New Multilinear Maps over the Integers

Jean-Sébastien Coron[1], Tancrède Lepoint[2], and Mehdi Tibouchi[3]

[1] University of Luxembourg, `jean-sebastien.coron@uni.lu`
[2] CryptoExperts, `tancrede.lepoint@cryptoexperts.com`
[3] NTT Secure Platform Laboratories, `tibouchi.mehdi@lab.ntt.co.jp`

February 25, 2015

Abstract. In the last few years, cryptographic multilinear maps have proved their tremendous potential as building blocks for new constructions, in particular the first viable approach to general program obfuscation. After the first candidate construction by Garg, Gentry and Halevi (GGH) based on ideal lattices, a second construction over the integers was described by Coron, Lepoint and Tibouchi (CLT). However the CLT scheme was recently broken by Cheon et al.; the attack works by computing the eigenvalues of a diagonalizable matrix over $\mathbb{Q}$ derived from the multilinear map.

In this paper we describe a new candidate multilinear map over the integers. Our construction is based on CLT but with a new arithmetic technique that makes the zero-testing element non-linear in the encoding, which prevents the Cheon et al. attack. Our new construction is relatively practical as its efficiency is comparable to the original CLT scheme. Moreover the subgroup membership and decisional linear assumptions appear to hold in the new setting.

1 Introduction

Multilinear maps. Since the breakthrough construction of Garg, Gentry and Halevi [GGH13a], there has been a growing interest in *cryptographic multilinear maps*. They have spurred scores of new cryptographic applications. Chiefly among them is possibly the first proposed approach to general program obfuscation [GGH$^+$13b]. Currently only three candidate constructions are known. Shorty after the first candidate construction of multilinear maps based on ideal lattices [GGH13a] (which we will refer to as GGH), Coron, Lepoint and Tibouchi proposed a second construction over the integers (CLT) using the same general paradigm [CLT13]. Recently, Gentry, Gorbunov and Halevi proposed another multilinear maps in which the map is defined with respect to a directed acyclic graph [GGH14].

A straightforward application of multilinear maps is multipartite Diffie-Hellman key exchange with $N = \kappa + 1$ users, where κ is the maximum level of the multilinear map scheme. Initially each user publishes a level-1 encoding of a random element while keeping a level-0 encoding of the same element private. Then each user can compute the product its level-0 by the product of the level-1 encodings of the other users. With $N = \kappa + 1$ users this gives a level-κ encoding from which the same secret value can be extracted by all users. The security of the protocol relies on a new hardness assumption which is a natural extension of the Decisional Diffie-Hellman assumption.

The CLT multilinear map over the integers. We recall the multilinear maps scheme over the integers from [CLT13]. One generates n secret primes p_i and publishes $x_0 = \prod_{i=1}^{n} p_i$ (where n is large enough to ensure security); one also generates n small secret primes g_i and a random secret integer z modulo x_0. The message space is $R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$. A level-$k$ encoding of a vector

$\boldsymbol{m} = (m_i) \in R$ is then an integer c such that for all $1 \leqslant i \leqslant n$:

$$c \equiv \frac{r_i \cdot g_i + m_i}{z^k} \pmod{p_i} \tag{1}$$

for some small random integers r_i; the integer c is therefore defined modulo x_0 by CRT. Encodings can then be added and multiplied modulo x_0, as long as the noise r_i is such that $r_i \cdot g_i + m_i < p$ for each i. The multiplication of a level-i encoding by a level-j encoding gives an encoding at level $i+j$.

For level-κ encodings one defines a zero-testing parameter p_{zt} with:

$$p_{zt} = \sum_{i=1}^{n} h_i \cdot \left(z^\kappa \cdot g_i^{-1} \bmod p_i \right) \cdot \frac{x_0}{p_i} \bmod x_0$$

for some small integers h_i. Given a level-κ encoding c as in (1), as a zero-testing procedure one computes $\omega = p_{zt} \cdot c \bmod x_0$ which gives:

$$\omega = \sum_{i=1}^{n} h_i \cdot \left(r_i + m_i \cdot (g_i^{-1} \bmod p_i) \right) \cdot \frac{x_0}{p_i} \bmod x_0 \ . \tag{2}$$

If $m_i = 0$ for all i, since the r_i's and h_i's are small, we obtain that ω is small compared to x_0; this enables to test whether c is an encoding of $\mathbf{0}$ or not. Moreover for non-zero encodings the leading bits of ω only depend on the m_i's and not on the noise r_i; for level-κ encodings this enables to extract a function of the m_i's only, which eventually defines a degree-κ multilinear map.

Cheon et al. attack. The CLT scheme above was completely broken by a recent attack from Cheon, Han Lee, Ryu and Stehlé [CHL$^+$14]; the attack runs in polynomial time, and recovers all secret parameters. The attack works by computing the eigenvalues of a diagonalizable matrix over $\mathbb{Q}$ derived from the multilinear map. More precisely, when applying the zero-testing procedure to the product of two encodings x and x', where x is an encoding of 0, the resulting ω in (2) can be seen as a diagonal quadratic form over $\mathbb{Z}$ in the CRT components $x \bmod p_i$ and $x' \bmod p_i$. By computing the values ω_{jk} of the quadratic form for n^2 product pairs of encodings $x_j \cdot x'_k$, one can then recover the coefficients of the quadratic form using eigendecomposition, which reveals all the secret p_i's and completely breaks the scheme. We recall the attack in more details in Section 3.

Tentative fixes. Shortly after Cheon et al. attack, two independent approaches to fix the CLT scheme have been proposed on the Cryptology ePrint Archive, due to Garg, Gentry, Halevi and Zhandry on the one hand [GGHZ14, Sec. 7][1], and Boneh, Wu and Zimmerman on the other [BWZ14]. However, both countermeasures were shown to be insecure in [CLT14]. Indeed, although these countermeasures do not expose encodings of zero, the value ω from the zero-testing procedure can still be expressed as a quadratic form in the CRT components of encodings. As a result, they can both be broken by a variant of the original Cheon et al. attack.

Our new construction. Our new construction keeps the same CLT encodings but departs from the two previous countermeasures by modifying the zero-testing procedure itself. Namely, we modify the definition of the zero-testing element p_{zt} so that ω cannot be expressed as a quadratic form

[1] We refer to the revised version of [GGHZ14] of November 12 2014, accessible on the Cryptology ePrint Archive.

anymore. For this we use a new arithmetic technique that maps the n CRT components $c \bmod p_i$ to some value modulo an independent integer N, so that the resulting ω in the zero-testing procedure depends on the CRT components in a non-linear way, rather than linearly as in (2).

The technique works as follows. Consider a level-κ encoding c as in (1); by the Chinese Remainder Theorem, we can write a relation of the form:

$$c = \sum_{i=1}^{n} \left(r_i + m_i \cdot (g_i^{-1} \bmod p_i)\right) \cdot u_i - a \cdot x_0 \tag{3}$$

over $\mathbb{Z}$ for some $a \in \mathbb{Z}$, where the u_i's are the CRT coefficients corresponding to the primes p_i's, and scaled by $g_i \cdot z^{-\kappa}$ for each i. Let N be a large integer and let $p_{zt} \in \mathbb{Z}_N$. For the zero-testing procedure we compute $\omega = p_{zt} \cdot c \bmod N$ which gives from (3):

$$\omega \equiv \sum_{i=1}^{n} \left(r_i + m_i \cdot (g_i^{-1} \bmod p_i)\right) \cdot v_i - a \cdot v_0 \pmod N \tag{4}$$

where $v_i := p_{zt} \cdot u_i \bmod N$ and $v_0 := p_{zt} \cdot x_0 \bmod N$. Assume now that we can generate p_{zt} and N such that all the v_i's are small compared to N, including v_0. Now if $m_i = 0$ for all i, since the r_i's are small, the integer a in (3) is also small, which implies that ω in (4) will also be small compared to N. This enables to test whether c is an encoding of 0 or not. As previously for level-κ encodings one can then extract a function of the m_i's only, which gives a degree-κ multilinear map. We show that such an element p_{zt} can be efficiently generated for any large enough N, owing to the particular structure of the CRT coefficients u_i.

Security analysis. By comparing equations (2) and (4), we see that the original CLT scheme is actually a particular case, with $N = x_0$ and $v_0 = 0$. Therefore the main difference in the new scheme is that $v_0 \neq 0$, which causes the value ω in (4) to depend on the integer a in (3). But that integer a depends on the CRT components r_i in a non-linear way. As a result, it is no longer true that the value ω computed from encoding products $x_j \cdot x'_k$ can be expressed as a quadratic form in the CRT components of x_j and x'_k, and the Cheon et al. attack is thus thwarted.

Another difference with the original CLT scheme is that we cannot publish $x_0 = \prod_{i=1}^{n} p_i$ anymore. Namely for encodings of 0 we get a small ω and therefore (4) holds over $\mathbb{Z}$. Therefore from x_0 one could compute $v_0 = p_{zt} \cdot x_0 \bmod N$ and apply the Cheon et al. attack modulo v_0 instead of over $\mathbb{Z}$. It is not a problem to keep x_0 private, however, as we can mimic the technique introduced by van Dijk et al. for their fully homomorphic encryption scheme over the integers [DGHV10] and approximate modular reduction by x_0 with a ladder of encodings of zero of increasing sizes.

We provide a detailed security analysis of our new construction in Section 3 (for the Cheon et al. attack and its variants) and Section 4 (for lattice attacks). We also explain why the subgroup membership (SubM) and decisional linear (DLIN) problems, which are known to be easy in the GGH scheme [GGH13a], seem to be hard in our new setting.

Implementation. We describe an implementation of our scheme, with a few optimizations. Instead of using a ladder of encodings of 0 at every level, we publish a small multiple x'_0 of x_0 so that intermediate encodings can be reduced modulo x'_0; only at the last level do we use a ladder of a few level-κ encodings of 0. Additionally, to reduce the size of public parameters, we store only a small subset of the public elements needed for re-randomization and combine them pairwise to generate the full public parameters, as in [CLT13]; such an optimization was originally described in [GH11]. With these optimizations our scheme is relatively practical; for reasonable security parameters a multipartite Diffie-Hellman computation with 7 users requires about 30 seconds, with a public parameter size of roughly 6 GBytes; a proof-of-concept implementation is available at [Lep15].

