{g}^{-1}\|$ will also be $O(\mathsf{poly}(n))$ except with negligible probability.

However, especially given that some of the attacks in Section 7 rely on the fact that $\mathcal{I}$ is a principal ideal, it makes sense to seek a scheme that can use also "generic" (non-principal) ideals according to a nice canonical distribution. Unfortunately, we do not know how to do this, since we do not know how to generate a general ideal $\mathcal{I}$ according to a nice distribution together with short vectors (e.g., within $\mathsf{poly}(n)$ of the first minima) in each of $\mathcal{I}$ and $\mathcal{I}^{-1}$.

We note that we can at least adapt the zero-test to general ideals, should the other problems be resolved. We can replace the single zero-test parameter $\mathbf{p}_{zt} = [\boldsymbol{h}\mathbf{z}^\kappa/\mathbf{g}]_q$ by n parameters, $\mathbf{p}_{\mathrm{zt},i} = [\boldsymbol{h}_i\mathbf{z}^\kappa \cdot \boldsymbol{f}_i]_q$, where the vectors $\boldsymbol{f}_i$ are "in spirit" just a small basis of the fractional ideal $\mathcal{I}^{-1}$ (but they are mapped to R_q via $\frac{1}{\boldsymbol{x}} \in \mathbb{K} \mapsto \boldsymbol{x}^{-1} \in R_q$). We note that a similar approach also addresses the (small) possibility that $\|\mathbf{g}^{-1}\|$ is not small. Since $\mathbf{g}^{-1} \subset R$, we can reduce $\mathbf{g}^{-1}$ modulo the integral basis of R to obtain short elements of $\mathcal{I}^{-1}$, and hence zero-testers that are sufficiently small.

4.4 Security of Our Constructions

The security of our graded encoding systems relies on new, perhaps unconventional assumptions, and at present it seems unlikely that they can be reduced to more established assumptions, such as learning-with-errors (LWE) [Reg05], or even the NTRU hardness assumption [HPS98]. Given that the construction of multilinear maps has been a central open problem now for over a decade, we feel that exploring unconventional assumptions for this purpose is well worth the effort, as long as this exploration is informed by extensive cryptanalysis.

We attempted an extensive cryptanalysis of our scheme, including some new extensions of tools from the literature that we devised in the course of this work. These attempts are described at length in Appendices 6 and 7. Here, we provide a brief overview.

Hardness of the CDH/DDH analogs. In our construction, the attacker sees the public parameters $\mathsf{params} = (\mathbf{y}, \{\mathbf{x}_i\}_{i=1}^m)$, where $\mathbf{y} = [\mathbf{a}/\mathbf{z}]_q$ is a level-1 encoding of $1+\mathcal{I}$ and each $\mathbf{x}_i = [\mathbf{b}_i/\mathbf{z}]_q$ is a level-1 encoding of $0+\mathcal{I}$. Recall that $\mathcal{I} = \langle \mathbf{g} \rangle$ where $\|\mathbf{g}\| = \text{poly}(n) = q^{o(1)}$, and a level-$i$ encoding of a coset $\alpha + \mathcal{I}$ is an element of the form $\boldsymbol{u} = [\boldsymbol{c}/\mathbf{z}^i]_q$ where $\boldsymbol{c} \in \alpha + \mathcal{I}$ is short, typically $\|\boldsymbol{c}\| = q^{o(1)}$ (and always $\|\boldsymbol{c}\| < q^{1/8}$). In addition the attacker also sees a zero-testing parameter at level κ of the form $\mathbf{p}_{zt} = [\boldsymbol{h}\mathbf{z}^\kappa/\mathbf{g}]_q$ with $\|\boldsymbol{h}\| = q^{1/2+o(1)}$. Consider the following process, on parameters $\lambda, n, q, \kappa, \sigma = \text{poly}(n), \sigma^* = \sigma \cdot 2^\lambda$ (as described in Section 4):

1. $(\mathbf{y}, \{\mathbf{x}_i\}_i, \mathbf{p}_{zt}) \leftarrow \mathsf{InstGen}(1^n, 1^\kappa)$
2. For $i = 0, \ldots, \kappa$
3. Choose $\boldsymbol{e}_i \leftarrow D_{\mathbb{Z}^n,\sigma}$ and $\boldsymbol{f}_i \leftarrow D_{\mathbb{Z}^n,\sigma}$ // $\boldsymbol{e}_i, \boldsymbol{f}_i$ in random $\eta_i + \mathcal{I}, \phi_i + \mathcal{I}$
4. Set $\boldsymbol{u}_i = \left[\boldsymbol{e}_i \mathbf{y} + \sum_j r_{ij}\mathbf{x}_j\right]_q$ where $r_{ij} \leftarrow D_{Z,\sigma^*}$ // encode only the η_i's
5. Set $\boldsymbol{u}^* = [\prod_{i=1}^\kappa \boldsymbol{u}_i]_q$ // level-κ encoding
6. Set $\boldsymbol{v} = [\boldsymbol{e}_0 \cdot \boldsymbol{u}^*]_q$ // encoding of the right product
7. Set $\boldsymbol{v}' = [\boldsymbol{f}_0 \cdot \boldsymbol{u}^*]_q$ // encoding of a random product

Definition 4 (GCDH/GDDH)**.** *The (graded) CDH problem (GCDH) is, on input* $((\mathbf{y}, \{\mathbf{x}_i\}_i, \mathbf{p}_{zt}), \boldsymbol{u}_0, \ldots, \boldsymbol{u}_\kappa)$ *to output a level-*κ *encoding of* $\prod_i \boldsymbol{e}_i + \mathcal{I}$*, specifically* $\boldsymbol{w} \in R_q$ *such that* $\|[\mathbf{p}_{zt}(\boldsymbol{v} - \boldsymbol{w})]_q\| < q^{3/4}$*.* [2] *The graded DDH problem (GDDH) is to distinguish between* $\boldsymbol{v}$ *and* $\boldsymbol{v}'$*, or more formally between the distributions*

$$\mathcal{D}_{GDDH} = \{(\mathbf{y}, \{\mathbf{x}_i\}_i, \mathbf{p}_{zt}), \boldsymbol{u}_0, \ldots, \boldsymbol{u}_\kappa, \boldsymbol{v}\} \quad \textit{and} \quad \mathcal{D}_{RAND} = \{(\mathbf{y}, \{\mathbf{x}_i\}_i, \mathbf{p}_{zt}), \boldsymbol{u}_0, \ldots, \boldsymbol{u}_\kappa, \boldsymbol{v}'\}.$$

We begin our cryptanalysis (in Section 6) with a "sanity check". In particular, we consider "simplistic" generic attacks that operate on the encodings of params and the problem instance using only simple operations – add, subtract, multiply, divide. That is, we model attackers as *arithmetic straight-line programs* (ASLPs). This model is analogous to the generic group model [Sho97b], which is often used as a "sanity check" in the analysis of group-based cryptosystems. As an example in our case, an ASLP can generate the element $\mathbf{p}_{zt}\mathbf{x}_i^\kappa$, which equals $\boldsymbol{h}\mathbf{g}^{\kappa-1}\mathbf{b}_i'^\kappa$ where $\mathbf{b}_i' = \mathbf{b}_i/\mathbf{g}$. We want to check that an ASLP cannot generate anything "dangerous".

We prove that an ASLP cannot solve GCDH. We do this by defining a weight function w for rational functions, such that everything in the GCDH instance has weight zero, but a GCDH solution has weight 1. The weight function behaves much like polynomial degree. For example, the term $[\mathbf{a}/\mathbf{z}]_q$ in params has weight 0, since we set $w(\mathbf{a}) = 1 = w(\mathbf{z})$. As another example, $w(\mathbf{p}_{zt}) = w(\boldsymbol{h}) + \kappa \cdot w(\mathbf{z}) - w(\mathbf{g})$, which equals 0, since we set $w(\mathbf{g}) = 1$ and $w(\mathbf{p}_{zt}) = 1 - \kappa$. To vastly oversimplify the remainder of our analysis, we show that, given terms of weight 0 (as in the GCDH instance), an ASLP attacker can only produce more terms of weight 0, and thus not a GCDH solution. (See Lemma 5 for a more accurate statement.)

More realistically, we consider (non-generic) averaging, algebraic and lattice attacks. Section 7 provides an extensive survey of these attacks, many of which arose in the cryptanalysis of NTRU signature schemes [HKL+00, HPS01, HHGP+03], but a couple of which are new (and will be of broader interest).

Averaging attacks – described in Sections 7.2 through 7.5 – allow us, after seeing many elements of the form $\boldsymbol{r}_i \cdot \boldsymbol{a}$ for the same $\boldsymbol{a}$ but many different "random" $\boldsymbol{r}_i$'s, to get a good approximation of $\boldsymbol{a}$ (or some related quantities from which we can derive $\boldsymbol{a}$). In our case, one might attempt

[2] This formulation allows the adversary to output even an *invalid encoding*, as long as it passes the equality check.

to mount such an averaging attack on the (possibly many) encodings of 0 $\{\mathbf{x}_i = \mathbf{b}_i'\mathbf{g}/\mathbf{z}\}$ that we provide in params. Fortunately, Gentry, Peikert and Vaikuntanathan [GPV08] already provide a countermeasure to this attack and similar "transcript" attacks. One of the key conceptual insights of [GPV08] is that a signer with *any* good basis B of a lattice L (e.g., a lattice where $\|B\|$ is less than some bound β) can generate signatures according to a *canonical* Gaussian distribution (with deviation tightly related to β). Thus, the signatures do not reveal anything about the signer's *particular* basis B aside from an upper bound on $\|B\|$. Our encoding systems (Section 4) use a similar approach, where we derive all the elements in the public parameters from a small set of elements, using a GPV-type procedure.

Not surprisingly, our constructions can be broken using a very good lattice reduction oracle (e.g., one that approximates SVP to within polynomial factors). For example, one attack begins with the fact (shown in Section 6.3.3) that an attacker *can* efficiently compute the *coset* $\boldsymbol{e_i} + \mathcal{I}$ used by any of the GDH participants. However, to convert this knowledge into a direct attack on GCDH – that is, to put himself into the position of one of the legitimate players – the attacker must find a *short* representative $\boldsymbol{e}$ of the coset $\boldsymbol{e_i} + \mathcal{I}$. It can do this with a good lattice reduction oracle, or with any short-enough element of $\mathcal{I}$, but it is unclear how such an attack could be implemented efficiently.

Our new algebraic/lattice attacks are extensions of an algorithm by Gentry and Szydlo [GS02], which combines lattice reduction and Fermat's Little Theorem in a clever way to solve a relative norm equation in a cyclotomic field. Our new attacks include a *dimension-halving attack* on principal ideal lattices (Section 7.8.1), demonstrating that one needs to double the dimension of principal ideal lattices (compared to general ideal lattices) to preserve security. Principal ideal lattices certainly arise in our scheme – in particular, it is straightforward from our params to generate a bases of principal ideals such as $\mathcal{I} = \langle \mathbf{g} \rangle$ – and therefore we need to set our parameters to be resilient to this new attack. Also, in Section 7.6, we provide a polynomial-time algorithm to solve the *closest principal ideal generator problem* in certain cases. Specifically, we can recover a generator of a principal ideal $\mathcal{I} = \langle \mathbf{g} \rangle$ from a basis of $\mathcal{I}$ and an ϵ-approximation of the generator $\mathbf{g}$, when $\epsilon \leq n^{-\Omega(\log \log n)}$ (slightly inverse quasi-polynomial). The latter attack emphasizes even more the necessity of preventing averaging attacks that could obtain such useful approximations. Undoubtedly there is a lot of meat here for cryptanalysts. But the bottom line is that we have extended the best known attacks and still not found an attack that is threatening to our constructions.

Easiness of other problems. In light of the apparent hardness of our CDH/DDH analog, we could optimistically hope to get also the analog of other hardness assumptions in bilinear maps, such as decision-linear, subgroup membership, etc. Unfortunately, these problems turn out to be easy in our setting, at least with the simple encoding methods from above.

To see why, observe that publishing level-1 encodings of 0 and 1 enables some "weak discrete log" computation at any level strictly smaller than κ. Specifically, consider one particular encoding of zero $\mathbf{x}_j = [\mathbf{b}_j/\mathbf{z}]_q$ (where $\mathbf{b}_j = \boldsymbol{c}_j\mathbf{g}$ for some $\boldsymbol{c}_j$), which is given in the public parameters together with an encoding of one $\mathbf{y} = [\mathbf{a}/\mathbf{z}]_q$ and the zero-testing parameter $\mathbf{p}_{zt} = [\boldsymbol{h}\mathbf{z}^\kappa/\mathbf{g}]_q$. Given a level-$i$

encoding with $1 \le i \lneq \kappa$, $\boldsymbol{u} = [\boldsymbol{d}/\mathbf{z}^i]_q$, we can multiply it by $\mathbf{x}_j$, $\mathbf{p}_{zt}$, and some power of $\mathbf{y}$ to get

$$\begin{aligned} \boldsymbol{f} &= [\boldsymbol{u} \cdot \mathbf{x}_j \cdot \mathbf{p}_{zt} \cdot \mathbf{y}^{\kappa-i-1}]_q = \left[\frac{\boldsymbol{d}}{\mathbf{z}^i} \cdot \frac{\boldsymbol{c}_j \cdot \mathbf{g}}{\mathbf{z}} \cdot \frac{\boldsymbol{h}\mathbf{z}^\kappa}{\mathbf{g}} \cdot \frac{\mathbf{a}^{\kappa-i-1}}{\mathbf{z}^{\kappa-i-1}}\right]_q \\ &= \underbrace{\boldsymbol{d} \cdot \boldsymbol{c}_j \cdot \boldsymbol{h} \cdot \mathbf{a}^{\kappa-i-1}}_{\ll q} = \boldsymbol{d} \cdot \underbrace{\boldsymbol{c}_j \cdot \boldsymbol{h}}_{\Delta_j} \pmod{\mathcal{I}}. \end{aligned}$$

We stress that the right-hand-side of the equality above is *not reduced modulo* q. This means that from a level-i encoding $\boldsymbol{u}$ of an element $\boldsymbol{d} + \mathcal{I}$, we can get a "plaintext version" of $\boldsymbol{d} \cdot \Delta_j$ from some fixed Δ_j (that depends only on the public parameters but not on $\boldsymbol{u}$). This "plaintext version" is not small enough to be a valid level-zero encoding (because Δ_j is roughly the size of $\boldsymbol{h}$, so in particular $\Delta_j > \sqrt{q}$). Nonetheless, we can still use it in attacks.

For starters, we can apply the above procedure to many of the level-one encodings of zero from the public parameters, thereby getting many elements in the ideal $\mathcal{I}$ itself. This by itself still does not yield a basis of $\mathcal{I}$ (since all these elements have the extra factor of h), but in Section 6.3.1we show how to remove this extra factor and nonetheless compute a basis for $\mathcal{I}$. This is not a small basis of course, but it tells us that we cannot hope to hide the plaintext space $R/\mathcal{I}$ itself.

Next, consider the subgroup membership setting, where we have $\mathbf{g} = \mathbf{g}_1 \cdot \mathbf{g}_2$, we are given a level-1 encoding $\boldsymbol{u} = [\boldsymbol{d}/\mathbf{z}]_q$ and need to decide if $\boldsymbol{d} \in \langle \mathbf{g}_1 \rangle$. Using the procedure above we can get $\boldsymbol{f} = \boldsymbol{d} \cdot \Delta_j$, which belongs to the ideal $\langle \mathbf{g}_1 \rangle$ if $\boldsymbol{d}$ does. Taking the GCD of the ideals $\langle \boldsymbol{f} \rangle$ and $\mathcal{I}$ will then give us the factor $\langle \mathbf{g}_1 \rangle$ with high probability. It follows that the subgroup membership problem is easy for the encoding method above.

Finally, consider getting a matrix of elements $A = (\boldsymbol{a}_{i,j})_{i,j}$, all encoded at some level $i \lneq \kappa$. Using the method above we can get a "plaintext version" of $\Delta_j \cdot M$, which has the same rank as A. Since the decision linear problem is essentially a matrix rank problem, this means that this problem too is easy for this encoding method.

At this point it is worth stressing again that these attacks do not seem to apply to the GDDH problem, specifically because in that problem we need to make a decision about a level-κ encoding, and the "weak discrete log" procedure from above only applies to encoding at levels strictly below κ. The attacks above make it clear that providing encodings of zero in the public parameters (in conjunction with the zero-testing parameter) gives significant power to the adversary. One interesting direction to counter these attacks is to find different randomization tools that can be applied even when we do not have these encodings of zero in the public parameters.

5 One-Round N-way Diffie-Hellman Key Exchange

Joux [Jou00] constructed the first one-round 3-way secret key exchange protocol using Weil and Tate pairings. Boneh and Silverberg [BS03] showed how this result could be extended to get a one-round N-way secret key exchange protocol if multi-linear maps existed. Our encoding schemes easily support the Boneh-Silverberg construction, with one subtle difference: Since our public parameters hide some secrets (i.e., the elements $\mathbf{g}, \boldsymbol{h}, \mathbf{z}$) then we get a one-round N-way secret key exchange protocol in the common reference string model.

5.1 Definitions

Consider a setting with N parties who wish to set up a shared key using a one-round protocol. The "one-round" refers to the setting in which each party is only allowed to broadcast one value to all other parties. Furthermore all N broadcasts occur simultaneously. Once all the N parties broadcast their values, each party should be able to locally compute a global shared secret s. Using the notation from [BS03], a one-round N-way key exchange scheme consists of the following three randomized PPT algorithms:

- $\mathsf{Setup}(\lambda, N)$: Takes a security parameter $\lambda \in \mathbb{Z}^+$ and the number of participants N. It runs in polynomial time in λ, N and outputs public parameters params.
- $\mathsf{Publish}(\mathsf{params}, i)$: Given an input $i \in \{1, \ldots, N\}$, the algorithm outputs a pair $(pub_i, priv_i)$, with both in $\{0,1\}^*$. Party i broadcasts pub_i to all other parties, and keeps $priv_i$ secret.
- $\mathsf{KeyGen}(\mathsf{params}, j, priv_j, \{pub_i\}_{i \neq j})$: Party $j \in \{1, \ldots N\}$ collects the public broadcasts sent by all other parties and executes KeyGen on all these public values and its secret value $priv_j$. On this execution the algorithm KeyGen outputs a key s_j.

The consistency requirement for the above scheme is that all N parties generate the same shared key whp. The scheme is secure if no polynomial time algorithm, given all N public values $(pub_1, \ldots pub_N)$, can distinguish the true shared key s from random.

5.2 Our Construction.

We present a one-round N-way key exchange protocol using an encoding schemes with $\kappa = (N-1)$, under the GDDH assumption. The construction is a straightforward adaptation of [BS03]:

$\mathsf{Setup}(1^\lambda, 1^N)$**.** We just run the $\mathsf{InstGen}$ algorithm of the underlying encoding scheme, getting $(\mathsf{params}, \mathbf{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^{N-1})$, and output $(\mathsf{params}, \mathbf{p}_{zt})$ as the public parameter. Note that $\mathbf{p}_{zt}$ is a level-$N-1$ zero-test parameter. Let q, n, σ be the corresponding parameters of the encoding scheme. Note also that in this construction we insist that the order of the quotient ring $R/\mathcal{I}$ be a large prime (or at least that it does not have any small divisors).

$\mathsf{Publish}(\mathsf{params}, \mathbf{p}_{zt}, i)$**.** Each party i chooses a random level-zero encoding $\boldsymbol{d} \leftarrow \mathsf{samp}(\mathsf{params})$ as a secret key, and publishes the corresponding level-one public key $\boldsymbol{w}_i \leftarrow \mathsf{enc}(\mathsf{params}, 1, \boldsymbol{d})$.

$\mathsf{KeyGen}(\mathsf{params}, \mathbf{p}_{zt}, j, \boldsymbol{d}_j, \{\boldsymbol{w}_i\}_{i \neq j})$**.** Each party j multiplies its secret key $\boldsymbol{d}_j$ by the public keys of all its peers, $\boldsymbol{v}_j \leftarrow \boldsymbol{d}_j \cdot \prod_{i \neq j} \boldsymbol{w}_i$, thus getting a level-$N-1$ encoding of the product coset $\prod_i \boldsymbol{d}_i + \mathcal{I}$. Then the party uses the extraction routine to compute the key, $s_j \leftarrow \mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{v}_j)$. (In out case extraction consists of multiplying by the zero-test parameter and outputting the high-order bits.)

The consistency requirement follows directly from the agreement property of the extraction procedure in the underlying encoding scheme: Notice that all the parties get valid encodings of the same uniformly-chosen coset, hence the extraction property implies that they should extract the same key whp.

Similarly, security follows directly from a combination of the GDDH assumption and the randomness property of the extraction property of the extraction procedure in the underlying encoding scheme.

Theorem 2. *The protocol described above is a one-round N-way Diffie-Hellman Key Exchange protocol if the GDDH assumption holds for the underlying encoding scheme.*

Proof. We need to show that an attacker that sees all the public keys cannot distinguish the output of the first party (say) from a uniformly random string. By GDDH, the adversary cannot distinguish between the level-$(N-1)$ encoding $\boldsymbol{v}_1 \leftarrow \boldsymbol{d}_1 \cdot \prod_{i>1} \boldsymbol{w}_i$ that Party 1 compute and an element $\boldsymbol{v}'_1 \leftarrow \boldsymbol{d}'_1 \cdot \prod_{i>1} \boldsymbol{w}_i$ that is obtained for a random and independent $\boldsymbol{d}'_1 \leftarrow \mathsf{samp}(\mathsf{params})$ (which is a level-$N-1$ encoding of the coset $(\boldsymbol{d}'_1 \cdot \prod_{i>1} \boldsymbol{d}_i) + \mathcal{I}$).

By the randomness property of the sampling procedure, $\boldsymbol{d}'_1$ is nearly uniformly distributed among the cosets of $\mathcal{I}$. Since $|R/\mathcal{I}|$ is a large prime then whp $\prod_{i>1} \boldsymbol{d}_i \not\equiv 0 \pmod{\mathcal{I}}$, and thus $\boldsymbol{d}'_1 \cdot \prod_{i>1} \boldsymbol{d}_i$ is also nearly uniformly distributed among the cosets of $\mathcal{I}$. We can now use the randomness property of the extraction function to conclude that $\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{v}'_1)$ is a nearly uniform string, completing the proof. □

6 Cryptanalysis

In this section we describe our attempts at cryptanalysis of our encoding schemes, and propose plausible countermeasures against the more "promising" lines of possible attacks. Despite significant effort, we do not have a polynomial-time attack even against a simplistic scheme that does not use any of these countermeasures. But the best (heuristic) attacks that we have come very close, they may be able to break the simplistic scheme in slightly super-polynomial $n^{O(\log \log n)}$ time. We stress, however, that we do not have such attacks against the "main scheme" as described in Section 4.1.

In this section we first "cover the basics," arguing that simplistic attacks that only compute rational functions in the system parameters cannot recover any "interesting quantities", and in particular cannot break our DDH analog. (This is somewhat analogous to generic-group-model analysis for pairing groups.) Then we move to more sophisticated settings, identifying seemingly useful quantities that can be computed from the public parameters, and other quantities that if we could compute them then we could break the scheme. We describe averaging and lattice-reduction attacks that can perhaps be useful in recovering some of these "interesting targets," and propose countermeasures to render these attacks less dangerous.