Extension to GGH. We briefly mention in Appendix G that our arithmetic technique can be adapted to GGH, in order to prevent attacks against base group assumptions like SubM and DLIN. However, a thorough security analysis of that GGH variant is beyond the scope of this paper.

2 New Multilinear Map over the Integers

In this section we define our new multilinear scheme. Our scheme is actually a *graded encoding scheme* (GES) as in previous works [GGH13a,CLT13]; we recall the notion of GES in Appendix A. As explained in introduction, our new multilinear map scheme keeps the same CLT encodings as given by (1), with two main differences:

1. The zero-testing parameter $\boldsymbol{p}_{zt}$ is computed differently, so that the CRT components modulo p_i of a level-κ encoding c are mapped to some value modulo an independent integer N, instead of modulo x_0. The resulting $\boldsymbol{\omega}$ in the zero-testing procedure then depends on those CRT components in a non-linear way, rather than linearly in the original CLT scheme, which prevents the Cheon et al. attack.
2. The integer $x_0 = \prod_{i=1}^{n} p_i$ is kept private. For re-randomization, this implies that we must slightly modify the proof of statistical indistinguishability. To reduce the size of intermediate encodings back to the size of x_0, we publish a ladder of encodings of 0. In Section 5 we describe a simple optimization with a public multiple x'_0 of x_0.

2.1 Scheme Description

System parameters. The system parameters are similar to the original CLT scheme. One first defines the security parameter λ and the required multilinearity level $\kappa \leqslant \mathsf{poly}(\lambda)$. Based on λ and κ, we choose:

- n: the vector dimension
- η: the bit-size of the primes p_i
- α: the bit-size of the primes g_i
- ρ: the bit-size of the randomness used in encodings

and various other parameters that will be specified later. The constraints that these parameters must satisfy are described in Section 2.2. For integers z, p we denote the reduction of z modulo p by $(z \bmod p)$ or $[z]_p$ with $-p/2 < [z]_p \leqslant p/2$. For integers $x_1, \dots, x_n$ we denote $\mathsf{CRT}_{p_1,\dots,p_n}(x_1, \dots, x_n)$ the unique integer x such that $x \equiv x_i \bmod p_i$ for all $1 \leqslant i \leqslant n$ and $0 \leqslant x < \prod_{i=1}^{n} p_i$.

As in the original CLT scheme a level-k encoding of a vector $\boldsymbol{m} = (m_i)$ is an integer c such that for all $1 \leqslant i \leqslant n$:

$$c \equiv \frac{r_i \cdot g_i + m_i}{z^k} \pmod{p_i} \tag{5}$$

where the r_i's are ρ-bit random integers (specific to the encoding c), with the following secret parameters: the p_i's are random η-bit prime integers, the g_i's are random α-bit primes, and the denominator z is a random (invertible) integer modulo $x_0 = \prod_{i=1}^n p_i$. The integer c is therefore defined by CRT modulo x_0, but as opposed to the original CLT scheme, x_0 is kept secret. We denote by γ the size of x_0 in bits. As in the CLT scheme the domain is the ring $R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$, so that for $\boldsymbol{m} = (m_i) \in R$ the components m_i are defined modulo g_i for all $1 \leqslant i \leqslant n$.

Instance generation: $(\mathsf{pp}, \boldsymbol{p}_{zt}) \leftarrow \mathsf{instGen}(1^\lambda, 1^\kappa)$. Instance generation is similar to [CLT13], except for the generation of $\boldsymbol{p}_{zt}$; moreover x_0 is kept private. We generate n secret random η-bit primes p_i and compute $x_0 = \prod_{i=1}^n p_i$. We generate a random invertible integer z modulo x_0. We generate n random α-bit prime integers g_i, and various other parameters that will be specified later. We publish the parameters $(\mathsf{pp}, \boldsymbol{p}_{zt})$ with

$$\mathsf{pp} = \Big(n, \eta, \alpha, \rho, \beta, \tau, \ell, \mu, y, \{x'_j\}_{j=1}^{\ell}, \{X_i^{(j)}\}, \{x_j\}_{j=1}^{\tau}, \{\Pi_j\}_{j=1}^{n+1}, s\Big).$$

Sampling level-zero encodings: $c \leftarrow \mathsf{samp}(\mathsf{pp})$. Since the primes p_i's in (5) must remain secret, the user cannot encode a vector $\boldsymbol{m} \in R$ by CRT directly from (5). Instead, as in [CLT13], a level-0 encoding c is generated as a random subset sum of random level-0 encodings x'_j from the public parameters. The only difference with [CLT13] is that the random subset-sum is computed over $\mathbb{Z}$ instead of modulo x_0, since x_0 is not public.

Therefore we publish as part as our instance generation a set of ℓ integers x'_j, where each x'_j encodes at level-0 the column vector $\boldsymbol{a}_j \in \mathbb{Z}^n$ of a secret matrix $\boldsymbol{A} = (a_{ij}) \in \mathbb{Z}^{n \times \ell}$, where each component a_{ij} is randomly generated in $[0, g_i) \cap \mathbb{Z}$. More precisely, using the CRT modulo x_0 we generate integers x'_j such that:

$$1 \leqslant j \leqslant \ell, \qquad x'_j \equiv r'_{ij} \cdot g_i + a_{ij} \pmod{p_i} \tag{6}$$

where the r'_{ij}'s are randomly generated in $(-2^\rho, 2^\rho) \cap \mathbb{Z}$.

To generate a level-0 encoding c, we first generate a random binary vector $\boldsymbol{b} = (b_j) \in \{0,1\}^\ell$ and output the level-0 encoding

$$c = \sum_{j=1}^{\ell} b_j \cdot x'_j \,.$$

From (6), this gives $c \equiv (\sum_{j=1}^{\ell} r'_{ij} b_j) \cdot g_i + \sum_{j=1}^{\ell} a_{ij} b_j \pmod{p_i}$; as required the output c is a level-0 encoding:

$$c \equiv r_i \cdot g_i + m_i \pmod{p_i} \tag{7}$$

of some vector $\boldsymbol{m} = \boldsymbol{A} \cdot \boldsymbol{b} \in R$ which is a random subset-sum of the column vectors $\boldsymbol{a}_j$. We note that for such level-0 encodings we get $|r_i \cdot g_i + m_i| \leqslant \ell \cdot 2^{\rho+\alpha}$ for all i. As in [CLT13] by applying the leftover hash lemma over $R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$ the distribution of $\boldsymbol{m}$ can be made statistically close to uniform over R.

Lemma 1 ([CLT13]). *Let $c \leftarrow \mathsf{samp}(\mathsf{pp})$ and write $c \equiv r_i \cdot g_i + m_i \pmod{p_i}$. Assume $\ell \geqslant n \cdot \alpha + 2\lambda$. The distribution of $(\mathsf{pp}, \boldsymbol{m})$ is statistically close to the distribution of $(\mathsf{pp}, \boldsymbol{m}')$ where $\boldsymbol{m}' \leftarrow R$.*

As opposed to [CLT13] we cannot reduce c modulo x_0; we only have the upper-bound $|c| \leqslant \ell \cdot 2^\gamma$, where γ is the size of x_0 in bits. In Appendix F, we show that instead of random sampling one can also publicly encode elements from the domain R, using a technique described in [BWZ14].

Encoding at higher levels: $c_k \leftarrow \mathsf{enc}(\mathsf{pp}, k, c)$. As in [CLT13], to allow encoding at higher levels, we publish as part of our instance-generation a level-one random encoding of $\mathbf{1}$, namely an integer y such that:

$$y \equiv \frac{r_i \cdot g_i + 1}{z} \pmod{p_i}$$

for random $r_i \in (-2^\rho, 2^\rho) \cap \mathbb{Z}$; as previously the integer y is computed by CRT modulo x_0. Given a level-0 encoding c of $\boldsymbol{m} \in R$ as given by (7), we can then compute a level-1 encoding of the same $\boldsymbol{m}$ by computing over $\mathbb{Z}$:

$$c_1 = c \cdot y.$$

Namely we obtain as required:

$$c_1 \equiv \frac{r_i' \cdot g_i + m_i}{z} \pmod{p_i}$$

for some integers r_i'. From $|c| \leqslant \ell \cdot 2^\gamma$, we obtain $|c_1| \leqslant \ell \cdot 2^{2\gamma}$.

The difference with [CLT13] is that we cannot reduce c_1 modulo x_0. Instead we provide a ladder of level-1 encodings of zero $X_i^{(1)}$ of increasing size, so that the size of a level-1 encoding can be progressively reduced down to the size of x_0, as in the DGHV scheme [DGHV10, Sec. 3.3.1]. Specifically, for $j = 0, \ldots, \gamma + \lfloor \log_2 \ell \rfloor$, we set:

$$X_j^{(1)} = \mathsf{CRT}_{p_1,\ldots,p_n}\left([r_{1j} \cdot g_1/z]_{p_1}, \ldots, [r_{nj} \cdot g_n/z]_{p_n}\right) + q_i \cdot x_0$$

where $r_{ij} \leftarrow (-2^\rho, 2^\rho) \cap \mathbb{Z}$ and $q_i \leftarrow [2^{\gamma+i-1}/x_0, 2^{\gamma+i}/x_0) \cap \mathbb{Z}$.

We can then iteratively reduce the size of c_1 down to the size of x_0, first by $X^{(1)}_{\gamma+\lfloor \log_2 \ell \rfloor}$ and eventually by $X_0^{(1)}$. Since the size reduction is done bit-by-bit, at each step some integer $b_j \cdot X_j^{(1)}$ is subtracted from c_1, for $b_j \in \{0, 1\}$. Therefore the noise increases additively by at most $(\gamma + \lfloor \log_2 \ell \rfloor + 1) \cdot 2^\rho$ in absolute value. After reduction, the resulting encoding $\hat{c}_1$ will be such that

$$\hat{c}_1 \equiv (\hat{r}_i \cdot g_i + m_i)/z \pmod{p_i}, \tag{8}$$

with $|\hat{r}_i \cdot g_i + m_i| \leqslant \ell \cdot 2^{\rho+\alpha} \cdot 2^{\rho+\alpha} + (\gamma + \lfloor \log_2 \ell \rfloor + 1) \cdot 2^\rho \leqslant 2\ell \cdot 2^{2\rho+2\alpha}$ for all i.

More generally to generate a level-k encoding we compute $c_k = c_0 \cdot y^k$, and the size of c_k can be iteratively reduced after each multiplication by y using ladders of similarly designed level-j encodings $\{X_i^{(j)}\}_{i=0}^{\gamma+\lfloor \log_2 \ell \rfloor}$ for levels $j = 1, \ldots, k$.

Re-randomization: $c' \leftarrow \mathsf{reRand}(\mathsf{pp}, k, \hat{c}_k)$. Our re-randomization procedure is similar to [CLT13] except that again we cannot reduce the encodings modulo x_0. We describe the re-randomization of encodings at level $k = 1$; the procedure can be easily adapted to randomize at level $k > 1$. We publish as part of our instance-generation a set of $n + 1$ integers $\varPi_j$:

$$1 \leqslant j \leqslant n + 1, \qquad \varPi_j = \sum_{i=1}^{n} \varpi_{ij} \cdot g_i \cdot u_i + \varpi_{n+1,j} \cdot x_0$$

where the u_i's are appropriate CRT coefficients so that the $\varPi_j$'s are all level-1 random encodings of zero:

$$1 \leqslant j \leqslant n + 1, \qquad \varPi_j \equiv \frac{\varpi_{ij} \cdot g_i}{z} \pmod{p_i}.$$

Namely, we let for all $1 \leqslant i \leqslant n$:

$$u_i := \left(z^{-1} \cdot \left(\frac{x_0}{p_i} \right)^{-1} \bmod p_i \right) \cdot \frac{x_0}{p_i} \tag{9}$$

The matrix $\boldsymbol{\mathit{\Pi}} = (\varpi_{ij}) \in \mathbb{Z}^{(n+1)\times(n+1)}$ is a diagonally dominant matrix generated as follows: the non-diagonal entries are randomly and independently generated in $(-2^\rho, 2^\rho) \cap \mathbb{Z}$, while the diagonal entries are randomly generated in $((n+1)2^\rho, (n+2)2^\rho) \cap \mathbb{Z}$.

We also publish as part of our instance-generation a set of τ integers x_j:

$$1 \leqslant j \leqslant \tau, \qquad x_j = \sum_{i=1}^{n} r_{ij} \cdot g_i \cdot u_i + r_{n+1,j} \cdot x_0$$

so that each x_j is a level-1 random encoding of zero:

$$1 \leqslant j \leqslant \tau, \qquad x_j \equiv \frac{r_{ij} \cdot g_i}{z} \pmod{p_i}$$

and where the column vectors of the matrix $\boldsymbol{X} = (r_{ij}) \in \mathbb{Z}^{(n+1)\times\tau}$ are randomly and independently generated in the half-open parallelepiped spanned by the columns of the previous matrix $\boldsymbol{\mathit{\Pi}}$; an algorithm to generate such r_i's is described in [CLT13, App. E]; we obtain $|r_{ij} \cdot g_i| \leqslant 3n2^{\rho+\alpha}$ for all i, j.

Given as input a (reduced) level-1 encoding $\hat{c}_1$ as given by Equation (8), we randomize $\hat{c}_1$ with a random subset-sum of the x_j's and a linear combination of the $\mathit{\Pi}_j$'s, over $\mathbb{Z}$:

$$c'_1 = \hat{c}_1 + \sum_{j=1}^{\tau} b_j \cdot x_j + \sum_{j=1}^{n+1} b'_j \cdot \mathit{\Pi}_j \tag{10}$$

where $b_j \leftarrow \{0, 1\}$, and $b'_j \leftarrow [0, 2^\mu) \cap \mathbb{Z}$, where $\mu := \rho + \alpha + \lambda$. The following Lemma shows that as required the distribution of c'_1 is nearly independent of the input (as long as it encodes the same $\boldsymbol{m}$). This essentially follows from the "leftover hash lemma over lattices" of [CLT13, Sec. 4.2]; we refer to Appendix B for the proof.

Lemma 2. *Let the encodings $c \leftarrow \mathsf{samp}(\mathsf{pp})$, $\hat{c}_1 \leftarrow \mathsf{enc}(\mathsf{pp}, 1, c)$, and c'_1 as given by (10). Write $c'_1 \equiv (r_i \cdot g_i + m_i)/z \pmod{p_i}$ for all $1 \leqslant i \leqslant n$ and $r_{n+1} = (c'_1 - \sum r_i \cdot g_i \cdot u_i)/x_0$, and define $\boldsymbol{r} = (r_1, \ldots, r_n, r_{n+1})^T$. If $2(\rho + \alpha + \lambda) \leqslant \eta$ and $\tau \geqslant (n+2) \cdot \rho + 2\lambda$, then the distribution of $(\mathsf{pp}, \boldsymbol{r})$ is statistically close to that of $(\mathsf{pp}, \boldsymbol{r'})$, where $\boldsymbol{r'} \in \mathbb{Z}^{n+1}$ is randomly generated in the half-open parallelepiped spanned by the column vectors of $2^\mu \boldsymbol{\mathit{\Pi}}$. Moreover we have $|r_i \cdot g_i + m_i| \leqslant 4n^2 \cdot 2^{2\rho+2\alpha+\lambda}$ for all $1 \leqslant i \leqslant n$.*

Finally, we can reduce the size of c'_1 down to the size of x_0 using the ladder $\{X_i^{(1)}\}$, and we obtain an encoding $\hat{c}'_1$. Writing $\hat{c}'_1 \equiv (\hat{r}'_i \cdot g_i + m_i)/z \pmod{p_i}$, we obtain

$$|\hat{r}'_i \cdot g_i + m_i| \leqslant 4n^2 \cdot 2^{2\rho+2\alpha+\lambda} + (\gamma + \lfloor \log_2 \ell \rfloor + 1) \cdot 2^\rho \leqslant 5n^2 \cdot 2^{2\rho+2\alpha+\lambda}\,.$$

Adding, negating and multiplying encodings. As in [CLT13] we can add, negate and multiply encodings. The difference is that we do those operations over $\mathbb{Z}$ instead of modulo x_0. More precisely, given level-one encodings v_j of vectors $\boldsymbol{m}_j \in \mathbb{Z}^n$ for $1 \leqslant j \leqslant \kappa$, with $v_j \equiv (r_{ij} \cdot g_i + m_{ij})/z \pmod{p_i}$, we compute over $\mathbb{Z}$:

$$v = \prod_{j=1}^{\kappa} v_j \, .$$

This gives:

$$v \equiv \frac{\prod\limits_{j=1}^{\kappa} (r_{ij} \cdot g_i + m_{ij})}{z^\kappa} \equiv \frac{r_i \cdot g_i + \left(\prod\limits_{j=1}^{\kappa} m_{ij} \right) \bmod g_i}{z^\kappa} \pmod{p_i}$$

for some integers $r_i \in \mathbb{Z}$. Hence we obtain a level-κ encoding of the vector $\boldsymbol{m}$ obtained by componentwise product of the vectors $\boldsymbol{m}_j$, as long as the components do not wrap modulo p_i, that is $\prod_{j=1}^{\kappa}(r_{ij} \cdot g_i + m_{ij}) < p_i$ for all i. Then, using the ladder $X_i^{(\kappa)}$ one can reduce its size down to the size of x_0, at the cost of an additive increase in absolute value of the noise.

In multipartite Diffie-Hellman key exchange we compute the product of κ level-1 encodings from reRand and one level-0 encoding from samp, which gives from previous bounds for all i:

$$|r_i| \leqslant (6n^2 2^{2\rho+2\alpha+\lambda})^\kappa \cdot \ell \cdot 2^{\rho+1}$$

In Section 5 we describe an optimization in which we publish a multiple x_0' of x_0; then all intermediate encodings can be reduced modulo x_0', instead of using a ladder of encodings of zero; only at the last stage do we need a ladder of a few level-κ encodings of zero.

Zero testing. $\mathsf{isZero}(\mathsf{pp}, \boldsymbol{p}_{zt}, c) \overset{?}{=} 0/1$. To prevent the Cheon et al. attack, we keep the same encoding as in (1) but we compute the p_{zt} differently; this is the most important difference. Let c be a level-κ encoding. We assume $0 \leqslant c < x_0$, as a result of approximate modular reduction using a ladder of level-κ encodings of 0. From (5) we can write by CRT:

$$c \equiv \sum_{i=1}^{n} \left(\frac{r_i \cdot g_i + m_i}{z^\kappa} \bmod p_i \right) \cdot \left(\left(\frac{x_0}{p_i} \right)^{-1} \bmod p_i \right) \cdot \frac{x_0}{p_i} \pmod{x_0}$$
$$c \equiv \sum_{i=1}^{n} \left(r_i + m_i \cdot g_i^{-1} \bmod p_i \right) \cdot \left(g_i \cdot z^{-k} \cdot \left(\frac{x_0}{p_i} \right)^{-1} \bmod p_i \right) \cdot \frac{x_0}{p_i} \pmod{x_0}$$

Therefore we can write over the integers:

$$c = \sum_{i=1}^{n} \left(r_i + m_i \cdot g_i^{-1} \bmod p_i \right) \cdot u_i' - a \cdot x_0 \tag{11}$$

for some integer a, where the u_i''s are the scaled CRT coefficients:

$$u_i' = \left(g_i \cdot z^{-k} \cdot \left(\frac{x_0}{p_i} \right)^{-1} \bmod p_i \right) \cdot \frac{x_0}{p_i} \tag{12}$$

We generate a random prime integer N of size $\gamma + 2\eta + 1$ bits. Using LLL in dimension 2, we obtain[2] pairs of nonzero integers (α_i, β_i) satisfying:

$$|\alpha_i| < 2^{\eta-1} \qquad |\beta_i| \leqslant \frac{4}{3} \cdot \frac{N}{2^{\eta-1}} < 2^{2-\eta} \cdot N \qquad \beta_i \equiv \alpha_i \cdot (u_i'/p_i) \pmod N.$$