A survey of the cryptanalysis tools that we use (including some new tools that we developed in the course of this work) can be found in Section 7.

6.1 Cryptanalytic Landscape for Our Constructions

In our constructions, the attacker sees the public parameters $\mathsf{params} = (\mathbf{y}, \{\mathbf{x}_i\}_i)$, where $\mathbf{y} = [\mathbf{a}/\mathbf{z}]_q$ is a level-1 encoding of $1 + \mathcal{I}$ and each $\mathbf{x}_i = [\mathbf{b}_i/\mathbf{z}]_q$ is a level-1 encoding of $0 + \mathcal{I}$. Recall that $\mathcal{I} = \langle \mathbf{g} \rangle$ where $\|\mathbf{g}\| = \mathrm{poly}(n) = q^{o(1)}$, and a level-$i$ encoding of a coset $\alpha + \mathcal{I}$ is an element of the form $\boldsymbol{u} = [\boldsymbol{c}/\mathbf{z}^i]_q$ where $\boldsymbol{c} \in \alpha + \mathcal{I}$ is short, typically $\|\boldsymbol{c}\| = q^{o(1)}$ (and always $\|\boldsymbol{c}\| < q^{1/8}$). In addition the attacker also sees a zero-testing parameter at level κ of the form $\mathbf{p}_{zt} = [\boldsymbol{h}\mathbf{z}^\kappa/\mathbf{g}]_q$ with $\|\boldsymbol{h}\| = q^{1/2+o(1)}$. Expressing the abstract GDDH assumption from Section 2 in terms of our specific construction, we get the following computational assumptions (below we state both the search and the decision versions). Consider the following process, on parameters $\lambda, n, q, \kappa, \sigma = \mathrm{poly}(n), \sigma^* = \sigma \cdot 2^\lambda$ (as described in Section 4):

1. $(\mathbf{y}, \{\mathbf{x}_i\}_i, \mathbf{p}_{zt}) \leftarrow \mathsf{InstGen}(1^n, 1^\kappa)$
2. For $i = 0, \ldots, \kappa$
3. Choose $\boldsymbol{e}_i \leftarrow D_{\mathbb{Z}^n,\sigma}$ and $\boldsymbol{f}_i \leftarrow D_{\mathbb{Z}^n,\sigma}$ // random $\eta_i + \mathcal{I}, \phi_i + \mathcal{I}$
4. Set $\boldsymbol{u}_i \leftarrow \left[\boldsymbol{e}_i \mathbf{y} + \sum_j r_{ij}\mathbf{x}_j\right]_q$ where $r_{ij} \leftarrow D_{Z,\sigma^*}$ // encode only the α_i's
5. Set $\boldsymbol{u}^* = [\prod_{i=1}^{\kappa} \boldsymbol{u}_i]_q$ // level-κ encoding
6. Set $\boldsymbol{v} = [\boldsymbol{e}_0 \cdot \boldsymbol{u}^*]_q$ // encoding of the right product
7. Set $\boldsymbol{v}' = [\boldsymbol{f}_0 \cdot \boldsymbol{u}^*]_q$ // encoding of a random product

Definition 5 (GCDH/GDDH)**.** *The graded CDH problem (GCDH) is, on input* $((\mathbf{y}, \{\mathbf{x}_i\}_i, \mathbf{p}_{zt}), \boldsymbol{u}_0, \ldots, \boldsymbol{u}_\kappa)$ *to output a level-*κ *encoding of* $\prod_i \boldsymbol{e}_i + \mathcal{I}$*, specifically* $\boldsymbol{w} \in R_q$ *such that* $\|[\mathbf{p}_{zt}(\boldsymbol{v} - \boldsymbol{w})]_q\| < q^{3/4}$*.* [3] *The graded DDH problem (GDDH) is to distinguish between* $\boldsymbol{v}$ *and* $\boldsymbol{v}'$*, or more formally between the distributions*

$$\mathcal{D}_{GDDH} = \{(\mathbf{y}, \{\mathbf{x}_i\}_i, \mathbf{p}_{zt}), \boldsymbol{u}_0, \ldots, \boldsymbol{u}_\kappa, \boldsymbol{v}\} \quad \textit{and} \quad \mathcal{D}_{RAND} = \{(\mathbf{y}, \{\mathbf{x}_i\}_i, \mathbf{p}_{zt}), \boldsymbol{u}_0, \ldots, \boldsymbol{u}_\kappa, \boldsymbol{v}'\}.$$

6.2 Simplistic Models of Attacks

We begin our cryptanalysis effort by considering "simplistic" generic attacks. Roughly, these are attacks in which we just take the terms the public parameters, add, subtract, multiply, and divide them, and hope to get something useful out of it. In other words, we consider *arithmetic straight-line programs* (ASLP) over the ring R_q as our model of attack.

We argue that such simplistic attacks are inherently incapable of solving GCDH. To that end we consider the different terms from the public parameters as formal variables, and show that all of the rational functions that the attacker can derive have a special form. Then we argue that any term of this form that expresses a solution to GCDH must refer to polynomials of size larger than q, hence it cannot be a correct solution.

Before presenting this analysis, we remark that a slightly less simplistic attack model is the *black-box field* (BBF) model of Boneh and Lipton [BL96]. In that model, the attacker can still compute terms that are rational functions in the given parameters, but now it can also test whether two terms are equal (and in our case perhaps also see the results of applying the zero test on two terms). Although we do not have any bounds on the security of our scheme in this model, we note that Boneh and Lipton's generic BBF algorithm for solving discrete log does not extend to our setting to solve our "discrete log" problem. The reason is that their algorithm requires black-box exponentiations of high (exponential) degree, whereas our encodings only permit the evaluation of polynomially-bounded degree, after which the "noise" in our encodings overwhelms the signal.

6.2.1 Hardness of GCDH in the Arithmetic Straight-Line Program Model

Our ASLP analysis resembles potential-function analysis to some extent: We assign some *weight* to terms from the public parameters and the GCDH instance that the attacker gets as input (and think of this weight as our "potential"). We then characterize the weight of the terms that the attacker can compute using an ASLP on these input terms, and argue that terms of this weight are not useful for solving GCDH.

First, we establish some terminology. Recall that a rational function is a ratio of two (multivariate) polynomials, and that the set of rational functions in some variables is closed under

[3] This formulation allows the adversary to output even an *invalid encoding*, as long as it passes the equality check.

addition, subtraction, multiplication and division. We denote the rational functions over the set of variables V over a ring R by $\mathcal{R}_R(V)$.

Definition 6 (Weight of Variables and Rational Functions)**.** *Consider a set of variables $V = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t\}$ over some ring R, and a* weight function *on these variables $w : V \to \mathbb{Z}$. This weight function is inductively extended rational functions in these variables over R, $w^* : \mathcal{R}_R(V) \to \mathbb{Z}$ as follows:*

- *For any constant $c \in R$, $w^*(c) = 0$, and for any variable $\boldsymbol{x} \in V$ $w^*(\boldsymbol{x}) = w(\boldsymbol{x})$;*
- *$\forall a \in \mathcal{R}_R(V)$, $w^*(-a) = w^*(a)$ and if $a \not\equiv 0$ then $w^*(1/a) = -w^*(a)$;*
- *$\forall a, b \in \mathcal{R}_R(V)$, s.t. $a+b$ is not equivalent to any simpler function, $w^*(a+b) = \max\{w^*(a), w^*(b)\}$.*
- *$\forall a, b \in \mathcal{R}_R(V)$, s.t. ab is not equivalent to any simpler function, $w^*(ab) = w^*(a) + w^*(b)$.*

Using the fundamental theorem of algebra, it can be shown that this extension w^* is well defined over the ring of integers in any number field. One example of such a weight function is the degree, $w(\boldsymbol{x}) = 1$ for all $\boldsymbol{x} \in V$. Below we identify w^* with w and denote both by $w(\cdot)$.

Definition 7 (Homogeneous Weight-Balanced Rational Function)**.** *We say that a rational function $r(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t) = p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)/q(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)$ is weight-homogeneous if p and q are both homogeneous in the sense that all of their monomials have the same weight. We say that r is (homogeneous)* weight-balanced *for weight function $w(\cdot)$ if it is weight-homogeneous and has weight zero.*

We use the following easy fact:

Fact 1. *Let $r_1(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)$ and $r_2(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)$ be balanced rational functions for weight function $w(\cdot)$. Then $-r_1$, $1/r_1$, $r_1 + r_2$ and $r_1 \cdot r_2$ are all balanced rational functions for weight function $w(\cdot)$.*

Using the above definitions, our basic strategy will be to treat the relevant elements in out scheme as formal variables and assign a weight to them such that all the terms that the adversary sees are weight-balanced rational functions. Fact 1 then implies that the terms that an ASLP attacker can produce must also be weight-balanced rational function. In addition every element is also assigned a *size* (lower-bounding the expected size of that element in the actual scheme). The main lemma in our analysis asserts that any element obtained as weight-balanced rational function, which is equivalent to $\prod_{i=0}^{\kappa} \boldsymbol{e}_i/\mathbf{z}^{\kappa} \pmod{\mathcal{I}}$, must have numerator of size more than $\sqrt{q}$. This means that when multiplied by the zero-testing parameter we get reduction modulo q, hence such term will not pass the equality test.

Size of terms. Below we use the following rules for the evolution of the size: If a, b are an elements of size $\mathsf{sz}(a), \mathsf{sz}(b)$, respectively, then we have $\mathsf{sz}(-a) = \mathsf{sz}(a)$, $\mathsf{sz}(1/a) = q$, $\mathsf{sz}(a+b) = \mathsf{sz}(a) + \mathsf{sz}(b)$ and $\mathsf{sz}(ab) = \mathsf{sz}(a) \cdot \mathsf{sz}(b)$. (The convention of $\mathsf{sz}(1/a) = q$ captures the intuition that the inverse of a small R_q element has size roughly q.)

Weight and size of elements in our scheme. Recall that a GCDH attacker gets as input the terms $\mathbf{a}/\mathbf{z}, \{\mathbf{b}_i/\mathbf{z}\}_{i=1}^m, \boldsymbol{h}\mathbf{z}^\kappa/\mathbf{g}$, and $\{\boldsymbol{e}_j/\mathbf{z}\}_{j=0}^\kappa$ (all in R_q), where we have $\mathcal{I} = \langle \mathbf{g} \rangle$, $\mathbf{b}_i \in \mathcal{I}$ for all i and $\mathbf{a} - 1 \in \mathcal{I}$.

To ensure that all the terms that the attacker gets are weight-balanced rational functions, we set $w(\mathbf{z}) = w(\mathbf{g}) = w(\mathbf{a}) = 1$ and also $w(\mathbf{b}_i) = 1$ for all i and $w(\boldsymbol{e}_j) = 1$ for all j. Finally, to make the zero-test parameter weight-balanced we set $w(\boldsymbol{h}) = 1 - \kappa$. We note that $\boldsymbol{h}$ is the only element that has negative weight. (If we wish to consider the decomposition $\mathbf{b}_i = \boldsymbol{r}_i\mathbf{g}$, then $w(\boldsymbol{r}_i) = 0$, and similarly if we decompose $\mathbf{a} = \boldsymbol{r}\mathbf{g} + 1$ then $w(\boldsymbol{r}) = 0$.)

For our analysis below, it is sufficient to assign size 1 to all the "small" elements, size just over $\sqrt{q}$ to the mid-size element $\boldsymbol{h}$, and size q to the random element $\mathbf{z}$. Namely we have $\mathsf{sz}(\mathbf{z}) = q$, $\mathsf{sz}(\mathbf{g}) = \mathsf{sz}(\mathbf{a}) = 1$, $\mathsf{sz}(\mathbf{b}_i) = 1$ for all i, $\mathsf{sz}(\boldsymbol{e}_j) = 1$ for all j and $\mathsf{sz}(\boldsymbol{h}) = \sqrt{q}$.

Lemma 5. *Consider the GCDH instance $\Gamma = (\mathbf{a}/\mathbf{z}, \{\mathbf{b}_i/\mathbf{z}\}_{i=1}^m, \boldsymbol{h}\mathbf{z}^\kappa/\mathbf{g}, \{\boldsymbol{e}_j/\mathbf{z}\}_{j=0}^\kappa)$ with weights and sizes as above. Assume that q is a prime. Let $\mathcal{A}$ be an arithmetic straight-line program. If $\mathcal{A}(\Gamma) = \boldsymbol{c}/\mathbf{z}^k$ such that $[\boldsymbol{c}]_q \equiv \prod_{j=0}^\kappa \boldsymbol{e}_j \pmod{\mathcal{I}}$ then $\mathsf{sz}([\boldsymbol{c}]_q) > \sqrt{q}$.*

Proof. By Fact 1 and the weights of elements in Γ, $\mathcal{A}$ can produce only weight-balanced rational functions of the variables. Since $w(\mathbf{z}) = 1$, this implies $w(\mathbf{c})$ is κ. Going forward, the intuition is since $\prod_{j=0}^\kappa \boldsymbol{e}_j$ has weight $\kappa + 1$, the only way to get $\mathbf{c}$ to have the correct weight is to make it divisible by $\boldsymbol{h}$, since it is the only variable with negative weight. But this makes the size of $\mathbf{c}$ at least $\sqrt{q}$.

Formally we prove below that any homogeneous rational function $\boldsymbol{d}$ that satisfies $\boldsymbol{d} \equiv \boldsymbol{c} \pmod{q}$ and $\boldsymbol{d} \equiv \prod_{j=0}^\kappa \boldsymbol{e}_j \pmod{\mathcal{I}}$ much have size at least $\sqrt{q}$, so in particular this must hold for $[\boldsymbol{c}]_q$.

Since $\boldsymbol{c}$ and $\boldsymbol{d}$ are homogeneous and $\boldsymbol{d} \equiv \boldsymbol{c} \pmod{q}$, there exist two homogeneous rational functions $\boldsymbol{s}, \boldsymbol{s'}$ such that $\boldsymbol{c} = \boldsymbol{s}\boldsymbol{d} + \boldsymbol{s'}$ with $\boldsymbol{s} \equiv 1 \pmod{q}$ and $\boldsymbol{s'} \equiv 0 \pmod{q}$. Since $\boldsymbol{c}$ is homogeneous, $w(\boldsymbol{c}) = w(\boldsymbol{s}) + w(\boldsymbol{d}) = w(\boldsymbol{s'})$.

Similarly since $\boldsymbol{d} \equiv \prod_{j=0}^\kappa \boldsymbol{e}_j \pmod{\mathcal{I}}$ then we must have $\boldsymbol{d} = \boldsymbol{r}\prod_{j=0}^\kappa \boldsymbol{e}_j + \boldsymbol{r'}$ for homogeneous rational functions $\boldsymbol{r}, \boldsymbol{r'}$ that satisfy $\boldsymbol{r} \equiv 1 \pmod{\mathcal{I}}$ and $\boldsymbol{r'} \equiv 0 \pmod{\mathcal{I}}$, and again we have

$$w(\boldsymbol{d}) = w(\boldsymbol{r}) + \kappa + 1.$$

Putting the two inequalities together, we thus have $w(\boldsymbol{d}) = w(\boldsymbol{s}) + w(\boldsymbol{r}) + \kappa + 1$. At the same time, by Fact 1 we know that $\mathcal{A}$ can only produce weight-balanced rational terms, so $w(\boldsymbol{c}/\mathbf{z}^\kappa) = 0$. Therefore $w(\boldsymbol{c}) = w(\mathbf{z}^\kappa) = \kappa$, which implies that $w(\boldsymbol{s}) + w(\boldsymbol{r}) = -1$.

Considering the size of $\boldsymbol{d}$, we first note that if $\boldsymbol{d} = \boldsymbol{p}/\boldsymbol{p'}$ for a nontrivial denominator $\boldsymbol{p'}$ then $\mathsf{sz}(\boldsymbol{d}) \geq q$ and there is nothing more to prove. Below we therefore assume that the denominator $\boldsymbol{p'}$ is trivial, i.e. $\boldsymbol{d}$ is a simple polynomial. Since $\boldsymbol{d} = \boldsymbol{r}\prod_{j=0}^\kappa \boldsymbol{e}_j + \boldsymbol{r'}$, then also $\boldsymbol{r'}$ is a simple polynomial and the only terms that we can have in the denominator of $\boldsymbol{r}$ are the $\boldsymbol{e}_j$'s. But we know that $\boldsymbol{r} \equiv 1 \pmod{\mathcal{I}}$ so the same $\boldsymbol{e}_j$'s must be in its numerator, making $\boldsymbol{r}$ too a simple polynomial. We conclude that $\boldsymbol{r}, \boldsymbol{r'}$ must both be simple polynomials, and $\mathsf{sz}(\boldsymbol{d}) = \mathsf{sz}(\boldsymbol{r}) \cdot \mathsf{sz}(\prod_j \boldsymbol{e}_j) + \mathsf{sz}(\boldsymbol{r'})$.

Returning to the weight, we now have two cases to analyze: either $w(\boldsymbol{s}) < 0$ or $w(\boldsymbol{r}) < 0$. If $w(\boldsymbol{r}) < 0$, then since the only variable with non-positive weight in our scheme is $\boldsymbol{h}$, it must be that $\boldsymbol{h}$ divides $\boldsymbol{r}$. Hence we get $\mathsf{sz}(\boldsymbol{d}) > \mathsf{sz}(\boldsymbol{c}) \geq \mathsf{sz}(\boldsymbol{r}) \geq \mathsf{sz}(\boldsymbol{h}) \geq \sqrt{q}$.

Considering the other case $w(\boldsymbol{s}) < 0$, we note $\boldsymbol{s} \equiv 1 \pmod{q}$ but none of the terms in our system are equivalent to 1 modulo q. The only way to get a homogeneous rational function $\boldsymbol{s} \equiv 1 \pmod{q}$ is if $w(\mathbf{s})$ is divisible by $q - 1$. Since the weight of $\boldsymbol{s}$ is negative and divisible by $q - 1$, then in particular we have $w(\boldsymbol{s}) \leq -q + 1$. Therefore, $w(\mathbf{r}) \geq q - 2$. For Γ, weights, and sizes as defined above, clearly $\mathsf{sz}(\boldsymbol{r})$, and hence $\mathsf{sz}(\boldsymbol{d})$, exceeds $\sqrt{q}$. □

6.3 Cryptanalysis Beyond the Generic Models

Below we attempt "real cryptanalysis" of our scheme, using state of the art tools in algebraic cryptanalysis and lattice reduction. Throughout this section we consider in particular the GDDH assumption, hence we assume that the attacker is given the following inputs, all relative to the random element $\mathbf{z} \in R_q$ and the ideal $\mathcal{I} = \langle \mathbf{g} \rangle \subset R$, with $\|\mathbf{g}\| \leq \sigma' \sqrt{n}$.

- $\mathbf{y} = [\mathbf{a}/\mathbf{z}]_q$, a level-one encoding of 1, namely $\mathbf{a} \in 1 + \mathcal{I}$ and $\|\mathbf{a}\| < \sigma\sqrt{n}$.
- $\mathbf{x}_i = [\mathbf{b}_i/\mathbf{z}]_q$, m randomizing terms s.t. $\forall i,\ \mathbf{b}_i \in \mathcal{I}$ and $\|\mathbf{b}_i\| < \sigma\sqrt{n}$. Below it will be convenient to denote $\mathbf{b}_i = \mathbf{b}'_i \cdot \mathbf{g}$, where $\mathbf{b}'_i$ is of size similar to $\mathbf{b}_i$.
- $\mathbf{p}_{zt} = [\boldsymbol{h}\mathbf{z}^k/\mathbf{g}]_q$ the zero-test parameter with $\sqrt{q} < \|\boldsymbol{h}\| < \sqrt{qn}$;
- $\boldsymbol{u}_j = [\boldsymbol{e}_j/\mathbf{z}]_q$, $\kappa + 1$ level-one encodings of random elements where $\forall j,\ \|\boldsymbol{e}_j\| < 2^\lambda \sigma\sqrt{n}$;
- $\boldsymbol{w} = [\boldsymbol{c}/\mathbf{z}^k]_q$, the "challenge element" with allegedly $\|\boldsymbol{c}\| < q^{1/8}$ and $\boldsymbol{c} \equiv \prod_{j=0}^{\kappa} \boldsymbol{e}_i \pmod{\mathcal{I}}$.

Our parameter setting is $n = \tilde{O}(\kappa\lambda^2)$ and $q \approx 2^{n/\lambda}$. In the analysis below we consider as a "real break" any method that has a heuristically significant chance of distinguishing the challenge $\boldsymbol{w}$ from a level-κ encoding of a random element different from $\prod_j \boldsymbol{e}_j$.

6.3.1 Easily computable quantities

Using only algebraic transformations (with no need for lattice reduction), it is easy to compute from the given parameters also the following quantities:

- Taking different κ-products including some number $r \geq 1$ of the $\mathbf{x}_i$'s, some number $s \geq 0$ of the $\boldsymbol{u}_j$'s and some power of $\mathbf{y}$, and multiplying these products by the zero-test parameter $\mathbf{p}_{zt}$, we get many different elements of the form

$$\boldsymbol{v} = \left[\left(\prod_{k=1}^{r} \mathbf{x}_{i_k} \right) \cdot \left(\prod_{k=1}^{s} \boldsymbol{u}_{j_k} \right) \cdot \mathbf{y}^{\kappa - r - s} \cdot \mathbf{p}_{zt} \right]_q = \left(\prod_{k=1}^{r} \mathbf{b}'_{i_k} \right) \cdot \mathbf{g}^{r-1} \cdot \left(\prod_{k=1}^{s} \boldsymbol{e}_{j_k} \right) \cdot \mathbf{a}^{\kappa - r - s} \cdot \boldsymbol{h} \quad (3)$$

  Importantly, the right-hand-side in Equation (3) is *not reduced modulo* q, because it is a product of the mid-size $\boldsymbol{h}$ by exactly κ short elements, hence its size is smaller than q.

- All the $\boldsymbol{v}$'s of the form of Equation (3) have a common factor $\boldsymbol{h}$, but if we choose the other elements at random then whp they will have no other common factors. Hence after seeing enough of them we can expect to get a basis for the principal ideal lattice $\langle \boldsymbol{h} \rangle$.

  A similar argument implies that we can also compute bases for the principal ideals $\langle \boldsymbol{h} \cdot \boldsymbol{e}_j \rangle$ for every $j \in \{0, 1, \ldots, \kappa\}$ and also bases for $\langle \boldsymbol{h} \cdot \mathbf{g} \rangle$ and $\langle \boldsymbol{h} \cdot \mathbf{a} \rangle$.

- Given a basis for $\langle \boldsymbol{h} \rangle$, we can get a basis for the fractional principal ideal $\langle 1/\boldsymbol{h} \rangle$ (where $1/\boldsymbol{h}$ is the inverse of $\boldsymbol{h}$ in the number field $\mathbb{K} = Q[X]/F(X)$).

- Using the bases for $\langle \boldsymbol{h} \cdot \mathbf{g} \rangle$ and $\langle 1/\boldsymbol{h} \rangle$, we can compute a basis for our principal ideal $\mathcal{I} = \langle \mathbf{g} \rangle$. Similarly we can also compute a basis for $\langle \mathbf{a} \rangle$ and bases for all the principal ideals $\langle \boldsymbol{e}_j \rangle$.