We also generate as in [CLT13] an integer matrix $\boldsymbol{H} = (h_{ij}) \in \mathbb{Z}^{n \times n}$ such that $\boldsymbol{H}$ is invertible in $\mathbb{Z}$ and both $\|\boldsymbol{H}^T\|_\infty \leqslant 2^\beta$ and $\|(\boldsymbol{H}^{-1})^T\|_\infty \leqslant 2^\beta$, for some parameter β specified later; here $\|\cdot\|_\infty$ is the operator norm on $n \times n$ matrices with respect to the ℓ^∞ norm on $\mathbb{R}^n$. A technique for generating such $\boldsymbol{H}$ is discussed in Appendix C. We then publish as part of our instance generation the following zero-testing vector $\boldsymbol{p}_{zt} \in \mathbb{Z}^n$:

$$(\boldsymbol{p}_{zt})_j = \sum_{i=1}^{n} h_{ij} \cdot \alpha_i \cdot p_i^{-1} \bmod N \tag{13}$$

To determine whether a level-κ encoding c is an encoding of zero or not, we compute the vector $\boldsymbol{\omega} = c \cdot \boldsymbol{p}_{zt} \bmod N$ and test whether $\|\boldsymbol{\omega}\|_\infty$ is small:

$$\mathsf{isZero}(\mathsf{pp}, \boldsymbol{p}_{zt}, c) = \begin{cases} 1 & \text{if } \|c \cdot \boldsymbol{p}_{zt} \bmod N\|_\infty < N \cdot 2^{-\nu} \\ 0 & \text{otherwise} \end{cases}$$

for some parameter ν specified later.

Namely for a level-κ ciphertext c we obtain from (11):

$$\begin{aligned}(\boldsymbol{\omega})_j = (c \cdot \boldsymbol{p}_{zt} \bmod N)_j &= \sum_{i=1}^{n} h_{ij} \cdot \alpha_i \cdot p_i^{-1} \cdot c \bmod N \\ &= \sum_{i=1}^{n} h_{ij} \cdot \alpha_i \cdot p_i^{-1} \cdot \left(\sum_{k=1}^{n} \left(r_k + m_k \cdot g_k^{-1} \bmod p_k \right) \cdot u_k' - a \cdot x_0 \right) \bmod N\end{aligned}$$

which gives:

$$\begin{aligned}(\boldsymbol{\omega})_j = &\sum_{i=1}^{n} h_{ij} \cdot \Big(\left(r_i + m_i \cdot g_i^{-1} \bmod p_i \right) \cdot \beta_i + \\ &\alpha_i \cdot \sum_{k=1,\, k \neq i}^{n} \left(r_k + m_k \cdot g_k^{-1} \bmod p_k \right) \cdot \frac{u_k'}{p_i} - a \cdot \alpha_i \cdot \frac{x_0}{p_i} \Big) \bmod N\end{aligned} \tag{14}$$

Recall that α_i is at most $\eta - 1$ bits, therefore $\alpha_i \cdot u_k'/p_i$ has size at most $\eta - 1 + \gamma - (\eta - 1) = \gamma$ bits; the integer $\alpha_i \cdot x_0/p_i$ has size also at most γ bits; moreover β_i is at most $|N| - \eta + 1$ bits. Therefore in Equation (14) the integers β_i, $\alpha_i \cdot u_k/p_i$ and $\alpha_i \cdot x_0/p_i$ are all small compared to N. This implies that if $m_i = 0$ for all $1 \leqslant i \leqslant n$, then ω_j will be small compared to N, when the r_i's are small enough, i.e. a limited number of additions/multiplications on encodings has been performed. Conversely if $m_i \neq 0$ for some i we show that $\|\boldsymbol{\omega}\|_\infty$ must be large. This shows the correctness of our zero-testing procedure. More precisely we prove the following lemma in Appendix D.

[2] More precisely, we apply Legendre reduction to the 2-dimensional lattice generated by the rows of $\begin{pmatrix} \lceil N/B^2 \rceil & u_i'/p_i \bmod N \\ 0 & N \end{pmatrix}$, where $B = (3/4)^{1/4} 2^{\eta-1}$. The shortest vector is of the form $(\alpha_i \lceil N/B^2 \rceil, \beta_i)$.

Lemma 3. *Let n, η, α and β be as in our parameter setting. Let ρ_f be such that $\alpha + \log_2 n < \rho_f \leqslant \eta - 2\beta - 2\alpha - \lambda - 8$, and let $\nu = \eta - \rho_f - \beta - \lambda - 3 \geqslant 2\alpha + \beta + 5$. Let c be such that $c \equiv (r_i \cdot g_i + m_i)/z^\kappa \pmod{p_i}$ for all $1 \leqslant i \leqslant n$, where $0 \leqslant m_i < g_i$ for all i. Let $\boldsymbol{r} = (r_i)_{1 \leqslant i \leqslant n}$ and assume that $\|\boldsymbol{r}\|_\infty < 2^{\rho_f}$. If $\boldsymbol{m} = 0$ then $\|\boldsymbol{\omega}\|_\infty < 2^{-\nu-\lambda} \cdot N$. Conversely if $\boldsymbol{m} \neq 0$ then $\|\boldsymbol{\omega}\|_\infty > 2^{-\nu+2} \cdot N$.*

Extraction. $sk \leftarrow \mathsf{ext}(\mathsf{pp}, \boldsymbol{p}_{zt}, u_\kappa)$. This part is essentially the same as in [GGH13a]. To extract a random function of the vector $\boldsymbol{m}$ encoded in a level-κ encoding c, we multiply c by the zero-testing parameter $\boldsymbol{p}_{zt}$ modulo N, collect the ν most significant bits of each of the n components of the resulting vector, and apply a strong randomness extractor (using the seed s from pp):

$$\mathsf{ext}(\mathsf{pp}, \boldsymbol{p}_{zt}, c) = \mathsf{Extract}_s\big(\mathsf{msbs}_\nu(c \cdot \boldsymbol{p}_{zt} \bmod N)\big)$$

where msbs_ν extracts the ν most significant bits of the result.

Namely if two encodings c and c' encode the same $\boldsymbol{m} \in \mathbb{Z}^n$ then from Lemma 3 we have $\|(c - c') \cdot \boldsymbol{p}_{zt} \bmod N\|_\infty < N \cdot 2^{-\nu-\lambda}$, and therefore we expect that $\boldsymbol{\omega} = c \cdot \boldsymbol{p}_{zt} \bmod N$ and $\boldsymbol{\omega'} = c' \cdot \boldsymbol{p}_{zt} \bmod N$ agree on their ν most significant bits, and therefore extract to the same value.

Conversely if c and c' encode different vectors then by Lemma 3 we must have $\|(c - c') \cdot \boldsymbol{p}_{zt} \bmod N\|_\infty > N \cdot 2^{-\nu+2}$, and therefore the ν most significant bits of the corresponding $\boldsymbol{\omega}$ and $\boldsymbol{\omega'}$ must be different. This implies that for random $\boldsymbol{m} \in R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$ the min-entropy of $\mathsf{msbs}_\nu(c \cdot \boldsymbol{p}_{zt} \bmod N)$ when c encodes $\boldsymbol{m}$ is at least $\log_2 |R| \geqslant n(\alpha - 1)$. Therefore we can use a strong randomness extractor to extract a nearly uniform bit-string of length $\lfloor \log_2 |R| \rfloor - \lambda$.

This concludes the description of our new multilinear encoding scheme.

Remark 1. By comparing equations (2) and (4) we see that the original CLT scheme is a particular case with $N = x_0$ and $\alpha_i = 0$ for all $1 \leqslant i \leqslant n$. Therefore the main difference of our construction is that it incorporates the additional term a, which depends on the r_i's in a non-linear way; this is to prevent the Cheon et al. attack (see Section 3).

2.2 Setting the Parameters

The constraints on the system parameters are similar to [CLT13].

- The bit-size ρ of the randomness used for encodings must satisfy $\rho = \Omega(\lambda)$ to avoid brute force attack on the noise. The improved attacks from [CN12] and [LS14] both have complexity $\tilde{\mathcal{O}}(2^{\rho/2})$, but with a large overhead, so in practice we can take $\rho = \lambda$.
- The bit-size α of the primes g_i must be large enough so that the order of the group $R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$ does not contain small prime factors (see Appendix E). One can take $\alpha = \lambda$.
- The parameter n must be large enough to thwart lattice-based attacks on the encodings, namely $n = \omega(\eta \log \lambda)$; see Section 4.
- The number ℓ of level-0 encodings x'_j for samp must satisfy $\ell \geqslant n \cdot \alpha + 2\lambda$ in order to apply the leftover hash lemma; see Lemma 1.
- The number τ of level-1 encodings x_j must satisfy $\tau \geqslant (n + 2) \cdot \rho + 2\lambda$ in order to apply the leftover hash lemma over lattices; see Lemma 2.
- As a conservative security precaution, we take $\beta = 3\lambda$ (see Section 4.3).

- The bit-size η of the primes p_i must satisfy $\eta \geqslant \rho_f + 2\alpha + 2\beta + \lambda + 8$, where ρ_f is the maximum bit size of the randoms r_i a level-κ encoding (see Lemma 3). When computing the product of κ level-1 encodings and an additional level-0 encoding (as in a multipartite Diffie-Hellman key exchange with $\kappa + 1$ users), one obtains $\rho_f = \kappa \cdot (2\rho + 2\alpha + \lambda + 2\log_2 n + 3) + \rho + \log_2 \ell + 1$ (see previous Section).
- We set $\nu = \eta - \rho_f - \lambda - \beta - 3$ for the number of most significant bits to extract (see Lemma 3).

2.3 Security of Our Construction

As in the original CLT scheme [CLT13] and in the GGH scheme [GGH13a] the security of our construction does not seem to be reducible to more classical assumptions, such as for example the Approximate-GCD problem. To prove the security of the one-round $(\kappa + 1)$-way Diffie-Hellman key exchange protocol, as in [GGH13a] one must therefore make the assumption that solving the Graded DDH problem (GDDH) is hard in our scheme; see Appendix A.

3 Cheon et al. Attack

The goal of this section is to argue that the Cheon et al. attack [CHL$^+$14] is prevented in our new construction.