The above tells us that we cannot expect to hide the ideal $\mathcal{I}$ itself, or the ideals generated by any of the other important elements in our scheme. It may still be hard, however, to find the short generators for these ideals, or any short elements in them. Indeed this difficulty is the sole reason for the conjectured security of our schemes.

6.3.2 Using averaging attacks

Averaging attacks are described in Sections 7.2 through 7.5, roughly speaking they allow us, after seeing many elements of the form $\boldsymbol{r}_i \cdot \boldsymbol{a}$ for the same $\boldsymbol{a}$ but many different "random" $\boldsymbol{r}_i$'s (e.g., that are independent of $\boldsymbol{a}$), to get a good approximation of $\boldsymbol{a}$ (or some related quantities from which we can derive $\boldsymbol{a}$).

In our case, if we use simplistic Gaussian distributions to choose all our public parameters, then we expect to be able to apply these tools with elements from Equation (3), in order to get approximations for $\boldsymbol{h}$ or $\boldsymbol{h} \cdot \mathbf{g}^r$ for various r's. The tools from the literature do not quite work "right out of the box" because the terms that we want to recover are not very short. Specifically they have size more than $\sqrt{q}$, so techniques from the literature may need to average super-polynomial (or even exponential) number of samples to get useful approximations.

In Section 7.6, however, we describe a new method that can recover elements such as $\boldsymbol{h}$ or $\boldsymbol{h} \cdot \mathbf{g}^r$ from approximations that are not very accurate. The level of accuracy needed to apply Theorem 6 still requires super-polynomial number of samples, but only just: It is heuristically enough to use only $n^{O(\log \log n)}$ samples. Indeed this potential attack is the reason for the slightly involved method of choosing the randomizers in Section 4.1, which is based on the countermeasures discussed in Section 6.4 below.

We mention that another potential problem is that our public parameters only include a small number of terms, whereas averaging attacks typically need a much larger number of samples. However, the attacker can get many more samples by taking sums and products of terms from the public parameters, and it seems likely that such samples will be "independent enough" to serve in the averaging attacks.

Below we show how recovering (small multiples of) the terms $\mathbf{g}$ or $1/\boldsymbol{h}$, can be used to break our scheme, and also a plausible method of using a small multiple of $\boldsymbol{h} \cdot \mathbf{g}^r$ for a large value of r. We remark that for the cases of having a small multiple of $\mathbf{g}$ or $1/\boldsymbol{h}$ we can show a real working attack, but for the case of having a small multiple of $\boldsymbol{h} \cdot \mathbf{g}^r$ we only have a "somewhat plausible approach" that does not seem to lead to a real attack.

6.3.3 Cryptanalysis with extra help

A short element in $\langle \mathbf{g} \rangle$. We begin by showing that knowing any short element in the ideal $\mathcal{I} = \langle \mathbf{g} \rangle$ would enable the attacker to break our scheme. Any short element in $\mathcal{I}$ has the form $\boldsymbol{d} \cdot \boldsymbol{g}$ for a short $\boldsymbol{d}$ (because $1/\mathbf{g} \in \mathbb{K}$ is short). We begin the attack by multiplying in R_q the short $\boldsymbol{d} \cdot \boldsymbol{g}$ by the zero-test parameter $\mathbf{p}_{zt}$, thus getting the modified zero-test parameter $\mathbf{p}'_{zt} = [\boldsymbol{d} \cdot \boldsymbol{h} \cdot \mathbf{z}^\kappa]_q$. Then we multiply the modified zero-test parameter by both the "challenge element" $\boldsymbol{w}$ and by the product of κ of the random encodings $\boldsymbol{u}_j$.

In the case where $\boldsymbol{w}$ is indeed an encoding of the right product, we would have $\boldsymbol{w} = (\boldsymbol{c}\mathbf{g} + \prod_{j=0}^{\kappa} \boldsymbol{e}_i)/\mathbf{z}^\kappa$ for some not-too-big $\boldsymbol{c}$ (i.e., $\|\boldsymbol{c}\| < q^{1/8}$). Hence in this case we would get the two

elements

$$\boldsymbol{v}_1 \;:=\; [\mathbf{p}'_{zt}\cdot\boldsymbol{w}]_q \;=\; \boldsymbol{d}\cdot\boldsymbol{h}\cdot\left(\boldsymbol{c}\cdot\mathbf{g}+\prod_{j=0}^{\kappa}\boldsymbol{e}_j\right) \quad\text{and}\quad \boldsymbol{v}_2 \;:=\; \left[\mathbf{p}'_{zt}\cdot\prod_{j=1}^{\kappa}\boldsymbol{u}_j\right]_q \;=\; \boldsymbol{d}\cdot\boldsymbol{h}\cdot\prod_{j=1}^{\kappa}\boldsymbol{e}_j.$$

Our next goal is to "divide $\boldsymbol{v}_1$ by $\boldsymbol{v}_2$ modulo $\mathcal{I}$" in order to isolate the element $\boldsymbol{e}_0$. For that purpose, we use our knowledge of a basis of $\mathcal{I}$ and compute the Hermite normal form (HNF) of that lattice. Recall that the HNF basis has the form of a upper-triangular matrix, and with good probability the first entry on the main diagonal is the norm of $\mathcal{I}$ and all the other entries are 1. Below we assume that this is indeed the case, and denote the norm of $\mathcal{I}$ by $N(\mathcal{I})$.

We can reduce both $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ modulo the HNF basis of $\mathcal{I}$, and if the basis has the above special form then we get two integers $\nu_1 = [\boldsymbol{v}_1]_{\mathrm{HNF}(\mathcal{I})} \in \mathbb{Z}$ and $\nu_1 = [\boldsymbol{v}_1]_{\mathrm{HNF}(\mathcal{I})} \in \mathbb{Z}$. Clearly we have

$$\nu_1 \equiv \boldsymbol{v}_1 \equiv \boldsymbol{dh}\prod_{j=0}^{\kappa}\boldsymbol{e}_j \pmod{\mathcal{I}}, \quad\text{and}\quad \nu_2 \equiv \boldsymbol{v}_2 \equiv \boldsymbol{dh}\prod_{j=1}^{\kappa}\boldsymbol{e}_j \pmod{\mathcal{I}}$$

Assuming that ν_2 is co-prime to $N(\mathcal{I})$, we can now compute over the integers $\eta = \nu_1\cdot\nu_2^{-1} \bmod N(\mathcal{I})$. Observing that we always have $N(\mathcal{I}) \in \mathcal{I}$, we therefore get (for some $\tau \in \mathbb{Z}$)

$$\eta\cdot\nu_2 = \nu_1 + \tau\cdot N(\mathcal{I}) \equiv \nu_1 \pmod{\mathcal{I}}.$$

At the same time we also have

$$\boldsymbol{e}_0\cdot\nu_2 \equiv \boldsymbol{e}_0\cdot\boldsymbol{v}_2 \equiv \boldsymbol{v}_1 \equiv \nu_1 \pmod{\mathcal{I}}.$$

Since μ_2 is co-prime with $N(\mathcal{I})$ then it is also co-prime with the ideal generator $\mathbf{g}$, and hence the two equalities above imply that $\eta \equiv \boldsymbol{e}_0 \pmod{\mathcal{I}}$.

Finally, we can reduce η modulo the rotation basis of $\boldsymbol{d}\cdot\mathbf{g}$, which is a basis consisting of only short vectors (because $\boldsymbol{d}\cdot\mathbf{g}$ itself is short). This yields a short element $\boldsymbol{e}'_0 = \eta + \boldsymbol{t}\cdot\boldsymbol{d}\mathbf{g} \equiv \eta \equiv \boldsymbol{e}_0 \pmod{\mathcal{I}}$. We observe that the short $\boldsymbol{e}'_0$ is functionally equivalent to the secret key $\boldsymbol{e}$ which was encoded in $\boldsymbol{u}_0$. (At least, it is functionally equivalent when $\boldsymbol{d}\cdot\mathbf{g}$ is short enough; if it is not short enough, the attack may fail.)

In particular we can use it to verify that the challenge element is indeed an encoding of the right product: we just multiply $\boldsymbol{u}'_0 = \boldsymbol{e}'_0\cdot\mathbf{y}$ to get a level-one encoding, then check that $\boldsymbol{u}_0 - \boldsymbol{u}'_0$ is a level-one encoding of zero. (Or course this test will fail in the random case, since the element that we recover will be in the coset of $\boldsymbol{f}_0$ not in the coset of $\boldsymbol{e}_0$.)

A small multiple of $1/\boldsymbol{h}$. Recall that we can compute from the public parameters a basis for the fractional ideal $\langle 1/\boldsymbol{h}\rangle$. If we could find a "somewhat short" element in that lattice, namely an element $\boldsymbol{v} = \boldsymbol{d}/\boldsymbol{h}$ with $\|\boldsymbol{d}\| \ll \sqrt{q}$, then we can mount the following simple attack:

Multiplying the zero-test parameter by $\boldsymbol{v}$, we get the "higher-quality" zero-test parameter $\mathbf{p}'_{zt} = [\mathbf{p}_{zt}\cdot\boldsymbol{v}]_q = [\boldsymbol{d}\mathbf{z}^\kappa/\mathbf{g}]$. Once we have this higher-quality parameter, we can square it and multiply by one of the randomizers to get

$$\mathbf{p}''_{zt} = [(\mathbf{p}'_{zt})^2\mathbf{x}_0]_q = [\boldsymbol{d}^2\mathbf{z}^{2\kappa}/\mathbf{g}^2\cdot\mathbf{b}'_0\mathbf{g}]_q = [\boldsymbol{d}^2\mathbf{b}'_0\mathbf{z}^{2\kappa}/\mathbf{g}]_q.$$

If $\|\boldsymbol{d}\|$ is sufficiently short so that $\|\boldsymbol{d}^2\mathbf{b}'_0\| \ll q$, then we can use $\mathbf{p}''_{zt}$ as a zero-test parameter at level 2κ. In particular we can distinguish whether the challenge element is an encoding of the right product or a random product by computing the level-$(\kappa+1)$ encoding of the product $\prod_{j=0}^{\kappa} \boldsymbol{u}_j$, mapping $\boldsymbol{w}$ to level $\kappa+1$ by multiplying with $\mathbf{y}$, then use the level-2κ zero-test parameter $\mathbf{p}''_{zt}$ to check if these two elements are indeed in the same coset.

A small multiple of $\boldsymbol{h}\mathbf{g}^r$. If we could compute an element $\boldsymbol{h}\mathbf{g}^r$ (for a large value of r) or a not-too-big multiple of it, say $\boldsymbol{v} = \boldsymbol{d}\boldsymbol{h}\mathbf{g}^r$ such that $\|\boldsymbol{v}\| < q$, then the following line of attack becomes plausible.

Extracting the $\boldsymbol{r}$'th root of $\boldsymbol{v}$ we get $\boldsymbol{v}' = \sqrt[r]{\boldsymbol{d}\boldsymbol{h}} \cdot \mathbf{g}$. We note that when $\boldsymbol{d}\boldsymbol{h}$ is "random and independent of $\mathbf{g}^r$", then $\sqrt[r]{\boldsymbol{d}\boldsymbol{h}}$ (over the number-field $\mathbb{K}$) tends to a (known) constant as r increases.[4] We can therefore hope that for a large enough value of r the fractional element $\sqrt[r]{\boldsymbol{v}}$ will provide a good enough approximation of $\mathbf{g}$, and then we could perhaps use an algorithm such as the one from Section 7.6 to recover $\mathbf{g}$ exactly.

It seems, however, that this line of attack as described does not work in our case. The reason is that we cannot hope to get approximations of $\boldsymbol{h}\mathbf{g}^r$ for $r \geq \kappa - 1$, and our dimension n is always much larger than κ, so this method inherently cannot produce good enough approximations. Still perhaps it can be used in conjunction with other tools.

6.4 Some Countermeasures

As explained above, the most potent attacks that we found against our scheme make use of averaging attacks, using samples that we get by multiplying the zero-test parameter by products of κ other elements from the public parameters. (See Section 7.2 and 7.4 for details on averaging attacks, and a discussion of how devastating they are.) We note that for the purpose of defending against averaging attacks we can ignore the GDDH instance, since it can be generated by the attacker itself just from the public parameters. (At least as long as the averaging part does not use the challenge element $\boldsymbol{w}$.)

Fortunately, Gentry, Peikert and Vaikuntanathan (GPV) [GPV08] have already given us an approach to defeat this sort of averaging attack. One of the key conceptual insights of [GPV08] is that a signer with *any* good basis B of a lattice L (e.g., a lattice where $\|B\|$ is less than some bound β) can generate signatures according to a *canonical* Gaussian distribution (with deviation tightly related to β). Thus, the signatures do not reveal anything about the signer's *particular* basis B aside from an upper bound on $\|B\|$. We will use a similar approach, where we derive all the elements in the public parameters from a small set of elements, using a GPV-type procedure.

Specifically, we give out (potentially many) encodings of 0 $\{\mathbf{x}'_i = \mathbf{b}'_i \cdot \mathbf{g}/\mathbf{z}\}$. Let us ignore, for the moment, the fact that these encodings live in R_q, and instead pretend that we present them to the attacker as elements $\mathbf{b}'_i\mathbf{g}/\mathbf{z}$ in the overlying cyclotomic field. (Of course, we are giving the attacker an additional advantage here.) Then, all of the encodings are in the fractional principal ideal lattice $\mathcal{J} = \langle \mathbf{g}/\mathbf{z} \rangle$. If we simply chose the $\mathbf{b}'_i$ values randomly and independently, it is conceivable that an averaging/transcript attack could recover $\mathbf{g}/\mathbf{z}$. However, we instead follow [GPV08] by generating the encodings $\{\mathbf{b}_i\}$ according to a Gaussian distribution over the fractional ideal lattice, using an efficient discrete Gaussian sampler [GPV08, Pei10, DN12a]. By the same argument as [GPV08],

[4] An easy example: If $\mathcal{U} \in_R [0, B]$ then $\Pr[\mathcal{U} > \frac{9}{10}B] = 0.1$. However if $\mathcal{U} \in_R [0, B^{100}]$ then $\Pr[\sqrt[100]{\mathcal{U}} > \frac{9}{10}B] \approx 1$.

such encodings (presented in characteristic zero) reveal nothing in particular about the term $\mathbf{g}/\mathbf{z}$ that is being used to generate the encodings. (We omit a formal proof.)

More concretely, one can show that with probability $1 - o(1)$ over the choice of $\mathbf{z}$ we we have $\|1/\mathbf{z}\| < n/q$, so in our instance generation we re-choose $\mathbf{z}$ until this condition is met. Similarly when choosing $\mathbf{g} \leftarrow D_{\mathbb{Z}^n,\sigma}$ we get $\|1/\mathbf{g}\| < n/\sigma$ with probability $1 - o(1)$, so here too we re-choose $\mathbf{g}$ until this condition is met. When the first condition is met, then we have $\|\mathbf{g}/\mathbf{z}\| < \sigma n^{1.5}/q$, so we can use the GPV procedure to sample elements from $\mathcal{J}$ according to the Gaussian distribution $\mathbf{x}'_i \leftarrow D_{\mathcal{J},s}$ with parameter $s = \sigma n^2/q$ (say).

We note that the elements that we draw are of the form $\mathbf{x}'_i = \mathbf{b}'_i \cdot \mathbf{g}/\mathbf{z}$ for some (integral) $\kappa' \in R$. Moreover we can bound the size of the $\mathbf{b}'_i$'s by $\|\mathbf{b}'_i\| = \|\mathbf{x}'_i\| \cdot \|\mathbf{z}\| \cdot \|1/\mathbf{g}\| < (\sigma n^{2.5}/q) \cdot q \cdot n/\sigma = n^{3.5}$.

Next we map these elements to R_q by setting $\mathbf{x}_i = [\mathbf{b}'_i \mathbf{g}/\mathbf{z}]_q$. Denoting the denominator by $\mathbf{b}_i = \mathbf{b}'_i \mathbf{g}$, we can bound its size by $\|\mathbf{b}_i\| = \|\mathbf{b}'_i\| \cdot \|\mathbf{g}\| < n^{3.5} \cdot \sigma\sqrt{n} = \sigma n^4$. Sampled this way, we know that the randomizers $\mathbf{x}_i$ do not provide any more power to the attacker beyond the ability to sample elements from $\mathcal{J}$ according to $D_{\mathcal{J},s}$. [5]

We set $\boldsymbol{h}$ in a similar way. Again, we use [GPV08] to prevent the attacker analyzing the zero-tester $\boldsymbol{h} \cdot \boldsymbol{z}^\kappa / \boldsymbol{g}$ geometrically to extract useful information about $\boldsymbol{h}$, or the other terms, individually. Roughly, once $\mathbf{g}$ and $\mathbf{z}$ are chosen, one chooses $\boldsymbol{h}$ according to an ellipsoid Gaussian of the same "shape" as $\boldsymbol{g}/\boldsymbol{z}^\kappa$, so that the distribution of the zero-tester is a spherical Gaussian.

Although we prefer to use the GPV-type approach above, we note for completeness that another plausible line of defense against averaging attacks is to actually decrease the number of elements made public, perhaps as few as only two. Namely we can publish only two elements $\mathbf{x}_1 = [\mathbf{b}'_1 \mathbf{g}/\mathbf{z}]_q$ and $\mathbf{x}_2 = [\mathbf{b}'_2 \mathbf{g}/\mathbf{z}]_q$, perhaps chosen according to the procedure above conditioned on $\mathbf{b}'_1, \mathbf{b}'_2$ being co-prime. To re-randomize a level-one encoding $\boldsymbol{u}$, we can then choose two small elements $\boldsymbol{a}_1, \boldsymbol{a}_2$ and set $\boldsymbol{u}' = \boldsymbol{u} + \boldsymbol{a}_1 \cdot \mathbf{x}_1 + \boldsymbol{a}_2 \cdot \mathbf{x}_2$. One drawback of this method is that we can no longer use Theorem 1 to argue that the output distribution of reRand is nearly independent of its input, instead we need to use yet another computational assumption (and a rather awkward one at that). Another drawback is that it is not at all clear that the attacker cannot just take many terms of the form $\boldsymbol{a}_1 \cdot \mathbf{x}_1 + \boldsymbol{a}_2 \cdot \mathbf{x}_2$ (for many random pairs $(\boldsymbol{a}_1, \boldsymbol{a}_2)$) to use for the samples of the averaging attacks.

7 Survey of Lattice Cryptanalysis

Here we provide a survey of relevant cryptanalysis techniques from the literature, and also provide two new attacks that we developed in the course of this work. Our new attacks are extensions of techniques that were developed in [GS02] for attacking NTRU signatures: In Section 7.8.1 we describe a "dimension-halving attack" on principal ideal lattices, demonstrating that one needs to double the dimension of principal ideal lattices (compared to general ideal lattices) to preserve security. Then in Section 7.6 we provide a polynomial-time algorithm that solves the *closest principal ideal generator* problem in certain cases. Specifically, it can recover a generator of a principal ideal $\mathcal{I} = \langle \mathbf{g} \rangle$ from a basis of $\mathcal{I}$ and an ϵ-approximation of the generator $\mathbf{g}$, for small enough ϵ – namely, $\epsilon \leq n^{-\Omega(\log \log n)}$.

[5] We expect it be even slightly less powerful, since these samples are mapped into R_q before the attacker sees them.

7.1 The Geometry of Number Fields

We provide some background on ideal lattices and cyclotomic fields that will prove useful when we begin describing attacks. Much of our description here follows [LPR10].

An *algebraic integer* $\zeta \in \mathbb{C}$ is a root of a monic irreducible polynomial $f(x) \in \mathbb{Z}[x]$, called ζ's *minimal polynomial*. Let n be the degree of $f(x)$. The *conjugates* of ζ are the n roots of $f(x)$.

A *number field* is a field extension $K = Q(\zeta)$ obtained by adjoining an algebraic integer ζ to $\mathbb{Q}$. There are exactly n field homomorphisms (*embeddings*) $\sigma_i : K \to \mathbb{C}$ that fix $\mathbb{Q}$, given by mapping ζ to its conjugates. When $\sigma_i(\zeta) \in \mathbb{R}$, we say σ_i is a real embedding; otherwise, it is complex. Since the roots of $f(x)$ in $\mathbb{C} \setminus \mathbb{R}$ come in *conjugate pairs*, so do the complex embeddings. The *signature* of K is (s_1, s_2), where s_1 is the number of real embeddings and $2s_2$ is the number of complex embeddings; we have $n = s_1 + 2s_2$. By convention, we order the embeddings so that $\{\sigma_j\}_{j \in [s_1]}$ are the real ones, and $\sigma_{s_1+s_2+j} = \overline{\sigma_{s_1+j}}$ for $j \in [s_2]$.

The *canonical embedding* $\sigma : K \to \mathbb{R}^{s_1} \times \mathbb{C}^{2s_2}$ is defined as

$$\sigma(\mathbf{a}) = (\sigma_1(\mathbf{a}), \ldots, \sigma_n(\mathbf{a})).$$

It is a field homomorphism from K to $\mathbb{R}^{s_1} \times \mathbb{C}^{2s_2}$, where multiplication and addition in $\mathbb{R}^{s_1} \times \mathbb{C}^{2s_2}$ are component-wise: we write $\sigma(\mathbf{ab}) = \sigma(\mathbf{a})\sigma(\mathbf{b})$. Due to the pairing of the complex embeddings, σ maps into the following space $H \subseteq \mathbb{R}^{s_1} \times \mathbb{C}^{2s_2} \subset \mathbb{C}^n$:

$$H = \{(x_1, \ldots, x_n) \in \mathbb{R}^{s_1} \times \mathbb{C}^{2s_2} : x_{s_1+s_2+j} = \overline{x_{s_1+j}}, \forall j \in [s_2]\}.$$

The space H is isomorphic to $\mathbb{R}^n$ as an inner product space. As in [LPR10], one can show this explicitly by defining an orthonormal basis $\{\overrightarrow{h_i}\}_{i \in [n]}$ of H. For two vectors $\overrightarrow{u}, \overrightarrow{v} \in H$, we use the *Hermitian inner product* $\langle \overrightarrow{u}, \overrightarrow{v} \rangle = \sum u_i \overline{v_i}$, which has the usual properties of a bilinear form and is positive definite – i.e., $\langle \overrightarrow{u}, \overrightarrow{u} \rangle$ is a real non-negative number. Moreover, for any $\overrightarrow{u}, \overrightarrow{v} \in H$, their Hermitian inner product is a real number – in particular, it is the same real number that one obtains when one instead interprets $\overrightarrow{u}, \overrightarrow{v}$ as vectors in $\mathbb{R}^n$ corresponding to their associated $\mathbb{R}$-linear combinations of the $\{\overrightarrow{h_i}\}_{i \in [n]}$ basis and we compute $\langle \overrightarrow{u}, \overrightarrow{v} \rangle$ in usual way.

There is also a *coefficient embedding* $\tau : K \to \mathbb{Q}^n$. In particular, since $f(\zeta) = 0$, there is an isomorphism between $\mathbb{Q}[x]$ modulo $f(x)$ and K given by $x \to \zeta$. So, K can be represented as a n-dimensional vector space over $\mathbb{Q}$ using the *power basis* $\{1, \zeta, \ldots, \zeta^{n-1}\}$, and τ maps an element of K to its associated coefficient vector. Occasionally, we map between the canonical and coefficient embeddings, which is a linear transformation depending only on K.