3.1 Attack Description

We first recall the Cheon et al. attack against the original CLT scheme. This attack makes use of low-level encodings of 0: if such encodings are made public, one can recover in polynomial time all secret parameters. In the CLT scheme such encodings of 0 are used for the rerandomization procedure, therefore the Cheon et al. attack leads to a complete break of CLT.

In the following we describe a slight simplification of [CHL$^+$14] in which only a single ciphertext c is used instead of two ciphertexts c_0 and c_1; this enables to obtain as eigenvalues directly the CRT components of c, instead of the ratios of the CRT components of c_0 and c_1. For simplicity we assume $\kappa = 2$; the attack is easily extended to any $\kappa > 2$. Let c be a level-0 encoding with $c \equiv c_i \pmod{p_i}$. Let x be a level-1 encoding with $x \equiv x_i/z \pmod{p_i}$, and let x' be a level-1 encoding of 0 with $x' \equiv r'_i \cdot g_i/z \pmod{p_i}$. Let c' be the level-κ product encoding

$$c' = x \cdot c \cdot x' \bmod x_0$$

From $c' \equiv x_i \cdot c_i \cdot r'_i \cdot g_i \cdot z^{-2} \pmod{p_i}$, we obtain by CRT:

$$c' \equiv \sum_{i=1}^{n} x_i \cdot c_i \cdot r'_i \cdot u_i \pmod{x_0} \tag{15}$$

with the CRT coefficients:

$$u_i = \left(g_i \cdot z^{-2} \cdot \left(\frac{x_0}{p_i} \right)^{-1} \bmod p_i \right) \cdot \frac{x_0}{p_i}$$

In the original CLT scheme, the zero-testing parameter p_{zt} is given by

$$p_{zt} = \sum_{i=1}^{n} h_i \cdot \left(z^2 \cdot g_i^{-1} \bmod p_i \right) \cdot \frac{x_0}{p_i} \bmod x_0$$

Using $p_{zt} \cdot u_i \equiv h_i \cdot x_0/p_i \pmod{x_0}$ for all $1 \leqslant i \leqslant n$, we obtain from (15):

$$\omega = [p_{zt} \cdot c']_{x_0} = \sum_{i=1}^{n} x_i \cdot c_i \cdot r'_i \cdot h_i \cdot x_0/p_i \tag{16}$$

where the last equality holds over $\mathbb{Z}$ because c' is an encoding of 0.

More generally, let x_j be level-1 encodings with $x_j \equiv x_{ij}/z \pmod{p_i}$, and let x'_k be a level-1 encodings of 0 with $x'_k \equiv r'_{ik} \cdot g_i/z \pmod{p_i}$. One can therefore compute for $1 \leqslant j, k \leqslant n$:

$$\omega_{jk} = [(x_j \cdot c \cdot x'_k) \cdot p_{zt}]_{x_0} \tag{17}$$

which gives as previously:

$$\omega_{jk} = \sum_{i=1}^{n} x_{ij} \cdot c_i \cdot r'_{ik} \cdot h_i \cdot x_0/p_i \tag{18}$$

over the integers. We note that ω_{jk} is a diagonal quadratic form over $\mathbb{Z}$ in the x_{ij}'s and the r'_{ik}'s. By spanning $1 \leqslant j, k \leqslant n$, one can construct a matrix $\boldsymbol{W_c} = (\omega_{jk})_{1 \leqslant j,k \leqslant n}$ such that

$$\boldsymbol{W_c} = \boldsymbol{X} \times \boldsymbol{C} \times \boldsymbol{R}, \tag{19}$$

where $\boldsymbol{C} = \mathsf{diag}(c_1, c_2, \ldots, c_n)$, $\boldsymbol{X} = (x_{ij} \cdot h_i \cdot x_0/p_i)_{1 \leqslant j,i \leqslant n}$ and $\boldsymbol{R} = (r'_{ik})_{1 \leqslant i,k \leqslant n}$.

We perform the same computation with $c = 1$ in (17); one can therefore compute a matrix $\boldsymbol{W_1}$ such that $\boldsymbol{W_1} = \boldsymbol{X} \times \boldsymbol{I} \times \boldsymbol{R}$, where $\boldsymbol{I}$ is the $n \times n$ identity matrix. Finally, one can publicly compute:

$$\boldsymbol{W} = \boldsymbol{W_c} \cdot \boldsymbol{W_1}^{-1} = \boldsymbol{X} \times \boldsymbol{C} \times \boldsymbol{X}^{-1}.$$

Since $\boldsymbol{C}$ is a diagonal matrix, by computing the eigenvalues of $\boldsymbol{W}$ one can recover the c_i's, and then the p_i's. Finally, Cheon et al. describe how to recover all the other secret values in [CHL$^+$14].

Extension. A similar attack applies against two independent approaches to fix the CLT scheme, [GGHZ14, Sec. 7] and [BWZ14], proposed shortly after the Cheon et al. attack. Namely, although the two countermeasures do not expose encodings of zero, the value ω from the zero-testing procedure can still be expressed as a diagonal quadratic form in the CRT components of encodings, as in Equation (18), hence the two countermeasure can be broken by the same technique; we refer to [CLT14] for a description of the modified attacks.

3.2 Non-Applicability of Cheon et al. Attack

In this section we explain why the above attack does not apply against our new scheme. As previously we let x be a level-1 encoding with $x \equiv x_i/z \pmod{p_i}$, and let x' be a level-1 encoding of 0 with $x' \equiv r'_i \cdot g_i/z \pmod{p_i}$. We consider as previously the level-κ product encoding, with $\kappa = 2$:

$$c' = x \cdot c \cdot x'$$

Here we cannot reduce c' modulo x_0 since x_0 is kept private; instead we must use a ladder of level-2 encodings of zero. Let c'' be the resulting encoding, with $0 \leqslant c'' < x_0$; we obtain:

$$c'' \equiv c' + \frac{s_i \cdot g_i}{z^2} \pmod{p_i}$$

for some integers s_i of size roughly ρ bits. Therefore instead of (15) we obtain over the integers:

$$c'' = \sum_{i=1}^{n} \left(x_i \cdot c_i \cdot r'_i + s_i \right) \cdot u_i - a \cdot x_0 \tag{20}$$

for some integer a. Using the new definition of $p_{zt} \in \mathbb{Z}_N$, and letting $v_i = p_{zt} \cdot u_i \bmod N$ for all $1 \leqslant i \leqslant n$ and $v_0 = p_{zt} \cdot x_0 \bmod N$, we obtain from (20):

$$\omega = [p_{zt} \cdot c'']_N = \sum_{i=1}^{n} (x_i \cdot c_i \cdot r'_i + s_i) \cdot v_i - a \cdot v_0 \tag{21}$$

where as previously the last equality holds over $\mathbb{Z}$.

Now comparing equalities (16) and (21), we see that we obtain two additional terms: the s_i's and the integer a. The s_i's come from reducing c' with the ladder of level-κ encodings of 0, so that eventually $0 \leqslant c'' < x_0$; therefore the s_i's depend on $x \cdot c \cdot x'$ in a non-linear way. Similarly the integer a in (21), which is the quotient of the division of $\sum_{i=1}^{n} (x_i \cdot c_i \cdot r'_i + s_i) \cdot u_i$ by x_0, depends on the $x_i \cdot c_i \cdot x'_i$ in a non-linear way. Therefore, if we apply Cheon et al. attack, we do not obtain a quadratic form as in (18) anymore.

More precisely, we can let as previously x_j be level-1 encodings with $x_j \equiv x_{ij}/z \pmod{p_i}$, and let x'_k be a level-1 encodings of 0 with $x'_k \equiv r'_{ik} \cdot g_i/z \pmod{p_i}$. As previously for all $1 \leqslant j, k \leqslant n$, we can compute the product encodings $c'_{jk} = x_j \cdot c \cdot x'_k$ and we let c''_{jk} be the encodings obtained after reducing c'_{jk} such that $0 \leqslant c''_{jk} < x_0$, using the ladder of level-κ encodings of zero. This gives:

$$\omega_{jk} = [p_{zt} \cdot c''_{jk}]_N = \sum_{i=1}^{n} (x_{ij} \cdot c_i \cdot r'_{ik} + s_{ijk}) \cdot v_i - a_{jk} \cdot v_0 \tag{22}$$

for integers s_{ijk} and a_{jk}. Compared to (18), we see that the previous equation has two additional terms s_{ijk} and a_{jk}. As previously we can write:

$$\boldsymbol{W_c} = \boldsymbol{X} \times \boldsymbol{C} \times \boldsymbol{R} + \boldsymbol{S} - \boldsymbol{A} \cdot v_0 \tag{23}$$

for some matrices $\boldsymbol{S}$ and $\boldsymbol{A}$. However we see that the previous attack does not apply, because of the additional terms $\boldsymbol{S}$ and $\boldsymbol{A} \cdot v_0$. Namely if as previously we perform the same computation with $c = 1$, we obtain:

$$\boldsymbol{W_1} = \boldsymbol{X} \times \boldsymbol{I} \times \boldsymbol{R} + \boldsymbol{S'} - \boldsymbol{A'} \cdot v_0 \tag{24}$$

but as opposed to the CLT scheme we cannot get a simple expression for $\boldsymbol{W} = \boldsymbol{W_c} \times \boldsymbol{W_1}^{-1}$. More generally, as opposed to the CLT case, it seems difficult to extract useful information about $\boldsymbol{C}$ from the matrices $\boldsymbol{W_c}$ and $\boldsymbol{W_1}$, since in equations (23) and (24) all terms $\boldsymbol{X}$, $\boldsymbol{R}$, $\boldsymbol{S}$, $\boldsymbol{S'}$, $\boldsymbol{A}$, $\boldsymbol{A'}$ and v_0 are unknown.

Remark 2. If we do not reduce c'_{jk} with the ladder of encodings, the s_{ijk} terms disappear but the integers a_{jk} becomes too large and (22) does not hold over $\mathbb{Z}$ anymore. The equation still holds modulo N, however there is still the additional term a_{jk} that prevents the Cheon et al. attack.

3.3 Attack with Known x_0.

In this section we describe an extension of the Cheon et al. attack against our scheme when x_0 is known; this explains why x_0 must be kept secret in our scheme.

When x_0 is known, we can reduce the previous ciphertexts c'_{jk} modulo x_0, and therefore the s_{ijk} terms in (22) disappear. Moreover $v_0 = [p_{zt} \cdot x_0]_N$ is known. Therefore we can compute the $\boldsymbol{W_c}$ matrix as previously, and we obtain from (23) with $\boldsymbol{S} = 0$:

$$\boldsymbol{W_c} = \boldsymbol{X} \times \boldsymbol{C} \times \boldsymbol{R} \bmod v_0$$

which is the same equation as (19) in the original attack except that it holds modulo v_0 instead of over $\mathbb{Z}$.

Therefore we can apply the Cheon et al. attack modulo v_0 instead of over $\mathbb{Z}$. If v_0 is prime, one can recover the eigenvalues of $\boldsymbol{W} = \boldsymbol{W_c} \cdot \boldsymbol{W_1}^{-1} \bmod v_0$ by factoring the characteristic polynomial modulo v_0, which reveals the c_i's as previously. If a prime p can be extracted from v_0, one can still apply the attack modulo p and recover the c_i's modulo p; for large enough p this reveals the c_i's; alternatively for sufficiently many such primes p, the c_i's could be recovered by CRT.

Actually the attack also works even if v_0 is hard to factor and no prime can be extracted. Namely the eigenvalues c_i's are small, so to recover the roots of the characteristic polynomial one can use Coppersmith's first theorem for finding small roots of polynomial equations modulo an integer of unknown factorization [Cop97]. Namely Coppersmith's bound applies: with a modulus v_0 of size roughly γ bits and a characteristic polynomial of degree n, the roots have size only roughly ρ bits, with $\rho \ll \eta \simeq \gamma/n$.

3.4 Attack for Small Multiple of x_0

In Section 5 we describe an optimization with a known multiple $x'_0 = q \cdot x_0$, in order to avoid the ladder of encodings of 0. Here we show that we cannot take a too small multiple x'_0, otherwise the attacker can compute:

$$v'_0 = [p_{zt} \cdot x'_0]_N = [p_{zt} \cdot q \cdot x_0]_N = q \cdot v_0 \bmod N$$

where, as in Section 3.3, we let $v_0 := p_{zt} \cdot x_0 \bmod N$. If the prime q is small enough then the previous equation holds over the integers, and the attacker obtains $v'_0 = q \cdot v_0$. Therefore the attacker can possibly extract a few primes from v'_0 and therefore from v_0. Letting b be a divisor of v_0, one could then apply the Cheon et al. attack modulo b instead of modulo v_0 and recover all secret parameters. Therefore one should make sure that $q \cdot v_0$ is greater than N. Letting η_q be the bitsize of q, this gives the condition $\eta_q + \gamma \geqslant \gamma + 2\eta + 1$. Therefore we can take $\eta_q = 2\eta + \lambda$.

3.5 The Subgroup Membership and Decision Linear Problems

In this section we also explain why the subgroup membership (SubM) and decisional linear (DLIN) problems, which are known to be easy in the GGH scheme [GGH13a], seem to be hard in our new setting.

We first describe the attacks against SubM and DLIN in the original CLT scheme, following [CHL$^+$14, Appendix A]. Obviously CLT is broken so the SubM and DLIN assumptions cannot hold for CLT, but this enables to argue why SubM and DLIN should be hard for our new scheme.

The SubM problem. Let $R = \mathbb{Z}_{g_1} \times \ldots \times \mathbb{Z}_{g_n}$ and let G be a subgroup of R obtained by fixing some of the n components to zero; for example we can take $G = \{0\} \times \mathbb{Z}_{g_2} \times \ldots \times \mathbb{Z}_{g_n}$. The subgroup membership problem consists in distinguishing between $c_1 = \mathsf{enc}(m_1)$ and $c_2 = \mathsf{enc}(m_2)$, where $m_1 \leftarrow R$ and $m_2 \leftarrow G$. Let $c = \mathsf{enc}(0)$. We first describe the distinguishing attack against the original CLT scheme. The attack is based on the Cheon et al. attack. Namely as in (19) we can compute the matrices:

$$\begin{aligned}
\boldsymbol{W_c} &= \boldsymbol{X} \times \mathsf{diag}(\, g_1 \cdot r_1 \, , g_2 \cdot r_2 \, , \ldots , g_n \cdot r_n \,) \times \boldsymbol{R} \\
\boldsymbol{W_{c_1}} &= \boldsymbol{X} \times \mathsf{diag}(\, g_1 \cdot r_1^{(1)} + m_{11} \, , g_2 \cdot r_2^{(1)} + m_{12} \, , \ldots , g_n \cdot r_n^{(1)} + m_{1n} \,) \times \boldsymbol{R} \\
\boldsymbol{W_{c_2}} &= \boldsymbol{X} \times \mathsf{diag}(\, g_1 \cdot r_1^{(2)} \, , g_2 \cdot r_2^{(2)} + m_{22} \, , \ldots , g_n \cdot r_n^{(2)} + m_{2n} \,) \times \boldsymbol{R}
\end{aligned}$$

Therefore we obtain:

$$\begin{aligned}
\gcd(\det \boldsymbol{W_c}, \det \boldsymbol{W_{c_1}}) &= \Delta \cdot \det(\boldsymbol{X}) \cdot \det(\boldsymbol{R}) \\
\gcd(\det \boldsymbol{W_c}, \det \boldsymbol{W_{c_2}}) &= \Delta' \cdot g_1 \cdot \det(\boldsymbol{X}) \cdot \det(\boldsymbol{R})
\end{aligned}$$

for some integers Δ and Δ' of roughly the same size on average; this enables to distinguish between c_1 and c_2.

It seems difficult to adapt the above attack against our new multilinear scheme, because of the additional terms $\boldsymbol{S}$ and $\boldsymbol{A}$ in Equation (23); namely because of those additional terms we cannot extract useful information from the encodings by computing determinants and gcds; therefore we conjecture that SubM is hard in our setting.

The DLIN problem. Consider a matrix of elements $\boldsymbol{B} = (b_{ij}) \in R^{L \times L}$ and their encodings $\boldsymbol{T} = (\mathsf{enc}(b_{ij}))$, the DLIN problem consists, given the matrix of encodings $\boldsymbol{T}$, in distinguishing between rank L and rank $L-1$ matrices $\boldsymbol{B}$. In the CLT scheme, the attack is similar to the above. Namely from the matrix of encodings $\boldsymbol{T} = (T_{ij})$ one can compute the matrix of matrices $\boldsymbol{V} = (\boldsymbol{W}_{T_{ij}})$. One can see that if $\boldsymbol{B}$ is a rank $L-1$ matrix, then $\det \boldsymbol{V}$ will contain the factor $\prod_i g_i$, whereas if $\boldsymbol{B}$ is full rank, $\det \boldsymbol{V}$ will not contain this factor; this enables to solve the DLIN problem in case of CLT.

As previously it seems difficult to adapt the above attack against our new multilinear scheme, because of the additional terms $\boldsymbol{S}$ and $\boldsymbol{A}$ in Equation (23); therefore we conjecture that DLIN is also hard in our setting.

4 Lattice Attacks

4.1 Lattice Attack on the Encodings

The first attack considered in [CLT13] against the original CLT scheme was based on computing a short basis for the lattice of vectors orthogonal modulo x_0 to $\boldsymbol{x} = (x_j)_{1 \leqslant j \leqslant t}$, where the x_j's are level-0 encodings of zero [CLT13, Sec. 5.1]. If the reduced basis vectors are short enough, they can reveal the noise values of the x_j's and hence break the scheme.

The attack does not apply directly to our modified scheme, because x_0 is now secret, and it is therefore no longer possible to compute a basis for the lattice of vectors orthogonal to $\boldsymbol{x}$ modulo x_0. However, we can also mount the attack using the lattice $\boldsymbol{x}^\perp$ of vectors orthogonal to $\boldsymbol{x}$ over $\mathbb{Z}$, or the lattice of vectors orthogonal to $\boldsymbol{x}$ modulo some multiple x_0' of x_0 when using the optimization suggested in Section 5 below.

Just as in [CLT13, Sec. 5.1], though, the complexity of these extended attacks remains exponential in n; it is in fact slightly worse, because the new lattice has slightly longer vectors for a given choice of the lattice dimension t. In particular, the complexity lower bound of $2^{\Omega(\gamma/\eta^2)}$ applies *a fortiori*. The attack is therefore defeated by letting $n = \omega(\eta \log \lambda)$.

4.2 Lattice Attack against $\boldsymbol{p_{zt}}$

From $x_0 = \prod_{i=1}^{n} p_i$ and $(\boldsymbol{p}_{zt})_j = \sum_{i=1}^{n} h_{ij} \cdot \alpha_i \cdot p_i^{-1} \bmod N$, we obtain:

$$x_0 \cdot (\boldsymbol{p}_{zt})_j = \sum_{i=1}^{n} h_{ij} \cdot \alpha_i \cdot \frac{x_0}{p_i} \bmod N.$$

Now x_0 is of size γ bits, and the right-hand side of this congruence, which we denote by w_j, is bounded above by $n2^{\beta+\gamma}$: they are both small compared to N. Therefore, if we consider a vector $\boldsymbol{p}$ formed by a subset of the $(\boldsymbol{p}_{zt})_j$'s, say $\boldsymbol{p} = \big((\boldsymbol{p}_{zt})_j\big)_{1\leqslant j\leqslant t} \in \mathbb{Z}^t$, it may be possible to recover $\boldsymbol{w} = (w_j)_{1\leqslant j\leqslant t}$ as a short vector in the lattice generated by $\boldsymbol{p}$ and $N\mathbb{Z}^t$, and obtain x_0 accordingly.