We give elements of K *geometric norms* by identifying them with canonical or coefficient embeddings. The ℓ_p norm of $\mathbf{a} \in K$ under the canonical embedding is $\|\mathbf{a}\|_p = (\sum |\sigma_i(\mathbf{a})|^p)^{1/p}$, where $p \in [1, \infty]$ and $\|\mathbf{a}\|_\infty = \max\{|\sigma_i(\mathbf{a})|\}$. The coefficient norm $\|\mathbf{a}\|_{co}$ we define to be the ℓ_∞-norm of $\mathbf{a}$'s coefficient vector. For convenience when converting from the canonical to the coefficient embedding, we let γ_2 denote the maximal value of $\|\mathbf{a}\|_{co}/\|\mathbf{a}\|_2$ and γ_∞ denote the maximal value of $\|\mathbf{a}\|_{co}/\|\mathbf{a}\|_\infty$ (where the dependence of these values on K is understood).

The *ring of integers* $\mathcal{O}_K \subset K$ is the set of all algebraic integers in K. It forms a ring under addition and multiplication, and it is a n-dimensional vector subspace of K. In particular, $\mathcal{O}_K$ is a free $\mathbb{Z}$-module of rank n – namely, the set of all $\mathbb{Z}$-linear combinations of some *integral basis* $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subset \mathcal{O}_K$.

An *(integral) ideal* $\mathcal{I} \subseteq \mathcal{O}_K$ is a nontrivial (i.e., nonempty and nonzero) additive subgroup that is closed under multiplication by $\mathcal{O}_K$ – that is, $\mathbf{r} \cdot \mathbf{a} \in \mathcal{I}$ for any $\mathbf{r} \in \mathcal{O}_K$ and $\mathbf{a} \in \mathcal{I}$. It is finitely

generated as the set of all $\mathcal{O}_K$-linear combinations of some generators $\mathbf{a}_1, \mathbf{a}_2, \ldots \in \mathcal{O}_K$; we write $\mathcal{I} = \langle \mathbf{a}_1, \mathbf{a}_2, \ldots \rangle$. An ideal $\mathcal{I}$ is *principal* if $\mathcal{I} = \langle \mathbf{a} \rangle$ for $\mathbf{a} \in \mathcal{O}_K$ – that is, if one generator suffices. A *fractional ideal* $\mathcal{I} \subset K$ is a set such that $\mathbf{d} \cdot \mathcal{I}$ is an integral ideal for some $\mathbf{d} \in \mathcal{O}_K$. The *inverse* $\mathcal{I}^{-1}$ of an ideal $\mathcal{I}$ is the set $\{\mathbf{a} \in K : \mathbf{a} \cdot \mathcal{I} \subseteq \mathcal{O}_K\}$. A (fractional) ideal also can be represented as a free $\mathbb{Z}$-module of rank n – that is, it is generated as the set of all $\mathbb{Z}$-linear combinations of some basis $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subset \mathcal{O}_K$ (or $B \subset K$ when the ideal is fractional). We may refer to a (fractional) ideal $\mathcal{I}$ as an *ideal lattice* when viewing it as a $\mathbb{Z}$-module, and apply lattice notation, such as $\lambda_1(\mathcal{I})$.

The *sum* of two ideals is $\mathcal{I} + \mathcal{J} = \{\mathbf{a} + \mathbf{b} : \mathbf{a} \in \mathcal{I}, \mathbf{b} \in \mathcal{J}\}$. Two ideals $\mathcal{I}, \mathcal{J} \subseteq \mathcal{O}_K$ such that $\mathcal{I} + \mathcal{J} = \mathcal{O}_K$ are *relatively prime*.

The *product* of two ideals is $\mathcal{I}\mathcal{J}$ is the additive subgroup formed by the set $\{\mathbf{a} \cdot \mathbf{b} : \mathbf{a} \in \mathcal{I}, \mathbf{b} \in \mathcal{J}\}$. An ideal $\mathfrak{p} \subsetneq \mathcal{O}_K$ is *prime* if $\mathbf{a} \in \mathfrak{p}$ or $\mathbf{b} \in \mathfrak{p}$ whenever $\mathbf{ab} \in \mathfrak{p}$ and $\mathbf{a}, \mathbf{b} \in \mathcal{O}_K$. Every ideal $\mathcal{I} \subseteq \mathcal{O}_K$ can be expressed uniquely as a product of powers of prime ideals – that is, $\mathcal{O}_K$ has *unique factorization of ideals*.

The *norm* of an ideal $\mathcal{I}$ is its index as an additive subgroup of $\mathcal{O}_K$ – i.e., $\mathsf{N}(\mathcal{I}) = |\mathcal{O}_K/\mathcal{I}|$. The norm of an element $\mathbf{a} \in \mathcal{O}_K$ is $\mathsf{N}(\mathbf{a}) = \mathsf{N}_{K/\mathbb{Q}}(\mathbf{a}) = \prod_{i \in [n]} \sigma_i(\mathbf{a})$. For a principal ideal $\mathcal{I} = \langle \mathbf{a} \rangle$ with $\mathbf{a} \in \mathcal{O}_K$, we have $\mathsf{N}(\mathcal{I}) = \mathsf{N}(\mathbf{a})$. The set of fractional ideals form a group under multiplication, and the norm is a multiplicative homomorphism on this group – in particular, $\mathsf{N}(\mathcal{I}\mathcal{J}) = \mathsf{N}(\mathcal{I})\mathsf{N}(\mathcal{J})$ and $\mathsf{N}(\mathcal{I}/\mathcal{J}) = \mathsf{N}(\mathcal{I})/\mathsf{N}(\mathcal{J})$. The *relative norm* $\mathsf{N}_{K/L}(\mathbf{a})$ of an element $\mathbf{a} \in K$ over a subfield $L \subset K$ is $\prod_{\sigma_i \in S} \sigma_i(\mathbf{a})$, where S consists of the K-embeddings σ_i that fix every element in L.

The *unit group* $\mathcal{U}_K \subset \mathcal{O}_K$ is the group of invertible elements in $\mathcal{O}_K$. If $\mathbf{u} \in \mathcal{U}_K$, then the index $|\mathcal{O}_K / \langle \mathbf{u} \rangle| = 1$, and therefore $\mathsf{N}(\mathbf{u}) = \pm 1$. The unit group may contain *torsion units* (roots of unity) and *nontorsion units*. By the Dirichlet Unit Theorem, the group of nontorsion units is finitely generated and has rank $s_1 + s_2 - 1$ (where rank refers to multiplicative independence). The *logarithmic embedding* $\lambda : K^* \to \mathbb{R}^{s_1+s_2}$ is a homomorphism from a multiplicative group to an additive group given by $\lambda(\mathbf{a}) = (\ln|\sigma_1(\mathbf{a})|, \ldots, \ln|\sigma_{s_1+s_2}(\mathbf{a})|)$. The kernel of λ consists of the torsion units in K. For every unit $\mathbf{u} \in \mathcal{U}_K$ (including nontorsion units), since $\mathsf{N}(\mathbf{u}) = \pm 1$, we have $\sum_{i \in [s_1]} \ln|\sigma_i(\mathbf{u})| + 2\sum_{i \in [s_2]} \ln|\sigma_{s_1+i}(\mathbf{u})| = 0$ (hence the units have rank only $s_1 + s_2 - 1$). In short, viewed through the logarithmic embedding, the units are endowed with the geometry of a lattice. We call this lattice the Dirichlet unit lattice.

7.1.1 Cyclotomic Number Fields

For positive integer m, let $\zeta_m = e^{2\pi i/m} \in \mathbb{C}$ denote the principal m-th root of unity. The m-th *cyclotomic* number field is $K = \mathbb{Q}(\zeta_m)$. The m-th *cyclotomic polynomial* is $\Psi_m(x) = \prod_{k \in \mathbb{Z}_m^*} (x - \zeta_m^k)$. This polynomial is monic, irreducible, has degree $n = \phi(m)$, and its roots (the conjugates of ζ_m) are the primitive m-th roots of unity in $\mathbb{C}$. The field K has degree n. It is convenient to index the embeddings by elements of $\mathbb{Z}_m^*$ instead of $[n]$, where $\sigma_i(\zeta_m) = \zeta_m^i$. Since all of the embeddings are complex, they come in $s_2 = n/2$ pairs – in particular, for $i \in \mathbb{Z}_m^*$, σ_i is paired with σ_{-i}.

When $K = \mathbb{Q}(\zeta_m)$, the ring of integers $\mathcal{O}_K$ is simply $\mathbb{Z}[\zeta_m] = \mathbb{Z}[x]/\Phi_m(x)$.

The cyclotomic field $K = \mathbb{Q}(\zeta_m)$ has a *real subfield* $K^+ = \mathbb{Q}(\zeta_m + \zeta_m^{-1})$ – that is, $\zeta_m + \zeta_m^{-1}$, and thus all elements in K^+, are real numbers. It has index 2 in K; its degree is $n/2$. The ring of integers $\mathcal{O}_{K^+}$ of K^+ is simply $\mathbb{Z}[\zeta_m + \zeta_m^{-1}]$. The embeddings σ_1, σ_{-1} both fix every element in K^+, and the relative norm $\mathsf{N}_{K/K^+}(\mathbf{a})$ of $\mathbf{a} \in K$ is $\sigma_1(\mathbf{a}) \cdot \sigma_{-1}(\mathbf{a}) = \mathbf{a} \cdot \overline{\mathbf{a}}$.

The group of *cyclotomic units* $\mathcal{U}_K$ has rank $s_2 - 1 = n/2 - 1$. Since the signature of the real subfield K^+ is $(n/2, 0)$, the rank of the real cyclotomic units $\mathcal{U}_{K^+} = \mathcal{U}_K \cap \mathcal{O}_{K^+}$ is also $n/2 - 1$. For m a prime power, $\mathcal{U}_K$ is generated by ζ_m and $\mathcal{U}_{K^+}$. For m a prime power, an explicit set of

generators of $\mathcal{U}_K$ is $\{\pm\zeta_m, (1-\zeta_m^k)/(1-\zeta_m) : k \in \mathbb{Z}_m^*\}$. To see that $\epsilon = (1-\zeta_m^k)/(1-\zeta_m)$ is a unit, observe that $\epsilon = 1 + \zeta_m + \ldots + \zeta_m^{k-1} \in \mathcal{O}_K$ and $\mathsf{N}_{K/Q}(\epsilon) = \prod_{\ell \in \mathbb{Z}_m^*}(1-\zeta_m^\ell)/\prod_{\ell \in \mathbb{Z}_m^*}(1-\zeta_m^\ell) = 1$. Ramachandra [Ram67] explicitly described a full-rank set of independent units for the case that m not a prime power.

In the coefficient embedding, where $\mathbf{a} \in \mathcal{O}_K$ is viewed as a polynomial $\mathbf{a}(x) \in \mathbb{Z}[x]/\Phi_m(x)$, we have an extension of Fermat's Little Theorem: $\mathbf{a}(x)^Q = \mathbf{a}(x^Q) \bmod Q$ for any prime Q. When $Q = 1 \bmod m$, this becomes $\mathbf{a}^Q = \mathbf{a} \bmod Q$.

7.1.2 Some Computational Aspects of Number Fields and Ideal Lattices

An element $\mathbf{v} \in K$ can be represented in its canonical embedding conveniently in terms of the integral basis for $\mathcal{O}_K$. Given $\mathbf{v} \in K$ represented in its canonical embedding, it is efficient to convert it to its coefficient embedding, or vice versa – via linear transformations corresponding to multipoint interpolation and evaluation. "Efficient" means in time polynomial in n, $\log \Delta_K$, and the bit-length of $\mathbf{v}$. (Here, Δ_K is the *discriminant* of K. For the important case of the m-th cyclotomic field of degree $n = \phi(m)$, we have $\Delta_K \leq n^n$.) Given $\mathbf{v}_1, \mathbf{v}_2 \in K$, represented in either their canonical or their coefficient embeddings, it is efficient to compute $\mathbf{v}_1 + \mathbf{v}_2$, $\mathbf{v}_1 \cdot \mathbf{v}_2$, and $\mathbf{v}_1/\mathbf{v}_2$. To handle denominators, the inverse $1/\mathbf{v}_2$ can be represented as $\mathbf{v}_2'/\mathsf{N}(\mathbf{v}_2)$ where $\mathbf{v}_2' \in \mathcal{O}_K$.

A convenient way of representing a (fractional) ideal in field K of degree n is as a $\mathbb{Z}$-module (i.e., a lattice) of rank n, generated as the set of all $\mathbb{Z}$-linear combinations of some basis $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subset \mathcal{O}_K$ (or $B \subset K$ when the ideal is fractional). We call this lattice an *ideal lattice*. We may use notation like $\mathcal{I} = \mathcal{L}(B)$, where $\mathcal{L}(B)$ denotes the lattice generated by B.

Like all lattices, an ideal lattice has a canonical basis called its *Hermite Normal Form* (HNF). The HNF basis of a lattices is unique and can be computed efficiently from any other basis of the lattice. The HNF basis has nice efficiency properties – in particular, it can be expressed in at most $O(n \log d)$ bits, where d is the absolute value of the determinant of a basis of the lattice [Mic01]. It also has nice security properties, in the sense that it reveals no information that cannot be derived in polynomial time from any other basis [Mic01]. For ideal lattices in the canonical embedding, the HNF basis is an integer lattice representing a linear transformation of the integral basis of $\mathcal{O}_K$. The determinant of the HNF basis equals the norm of the ideal. Given HNF bases of ideals $\mathcal{I}_1, \mathcal{I}_2$, one can efficiently compute an HNF basis for the ideals $\mathcal{I}_1 + \mathcal{I}_2$, $\mathcal{I}_1 \cdot \mathcal{I}_2$, $\mathcal{I}_1/\mathcal{I}_2$. Various other natural operations on ideals and bases are also efficient. An example: one can efficiently reduce an element $\mathbf{v} \in K$ modulo a basis $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ of an ideal $\mathcal{I}$ – that is, find the element $\mathbf{w} \in K$ with $\mathbf{v} - \mathbf{w} \in \mathcal{I}$ and $\mathbf{w} \in \mathcal{P}(B)$, where $\mathcal{P}(B) = \{\sum x_i \cdot \mathbf{b}_i : x_i \in [-1/2, 1/2)\}$ is the parallelepiped associated to B.

7.1.3 Computational Hardness Assumptions over Number Fields

Hard problems involving ideal lattices often have both algebraic and geometric aspects.

Geometrically, we can specialize standard lattice problems – such as the shortest vector problem (SVP), shortest independent vector problem (SIVP), closest vector problem (SVP), the bounded distance decoding problem (BDDP), etc. – to ideal lattices. The celebrated LLL algorithm [LLL82] finds somewhat short vectors in (general) lattices:

Fact 2. *Let $B = \{\overrightarrow{b_1}, \ldots, \overrightarrow{b_n}\}$ be a basis of a lattice L. Given B, the LLL algorithm outputs a vector $\overrightarrow{v} \in L$ satisfying $\|\overrightarrow{v}\|_2 \leq 2^{n/2} \cdot \det(L)^{1/n}$. The algorithm runs in time polynomial in the size of its input.*

Schnorr and others have described other lattice reduction algorithms with a variety of tradeoffs; for example, [Sch87] proves the following:

Fact 3. *Let $B = \{\overrightarrow{b_1}, \ldots, \overrightarrow{b_n}\}$ be a basis of a lattice L. Given B and integer k, Schnorr's algorithm [Sch87] outputs a vector $\overrightarrow{v} \in L$ satisfying $\|\overrightarrow{v}\|_2 \leq k^{O(n/k)} \cdot \det(L)^{1/n}$ in time $k^{O(k)}$.*

The asymptotics of lattice reduction algorithms are still similar to [Sch87], and thus attacks on ideal lattices using purely geometric tools are limited.

Algebraically, we can consider problems such as the factorization of ideals, the structure of the class group and unit group, etc. Subexponential classical algorithms are known for factoring ideals, computing the class group and unit group, and computing a generator of a principal ideal (the Principal Ideal Generator Problem (PIGP)). Polynomial-time quantum algorithms are known for the latter three problems when the degree of the field is constant [Hal05, SV05].

Factoring ideals reduces to factoring integers, hence is subexponential-time classically [LLMP90] and polynomial-time quantumly [Sho97a]. In particular, for any monogenic ring $R = \mathbb{Z}[x]/(f(x))$ such as $\mathcal{O}_K$ for a cyclotomic field K, there is an efficient algorithm to find all of the prime ideals in R with norms that are a power of a prime p. The algorithm resorts to the following theorem.

Theorem 3 (Kummer-Dedekind, from [Ste08])**.** *Suppose $f(x) = \prod_i g_i(x)^{e_i} \bmod p$ for prime integer p. The prime ideals $\mathfrak{p}_i$ in $\mathbb{Z}[x]/(f(x))$ whose norms are powers of p are precisely $\mathfrak{p}_i = (p, g_i(x))$.*

There are polynomial time algorithms for factoring polynomials in $\mathbb{Z}_p[x]$ – e.g., by Kaltofen and Shoup [KS98]. Therefore, at least for monogenic rings, factoring an ideal with norm N efficiently reduces to factoring the integer N.

Peikert and Rosen [PR07] provided a reduction of an average-case lattice problem to the worst-case hardness of ideal lattice problem, where the lossiness of the reduction was only logarithmic over fields of small root discriminant. Gentry [Gen10] showed that ideal lattice problems are efficiently *self*-reducible (in some sense) in the quantum setting. This worst-case/average-case reduction exploited, among other things, efficient factorization of ideals via Kummer-Dedekind. Lyubashevsky, Peikert and Regev [LPR10] defined a decision problem called "ring learning with errors" (RLWE) and showed that an attacker that can solve RLWE on average can be used to solve ideal lattice problems, such as SIVP, in the worst case. (Earlier, Regev [Reg05] found an analogous worst-case/average-case connection between the learning with errors (LWE) problem and problems over general lattices.) They relied heavily on the algebraic structure of ideal lattice problems – in particular, on underlying ring automorphisms – to construct their search-to-decision reduction.

7.2 Averaging Attacks

In the so-called "averaging attack", the attacker is given a set $S = \{\mathbf{v} \cdot \mathbf{y}_i\}$, where $\mathbf{v}, \mathbf{y}_1, \mathbf{y}_2, \ldots$ are ring elements, and its goal is to use "averaging" to recover $\mathbf{v} \cdot \overline{\mathbf{v}}$, where $\overline{\mathbf{v}} = \mathbf{v}(x^{-1})$ is the conjugate of $\mathbf{v}$. It was used by Kaliski (in connection with patent [HKL$^+$00]) and Gentry and Szydlo [GS02] in attacks against NTRU signature schemes [HKL$^+$00, HPS01]. We review the averaging attack here. Along the way, we update the attack so that it works within the ring of integers of any cyclotomic field. (Previously, the attack focused on the ring $\mathbb{Z}[x]/(x^m - 1)$, as used by NTRU signature schemes.)

The averaging attack is relevant to our constructions in the sense that, for certain (ill-advised) distributions of our params, the attacker could use the averaging attack to recover nontrivial

information. For example, in one version of our constructions, params includes a zero-tester $\mathbf{p}_{zt} = [\boldsymbol{h}\mathbf{z}^\kappa/\mathbf{g}]_q$ and multiple terms $\{\mathbf{x}_i = [\mathbf{b}_i/\mathbf{z}]_q\}$ with $\mathbf{b}_i \in \langle \mathbf{g} \rangle$. Let $\mathbf{b}'_i = \mathbf{b}_i/\mathbf{g}$. From params, the attacker can derive the values $\{[\mathbf{p}_{zt}\mathbf{x}_i^\kappa]_q = \boldsymbol{h}\mathbf{g}^{\kappa-1} \cdot \mathbf{b}_i'^\kappa\}$. Conceivably, depending on the particular distributions of the parameters, the attacker could use averaging to remove the $\mathbf{b}'_i$'s and recover $\boldsymbol{h}\mathbf{g}^{\kappa-1}$.

We have a couple of defenses against this averaging attack. First, for our constructions it seems that $\boldsymbol{h}\mathbf{g}^{\kappa-1}$ (and other terms that could conceivably be obtained through averaging) do not seem to be useful to the attacker. Second, as described in Section 6.4, we choose our params according to distributions designed to make averaging attacks useless. More precisely, we adapt an observation of Gentry, Peikert and Vaikuntanathan [GPV08] in the context of lattice-based signatures – namely, that we can use a "good" lattice basis to generate a transcript of lattice points according to a *canonical* distribution that reveals nothing about the *particular* good basis that we are using (aside from the fact that it is "good"). We generate our params according to such canonical distributions.

Now, let us describe how the averaging attack works. The distributions of $\mathbf{v}$ and the $\mathbf{y}_i$'s may vary, but let us suppose for concreteness that the challenger samples $\mathbf{v}'$ and $\{\mathbf{y}'_i\}$ according to Gaussian distributions $\mathbf{v}' \leftarrow D_{\mathbb{Z}^m,\sigma}$ and $\mathbf{y}'_i \leftarrow D_{\mathbb{Z}^m,\sigma'}$, interprets these as coefficient vectors of polynomials in $\mathbb{Z}[x]/(x^m - 1)$, and finally sets $\mathbf{v} \leftarrow \mathbf{v}' \bmod \Phi_m(x)$ and $\mathbf{y}_i \leftarrow \mathbf{y}'_i \bmod \Phi_m(x)$.

Now, consider the average:

$$\mathbf{A}_r \quad = \quad (1/r)\sum_{i=1}^{r}(\mathbf{v}\cdot\mathbf{y}_i)\cdot\overline{(\mathbf{v}\cdot\mathbf{y}_i)} \quad = \quad (\mathbf{v}\cdot\overline{\mathbf{v}})\cdot\left((1/r)\sum_{i=1}^{r}\mathbf{y}_i\cdot\overline{\mathbf{y}_i}\right).$$

Under the canonical embedding, we have:

$$\sigma(\mathbf{A}_r) = \sigma(\mathbf{v}\cdot\overline{\mathbf{v}})\cdot\sigma(\mathbf{Y}_r), \quad \text{where} \quad \mathbf{Y}_r = \left((1/r)\sum_{i=1}^{r}\mathbf{y}_i\cdot\overline{\mathbf{y}_i}\right).$$

Toward understanding $\sigma(\mathbf{Y}_r)$, first consider a single vector $\sigma(\mathbf{y}_i \cdot \overline{\mathbf{y}_i})$ in the summation. Recall that, since we are working in a cyclotomic field, the embeddings are all complex and come in conjugate pairs (σ_j, σ_{-j}), where σ_j for $j \in \mathbb{Z}_m^*$ denotes the embedding $\sigma_j(\zeta_m) = \zeta_m^j$. Moreover, for any $\mathbf{a}$ in the cyclotomic field, the values $\sigma_j(\mathbf{a})$ and $\sigma_{-j}(\mathbf{a})$ are conjugate complex numbers, and therefore $\sigma_j(\mathbf{a}) \cdot \sigma_{-j}(\mathbf{a})$ is a non-negative real number. Now, notice that $\sigma_j(\mathbf{a}) \cdot \sigma_{-j}(\mathbf{a}) = \sigma_j(\mathbf{a}) \cdot \sigma_j(\overline{\mathbf{a}}) = \sigma_j(\mathbf{a} \cdot \overline{\mathbf{a}})$. This means that each vector $\sigma(\mathbf{y}_i \cdot \overline{\mathbf{y}_i})$ in the summation consists entirely of non-negative real numbers!