More precisely, $\boldsymbol{w}^+ = (\boldsymbol{w}, n2^\beta \cdot x_0) \in \mathbb{Z}^{t+1}$ is a vector of norm bounded by $\sqrt{t+1} \cdot n2^{\beta+\gamma}$ in the lattice L generated by the rows of the following matrix:

$$\begin{pmatrix} N & & & \\ & \ddots & & \\ & & N & \\ (\boldsymbol{p}_{zt})_1 & \cdots & (\boldsymbol{p}_{zt})_t & n2^\beta \end{pmatrix}.$$

The volume of L is $\operatorname{vol}(L) = n2^\beta \cdot N^t$, and therefore we expect $\boldsymbol{w}^+$ to be the shortest vector in L when:

$$\sqrt{t+1} \cdot n2^{\beta+\gamma} \ll \sqrt{\frac{t+1}{2\pi e}} \operatorname{vol}(L)^{\frac{1}{t+1}}.$$

Taking logarithms and neglecting logarithmic terms, this gives:

$$\begin{aligned} \beta + \gamma &\lesssim \frac{t \log_2 N + \beta}{t+1} \\ t &\gtrsim \frac{\gamma}{\log_2 N - \gamma - \beta} \approx \frac{\gamma}{2\eta + 1 - \beta} \approx \frac{n}{2} \end{aligned}$$

since N is of size $\gamma + 2\eta + 1$ bits. When that condition is satisfied, the gap between $\lambda_1(L) = \|\boldsymbol{w}^+\|$ and the second minimum $\lambda_2(L)$ of the lattice is expected to be roughly:

$$\frac{\lambda_2(L)}{\lambda_1(L)} \approx \frac{\big(\operatorname{vol}(L)/\|\boldsymbol{w}^+\|\big)^{1/t}}{\|\boldsymbol{w}^+\|} \approx \frac{N \cdot 2^{-\gamma/t}}{2^{\beta+\gamma}} \approx 2^{2\eta-\beta-\gamma/t} \lesssim 2^\eta$$

since we must have $t \leqslant n$ due to the size of the zero-testing vector. As a result, to find $\boldsymbol{w}^+$, we need to solve the Unique SVP problem within a factor at most 2^η in a lattice of dimension $t \geqslant n/2$. This requires a lattice reduction method with Hermite constant at most $2^{2\eta/n}$, hence a complexity lower bound of $2^{\Omega(n/\eta)} = 2^{\Omega(\gamma/\eta^2)}$, just as in Section 4.1. Thus, this attack is thwarted by our choice of parameters.

4.3 Attack against $\boldsymbol{p}_{zt}$

Another attack considered in [CLT13, Sec. 5.2], and later improved in [LS14], would recover a factor of x_0 when some of the secret coefficients h_{ij} of the zero-testing matrix were too small (the improved attack of Lee and Seo had complexity $\tilde{O}(2^{b/2})$ where b is the actual bit size of the h_{ij}'s, which is upper bounded by β but *not tightly so* in the original CLT construction).

The attack crucially requires x_0 as an input, however, so it doesn't seem to apply to our new scheme since x_0 is now kept secret. Therefore, we are not aware of an attack in the case the h_{ij}'s are small. Nevertheless, as a security precaution, we now suggest using an improved method for generating the matrix $\boldsymbol{H}$ in a way that yields much larger coefficients than previously (see Appendix C).

4.4 Hidden Subset Sum Attack on Zero Testing

One can also consider a hidden subset sum attack on the zero-testing parameter $\boldsymbol{p}_{zt}$, similar to the approach described in [CLT13, Sec. 5.3]. More precisely, we can write $\boldsymbol{p}_{zt}$ in the following way:

$$\boldsymbol{p}_{zt} = \sum_{i=1}^{n} \boldsymbol{h}_i \cdot \alpha_i \cdot p_i^{-1} \pmod N$$

as a hidden subset sum of the secret vectors $\boldsymbol{h}_i$. Therefore, the Nguyen-Stern orthogonal lattice technique [NS99] can in principle be used to recover the $\boldsymbol{h}_i$'s. Indeed, any vector $\boldsymbol{u}$ orthogonal to $\boldsymbol{p}_{zt}$ modulo N satisfies:

$$\sum_{i=1}^{n} \frac{\alpha_i}{p_i} \langle \boldsymbol{h}_i, \boldsymbol{u} \rangle = 0 \pmod N.$$

This means that the vector $\boldsymbol{v} = \big(\langle \boldsymbol{h}_i, \boldsymbol{u}\rangle\big)_{1\leqslant i\leqslant n}$ is orthogonal modulo N to $\boldsymbol{a} = \big(\alpha_i/p_i \bmod N\big)_{1\leqslant i\leqslant n}$. In particular, if $\boldsymbol{u}$ is short enough to ensure that $\boldsymbol{v}$ is shorter than the shortest vector of the lattice $L_{\boldsymbol{a}}$ of vectors orthogonal to $\boldsymbol{a}$ modulo N, then we get $\boldsymbol{v} = 0$ and hence $\boldsymbol{u}$ is orthogonal to each of the $\boldsymbol{h}_i$'s over $\mathbb{Z}$. If we can find sufficiently many such vectors $\boldsymbol{u}$, the $\boldsymbol{h}_i$'s can be retrieved by orthogonal lattice techniques again.

For this general approach to succeed, we need to be able to compute the very short vectors $\boldsymbol{u}$. Just like in the setting of [CLT13, Sec. 5.3], however, short enough vectors simply do no exist. Indeed, the length condition is that $2^{\beta}\|\boldsymbol{u}\| < \lambda_1(L_{\boldsymbol{a}})$, and the shortest vector of $L_{\boldsymbol{a}}$ is actually of the form $(\alpha'_1 p_1, \dots, \alpha'_n p_n)$ where $\sum \alpha_i \alpha'_i = 0$. This yields $\lambda_1(L_{\boldsymbol{a}}) \approx 2^{(1+1/n)\eta}$. Thus, we need to find $\boldsymbol{u}$ of length significantly less than $2^{(1+1/n)\eta}$. Now the lattice $L_{\boldsymbol{p}_{zt}}$ of vectors orthogonal to $\boldsymbol{p}_{zt}$ modulo N is of volume N, so we expect its successive minima to all be around $N^{1/n} > 2^{(1+2/n)\eta}$. Therefore, short enough vectors $\boldsymbol{u}$ do not exist, and the approach of Nguyen-Stern does not yield an attack in our setting.

A similar technique can be considered against the vector $\boldsymbol{\omega} = \boldsymbol{p}_{zt} \cdot c \bmod N$ that arises from applying the zero-testing algorithm to an encoding of 0 at level κ, but the approach similarly fails to yield an attack for exactly the same reason.

4.5 Attack on the Inverse Zero Testing Matrix

Similarly to the above, we can also consider an attack approach analogous to [CLT13, Sec. 5.4] using the inverse zero-testing matrix. From

$$(\boldsymbol{p}_{zt})_j = \sum_{i=1}^{n} h_{ij} \cdot \alpha_i \cdot p_i^{-1} \bmod N$$

and since the matrix $\boldsymbol{H}$ is invertible, with inverse $\boldsymbol{H}^{-1} = \boldsymbol{T}$, we obtain for all rows $\boldsymbol{t}_i$ of $\boldsymbol{T}$:

$$\langle \boldsymbol{t}_i, \boldsymbol{p}_{zt} \rangle = \alpha_i \cdot p_i^{-1} \pmod N$$

and therefore:

$$\langle p_i \boldsymbol{t}_i, \boldsymbol{p}_{zt} \rangle = \alpha_i \pmod N.$$

As a result, we may hope to recover $(p_i \boldsymbol{t}_i, 2^{\beta} \cdot \alpha_i)$ as a short vector (of length $\approx 2^{\eta+\beta}$) orthogonal to $\boldsymbol{p}^{+} = (\boldsymbol{p}_{zt}, -1/2^{\beta})$ modulo N. However, the lattice of vectors orthogonal to $\boldsymbol{p}^{+}$ modulo N is of rank $n+1$ and volume N, so its successive minima are expected to be around $N^{1/(n+1)} \approx 2^{((n+2)\eta+\lambda)/(n+1)}$. Therefore, we can only hope to recover the secret vectors if the size condition:

$$(n+1)(\eta+\beta) \lesssim (n+2)\eta + \lambda, \quad \text{i.e.} \quad \beta \lesssim \frac{\eta+\lambda}{n+1}$$

is satisfied, which is clearly not the case. Therefore, this approach does not actually yield an attack either.

5 Optimizations and Implementation

In this section we describe an implementation of our new multilinear map scheme in the one-round $(\kappa+1)$-way Diffie-Hellman key exchange protocol; we recall the protocol in Appendix E, following [BS03,GGH13a]. We use the following optimizations:

1. Integer p_{zt}: as in [CLT13] we use a single integer p_{zt} instead of a vector $\boldsymbol{p}_{zt}$ with n components, as this is enough for Diffie-Hellman key exchange. Moreover the integer N can be generated as the product of large enough prime integers, instead of being prime.
2. Known multiple of x_0: we publish a multiple $x_0' = q \cdot x_0$ of x_0, so that all intermediate encodings can be reduced modulo x_0', instead of using a ladder of encodings of 0 at each level.[3]
3. Quadratic re-randomization: as in [CLT13] we only store a small subset of encodings which are later combined pairwise to generate the full set of encodings. This implies that the randomization of encodings becomes heuristic only. We describe a slightly more efficient variant.

5.1 Zero-Testing Element

As in [CLT13] we can use a single integer p_{zt} instead of a vector with n components:

$$p_{zt} = \sum_{i=1}^{n} h_i \cdot \alpha_i \cdot p_i^{-1} \bmod N$$

[3] Note that Lem. 2 can be adapted, based on the Leftover-Hash-Lemma over lattices.

where the h_i's are random β-bit integers, and the α_i's are generated as previously. Therefore we obtain a single integer $\omega = p_{zt} \cdot c \bmod N$, with

$$\omega = \sum_{i=1}^{n} h_i \cdot \Big(\big(r_i + m_i \cdot g_i^{-1} \bmod p_i\big) \cdot \beta_i + \alpha_i \cdot \sum_{k=1,\, k\neq i}^{n} \big(r_k + m_k \cdot g_k^{-1} \bmod p_k\big) \cdot \frac{u'_k}{p_i} - a \cdot \alpha_i \cdot \frac{x_0}{p_i} \Big) \bmod N$$

As before, if $\|\boldsymbol{r}\|_\infty < 2^{\rho_f}$ we still have $|\omega| < x_0 \cdot 2^{-\nu-\lambda-2}$. However the converse is no longer true: we can have $|\omega| < x_0 \cdot 2^{-\nu}$ for an encoding of a non-zero vector $\boldsymbol{m}$. This implies that two encodings of different vectors can now extract to the same value. As in [CLT13], while it is actually easy to generate such collisions using LLL, this does not seem to give an attack against the GDDH problem.