It is clear that, for any j, the average $\sigma_j(\mathbf{Y}_r) = 1/r \sum_{i=1}^{r} \sigma_j(\mathbf{y}_i \cdot \overline{\mathbf{y}_i})$ converges toward some positive number (rather than tending toward 0). Moreover, by symmetry, it converges to the *same* positive number for all j. Therefore, $\mathbf{A}_r$ converges to $s \cdot \mathbf{v} \cdot \overline{\mathbf{v}}$ for some known positive real scalar s.

The imprecision of the average decreases with $1/\sqrt{r}$. If the coefficients of $\mathbf{v}$ are only polynomial in size, then the averaging attack needs only a polynomial number of samples to obtain all of the coefficients of $\mathbf{v} \cdot \overline{\mathbf{v}}$ to within less than $1/2$, whereupon the attacker can round to obtain $\mathbf{v} \cdot \overline{\mathbf{v}}$ exactly. As we describe in Section 7.6, even if the coefficients of $\mathbf{v}$ are large, a ϵ-approximation of $\mathbf{v} \cdot \overline{\mathbf{v}}$, together with a basis of the ideal $\langle \mathbf{v} \cdot \overline{\mathbf{v}} \rangle$, is sufficient to recover $\mathbf{v} \cdot \overline{\mathbf{v}}$ exactly when ϵ is some inverse-quasi-polynomial function of m. (Note that it is easy to generate a basis of the ideal $\langle \mathbf{v} \cdot \overline{\mathbf{v}} \rangle$ from a basis of the ideal $\langle \mathbf{v} \rangle$, and that the latter (as mentioned previously) can likely be generated from S.)

If the averaging attack is successful and we recover $\mathbf{v} \cdot \overline{\mathbf{v}}$, we can then use an algorithm by Gentry and Szydlo [GS02]. This algorithm takes $\mathbf{v} \cdot \overline{\mathbf{v}}$ and a basis of the ideal $\langle \mathbf{v} \rangle$, and outputs the actual element $\mathbf{v}$ in polynomial time. (See Section 7.3.)

Can the averaging attack be extended? Since $\mathbf{v} \cdot \overline{\mathbf{v}}$ is the *relative norm* $\mathsf{N}_{K/K^+}(\mathbf{v})$ of $\mathbf{v}$ with respect to the index-2 real subfield $K^+ = \mathbb{Q}(\zeta_m + \zeta_m^{-1})$ (see Section 7.1 for more details), it is natural to ask whether the attack can be extended to other relative norms: Can we use the samples $S = \{\mathbf{v} \cdot \mathbf{y}_i\}$ to recover $\mathsf{N}_{K/L}(\mathbf{v})$ for some subfield L' of K that is not a subfield of K^+?

7.3 Gentry-Szydlo: Recovering v from $\mathbf{v} \cdot \overline{\mathbf{v}}$ and $\langle \mathbf{v} \rangle$

Here, we describe an algorithm by Gentry and Szydlo [GS02] (the GS algorithm) that recovers $\mathbf{v}$ from $\mathbf{v} \cdot \overline{\mathbf{v}}$ and a basis of the ideal $\langle \mathbf{v} \rangle$. The algorithm runs in polynomial time.

Gentry and Szydlo used the algorithm in combination with the averaging attack above to break an NTRU signature scheme. They used a set of samples $S = \{\mathbf{v} \cdot \mathbf{y}_i\}$ to approximate $\mathbf{v} \cdot \overline{\mathbf{v}}$ with sufficient precision to compute it exactly via rounding, and then invoked (but did not implement) the GS algorithm to recover $\mathbf{v}$ (the secret signing key). In our setting, the idea would be to attack our params using a similar approach. The GS algorithm was originally designed to work in $\mathbb{Z}[x]/(x^p - 1)$ for prime p. Here, we adapt it to a more general setting over the ring of integers $\mathcal{O}_K$ of the m-th cyclotomic field K. For convenience, we use R to refer to $\mathcal{O}_K$, and R_P to denote $\mathbb{Z}_P[x]/\Phi_m(x)$.

We describe the GS algorithm in detail, with proofs, because in Section 7.6 we will extend the algorithm to address the setting where our approximation of some generator $\mathbf{u}$ of a principal ideal $\mathcal{I} = \langle \mathbf{u} \rangle$ (e.g., where $\mathbf{u}$ is $\mathbf{v} \cdot \overline{\mathbf{v}}$) is *not* precise enough to obtain the value $\mathbf{u}$ exactly via rounding; we give a polynomial-time algorithm to recover $\mathbf{u}$ from a ϵ-approximation of it when ϵ is inverse-quasi-polynomial.

Recall that the value $\mathbf{v} \cdot \overline{\mathbf{v}}$ is the relative norm of $\mathbf{v} \in K = \mathbb{Q}(\zeta_m)$ with respect to the subfield $K^+ = \mathbb{Q}(\zeta_m + \zeta_m^{-1})$ – i.e., $\mathbf{v} \cdot \overline{\mathbf{v}} = \mathsf{N}_{K/K^+}(\mathbf{v})$. The GS algorithm might be somewhat surprising, since we do not know how to recover $\mathbf{v}$ efficiently from the norm $\mathsf{N}_{K/\mathbb{Q}}(\mathbf{v})$ and a basis of $\langle \mathbf{v} \rangle$. Indeed, the value $\mathsf{N}_{K/\mathbb{Q}}(\mathbf{v})$ is superfluous, since it can be derived from the basis of $\langle \mathbf{v} \rangle$; therefore, finding $\mathbf{v}$ would solve the so-called Principal Ideal Generator Problem, which seems infeasible.

One might also be surprised that $\mathsf{N}_{K/K^+}(\mathbf{v})$ and $\langle \mathbf{v} \rangle$ are enough to uniquely define $\mathbf{v}$, given that $\mathsf{N}_{K/\mathbb{Q}}(\mathbf{v})$ and $\langle \mathbf{v} \rangle$ only define $\mathbf{v}$ up to an infinite group of units. (See 7.1.1 for a discussion of cyclotomic units.) Indeed, $\mathsf{N}_{K/K^+}(\mathbf{v})$ and $\langle \mathbf{v} \rangle$ are *not* enough to uniquely define $\mathbf{v}$ – in particular, if $\mathbf{v}' = \mathbf{v} \cdot \mathbf{u}$ for any *torsion unit* (root of unity) $\mathbf{u}$, we have $\mathsf{N}_{K/K^+}(\mathbf{v}') = \mathsf{N}_{K/K^+}(\mathbf{v})$ and $\langle \mathbf{v}' \rangle = \langle \mathbf{v} \rangle$. However, in attacks, it is typically sufficient to obtain $\mathbf{v}$ up to a small set of roots of unity. On the other hand, if $\mathbf{u}$ is not a torsion unit – e.g., if it is a nontrivial cyclotomic unit – then we will have $\mathsf{N}_{K/K^+}(\mathbf{u}) \neq 1$ and therefore $\mathsf{N}_{K/K^+}(\mathbf{v}') \neq \mathsf{N}_{K/K^+}(\mathbf{v})$. The reason we have $\mathsf{N}_{K/K^+}(\mathbf{u}) \neq 1$ for nontorsion units is that, up to multiplication by a torsion unit, all nontorsion units in K are already in the real subfield K^+ – i.e., $\mathbf{u} = \zeta_m^i \cdot \mathbf{u}'$ where $\mathbf{u}' \in K^+$ is a nontorsion unit. So, $\mathsf{N}_{K/K^+}(\mathbf{u}) = \mathbf{u} \cdot \overline{\mathbf{u}} = \mathbf{u}'^2 \neq 1$.

The essential strategy of the GS algorithm is to combine algebra (in particular, Fermat's Little Theorem) with lattice reduction (LLL). By an extension of Fermat's Little Theorem, for any prime $P = 1 \bmod m$, we have that $\mathbf{v}^P = \mathbf{v}$ over R_P. Unless $\mathbf{v}$ is a zero divisor in R_P (there are only $\mathsf{poly}(m, \log \mathsf{N}_{K/Q}(\mathbf{v}))$ primes P for which this can happen), we have $\mathbf{v}^{P-1} = 1$ over R_P. Now, suppose that we compute a LLL-reduced basis B of the ideal $\langle \mathbf{v}^{P-1} \rangle$; this we can do in time polynomial in m, P, and the bit-length of $\mathbf{v}$. The shortest element $\mathbf{w}$ in the reduced basis has the

form $\mathbf{v}^{P-1} \cdot \mathbf{a}$ for some $\mathbf{a}$. If it happens that $\|\mathbf{a}\|_{co} < P/2$ – i.e., if $\mathbf{a}$'s coefficients all have magnitude less than $P/2$ – then we obtain $\mathbf{a} = [\mathbf{w}]_P$ exactly, and thus $\mathbf{v}^{P-1}$. From $\mathbf{v}^{P-1}$, we can compute $\mathbf{v}$ in time polynomial in m, P, and the bit-length of $\mathbf{v}$.

The actual algorithm is more complicated than this, since the essential strategy above leaves two important issues unresolved.

- Issue 1 (How to Guarantee that $\mathbf{a}$ is small): LLL guarantees that it will find $\mathbf{w} \in \langle \mathbf{v}^{P-1} \rangle$ of length at most $2^{(n-1)/2} \cdot \lambda_1(\langle \mathbf{v}^{P-1} \rangle)$. But this does not imply that $\mathbf{a} = \mathbf{w}/\mathbf{v}^{P-1}$ has length at most $2^{(n-1)/2}$. Indeed, $\langle \mathbf{v}^{P-1} \rangle$ does not even define $\mathbf{v}$ uniquely (due to the group of units). Since these units can have arbitrarily high Euclidean norm, $\mathbf{a}$ could be arbitrarily long.

- Issue 2 (LLL needs P to be exponential): Let us suppose that we could somehow use LLL to ensure that $\|\mathbf{a}\|_{co} \leq 2^{(n-1)/2}$. Then, we need P to be at least $2^{(n+1)/2}$ for the strategy to work. But then $\mathbf{v}^{P-1}$ is so long that it takes exponential time even to write it down.

The algorithm resolves these two issues with the following two tools:

- Tool 1 (Implicit Lattice Reduction): We apply LLL *implicitly* to the *multiplicands* of $\mathbf{v}^{P-1}$ to ensure that $\mathbf{a} = \mathbf{w}/\mathbf{v}^{P-1}$ has length at most $2^{(n-1)/2}$. The idea is that the relative norm $\mathbf{v} \cdot \overline{\mathbf{v}}$ actually reveals a lot about the "geometry" of $\mathbf{v}$ (and hence of $\mathbf{v}^{P-1}$). We use the relative norm to "cancel" $\mathbf{v}^{P-1}$'s geometry so that LLL implicitly acts on the multiplicands.

- Tool 2 (Polynomial Chains): We use $P > 2^{(n+1)/2}$. However, we never compute on $\mathbf{v}^{P-1}$ directly. Instead, $\mathbf{v}^{P-1}$ and $\mathbf{w}$ are represented implicitly via a chain of polynomials that are computed using LLL. From this chain, we compute $\mathbf{a} = [\mathbf{w}]_P$ exactly. Next, we perform computations modulo a set of small primes $p_1, \ldots, p_t$ – specifically, we reduce $\mathbf{a}$ modulo the p_i's, and use the polynomial chain to compute $\mathbf{v}^{P-1}$ modulo the p_i's. We do the same thing for another large prime P' such that $\gcd(P-1, P'-1) = 2m$, and then use the Euclidean algorithm (in the exponent) to compute $\mathbf{v}^{2m}$ modulo the p_i's. We chose the p_i's so that $2\|\mathbf{v}^{2m}\|_{co} < \prod p_i$, so we obtain $\mathbf{v}^{2m}$ exactly, from which we can compute $\mathbf{v}$ efficiently.

Below, we discuss the GS algorithm in detail. We begin with implicit lattice reduction, as characterized by the following lemma.

Implicit Lattice Reduction.

Lemma 6 ([GS02])**.** *Let $\mathbf{v} \in R$. Given $\mathbf{v} \cdot \overline{\mathbf{v}}$ and the HNF basis B for the ideal lattice $\langle \mathbf{v} \rangle$, we can output an element $\mathbf{w} \in \langle \mathbf{v} \rangle$ such that $\mathbf{w} = \mathbf{v} \cdot \mathbf{a}$ and $\|\mathbf{a}\|_2 \leq 2^{(n-1)/2} \cdot \sqrt{n}$ in time polynomial in m and the bit-length of $\mathbf{v}$.*

Proof. Consider how LLL works. LLL maintains a sequence of n basis vectors $(\overrightarrow{w_1}, \ldots, \overrightarrow{w_n})$. In general, when LLL is deciding whether to perform an operation – a size-reduction step or a swap step – the only information that LLL requires are all of the mutual dot products $\langle \overrightarrow{w_i}, \overrightarrow{w_j} \rangle_{i,j \in [n]}$. In short, LLL needs only the Gram matrix corresponding to its reduced-so-far lattice basis.

Now, consider LLL in our setting, as applied to ideal lattices under the canonical embedding (without trying to do LLL implicitly yet). At a given stage, LLL has a sequence of vectors $(\sigma(\mathbf{w}_1), \ldots, \sigma(\mathbf{w}_n))$ where the $\mathbf{w}_i$'s are in $\langle \mathbf{v} \rangle$. LLL (as before) considers only the mutual (Hermitian) inner products of the vectors in deciding whether to perform a step. These inner products are of the form $\langle \sigma(\mathbf{w}_i), \sigma(\mathbf{w}_j) \rangle = \sum_{k \in \mathbb{Z}_m^*} \sigma_k(\mathbf{w}_i \overline{\mathbf{w}_j})$.

Now, to do LLL *implicitly* in the canonical embedding – i.e., to use LLL to reduce the multiplicands $\mathbf{a}_i = \mathbf{w}_i/\mathbf{v}$ – LLL needs the mutual Hermitian inner products for $i, j \in [n]$:

$$\langle \sigma(\mathbf{w}_i/\mathbf{v}), \sigma(\mathbf{w}_j/\mathbf{v}) \rangle = \sum_{k \in \mathbb{Z}_m^*} \sigma_k(\mathbf{w}_i/\mathbf{v})\overline{\sigma_k(\mathbf{w}_j/\mathbf{v})} = \sum_{k \in \mathbb{Z}_m^*} \sigma_k(1/\mathbf{v}\overline{\mathbf{v}})\sigma_k(\mathbf{w}_i\overline{\mathbf{w}_j}).$$

But all of the values $\sigma_k(1/\mathbf{v}\overline{\mathbf{v}})$ can be computed efficiently from $\mathbf{v} \cdot \overline{\mathbf{v}}$ (and the implicit LLL algorithm actually possesses all of the vectors $\{\sigma(\mathbf{w}_i)\}$). Therefore, LLL has all of the information it needs to decide whether to perform a step. To actually perform a step implicitly – size-reduction or swapping – it simply applies the linear transformation dictated by the step to the vectors $\{\sigma(\mathbf{w}_i)\}$ that it has in its hand.

The bound $\|\mathbf{a}\|_2 \leq 2^{(n-1)/2} \cdot \sqrt{n}$ follows from the guarantee of LLL and the fact $\|1\|_2 = \sqrt{n}$ in the canonical embedding. □

Polynomial Chains.

Lemma 7 (Theorem 1 in [GS02]). *Let $\mathbf{v}_0 \in R$. Let $k = \sum k_i 2^i$ with $k_i \in \{0, 1\}$ be an integer with $r = \lfloor \log_2 k \rfloor$. Let P be a prime such that $\mathbf{v}_0$ is not a zero divisor in R_P. Then, given the input $\mathbf{v}_0 \cdot \overline{\mathbf{v}_0}$ and a basis B_0 of $\langle \mathbf{v}_0 \rangle$, we may compute, in time polynomial in r, m, and the bit-length of the input, the chains:*

$$\{\mathbf{v}_0^{k_{r-1}} \cdot \mathbf{v}_0^2 \cdot \overline{\mathbf{v}_1}, \ldots, \mathbf{v}_0^{k_0} \cdot \mathbf{v}_{r-1}^2 \cdot \overline{\mathbf{v}_r}\} \quad \text{and}$$
$$\{\mathbf{v}_0 \cdot \overline{\mathbf{v}_0}, \ldots, \mathbf{v}_{r-1} \cdot \overline{\mathbf{v}_{r-1}}\},$$

where for all $i > 0$, no $\mathbf{v}_i$ is a zero divisor in R_P, and $\|\mathbf{v}_i\|_2 < 2^{(n-1)/2}\sqrt{n}$. Using these chains, we may compute $\mathbf{v}_0^k \cdot \overline{\mathbf{v}_r} \bmod P$ in polynomial time. If $k = P - 1 \geq 2^{(n+1)/2}\sqrt{n}\gamma_2$ with $P = 1 \bmod 2m$, we may compute $\overline{\mathbf{v}_r}$ exactly, and thereafter use the above chains to compute $\mathbf{v}_0^{P-1} \bmod Q$ in polynomial time for any prime Q such that $\overline{\mathbf{v}_r}$ is not a zero divisor in R_Q.

Proof. (Sketch) Consider the first term of the first chain: $\mathbf{v}_0^{k_{r-1}} \cdot \mathbf{v}_0^2 \cdot \overline{\mathbf{v}_1}$. For convenience, let $c = k_{r-1} + 2$. Given $\mathbf{v}_0 \cdot \overline{\mathbf{v}_0}$ and a basis B_0 for $\langle \mathbf{v}_0 \rangle$, we efficiently compute $\mathbf{v}_0^c \cdot \overline{\mathbf{v}_0}^c$ and a basis B_0' for the ideal $\langle \mathbf{v}_0^c \rangle$. Then, using implicit lattice reduction (Lemma 6), we efficiently compute $\mathbf{w} = \mathbf{v}_0^c \cdot \mathbf{a}$ with $\|\mathbf{a}\|_2 < 2^{(n-1)/2}\sqrt{n}$. We set $\mathbf{w}$ to be the first term of our chain and set $\mathbf{v}_1 \leftarrow \overline{\mathbf{a}}$. (Gentry and Szydlo provide techniques to handle the small possibility that $\mathbf{v}_1$ is a zero divisor in R_P.)

Now, we compute $\mathbf{v}_1 \cdot \overline{\mathbf{v}_1}$ as $\mathbf{w} \cdot \overline{\mathbf{w}}/(\mathbf{v}_0^c \cdot \overline{\mathbf{v}_0}^c)$. Also, we compute a basis B_1 of $\langle \mathbf{v}_1 \rangle$, as follows. Since B_0' generates $\langle \mathbf{v}_0^c \rangle$, the terms of the basis B_0' of $\langle \mathbf{v}_0^c \rangle$ have the form $\mathbf{b}_i = \mathbf{v}_0^c \cdot \mathbf{a}_i$, where $R = \langle \{\mathbf{a}_i\} \rangle$. Our basis B_1 of $\langle \mathbf{v}_1 \rangle$ consists of the terms $\mathbf{b}_i \cdot \overline{\mathbf{w}}/(\mathbf{v}_0^c \cdot \overline{\mathbf{v}_0}^c) = \mathbf{v}_1 \cdot \mathbf{a}_i$, which generates $\langle \mathbf{v}_1 \rangle$ since (again) $R = \langle \{\mathbf{a}_i\} \rangle$.

Now that we have $\mathbf{v}_1 \cdot \overline{\mathbf{v}_1}$ and a basis B_1 of $\langle \mathbf{v}_1 \rangle$, we continue the same process iteratively to compute all of the terms in the chains.

We compute $\mathbf{v}_0^k \cdot \overline{\mathbf{v}_r} \bmod P$ iteratively, as follows. For $s \leq r$, let $k^{(s)} \in [0, 2^{s+1} - 1]$ denote the $s + 1$ MSBs of k. Suppose, inductively, that we have computed $\mathbf{v}_0^{k^{(s)}} \cdot \overline{\mathbf{v}_s} \bmod P$. (For $s = 1$, this term already exists in the polynomial chain.) Then, we compute

$$\mathbf{v}_0^{k^{(s+1)}} \cdot \overline{\mathbf{v}_{s+1}} = (\mathbf{v}_0^{k^{(s)}} \cdot \overline{\mathbf{v}_s})^2 \cdot (\mathbf{v}_0^{k_{r-s-1}} \cdot \mathbf{v}_s^2 \cdot \overline{\mathbf{v}_{s+1}})/(\mathbf{v}_s \cdot \overline{\mathbf{v}_s})^2 \bmod P$$

where the latter two multiplicands on the right-hand-side come from the polynomial chains. (Notice that this iterative computation is rather similar to the repeated squaring approach to modular exponentiation.)

We compute $\overline{\mathbf{v}_r}$ exactly as $\mathbf{v}_0^{P-1} \cdot \overline{\mathbf{v}_r} \bmod P$. (This works since the coefficients of $\overline{\mathbf{v}_r}$ have magnitude at most $\|\mathbf{v}_i\|_2\gamma_2 \leq 2^{(n-1)/2}\sqrt{n}\gamma_2 < P/2$.) Thereafter, we clearly can compute $\mathbf{v}_0^{P-1}$ modulo any prime Q for which $\overline{\mathbf{v}_r}$ is not a zero divisor in R_Q. □

Remainders of the GS Algorithm.

Lemma 8 (Theorem 2 in [GS02]). *Let $\mathbf{v} \in R$. Then, given $\mathbf{v} \cdot \overline{\mathbf{v}}$ and a basis B of $\langle \mathbf{v} \rangle$, we may compute $\mathbf{v}^{2m}$ in time polynomial in m and the bit length of $\mathbf{v}$.*

Proof. We choose primes P and P' each large enough for Lemma 7, where $\gcd(P-1, P'-1) = 2m$ and $\mathbf{v}$ is not a zero divisor in either R_P or $R_{P'}$ (using Dirichlet's theorem on primes in arithmetic progression and the fact that $\mathbf{v}$ may be a zero divisor in R_Q for only a finite number of primes Q). By Lemma 7, we can compute chains that will allow us to compute $\mathbf{v}^{P-1} \bmod p_i$ and $\mathbf{v}^{P'-1} \bmod p_i$ in polynomial time for any prime p_i such that the values $\overline{\mathbf{v}_r}$ and $\overline{\mathbf{v}_r}'$ in the chains are not zero divisors in R_{p_i}. Choose a set of primes $p_1, \ldots, p_t$ that satisfy this condition and such that $2\|\mathbf{v}^{2m}\|_{co} < \prod p_i$. (We simply avoid the finite number of problematic primes.) Apply the Euclidean algorithm in the exponent to compute $\mathbf{v}^{2m}$ modulo each p_i, and ultimately $\mathbf{v}^{2m}$ exactly using the Chinese Remainder Theorem. □

Lemma 9 (Similar to [GS02]). *Let $\mathbf{v} \in R$. Let $\mathbf{w} = \mathbf{v}^r$ where $2m$ divides r. Then, given $\mathbf{w}$, we may output a list L of r values $\mathbf{v}_1, \ldots, \mathbf{v}_r$ in time polynomial in r and the bit length of $\mathbf{w}$, such that L includes $\mathbf{v}$.*

Lemma 9 may seem trivial, and it certainly would be if r and m were relatively prime. In this case, one could simply pick a prime $Q > 2\|\mathbf{v}\|_{co}$ with $\gcd(r, Q-1) = 1$, set $s = r^{-1} \bmod m(Q-1)$, and compute $\mathbf{w}^s = \mathbf{v}^{rs} = \mathbf{v}^{1+km(Q-1)} = \mathbf{v}$ in R_Q (by Fermat's Little Theorem), which yields $\mathbf{v}$ exactly. Things become more complicated when $\gcd(r, m) \neq 1$.

Proof. First, we observe that $\mathbf{w}$ does not uniquely determine $\mathbf{v}$. Specifically, for any $\mathbf{e} = \pm x^i \in R$ (the $2m$ values that are plus or minus an m-th root of unity in R), we have that $\mathbf{v} \cdot \mathbf{e}$ is also in R and $\mathbf{w} = (\mathbf{v} \cdot \mathbf{e})^r$. However, we show that fixing $\mathbf{v}$'s value at any (complex) primitive m-th root of unity ζ_m also fixes $\mathbf{v}$'s value at the other primitive m-th roots of unity, after which we may obtain $\mathbf{v}$ via interpolation. Given $\mathbf{w}(\zeta_m) = \mathbf{v}(\zeta_m)^r$, there are only r possibilities for $\mathbf{v}(\zeta_m)$. By iterating the procedure below for each possibility of $\mathbf{v}(\zeta_m)$, the procedure will eventually use the "correct" value, and the correct value of $\mathbf{v}$ will be included in the output.

For any prime Q, by an extension of Fermat's Little Theorem, we have that $\mathbf{a}(x)^Q = \mathbf{a}(x^Q)$ in the ring R_Q. Let $Q = cr - b$ be a prime for some positive integers $b < r$ and c such that $\mathbf{w}$ is not a zero divisor in R_Q and $\gamma_\infty \cdot \|\mathbf{w}\|_\infty < Q/2$. (Recall that γ_∞ denotes the maximal value of $\|\mathbf{a}\|_{co}/\|\mathbf{a}\|_\infty$ for $\mathbf{a} \in K$.) Given that m divides r, we compute that $(\mathbf{v}^r)^c = \mathbf{v}^Q\mathbf{v}^b = \mathbf{v}(x^Q)\mathbf{v}^b = \mathbf{v}(x^{-b})\mathbf{v}^b \bmod Q$. Since $\gamma_\infty \cdot \|\mathbf{v}(x^{-b})\mathbf{v}^b\|_\infty \leq \gamma_\infty \cdot \|\mathbf{w}\|_\infty < Q/2$, we efficiently recover the term $\mathbf{z}_b \leftarrow \mathbf{v}(x^{-b})\mathbf{v}^b$ exactly. This allows us to compute $\mathbf{v}(\zeta_m^{-b}) = \mathbf{z}_b(\zeta_m)/\mathbf{v}(\zeta_m)^b$. By choosing other Q's, we similarly compute $\mathbf{z}_b$ for each $b \in \mathbb{Z}_m^*$, thereby compute $\mathbf{v}(\zeta)$ for all complex primitive m-th roots of unity ζ, and thus recover $\mathbf{v}$. □

Theorem 4 ([GS02]). *Let $\mathbf{v} \in R$. Given $\mathbf{v} \cdot \overline{\mathbf{v}}$ and the HNF basis B for the ideal lattice $\langle \mathbf{v} \rangle$, we can compute $\mathbf{v}$ in time polynomial in m and the bit-length of $\mathbf{v}$.*

Proof. This follows from Lemmas 8 and 9. □

Some Extensions.

Howgrave-Graham and Szydlo [HGS04] observed that one can use the GS algorithm to recover $\mathbf{v}$ from the relative norm $\mathsf{N}_{K/K^+} = \mathbf{v} \cdot \overline{\mathbf{v}}$ *without* a basis of $\langle \mathbf{v} \rangle$, as long as one has a factorization of $\mathsf{N}_{K/Q}(\mathbf{v} \cdot \overline{\mathbf{v}}) = \mathsf{N}_{K/Q}(\mathbf{v})^2$. The idea is that, from $\mathsf{N}_{K/K^+} = \mathbf{v} \cdot \overline{\mathbf{v}}$ and the factorization, one can use Kummer-Dedekind (Theorem 3) to generate a basis of some $\mathbf{v}'$ such that $\mathbf{v}' \cdot \overline{\mathbf{v}'} = \mathbf{v} \cdot \overline{\mathbf{v}}$ ($\mathbf{v}$ may not be unique). If $\mathsf{N}_{K/Q}(\mathbf{v})$ is composite, one can compute its factorization using a classical sub-exponential factorization algorithm such as the number field sieve [LLMP90, LL93] or Shor's polynomial-time quantum algorithm [Sho97a].

Another way to view the GS and HS algorithms is the following. The averaging attack yields the *Gram matrix* (essentially the co-variance matrix) $B_{priv}^T \cdot B_{priv}$ associated to the secret lattice basis of the signer. In early NTRU signature schemes, this Gram matrix happened to have a very special form; it corresponded to the relative norm $\mathsf{N}_{K/K^+}(\mathbf{v}) = \mathbf{v} \cdot \overline{\mathbf{v}}$. The GS and HS algorithms are able to *factor the Gram matrix* in this special case (using the auxiliary information $\langle \mathbf{v} \rangle$ in the case of the GS algorithm).

The NTRUSign signature scheme [HHGP+03] was proposed shortly after the Gentry-Szydlo attack was announced. As noted in [GS02, HGS04], for NTRUSign, applying an averaging attack similar to that described in Section 7.2 still yields the Gram matrix $B_{priv}^T \cdot B_{priv}$ associated to the secret lattice basis of the signer. However, the Gram matrix in NTRUSign has a more complicated form than in previous NTRU signature schemes. In particular, it is a 2×2 block of ring elements:

$$B_{priv}^T \cdot B_{priv} = \begin{bmatrix} \mathbf{v} \cdot \overline{\mathbf{v}} + \mathbf{V} \cdot \overline{\mathbf{V}} & \mathbf{w} \cdot \overline{\mathbf{v}} + \mathbf{W} \cdot \overline{\mathbf{V}} \\ \mathbf{v} \cdot \overline{\mathbf{w}} + \mathbf{V} \cdot \overline{\mathbf{W}} & \mathbf{w} \cdot \overline{\mathbf{w}} + \mathbf{W} \cdot \overline{\mathbf{W}} \end{bmatrix}$$

where $\mathbf{v}$, $\mathbf{w}$, $\mathbf{V}$ and $\mathbf{W}$ are short elements that constitute the signer's private key. It remains an open problem to efficiently factor Gram matrices of this form (as well as general Gram matrices), even when given a basis (e.g., the HNF basis) of the lattice generated by B_{priv}. Szydlo [Szy03] showed that the Gram matrix factorization problem can be reduced to an oracle that distinguishes whether two Gram matrices are associated to bases of the same lattice, but it is unknown how to instantiate this oracle efficiently in general.

The GS algorithm suggests an open problem about other relative norms: Is it possible to efficiently recover $\mathbf{v}$ from $\langle \mathbf{v} \rangle$ and the relative norm $\mathsf{N}_{K/L}(\mathbf{v})$ when L is some subfield of K other than the index-2 real subfield K^+? When $L = \mathbb{Q}$, this is just the Principal Ideal Generator problem, which seems infeasible in general, but perhaps the problem is feasible when the index $[K : L]$ is small or smooth. For example, suppose K is the m-th cyclotomic field for $m = 2^k$ and L is an index-4 subfield. In this case, can we efficiently recover $\mathbf{v}$ from $\langle \mathbf{v} \rangle$ and $\mathsf{N}_{K/L}(\mathbf{v})$? Can we, perhaps, first recover $\mathsf{N}_{K/K^+}(\mathbf{v})$ from $\langle \mathbf{v} \rangle$ and $\mathsf{N}_{K/L}(\mathbf{v})$, and then use the GS algorithm to recover $\mathbf{v}$? It seems doubtful, since the GS algorithm relies implicitly on the fact that $\langle \mathbf{v} \rangle$ and $\mathsf{N}_{K/K^+}(\mathbf{v})$ define $\mathbf{v}$ uniquely up to torsion units, due to the special relationship between the cyclotomic units and the subfield K^+.

We remark that it is interesting that, while the GS algorithm clearly relies on the structure of the cyclotomic unit group, this reliance is implicit; it would be worthwhile to make the connection more explicit.

7.4 Nguyen-Regev: A Gradient Descent Attack

Nguyen and Regev [NR09] described how to extend averaging and key recovery attacks to signature schemes based on *general lattices* – in particular, to lattices underlying the GGH [GGH97] and NTRUSign [HHGP+03] signature schemes (for suggested parameters). These attacks show that averaging a transcript of lattice-based signatures can be a devastating attack in general, and further recommend the approach taken by [GPV08] of ensuring that the distribution of signatures has some canonical distribution (e.g., a Gaussian distribution) that is essentially independent of the particular lattice basis that the signer is using.

Their attack is designed to "learn a parallelepiped". That is, given samples $\{B_{priv} \cdot \overrightarrow{\mathbf{y}_i}\}$ where the $\mathbf{y}_i$'s are (discretely) uniform over a hypercube, their attack converges upon the shape of $\mathcal{P}(B_{priv})$ and ultimately outputs the private basis B_{priv}.

To understand the NR attack, it might help to understand why previous attacks failed to break GGH and NTRUSign. Previous attacks, were (in some sense) too modular. They divided the attack into two parts: 1) an averaging/covariance/second-moment attack which used samples $\{B_{priv} \cdot \overrightarrow{\mathbf{y}_i}\}$ to recover the Gram matrix $B_{priv}^T \cdot B_{priv}$ associated to the secret lattice basis of the signer, and 2) a "factoring" attack that either factored the relative norm [GS02, HGS04] or otherwise tried to factor the Gram matrix [Szy03]. The second step, the factoring attack, sometimes used a lattice basis as auxiliary information (as in the GS algorithm). But, crucially, the second step *did not use the samples*. After using the samples to obtain the Gram matrix (and a lattice basis), previous attacks simply discarded the samples. In this case, key recovery reduces to the Gram matrix factorization problem (with a lattice basis), for which no general polynomial-time algorithm is known.

In contrast, the NR algorithm is (in some sense) less modular. They use the samples throughout the attack. In particular, they first show that the 4-th moment (also known as the *kurtosis*) of a transcript of signatures defines a global minimum related to the secret key. (Recall that, for a set of vectors $B = \{\overrightarrow{b_1}, \ldots, \overrightarrow{b_n}\} \in GL_n(\mathbb{R})$, the k-th moment of the parallelepiped $\mathcal{P}(B)$ over a vector $\overrightarrow{w}$ is defined as $\mathrm{mom}_{B,k}(\overrightarrow{w}) = \mathrm{Exp}[\langle \overrightarrow{u}, \overrightarrow{w} \rangle^k]$ where $\overrightarrow{u}$ is chosen uniformly over $\mathcal{P}(B)$.)

Lemma 10 (Lemma 3 in [NR09])**.** *Let $B = \{\overrightarrow{b_1}, \ldots, \overrightarrow{b_n}\} \in GL_n(\mathbb{R})$. Then the global minimum of $\mathrm{mom}_{B,4}(\overrightarrow{w})$ over the unit sphere of $\mathbb{R}^n$ is $1/5$ and this minimum is obtained at $\pm\overrightarrow{b_1}, \ldots, \pm\overrightarrow{b_n}$. There are no other local minima.*

Then, they use *gradient descent* to find this global minimum approximately, using the samples at each stage of the descent to approximate the gradient function. This leads to the following theorem.

Theorem 5 (Theorem 4 in [NR09])**.** *For any $c_0 > 0$ there exists a $c_1 > 0$ such that given n^{c_1} samples uniformly distributed over some parallelepiped $\mathcal{P}(B)$, $B = \{\overrightarrow{b_1}, \ldots, \overrightarrow{b_n}\} \in GL_n(\mathbb{R})$, the approximate gradient descent algorithm outputs with constant probability a vector $B \cdot \tilde{\mathbf{e}}$ where $\tilde{\mathbf{e}}$ is within ℓ_2 distance n^{-c_0} of some standard basis vector $\mathbf{e}_i$.*

Assuming the approximate solution output by the NR algorithm is "good enough" – that is, good enough to obtain B exactly via rounding – the NR attack succeeds. The secret bases in GGH and NTRUSign have small entries (polynomial in the security parameter), and so the NR attack succeeds asymptotically with only a polynomial number of signatures, also also performs quite well in practice for suggested parameters.

One issue that the NR attack leaves somewhat unresolved is: What happens when the approximate solution output by the NR algorithm is not "good enough" to use rounding to get the

exact solution? Nguyen and Regev suggest using a CVP approximation algorithm, which they observe performs reasonably well in practice on suggested parameters, but which of course is not polynomial-time in general. This is a weakness also of the averaging attack described in Section 7.2. This weakness suggests an obvious way of fixing the schemes: choose the secret basis so that its entries are *super-polynomial* or even *sub-exponential* integers, so that averaging attacks cannot approximate the entries of the basis precisely enough to obtain them exactly via rounding. (Of course, this makes the cryptographic construction less practical, but still polynomial-time.)

In Section 7.6, we describe an attack that casts doubt on this fix, at least in the context of ideal lattices. We show that we can recover $\mathbf{v}$ from $\langle \mathbf{v} \rangle$ and a ϵ-approximation $\mathbf{u}$ of $\mathbf{v}$ when ϵ is inverse-quasi-polynomial, even when the coefficients of $\mathbf{v}$ are arbitrarily large.

7.5 Ducas-Nguyen: Gradient Descent over Zonotopes and Deformed Parallelepipeds

The Nguyen-Regev algorithm was designed to "learn a parallelepiped", Ducas and Nguyen [DN12b] showed how to extend the algorithm to learn more complicated shapes, including zonotopes and deformed parallelepipeds.

Recall that the parallelepiped associated to a basis $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ is the set $\mathcal{P}(B) = \{\sum x_i \cdot \mathbf{b}_i : x_i \in [-1/2, 1/2)\}$. Under certain circumstances (see Section 7.4), Nguyen-Regev learns the parallelepiped $\mathcal{P}(B)$ from samples of the form $\{B \cdot \boldsymbol{r}\}$, where $\boldsymbol{r} = (r_1, \ldots, r_n)$ is (discretely) uniform over an n-dimensional hypercube. This algorithm breaks certain signature schemes, such as the basic version of NTRUSign [HHGP$^+$03], where a transcript of signatures implicitly provides samples $\{B_{priv} \cdot \boldsymbol{r}\}$ where B_{priv} is the signer's private basis. A zonotope is a generalization of a parallelepiped to a dependent set of vectors. Let $M = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$ be a $n \times m$ matrix for $m > n$. The zonotope formed by M is the set $\mathcal{Z}(M) = \{\sum x_i \cdot \mathbf{b}_i : x_i \in [-1/2, 1/2)\}$. Even though the vectors of M are dependent and the zonotope has a shape that is "closer to spherical" than a parallelepiped (the corners typically have more obtuse angles), Ducas and Nguyen show the Nguyen-Regev algorithm can be extended to this setting, when the samples have the form $\{M \cdot \boldsymbol{r}\}$, where $\boldsymbol{r}$ is (discretely) uniform over an m-dimensional hypercube. Their new algorithm does not provably always work, but it works quite well in practice. They used their algorithm to break a version of NTRUSign with a "perturbations" countermeasure. In NTRUSign with perturbations, the signer uses perturbations to obscure its private basis, in such a way that a transcript of signatures induces the distribution of a zonotope rather than a parallelepiped.

Can the Nguyen-Regev and Ducas-Nguyen algorithms be extended even further? For example, suppose we have samples of the form $\{B \cdot \boldsymbol{r}\}$ or $\{M \cdot \boldsymbol{r}\}$, where $\boldsymbol{r}$ comes from a discrete Gaussian distribution. In these cases, assuming that the coordinates of $\boldsymbol{r}$ have moderate deviation, one can show [Pei10, AGHS12] that the samples also have a discrete Gaussian distribution over the lattice generated by B or M, where the Gaussian is ellipsoidal according to the shape of B or M. In the latter case, the ellipsoid get closer to a sphere as m gets larger relative to n (in the sense that the singular values of M get closer together). A discrete ellipsoidal Gaussian does not have any "corners" like a parallelepiped or zonotope, which are the local minima of the Nguyen-Regev and Ducas-Nguyen algorithms. This fact seems to prevent a direct application of Nguyen-Regev or Ducas-Nguyen. However, the shape of the ellipsoid still may provide some useful information.[6]

[6] For signature schemes, the signer can use the Gaussian samplers from [GPV08, Pei10] to get a perfectly spherical distribution, thus ensuring that the transcript of signatures "leaks no information at all."

Interestingly, the re-randomization algorithm of our construction (see Section 4) involves adding a term of the form $(M \cdot \boldsymbol{r})/\mathbf{z}$, where $\boldsymbol{r}$ has a spherical Gaussian distribution. Consequently, the numerator of this added term has an ellipsoidal Gaussian distribution, where the numerator's shape depends on the shape of M. Note that as opposed to the case of signatures, re-randomization in our construction is not supposed to hide M (in fact we give out $M/\mathbf{z}$ in the public parameters). Rather, the purpose of re-randomization in is just to "drown out" the initial value that is being randomized (while preserving its coset wrt the ideal $\mathcal{I}$).

7.6 A New Algorithm for the Closest Principal Ideal Generator Problem

As usual, let R be the ring of integers for the m-th cyclotomic field. Let $\mathbf{v} \in R$ and $\mathcal{I} = \langle \mathbf{v} \rangle$. Let $\mathbf{u}$ be a ϵ-approximation of $\mathbf{v}$ – i.e., $1/(1+\epsilon) \leq |\sigma_k(\mathbf{v})/\sigma_k(\mathbf{u})| \leq 1+\epsilon$ for all $k \in \mathbb{Z}_m^*$. How efficiently can we recover the principal ideal generator $\mathbf{v}$ from $\mathcal{I}$ and $\mathbf{u}$?

A cryptanalyst would hope that we can recover $\mathbf{v}$ whenever ϵ is bounded by some inverse-polynomial function, so that the averaging and Nguyen-Regev attacks become more devastating. Recall that the averaging and Nguyen-Regev attacks only output a $1/\mathsf{poly}$-approximate solution of $\mathbf{v}$ (or a related value) when given a polynomial number of samples; afterward, the attacks attempt to output an exact solution by rounding (or by solving approximate-CVP, but this is not efficient in general). Thus, the averaging and Nguyen-Regev attacks can easily be escaped by choosing $\mathbf{v}$ so that its coefficients are super-polynomial in size. However, a cryptanalyst could prevent this escape with an efficient algorithm to recover $\mathbf{v}$ from a $1/\mathsf{poly}$-approximation of $\mathbf{v}$, since this would break the scheme regardless of how large $\mathbf{v}$'s coefficients are.

Here, we show how to recover $\mathbf{v}$ in time polynomial in m and the bit-length of $\mathbf{v}$, assuming that ϵ is bounded by some inverse-quasi-polynomial function in m. This algorithm does not quite fulfill the cryptanalyst's dream, but it suggests a direction for future, possibly more devastating attacks. The algorithm that we describe here is a natural extension of the Gentry-Szydlo algorithm ([GS02], see Section 7.3). Whereas the GS algorithm uses the *exact* information about $\mathbf{v}$'s geometry provided by the relative norm $\mathsf{N}_{K/K^+}(\mathbf{v}) = \mathbf{v} \cdot \overline{\mathbf{v}}$, our algorithm here tries to make-do with the approximate information provided by $\mathbf{u}$.

The algorithm follows the algebraic strategy of the GS algorithm. In particular, it invokes Fermat's Little Theorem to assert that $\mathbf{v}^r = 1 \bmod P$ for prime P when $(P-1)$ and m divide r (as long as $\mathbf{v}$ is not a zero divisor in R_P). Next, it applies (implicit) lattice reduction to the lattice $\mathcal{I}^r$ to obtain a reduced element $\mathbf{w} = \mathbf{v}^r \cdot \mathbf{a}$. Finally, it tries to recover $\mathbf{a}$ (and hence $\mathbf{v}$) by using the fact that $\mathbf{a} = \mathbf{w} \bmod P$. The main differences between the GS algorithm and our algorithm are:

- We require r to be only quasi-polynomial (not exponential): The GS algorithm has exact information about $\mathbf{v}$'s geometry, which allows it to derive exact information about $\mathbf{v}^r$'s geometry even when r is exponential (though this information is represented implicitly in the polynomial chains). In contrast, we only have approximate information about $\mathbf{v}$'s geometry, and the accuracy of our information about $\mathbf{v}^r$'s geometry degrades exponentially with r. So, we cannot have r much bigger than $1/\epsilon$.

- We will work modulo the product of many primes: To compensate for the fact that r cannot be too large in our setting, we choose r so that $(p_i - 1)$ divides r for *many* primes p_i, and we work modulo $P = \prod p_i$. We heuristically estimate that we can achieve $P = 2^{\Omega(m)}$ when $r = 2^{O(\log m \log\log m)}$. (Similar to the GS algorithm, we need P to exceed the LLL approximation factor, and then some.)

Let us begin by considering how to set r and P. For some k to be determined, let $q_1, \ldots, q_k$ be the first k primes, and set $r_{k,m} = m \prod q_i$. Set $\mathcal{S}_{k,m}$ be the set of 2^k products of m with a subset product of $q_1, \ldots, q_k$. Set $\mathcal{T}_{k,m} = \{1 + s : s \in S_{k,m}\}$, $\mathcal{P}_{k,m} = \{\text{prime } p \in T_{k,m}\}$, and $P_{k,m} = \prod_{p \in \mathcal{P}_{k,m}} p$. We claim that $(r_{k,m}, P_{k,m})$ will tend to be a good choice for (r, P). Certainly it is true that $r_{k,m}$ is divisible by $p_i - 1$ for the primes that divide P; the remaining issue is the size of $r_{k,m}$ and $P_{k,m}$.

First, consider the size of $r_{k,m}$. We have:

$$\ln r_{k,m} = \ln m + \sum_{i=1}^{k} \ln q_i = \ln m + q_k + o(k) = \ln m + k \ln k + o(k \ln k),$$

where the second and third equalities follow from extensions of the Prime Number Theorem (see Corollaries 8.2.7 and 8.2.8 in [BS96]). Assuming $k \ln k$ dominates m, we have $r_{k,m} = 2^{(1+o(1))k \ln k}$.

Now, consider the size of $P_{k,m}$. Clearly, many elements of $\mathcal{T}_{k,m}$ are not prime. For example, $1 + s$ cannot be prime unless s is divisible by 2 – i.e., unless 2 is part of the subset product that forms s. Similarly, if s is a subset product not divisible by 3, then $1 + s$ has (roughly) only a $1/2$ (versus the usual $1/3$) probability of not being divisible by 3. But, aside from such observations, we would heuristically expect that, by the Prime Number Theorem, an element $t \in T_{k,m}$ has a $\Omega(1/\ln t)$ chance of being prime. With this heuristic, we calculate:

$$P_{k,m} = \prod_{p \in \mathcal{P}_{k,m}} p = \prod_{t \in \mathcal{T}_{k,m}} t^{\Omega(1/\ln t)} = 2^{\Omega(|\mathcal{T}_{k,m}|)} = 2^{\Omega(2^k)} .$$

Assuming these heuristic estimates of $r_{k,m}$ and $P_{k,m}$ are true, then for any constant c_1, there is a constant c_2, such that setting $k = \lfloor \ln m \rfloor + c_2$ ensures that $P_{k,m}$ is at least $2^{c_1 \cdot m}$. With this value of k, we have $r_{k,m} = 2^{(1+o(1)) \ln m \ln \ln m} = m^{(1+o(1)) \ln 2 \ln \ln m}$. In other words, while $P_{k,m}$ is exponential in m, $r_{k,m}$ is only slightly quasi-polynomial in m. For convenience, we capture these observations in the following claim.

Claim 1. *Let $\rho_m(x)$ denote the smallest positive integer such that there exist distinct primes $\{p_i\}$ such that $\prod p_i \geq x$ and $\rho_m(x)$ is divisible by m and $(p_i - 1)$ for all i. Then, for $x = 2^{\Omega(m)}$, we have $\rho_m(x) = 2^{(1+o(1)) \ln \ln x \ln \ln \ln x}$. For $x = 2^{\Theta(m)}$, we have $\rho_m(x) = m^{(1+o(1)) \ln \ln m}$. The "proof" of the claim is constructive – that is, one can (heuristically) generate a value $r_{k,m}$ that meets these asymptotic bounds of $\rho_m(x)$ by setting $r_{k,m}$ to be the product of m with the first $c + \ln \ln x$ primes for some constant c.*

Next, we revisit Lemma 7, adapting implicit lattice reduction and the polynomial chains of the GS algorithm to our setting.

Lemma 11 (Adaptation of Lemma 7)**.** *Let $\mathbf{v}_0 \in R$ and let B_0 be the HNF basis B_0 for the ideal lattice $\mathcal{I}_0 = \langle \mathbf{v}_0 \rangle$. Let $\mathbf{u}_0$ be an ϵ-approximation of $\mathbf{v}_0$ – i.e., $1/(1+\epsilon) \leq |\sigma_k(\mathbf{v}_0)/\sigma_k(\mathbf{u}_0)| \leq 1 + \epsilon$ for all $k \in \mathbb{Z}_m^*$. Let $k = \sum k_i 2^i$ with $k_i \in \{0, 1\}$ be an integer with $r = \lfloor \log_2 k \rfloor$. Let P be an integer such that $\mathbf{v}_0$ is not a zero divisor in R_P. Then, given the input $(B_0, \mathbf{u}_0)$, we may compute, in time polynomial in r, m, and the bit-length of the input, the chains:*

$$\{\mathbf{v}_0^{k_{r-1}} \cdot \mathbf{v}_0^2/\mathbf{v}_1, \ldots, \mathbf{v}_0^{k_0} \cdot \mathbf{v}_{r-1}^2/\mathbf{v}_r\}$$

where for all $i > 0$, no $\mathbf{v}_i$ is a zero divisor in R_P, and $\|\mathbf{v}_i\|_2 < 2^{(n-1)/2}\sqrt{n}(1+\epsilon)^{k^{(i)}}$, where $k^{(i)}$ is the integer formed by the $i + 1$ most significant bits of k. Using these chains, we may

compute $\mathbf{v}_0^k/\mathbf{v}_r \bmod P$ *in polynomial time. If* k *and* P *are such that* $\mathbf{v}_0^k = 1 \bmod P$ *and* $P > 2^{(n+1)/2}\sqrt{n}(1+\epsilon)^k\gamma_2$*, we may compute* $\mathbf{v}_r$ *exactly, and thereafter use the above chains to compute* $\mathbf{v}_0^k \bmod Q$ *in polynomial time for any prime* Q *such that* $\mathbf{v}_r$ *is not a zero divisor in* R_Q*.*

Proof. Consider the first term of the first chain: $\mathbf{v}_0^{k_{r-1}} \cdot \mathbf{v}_0^2/\mathbf{v}_1$. For convenience, let $c = 2k_r + k_{r-1}$. Given $(B_0, \mathbf{u}_0)$, we efficiently compute a basis B_0' for the ideal $\mathcal{I}_0' = \langle \mathbf{u}_0^c \rangle / \mathcal{I}^c$. Apply LLL to B_0'. Set $\mathbf{u}_1 \in \mathcal{I}_0'$ to be the element corresponding to the shortest vector in the reduced basis. Since $\mathcal{I}_0'$ is a principal (fractional) ideal, we have $\mathbf{u}_1 = (\mathbf{u}_0/\mathbf{v}_0)^c\mathbf{v}_1$ for some $\mathbf{v}_1 \in R$. (To handle the possibility that $\mathbf{v}_1$ is a zero divisor in R_P, use techniques by Gentry and Szydlo.) Since $\mathbf{v}_1 = \mathbf{u}_1 \cdot (\mathbf{v}_0/\mathbf{u}_0)^c$, we have that $\|\mathbf{v}_1\|_2 \leq 2^{(n-1)/2} \cdot \sqrt{n} \cdot (1+\epsilon)^c$ by the guarantee of LLL and the fact $\|\mathbf{v}_0^c/\mathbf{u}_0^c\|_\infty \leq (1+\epsilon)^c$. Include the term $\mathbf{u}_0^c/\mathbf{u}_1 = \mathbf{v}_0^c/\mathbf{v}_1$ in the polynomial chain. Observe that $\mathbf{u}_1$ is a $(1+\epsilon)^c$ approximation of $\mathbf{v}_1$. Also, we can efficiently generate a basis B_1 of the ideal $\mathcal{I}_1 = \langle \mathbf{v}_1 \rangle = \langle \mathbf{u}_1 \rangle / \mathcal{I}_0'$.

The second term in the chain is supposed to be $\mathbf{v}_0^{k_{r-2}} \cdot \mathbf{v}_1^2/\mathbf{v}_2$. Given $(B_0, B_1, \mathbf{u}_0, \mathbf{u}_1)$, we efficiently compute a basis B_1' for the ideal $\mathcal{I}_1' = \left\langle \mathbf{u}_0^{k_{r-2}}\mathbf{u}_1^2 \right\rangle / (\mathcal{I}_0^{k_{r-2}}\mathcal{I}_1^2)$. Apply LLL to B_1'. Set $\mathbf{u}_2 \in \mathcal{I}_1'$ to be the element corresponding to the shortest vector in the reduced basis. Since $\mathcal{I}_1'$ is a principal (fractional) ideal, we have $\mathbf{u}_2 = (\mathbf{u}_0/\mathbf{v}_0)^{k_{r-2}}(\mathbf{u}_1/\mathbf{v}_1)^2\mathbf{v}_2$ for some $\mathbf{v}_2 \in R$. (To handle the possibility that $\mathbf{v}_2$ is a zero divisor in R_P, use techniques by Gentry and Szydlo.) Since $\mathbf{v}_2 = \mathbf{u}_2 \cdot (\mathbf{v}_0/\mathbf{u}_0)^{k_{r-2}}(\mathbf{v}_1/\mathbf{u}_1)^2$, we have that $\|\mathbf{v}_2\|_2 \leq 2^{(n-1)/2} \cdot \sqrt{n} \cdot (1+\epsilon)^{4k_r+2k_{r-1}+k_{r-2}}$ by the guarantee of LLL and the fact $\|(\mathbf{v}_0/\mathbf{u}_0)^{k_{r-2}}(\mathbf{v}_1/\mathbf{u}_1)^2\|_\infty \leq (1+\epsilon)^{4k_r+2k_{r-1}+k_{r-2}}$. Include the term $\mathbf{u}_0^{k_{r-2}} \cdot \mathbf{u}_1^2/\mathbf{u}_2 = \mathbf{v}_0^{k_{r-2}} \cdot \mathbf{v}_1^2/\mathbf{v}_2$ in the polynomial chain. Observe that $\mathbf{u}_2$ is a $(1+\epsilon)^{4k_r+2k_{r-1}+k_{r-2}}$ approximation of $\mathbf{v}_2$. Also, we can efficiently generate a basis B_2 of the ideal $\mathcal{I}_2 = \langle \mathbf{v}_2 \rangle = \langle \mathbf{u}_2 \rangle / \mathcal{I}_1'$. One continues in this fashion until all the terms in the polynomial chain are computed.

The rest of the proof proceeds similar to the proof of Lemma 7. □

Since in Lemma 7 k may be super-polynomial, we prefer not to compute $\mathbf{v}_0^k$ directly. Instead, as in Lemma 8, we may compute $\mathbf{v}_0^{2m}$ by computing $\mathbf{v}_0^{k_1}$ and $\mathbf{v}_0^{k_2}$ for which $\gcd(k_1, k_2) = 2m$, and then applying the Euclidean algorithm in the exponent.

Lemma 12. *Let* $\mathbf{v} \in R$ *and let* B *be the HNF basis* B *for the ideal lattice* $\mathcal{I} = \langle \mathbf{v} \rangle$*. Let* $\mathbf{u}$ *be an* ϵ*-approximation of* $\mathbf{v}$ *– i.e.,* $1/(1+\epsilon) \leq |\sigma_k(\mathbf{v})/\sigma_k(\mathbf{u})| \leq 1+\epsilon$ *for all* $k \in \mathbb{Z}_m^*$*. Then, given* $\mathbf{u}$ *and* B*, we may compute* $\mathbf{v}^{2m}$ *in time polynomial in* m *and the bit length of* $\mathbf{v}$*.*

Proof. Similar to the proof of Lemma 8. □

Theorem 6. *Assuming Claim 1, there is an* $\epsilon = m^{-(1+o(1))\ln\ln m}$ *such that, given the HNF basis for the ideal lattice* $\mathcal{I} = \langle \mathbf{v} \rangle$ *for some* $\mathbf{v} \in R$ *and an* ϵ*-approximation* $\mathbf{u}$ *of* $\mathbf{v}$*, we can compute* $\mathbf{v}$ *in time polynomial in* m *and the bit-length of* $\mathbf{v}$*.*

Proof. This follows from Lemmas 12 and 9 and Claim 1. □

We remark that this algorithm implies that the bounded distance decoding problem (BDDP) is *easy* for the Dirichlet unit lattice Λ for surprisingly low approximation factors. (Recall from Section 7.1 that the Dirichlet unit lattice is the lattice formed by the image of the units under the map $\lambda : K^* \to \mathbb{R}^{s_1+s_2}$ given by $\lambda(\mathbf{a}) = (\ln|\sigma_1(\mathbf{a})|, \ldots, \ln|\sigma_{s_1+s_2}(\mathbf{a})|)$.) Specifically, by the above algorithm, given an ϵ-approximation $\mathbf{u}$ of a unit $\mathbf{v}$, we can recover $\mathbf{v}$ exactly. So, in the Dirichlet unit lattice, taking logarithms, given a vector $\lambda(\mathbf{u})$ whose ℓ_∞ distance from Λ is at most $\ln(1+\epsilon) \approx \epsilon$,

we can efficiently recover the vector in Λ-vector closest to $\lambda(\mathbf{u})$. Really, this corollary is not *so* surprising, since in the case of the m-th cyclotomic field for prime power m we already have in our hands a fairly short basis of Λ given by the basis $\{\lambda(\mathbf{b}_i) : \mathbf{b}_i = (1-\zeta_m^i)/(1-\zeta_m) : i \in \mathbb{Z}_m^*\}$, which gives more direct ways of achieving the same result. What is interesting is that, as with the GS algorithm, the algorithm above does not *explicitly* use the structure of the unit group, though of course it must be doing so implicitly; it would be interesting to make the connection more explicit.

7.7 Coppersmith Attacks

Coppersmith-type attacks [Cop96a, Cop96b] would seem to be ideally suited to ideal lattices, as these attacks elegantly combine algebra and geometry. Somewhat surprisingly, however, they have not yet resulted in attacks that are more effective than generic lattice reduction algorithms.

Cohn and Heninger [CH11] applied Coppersmith's method to solving the BDDP over ideal lattices. In the BDDP over ideal lattices, one is given a basis B of an ideal lattice $\mathcal{I} \subset R$ and an element $\mathbf{u} \in R$ that is very close to some $\mathbf{v} \in \mathcal{I}$; the task is to output $\mathbf{v}$. Following Coppersmith's method, and to oversimplify a bit, Cohn and Heninger let $\mathbf{x} = \mathbf{u} - \mathbf{v}$ be the small unknown offset, and generate numerous univariate polynomials that have $\mathbf{x}$ as a root modulo $\mathcal{I}^t$ for some large exponent t. For example, any polynomial of the form $\mathbf{a}^r \cdot (\mathbf{u} - X)^{t-r}$ with $\mathbf{a} \in \mathcal{I}$ evaluates at $\mathbf{x}$ to an element that is in $\mathcal{I}^t$, and therefore any linear combination of such polynomials does as well. These polynomials form a lattice, and they apply LLL to this lattice to find a polynomial $p(X)$ with (somewhat) small coefficients. They design the lattice so that $p(\mathbf{x})$ is small (by the smallness of p's coefficient vector and of $\|\mathbf{x}\|_\infty$), indeed smaller than any nonzero element in $\mathcal{I}^t$. Since $p(\mathbf{x}) = 0 \bmod \mathcal{I}^t$, they conclude that $p(\mathbf{x}) = 0$ exactly, whereupon they recover $\mathbf{x}$ with efficient characteristic-zero root finding techniques [Len83].

Coppersmith's method works well in many settings involving *integers* – e.g., finding small solutions of univariate equations [Cop96b], factoring when the MSBs of a factor are known [Cop96a], factoring numbers of the form $p^r q$ for large r [BDHG99], etc. The main obstacle to successfully applying this method to *ideals* appears to be that the Coppersmith lattices involved have too high dimension. The Coppersmith lattice used by Cohn and Heninger has $n \times n$ blocks where one would have only a single entry in the integer case. In short, the lattice dimension is multiplied by n versus the integer case, and consequently the lattice reduction step performs much worse.

We remark that the GS algorithm, as well as our algorithm for solving the closest principal ideal generator problem (see Section 7.6), have a strategy somewhat similar to Coppersmith's method. In particular, they use Coppersmith's strategy of using lattice reduction and smallness to convert a modular equation to an exact equation, and thereafter to extract roots in characteristic zero.

7.8 Principal Ideals with a Generator

Gentry [Gen01] observed that, given a generator $\mathbf{v}$ of a principal ideal $\mathcal{I}$ in the ring $\mathbb{Z}[x]/(x^m - 1)$, one can construct a sub-lattice of $\mathcal{I}$ of dimension only $\lfloor (m+1)/2 \rfloor$ that contains a vector of length $2 \cdot \lambda_1(\mathcal{I})$. Therefore, one can hope to find a short vector in $\mathcal{I}$ by reducing a lattice that has only *half* the usual dimension. We can update this observation to obtain the following results about principal ideals in the ring of integers $\mathcal{O}_K$ of the m-th cyclotomic field K.

Lemma 13. *Let B be a $\mathbb{Z}$-basis of a principal ideal $\mathcal{I} = \langle \mathbf{v} \rangle$ over the ring of integers $\mathcal{O}_K$ of the m-th cyclotomic field K. Let $n = \phi(m)$. Let Λ be the $n/2$-dimensional sub-lattice of $\mathcal{I}$ given by $\Lambda =$*

$\{\mathbf{v} \cdot \mathbf{r} : \mathbf{r} \in \mathcal{O}_{K^+}\}$, *where* $\mathcal{O}_{K^+}$ *is the ring of integers of the index-2 real subfield* $K^+ = \mathbb{Q}(\zeta_m + \zeta_m^{-1})$ *of* K. *Then,* $\lambda_1(\Lambda) \leq 2\lambda_1(\mathcal{I})$.

Proof. Let $\mathbf{z} \in \mathcal{I}$ be such that $\|\mathbf{z}\|_2 = \lambda_1(\mathcal{I})$ (in the canonical embedding). Since $\mathcal{I}$ is principal, $\mathbf{z} = \mathbf{v} \cdot \mathbf{a}$ for some $\mathbf{a} \in \mathcal{O}_K$. Let $\mathbf{z}' = \mathbf{v} \cdot \overline{\mathbf{a}}$, where $\overline{\mathbf{a}} = \mathbf{a}(x^{-1})$ is the conjugate of $\mathbf{a}$. Then

$$\|\mathbf{z}'\|^2 = \langle \sigma(\mathbf{z}'), \sigma(\overline{\mathbf{z}'}) \rangle = \sum_{k \in \mathbb{Z}_m^*} \sigma_k(\mathbf{z}')\sigma_k(\overline{\mathbf{z}'}) = \sum_{k \in \mathbb{Z}_m^*} \sigma_k(\mathbf{v})\sigma_k(\mathbf{a})\sigma_k(\overline{\mathbf{v}})\sigma_k(\overline{\mathbf{a}}) = \sum_{k \in \mathbb{Z}_m^*} \sigma_k(\mathbf{z})\sigma_k(\overline{\mathbf{z}}) = \|\mathbf{z}\|^2.$$

Thus, $\mathbf{z} + \mathbf{z}'$ is a $\mathcal{I}$-element with length at most $2\lambda_1(\mathcal{I})$, and it is contained in the sub-lattice Λ. □

Theorem 7. *Let* $\mathbf{v}$ *be a generator of a principal ideal* $\mathcal{I}$ *in the ring of integers* $\mathcal{O}_K$ *of the* m*-th cyclotomic field* K. *Given* $\mathbf{v}$, *we can efficiently construct a* $n/2$*-dimensional sub-lattice of* $\mathcal{I}$ *that contains some* $\mathbf{w} \in \mathcal{I}$ *of length at most* $2\lambda_1(\mathcal{I})$.

Proof. From $\mathbf{v}$, we can efficiently construct a lattice Λ that contains precisely all elements of the form $\mathbf{v} \cdot \mathbf{a}$ for $\mathbf{a} \in \mathcal{O}_{K^+}$. By Lemma 13, the lattice Λ has the desired properties. □

In fact, we can do slightly better. We can also consider the sub-lattice Λ^- that contains precisely all elements of the form $\mathbf{v} \cdot \mathbf{a}$ where $\mathbf{a}$ is in the $n/2$ dimensional lattice of elements that can be expressed as $\mathbf{b} - \overline{\mathbf{b}}$ for some $\mathbf{b} \in \mathcal{O}_K$. We can then show that either Λ or Λ^- has a $\mathcal{I}$-vector of length at most $\sqrt{2}\lambda_1(\mathcal{I})$.

In Section 7.8.1, we extend this dimension-halving attack on principal ideal lattices to the setting where the attacker is *not* given a generator of the ideal (rather only a $\mathbb{Z}$-basis of the ideal).

7.8.1 Dimension Halving in Principal Ideal Lattices

Is approximate-SVP for principal ideal lattices *easier* than it is for general ideal lattices (over the ring of integers of the m-th cyclotomic number field)? For general ideal lattices, currently the best known algorithm for approximate-SVP involves applying a lattice reduction algorithm (e.g., LLL [LLL82] or BKZ [Sch87]) to a lattice of dimension $n = \phi(m)$. However, as we will see, the GS algorithm implies that, for principal ideal lattices, we only need to reduce lattices of dimension $n/2$. In short, the GS algorithm gives *much stronger attacks on principal ideal lattices* than we currently have on general ideal lattices (albeit still exponential time for small approximation factors).

Theorem 8. *Let* $T(n, d, \gamma)$ *denote the (worst-case) complexity of computing a* γ*-approximate shortest vector in the lattice* $\mathcal{L}(B)$, *where* B *is the HNF basis* B *of an* n*-dimensional lattice of determinant at most* d. *Computing a* γ*-approximate shortest vector in the lattice* $\mathcal{L}(B)$, *where* B *is a HNF basis of a principal ideal lattice* $\mathcal{I}$ *of norm* d *in the ring of integers* $\mathbb{Z}[x]/\Phi_m(x)$ *of the* m*-th cyclotomic field, has worst-case complexity at most* $\mathsf{poly}(m, \log d) + T(\phi(m)/2, d, \gamma/2)$.

Proof. Let $\mathcal{I}_{\mathbf{u}} = \langle \mathbf{u} \rangle$ be the ideal lattice for which we want to solve approximate-SVP, presented as a $\mathbb{Z}$-basis of $\{\mathbf{b}_i\}_{i \in [n]}$ with $\mathbf{b}_i = \mathbf{u} \cdot \mathbf{a}_i$ and $\mathbf{a}_i \in R$. Formally set $\mathbf{v} = \mathsf{N}_{K/\mathbb{Q}}(\mathbf{u}) \cdot (\mathbf{u}/\overline{\mathbf{u}})$ – that is $\mathbf{v}$ is essentially the fraction $\mathbf{u}/\overline{\mathbf{u}}$, except that we multiply by an appropriate integer to eliminate denominators and ensure $\mathbf{v} \in R$. Observe that, from B, we can compute both a basis of $\mathcal{I}_{\mathbf{v}} = \langle \mathbf{v} \rangle$ and also the term $\mathbf{v} \cdot \overline{\mathbf{v}} = \mathsf{N}_{K/\mathbb{Q}}(\mathbf{u})^2$. Use the GS algorithm to recover $\mathbf{v}$ (and hence $\mathbf{u}/\overline{\mathbf{u}}$) in polynomial time.

From $\mathbf{u}/\overline{\mathbf{u}}$ and B, compute a $\mathbb{Z}$-basis $C = \{\mathbf{c}_i = \mathbf{b}_i(1 + \overline{\mathbf{u}}/\mathbf{u})\}_{i \in [n]}$ of the principal ideal lattice $\mathcal{I}_{\mathbf{u}+\overline{\mathbf{u}}} = \langle \mathbf{u} + \overline{\mathbf{u}} \rangle$. Observe that $\mathbf{u} + \overline{\mathbf{u}}$ is in the index-2 real subfield $K^+ = \mathbb{Q}(\zeta_m + \zeta_m^{-1})$. Project the

basis C down to a $n/2$-dimensional basis C_{K^+} of the ideal $\mathcal{I}_{\mathbf{u}+\overline{\mathbf{u}},K^+} = \mathcal{I}_{\mathbf{u}+\overline{\mathbf{u}}} \cap K^+ \subset \mathcal{O}_{K^+}$. Observe that C_{K^+} is a set of the form $\{(\mathbf{u}+\overline{\mathbf{u}}) \cdot \mathbf{r} : \mathbf{r} \in \mathcal{O}_{K^+}\}$. Multiply each of the elements in C_{K^+} by $\mathbf{u}/(\mathbf{u}+\overline{\mathbf{u}})$ to get a basis $B_{K^+} = \{\mathbf{u} \cdot \mathbf{r} : \mathbf{r} \in \mathcal{O}_{K^+}\}$ of the lattice $\Lambda = \mathcal{L}(B_{K^+})$.

By Lemma 7, Λ has a nonzero vector of length at most $2\lambda_1(\mathcal{I})$. Therefore, we can solve γ-approximate-SVP in $\mathcal{I}$ by solving $\gamma/2$-approximate-SVP in Λ, proving the theorem. □

Note that non-principal ideal lattices, which in general can be expressed in terms of *two* generators, do not appear to be vulnerable to this dimension-halving attack.

The params in our constructions implicitly reveal principal ideal lattices – e.g., the lattice $\left\langle \mathbf{h}_\kappa \cdot \mathbf{g}^{\kappa-1} \right\rangle$ will likely be generated as an R-linear combination of the terms $\mathbf{h}_\kappa \cdot \mathbf{b}_1^\kappa/\mathbf{g}$ and $\mathbf{h}_\kappa \cdot \mathbf{b}_2^\kappa/\mathbf{g}$ that can be computed from params. Therefore, we recommend using R of degree twice what one would normally use for general ideal lattices.

Previous schemes have also used, or raised the possibility of using, principal ideals, including fully homomorphic encryption schemes [Gen09, SV10, GH11], homomorphic signatures schemes [BF11], and key agreement schemes [Buc91].

References

[AGHS12] Shweta Agrawal, Craig Gentry, Shai Halevi, and Amit Sahai. Sampling discrete gaussians efficiently and obliviously. Cryptology ePrint Archive, Report 2012/714, 2012. http://eprint.iacr.org/.

[BDHG99] Dan Boneh, Glenn Durfee, and Nick Howgrave-Graham. Factoring n = p^rq for large r. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 326–337. Springer, 1999.

[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.

[BF11] Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Paterson [Pat11], pages 149–168.

[BL96] Dan Boneh and Richard J. Lipton. Algorithms for black-box fields and their application to cryptography (extended abstract). In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 1996.

[BS96] Eric Bach and Jeffrey Shallit. *Algorithmic Number Theory, Volume 1: Efficient Algorithms*. MIT Press, 1996.

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

[Buc91] Johannes Buchmann. Number theoretic algorithms and cryptology. In Lothar Budach, editor, *FCT*, volume 529 of *Lecture Notes in Computer Science*, pages 16–21. Springer, 1991.

[CH11] Henry Cohn and Nadia Heninger. Ideal forms of coppersmith's theorem and guruswami-sudan list decoding. In Bernard Chazelle, editor, *ICS*, pages 298–308. Tsinghua University Press, 2011.

[Cop96a] Don Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In Maurer [Mau96], pages 178–189.

[Cop96b] Don Coppersmith. Finding a small root of a univariate modular equation. In Maurer [Mau96], pages 155–165.

[CS97] Don Coppersmith and Adi Shamir. Lattice attacks on ntru. In Fumy [Fum97], pages 52–61.

[DN12a] L. Ducas and P. Q. Nguyen. Faster gaussian lattice sampling using lazy floating-point arithmetic. In *Advances in Cryptology – Proceedings of ASIACRYPT '12*, LNCS. Springer, 2012.

[DN12b] L. Ducas and P. Q. Nguyen. Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. In *Advances in Cryptology – Proceedings of ASIACRYPT '12*, LNCS. Springer, 2012.

[Fum97] Walter Fumy, editor. *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*. Springer, 1997.

[Gen01] Craig Gentry. Key recovery and message attacks on ntru-composite. In *Advances in Cryptology - EUROCRYPT'01*, volume 2045 of *Lecture Notes in Computer Science*, pages 182–194. Springer, 2001.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *STOC*, pages 169–178. ACM, 2009.

[Gen10] Craig Gentry. Toward basing fully homomorphic encryption on worst-case hardness. In Rabin [Rab10], pages 116–137.

[GF05] Harold N. Gabow and Ronald Fagin, editors. *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*. ACM, 2005.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. Springer, 1997.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. Cryptology ePrint Archive, Report 2013/128, 2013. `http://eprint.iacr.org/`.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, 2013.

[GH11] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In Paterson [Pat11], pages 129–148.

[GKP+13] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Succinct functional encryption and applications: Reusable garbled circuits and beyond. In *STOC*, 2013.

[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.

[GS02] Craig Gentry and Michael Szydlo. Cryptanalysis of the revised ntru signature scheme. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 299–320. Springer, 2002.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits. In *STOC*, 2013.

[Hal05] Sean Hallgren. Fast quantum algorithms for computing the unit group and class group of a number field. In Gabow and Fagin [GF05], pages 468–474.

[HGS04] Nick Howgrave-Graham and Michael Szydlo. A method to solve cyclotomic norm equations. In Duncan A. Buell, editor, *ANTS*, volume 3076 of *Lecture Notes in Computer Science*, pages 272–279. Springer, 2004.

[HHGP+03] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. Ntrusign: Digital signatures using the ntru lattice. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 122–140. Springer, 2003.

[HKL+00] Jeffrey Hoffstein, Burton S. Kaliski, Daniel Bennett Lieman, Matthew John Barton Robshaw, and Yiqun Lisa Yin. Secure user identification based on constrained polynomials. *US Patent 6,076,163*, 2000.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In Joe Buhler, editor, *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

[HPS01] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Nss: An ntru lattice-based signature scheme. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 211–228. Springer, 2001.

[Jou00] Antoine Joux. A one round protocol for tripartite diffie-hellman. In *Algorithmic Number Theory - ANTS'00*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000.

[KS98] Erich Kaltofen and Victor Shoup. Subquadratic-time factoring of polynomials over finite fields. *Math. Comput.*, 67(223):1179–1197, 1998.

[Lan03] Edmund Landau. Neuer beweis des primzahlsatzes und beweis des primidealsatzes. *Mathematische Annalen*, 56(4):645–670, 1903.

[Len83] Arjen K. Lenstra. Factoring polynominals over algebraic number fields. In J. A. van Hulzen, editor, *EUROCAL*, volume 162 of *Lecture Notes in Computer Science*, pages 245–254. Springer, 1983.

[LL93] Arjen K. Lenstra and Hendrik W. Lenstra. *The Development of the Number Field Sieve*, volume 1554 of *Lecture notes in mathematics*. Springer-Verlag, 1993.

[LLL82] A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982.

[LLMP90] Arjen K. Lenstra, Hendrik W. Lenstra, Mark S. Manasse, and J.M. Pollard. The number field sieve. In *STOC*, volume 1554 of *Lecture Notes in Computer Science*, pages 564–572. ACM, 1990.

[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.

[Mau96] Ueli M. Maurer, editor. *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*. Springer, 1996.

[Mic01] Daniele Micciancio. Improving lattice based cryptosystems using the hermite normal form. In Joseph H. Silverman, editor, *CaLC*, volume 2146 of *Lecture Notes in Computer Science*, pages 126–145. Springer, 2001.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Computing*, 37(1):267–302, 2007.

[NR06] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of ggh and ntru signatures. In *Advances in Cryptology - EUROCRYPT'06*, volume 4004 of *Lecture Notes in Computer Science*, pages 271–288. Springer, 2006.

[NR09] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of ggh and ntru signatures. *J. Cryptology*, 22(2):139–160, 2009.

[Pat11] Kenneth G. Paterson, editor. *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*. Springer, 2011.

[Pei10] Chris Peikert. An efficient and parallel gaussian sampler for lattices. In Rabin [Rab10], pages 80–97.

[PR07] Chris Peikert and Alon Rosen. Lattices that admit logarithmic worst-case to average-case connection factors. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing – STOC 2007*, pages 478–487. ACM, 2007.

[PTT10] Charalampos Papamanthou, Roberto Tamassia, and Nikos Triandopoulos. Optimal authenticated data structures with multilinear forms. In Marc Joye, Atsuko Miyaji, and Akira Otsuka, editors, *Pairing*, volume 6487 of *Lecture Notes in Computer Science*, pages 246–264. Springer, 2010.

[Rab10] Tal Rabin, editor. *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*. Springer, 2010.

[Ram67] K. Ramachandra. On the units of cyclotomic fields. *Acta Arith.*, 12:165–173, 1966/67.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Gabow and Fagin [GF05], pages 84–93.

[Rot13] Ron Rothblum. On the circular security of bit-encryption. In *TCC*, pages 579–598, 2013.

[RS09] Markus Rückert and Dominique Schröder. Aggregate and verifiably encrypted signatures from multilinear maps without random oracles. In Jong Hyuk Park, Hsiao-Hwa Chen, Mohammed Atiquzzaman, Changhoon Lee, Tai-Hoon Kim, and Sang-Soo Yeo, editors, *ISA*, volume 5576 of *Lecture Notes in Computer Science*, pages 750–759. Springer, 2009.

[Sch87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53:201–224, 1987.

[Sho97a] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[Sho97b] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Fumy [Fum97], pages 256–266.

[SOK00] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *SCIS 2000*, Okinawa, Japan, January 2000.

[Ste08] Peter Stevenhagen. The arithmetic of number rings. *Algorithmic Number Theory, Lattices, Number Fields, Curves and Cryptography*, 44:209–266, 2008.

[SV05] Arthur Schmidt and Ulrich Vollmer. Polynomial time quantum algorithm for the computation of the unit group of a number field. In Gabow and Fagin [GF05], pages 475–480.

[SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.

[Szy03] Michael Szydlo. Hypercubic lattice reduction and analysis of ggh and ntru signatures. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 433–448. Springer, 2003.

A Generalizing Graded Encoding Systems

Here we generalize the definitions of graded encodings schemes from Section 2.2 to deal with the "asymmetric case", where there are many different "level-one sets" (corresponding to the many different source groups). We view the different level-one sets as separate dimensions, and correspondingly replace the index i from the symmetric case by an index-vector $\boldsymbol{v} \in \mathbb{N}^\tau$ (with $\mathbb{N}$ the natural numbers and τ the equivalent of the number of different groups). The different level-one set correspond to the standard unit vectors $\boldsymbol{e}_i$, and an encoding of $\alpha \in R$ relative to the index $\boldsymbol{e}_i$ (i.e., an element $a \in S^{(\alpha)}_{\boldsymbol{e}_i}$) is playing a role analogous to $\alpha \cdot g_i$ in asymmetric multilinear maps.

Note that in our case we can have τ "different groups" and yet we can multiply upto some number κ of different encodings, potentially $\kappa \neq \tau$. Hence we can also get a mix of the symmetric and asymmetric cases. If $u_1, \ldots, u_\kappa$ are encodings of $\alpha_1, \ldots, \alpha_\kappa \in R$ relative to indexes $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\kappa \in \mathbb{N}^\tau$, respectively, then $u^* = u_1 \times \cdots \times u_\kappa$ is an encoding of the product $\alpha^* = \prod_i \alpha_i \in R$ relative to the sum of the indexes $\boldsymbol{v} = \sum_i \boldsymbol{v}_i \in \mathbb{N}^\tau$.

For this general setting, we replace the parameter κ by a subset $\Lambda \subset \mathbb{N}^\tau$ that includes the indexes for which we can get valid encodings, and we can have a subset of indexes where we can test for zero. Of course, we preclude encoding "above the zero-testing levels", since for those levels we cannot check equality of encodings. Hence the zero-test indexes implicitly define also the subset Λ. We begin by formalizing the notions of "above" and "below" for our indexes, which is defined entry-wise.

Definition 8 (Partial order on $\mathbb{N}^\tau$)**.** *For an integer $\tau > 0$ and two vector $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{N}^\tau$, we define*

$$\boldsymbol{v} \leq \boldsymbol{w} \;\Leftrightarrow\; \boldsymbol{v}[j] \leq \boldsymbol{w}[j] \text{ for all } j = 1, 2, \ldots, \tau.$$

As usual, we have $\boldsymbol{v} < \boldsymbol{w}$ if $\boldsymbol{v} \leq \boldsymbol{w}$ and $\boldsymbol{v} \neq \boldsymbol{w}$.

For an arbitrary subset of indexes $T \subset \mathbb{N}^\tau$ we denote the set of indexes "below T" as:

$$\Lambda(T) \stackrel{\text{def}}{=} \{\boldsymbol{v} \in \mathbb{N}^\tau : \exists \boldsymbol{w} \in T \text{ s.t. } \boldsymbol{v} \leq \boldsymbol{w}\}.$$

We can now extend Definition 2 to the asymmetric case by defining T-graded encoding systems, where we think of T as the subset of indexes that admit zero-testing.

Definition 9 (T-Graded Encoding System)**.** *Let $T \subset \mathbb{N}^\tau$ be a finite set (for some integer $\tau > 0$), and let R be a ring. A T-Graded Encoding System for R is a system of sets $\mathcal{S} = \{S^{(\alpha)}_{\boldsymbol{v}} \subset \{0,1\}^* : \boldsymbol{v} \in \Lambda(T), \alpha \in R\}$, with the following properties:*

1. *For every fixed index $\boldsymbol{v} \in \Lambda(T)$, the sets $\{S^{(\alpha)}_{\boldsymbol{v}} : \alpha \in R\}$ are disjoint (hence they form a partition of $S_{\boldsymbol{v}} \stackrel{\text{def}}{=} \bigcup_\alpha S^{(\alpha)}_{\boldsymbol{v}}$).*
2. *There are binary operations '$+$' and '$-$' (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every $\boldsymbol{v} \in \Lambda(T)$, and every $u_1 \in S^{(\alpha_1)}_{\boldsymbol{v}}$ and $u_2 \in S^{(\alpha_2)}_{\boldsymbol{v}}$, it holds that*

$$u_1 + u_2 \in S^{(\alpha_1+\alpha_2)}_{\boldsymbol{v}} \quad \text{and} \quad u_1 - u_2 \in S^{(\alpha_1-\alpha_2)}_{\boldsymbol{v}} \tag{4}$$

where $\alpha_1 + \alpha_2$ and $\alpha_1 - \alpha_2$ are addition and subtraction in R.

3. *There is an associative binary operation '$\times$' (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every $\boldsymbol{v}_1, \boldsymbol{v}_2$ with $\boldsymbol{v}_1 + \boldsymbol{v}_2 \in \Lambda(T)$, and every $u_1 \in S_{\boldsymbol{v}_1}^{(\alpha_1)}$ and $u_2 \in S_{\boldsymbol{v}_2}^{(\alpha_2)}$, it holds that*

$$u_1 \times u_2 \in S_{\boldsymbol{v}_1+\boldsymbol{v}_2}^{(\alpha_1 \cdot \alpha_2)}. \tag{5}$$

Here $\alpha_1 \cdot \alpha_2$ is multiplication in R, and $\boldsymbol{v}_1 + \boldsymbol{v}_2$ is vector addition in $\mathbb{N}^\tau$.

Clearly, Definition 9 implies that if we have a collection of n encodings $u_i \in S_{\boldsymbol{v}_i}^{(\alpha_i)}$, $i = 1, 2 \ldots, n$, then as long as $\sum_i \boldsymbol{v}_i \in \Lambda(T)$ we get $u_1 \times \cdots \times u_n \in S_{\sum_i \boldsymbol{v}_i}^{(\prod_i \alpha_i)}$. We note that symmetric κ-multilinear maps as per Definition 2 correspond to $\{\kappa\}$-graded encoding systems (with $\tau = 1$), the asymmetric bilinear case corresponds to $\{(1,1)\}$-graded systems (with $\tau = 2$), etc.

A.1 Efficient Procedures, the Dream Version

As before, we first describe a "dream version" of the efficient procedures and then explain how to modify them to deal with technicalities that arise from our use of lattices in the realization.

Instance Generation. The randomized $\mathsf{InstGen}(1^\lambda, \tau, T)$ takes as inputs the parameters λ, τ the subset $T \subset \mathbb{N}^\tau$. It outputs $(\mathsf{params}, \mathbf{p}_{zt})$, where params is a description of a T-Graded Encoding System as above, and $\mathbf{p}_{zt}$ is a set of zero-test parameters for the indexes in T.

Ring Sampler. The randomized $\mathsf{samp}(\mathsf{params})$ outputs a "level-zero encoding" $a \in S_{\mathbf{0}}^{(\alpha)}$ for a nearly uniform element $\alpha \in_R R$. (Note that we require that the "plaintext" $\alpha \in R$ is nearly uniform, but not that the encoding a is uniform in $S_{\mathbf{0}}^{(\alpha)}$.)

Encoding. The (possibly randomized) $\mathsf{enc}(\mathsf{params}, \boldsymbol{v}, a)$ takes a "level-zero" encoding $a \in S_{\mathbf{0}}^{(\alpha)}$ for some $\alpha \in R$ and index $\boldsymbol{v} \in \Lambda(T)$, and outputs the "level-$\boldsymbol{v}$" encoding $u \in S_{\boldsymbol{v}}^{(\alpha)}$ for the same α.

Addition and negation. Given params and two encodings relative to the same index, $u_1 \in S_{\boldsymbol{v}}^{(\alpha_1)}$ and $u_2 \in S_{\boldsymbol{v}}^{(\alpha_2)}$, we have $\mathsf{add}(\mathsf{params}, i, u_1, u_2) = u_1 + u_2 \in S_{\boldsymbol{v}}^{(\alpha_1+\alpha_2)}$, and $\mathsf{sub}(\mathsf{params}, i, u_1, u_2) = u_1 + u_2 \in S_{\boldsymbol{v}}^{(\alpha_1+\alpha_2)}$,

Multiplication. For $u_1 \in S_{\boldsymbol{v}_1}^{(\alpha_1)}$, $u_2 \in S_{\boldsymbol{v}_2}^{(\alpha_2)}$ with $\boldsymbol{v}_1 + \boldsymbol{v}_2 \in \Lambda(T)$, we have $\mathsf{mul}(\mathsf{params}, \boldsymbol{v}_1, u_1, \boldsymbol{v}_2, u_2) = u_1 \times u_2 \in S_{\boldsymbol{v}_1+\boldsymbol{v}_2}^{(\alpha_1 \cdot \alpha_2)}$.

Zero-test. The procedure $\mathsf{isZero}(\mathsf{params}, \boldsymbol{v}, u)$ output 1 if $\boldsymbol{v} \in T$ and $u \in S_{\boldsymbol{v}}^{(0)}$ and 0 otherwise. Note that in conjunction with the subtraction procedure, this lets us test if $u_1, u_2 \in S_{\boldsymbol{v}}$ encode the same element $\alpha \in R$.

Extraction. This procedure extracts a "canonical" and "random" representation of ring elements from their level-$\boldsymbol{v}$ encoding. Namely $\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u)$ outputs (say) $s \in \{0,1\}^\lambda$, such that:

(a) For any $\alpha \in R$, $\boldsymbol{v} \in T$ and two $u_1, u_2 \in S_{\boldsymbol{v}}^{(\alpha)}$, $\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{v}, u_1) = \mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{v}, u_2)$,
(b) For any $\boldsymbol{v} \in T$, the distribution $\{\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{v}, u) \ : \ \alpha \in_R R, u \in S_{\boldsymbol{v}}^{(\alpha)}\}$ is nearly uniform over $\{0,1\}^\lambda$.

A.2 Efficient Procedures, the Real-Life Version

As before, our real-life procedures have noise bounds and we are only ensured of their properties when the bounds are valid and small enough. Also as before, we relax the requirements on the zero-test and the extraction routines, as we now describe.

Zero-test. We sometime allow false positives for this procedure, but not false negatives. Namely, $\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{v}, u) = 1$ for every $\boldsymbol{v} \in T$ and $u \in S_{\boldsymbol{v}}^{(0)}$, but we may have $\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{v}, u) = 1$ also in other cases. Again our weakest functionality requirement that we make is that for a uniform random choice of $\alpha \in_R R$, we have for every $\boldsymbol{v} \in T$

$$\Pr_{\alpha \in_R R}\left[\exists\, u \in S_{\boldsymbol{v}}^{(\alpha)} \text{ s.t } \mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{v}, u) = 1\right] = \text{negligible}(\lambda). \tag{6}$$

Additional requirements are considered security features (that a scheme may or may not possess), and are discussed later in this section.

Extraction. Our construction from Section 4 does not support full canonicalization. Instead, we settle for $\mathsf{ext}(\Lambda, \mathbf{p}_{zt}, \boldsymbol{v}, u)$ that has a good chance of producing the same output when applied to different encoding of the same elements. Specifically, we replace properties (a)-(b) from above by the weaker requirements:

(a′) For a randomly chosen $a \leftarrow \mathsf{samp}(\mathsf{params})$ and every $\boldsymbol{v} \in T$, if we run the encoding algorithm twice to encode a at level $\boldsymbol{v}$ and then extract from both copies then we get:

$$\Pr\left[\begin{array}{l}\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{v}, u_1) \\ = \mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{v}, u_2)\end{array} \;:\; \begin{array}{l}a \leftarrow \mathsf{samp}(\mathsf{params}) \\ u_1 \leftarrow \mathsf{enc}(\mathsf{params}, \boldsymbol{v}, a) \\ u_2 \leftarrow \mathsf{enc}(\mathsf{params}, \boldsymbol{v}, a)\end{array}\right] \;\geq\; 1 - \text{negligible}(\lambda).$$

(b′) The distribution $\{\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{v}, u) : a \leftarrow \mathsf{samp}(\mathsf{params}), u \leftarrow \mathsf{enc}(\mathsf{params}, \boldsymbol{v}, a)\}$ is nearly uniform over $\{0,1\}^\lambda$.

We typically need these two conditions to hold even if the noise bound that the encoding routine takes as input is larger than the one output by samp (upto some maximum value).

A.3 Hardness Assumptions

The DDH analog for this case says that it is hard to recognize encoding of products, except relative to indexes in $\Lambda(T)$. One way to formalize it is by letting the adversary choose the level "above T" on which it wants to be tested. This is formalized by the following process. (Below we suppress the noise bounds for readability):

1. $(\mathsf{params}, \mathbf{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, \tau, T)$
2. $\boldsymbol{v}, \boldsymbol{v}^* \leftarrow \mathcal{A}(\mathsf{params}, \mathbf{p}_{zt})$ // $\boldsymbol{v} \in T$ and $\boldsymbol{v} \lneq \boldsymbol{v}*$
3. For $i = 1, \ldots, \tau$, for $j = 1, \ldots \boldsymbol{v}_i^*$:
4. Choose $a_{i,j} \leftarrow \mathsf{samp}(\mathsf{params})$ // level-0 encoding of random $\alpha_{i,j} \in_R R$
5. Set $u_{i,j} \leftarrow \mathsf{enc}(\mathsf{params}, \boldsymbol{e}_i, a_{i,j})$ // encoding of $\alpha_{i,j}$ w.r.t the i'th unit vector
6. Set $\tilde{a} = \prod_{i,j} a_{i,j}$ // level-0 encoding of the product
7. Choose $\hat{a} \leftarrow \mathsf{samp}(\mathsf{params})$ // level-0 encoding of a random element
8. Set $\tilde{u} \leftarrow \mathsf{enc}(\mathsf{params}, \boldsymbol{v}, \tilde{a})$ // level-κ encoding of the product
9. Set $\hat{u} \leftarrow \mathsf{enc}(\mathsf{params}, \boldsymbol{v}, \hat{a})$ // level-κ encoding of random

The adversary $\mathcal{A}$ then gets all the $u_{i,j}$'s and either $\tilde{u}$ or $\hat{u}$, and it needs to guess which is the case. It is considered successful if the guess is correct and in addition $\boldsymbol{v} \in T$ and $\boldsymbol{v} \lneq \boldsymbol{v}*$. The generalized GDDH problem is hard if any polynomial-time adversary can only succeed with probability negligibly larger than 1/2.

Zero-test security. Zero-testing security is defined exactly as in the symmetric case, except that we requite it relative to all the indexes $\boldsymbol{v} \in T$.