5.2 Optimization with a Known Multiple of x_0

As explained in Section 3.3 one should not publish x_0; however one can still publish a large enough multiple of x_0:

$$x'_0 = q \cdot x_0$$

where q is a prime integer of size η_q; since x_0 is of size γ bits, the size of x'_0 is therefore at most $\gamma + \eta_q$ bits. Then all intermediate encodings can be first reduced modulo x'_0, instead of using a ladder of encodings of 0 at each level, which is simpler and more efficient. For encodings of intermediate level $k < \kappa$, we can keep the encoding reduced modulo x'_0 only.

Eventually we obtain at level κ an encoding c of size $\gamma + \eta_q$. In principle the zero-testing procedure for an encoding c requires that $0 \leqslant c < x_0$; therefore only at this stage do we use a ladder of level-κ encodings of 0. Actually we show that it is not strictly necessary to ensure that $0 \leqslant c < x_0$. Namely from

$$c = \sum_{i=1}^{n} \big(r_i + m_i \cdot g_i^{-1} \bmod p_i\big) \cdot u'_i - a \cdot x_0$$

one obtains the upper bound when $m_i = 0$ for all i:

$$|a| \leqslant \frac{n \cdot 2^{\rho_f} \cdot x_0 + |c|}{x_0} \leqslant n \cdot 2^{\rho_f} + \frac{|c|}{x_0}$$

which gives $|a| \leqslant n \cdot 2^{\rho_f} + 1$ when $0 \leqslant c < x_0$. Therefore it is not necessary to ensure that $0 \leqslant c < x_0$; if we only require that c has at most $\gamma + \rho_f$ bits, we get the similar bound:

$$|a| \leqslant 3n \cdot 2^{\rho_f}$$

Therefore we must reduce the size of c by $\eta_q - \rho_f$ bits. For this we use a ladder of level-κ encodings of 0 of increasing size. We use encodings of only ρ bits of noise, and we are allowed to reach ρ_f bits of noise, so we can work by increment of $\rho_f - \rho$ bits. Therefore we need a ladder of

$$n_e = \left\lceil \frac{\eta_q - \rho_f}{\rho_f - \rho} \right\rceil$$

level-κ encodings of zero, of size $\gamma + \rho_f + i \cdot (\rho_f - \rho)$ for $0 \leqslant i < n_e$. As explained in Section 3.4 we can take $\eta_q = 2\eta + \lambda$, which gives:

$$n_e = \left\lceil \frac{2\eta + \lambda - \rho_f}{\rho_f - \rho} \right\rceil$$

From the parameter constraints in Section 2.2, with $\alpha = \lambda$ and $\beta = 3\lambda$ we can take $\eta = \rho_f + 10\lambda$; with $\rho = \lambda$ and using $\rho_f \geqslant 2 \cdot \kappa \cdot \rho$ we obtain:

$$n_e \leqslant \left\lceil \frac{\rho_f + 21\lambda}{\rho_f - \lambda} \right\rceil \leqslant \left\lceil \frac{2\kappa + 21}{2\kappa - 1} \right\rceil \leqslant 9$$

for $\kappa \geqslant 2$. Therefore only a small number of encodings is needed for the ladder.

5.3 Improved Quadratic Re-randomization

To reduce the public parameters size, in [CLT13] only a small subset of low level encodings is stored and then combined pairwise to generate the full public parameters; such optimization was originally described in [GH11].

In the original CLT scheme, one only stores $\Delta = \lfloor \sqrt{n} \rfloor$ level-0 encoding $x_j^{(0)}$ and also Δ level-1 encodings $x_j^{(1)}$ of 0. Given a level-1 encoding c, one randomizes it using a random subset-sum of pairwise products of the previous encodings:

$$c' = c + \sum_{i,j=1}^{\Delta} \alpha_{ij} \cdot x_i^{(0)} \cdot x_j^{(1)} \bmod x_0 \ ,$$

where the α_{ij}'s are random bits. As a further optimization, one can use as in [CMNT11] a sparse vector α_{ij}, with small Hamming weight θ; for example one can take $\theta = 16$.

In this paper we use a slightly faster variant; namely we compute c' as:

$$c' = c + \left(\sum_{i=1}^{\Delta} \alpha_i \cdot x_i^{(0)} \right) \cdot \left(\sum_{j=1}^{\Delta} \alpha'_j \cdot x_j^{(1)} \right) \ ,$$

where the α_i's and α'_j's are random bits. Therefore we have replaced $\theta = 16$ multiplications by $\Delta \simeq 100$ additions on average, and one multiplication, which is faster in practice.

After re-randomization the CRT components $c' \bmod p_i$ are still upper-bounded in absolute value by $\ell \cdot 2^{2(\rho+\alpha)} + \Delta^2 \cdot 2^{\rho+\alpha} \leqslant 2n \cdot 2^{2(\rho+\alpha)}$. Therefore for level-$\kappa$ encodings we obtain the bound on the noise:

$$\rho_f = \kappa \cdot (2\rho + 2\alpha + \log_2 n + 1) + \rho + \log_2 \ell + 1$$

which is slightly smaller than in Section 2.2.

5.4 Parameters and Timings

We have implemented a one-round $(\kappa + 1)$-way Diffie-Hellman key exchange protocol with $\kappa + 1 = 7$ users, in C++ using the GMP library [Gt14] to perform operations on large integers and fplll [ACPS] for LLL. We refer to Appendix E for a description of the protocol. We provide our concrete parameters and the resulting timings in Table 1, for security parameters ranging from 52 to 80 bits. As in

[CLT13], for a security level λ we expect that the best attack requires at least 2^λ clock cycles. We also provide in Table 2 a comparison with the initial CLT implementation [CLT13] and with the implementation of GGHLite recently described in [ACLL14]; GGHLite is an improvement of GGH proposed by Langlois, Stehlé and Steinfeld [LSS14].

The timings of Table 2 show that the implementation of our scheme improves upon the implementation in [CLT13], especially for the Setup phase, but is not as efficient as [ACLL14] at high security level. The KeyGen phase of the multipartite Diffie-Hellman protocol requires only a few seconds per user, but public parameter size is large, even with our optimizations.

Instantiation	λ	κ	n	η	Δ	ρ	$\gamma = n \cdot \eta$	pk size	Setup	Publish	KeyGen
Small	52	6	540	1679	23	52	$0.9 \cdot 10^6$	27 MB	5.9 s	0.10 s	0.17 s
Medium	62	6	2085	1989	45	62	$4.14 \cdot 10^6$	175 MB	36 s	0.33 s	1.06 s
Large	72	6	8250	2306	90	72	$19.0 \cdot 10^6$	1.2 GB	583 s	2.05 s	6.17 s
Extra	80	6	25305	2619	159	85	$66.3 \cdot 10^6$	6.1 GB	4528 s	7.8 s	23.9 s

Table 1. Parameters and timings to instantiate a *one-round* 7*-way Diffie-Hellman key exchange protocol* with $\kappa = 6$, $\ell = 2\lambda$ and $\alpha, \beta, \nu = \lambda$ on a 16-core computer (Intel Xeon E7-8837 at 2.67GHz). Setup was run in parallel on the 16 cores, while the other steps ran on a single core. Publish and KeyGen timings are per party.

			Setup		Publish (pp)		KeyGen (pp)	
Instantiation	λ	κ	time	# cores	time	# cores	time	#cores
[CLT13]	52	6	7 s	16	0.18 s	1	0.20 s	1
[ACLL14]	52	6	187 s	16	1.59 s	16	0.38 s	16
Ours	52	6	5.9 s	16	0.10 s	1	0.17 s	1
[CLT13]	80	6	27295 s	16	17.8 s	1	20.2 s	1
[ACLL14]	80	6	2532 s	16	7.31 s	16	1.75 s	16
Ours	80	6	4528 s	16	7.8 s	1	23.9 s	1

Table 2. Comparison with timings reported in [CLT13] and [ACLL14]. Both [CLT13] experiments and ours were run on Intel Xeon E7-8837 2.67Ghz and [ACLL14] experiments were run on Intel Xeon E5-2667 v2 3.30Ghz.

References

[ACLL14] Martin R. Albrecht, Catalin Cocis, Fabien Laguillaumie, and Adeline Langlois. The whole alphabet (and then some) agree on a key in one round: making multilinear maps practical. *IACR Cryptology ePrint Archive*, 2014:928, 2014.

[ACPS] M. Albrecht, D. Cadé, X. Pujol, and D. Stehlé. fplll-4.0, a floating-point LLL implementation. Available at `http://perso.ens-lyon.fr/damien.stehle`.

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

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

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

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

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

[CMNT11] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *LNCS*, pages 487–504. Springer, 2011.

[CN12] Yuanmi Chen and Phong Nguyen. Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *LNCS*, pages 502–519. Springer, 2012.

[Cop97] Don Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptology*, 10(4):233–260, 1997.

[DGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *LNCS*, pages 24–43. Springer, 2010.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *LNCS*, pages 1–17. Springer, 2013.

[GGH$^+$13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49. IEEE Computer Society, 2013.

[GGH14] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. *IACR Cryptology ePrint Archive*, 2014:645, 2014. To appear at TCC 2015.

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

[GH11] Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In Kenneth Paterson, editor, *EUROCRYPT*, volume 6632 of *LNCS*, pages 129–148. Springer, 2011.

[Gt14] Torbjörn Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 6.0.0 edition, 2014. `http://gmplib.org/`.

[HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In Ran Canetti and Juan A. Garay, editors, *CRYPTO*, pages 494–512. Springer, 2013.

[Lep15] Tancrède Lepoint. Proof-of-concept implementation of the "new" multilinear maps over the integers, 2015. `https://github.com/tlepoint/new-multilinear-maps`.

[LS14] Hyung Tae Lee and Jae Hong Seo. Security analysis of multilinear maps over the integers. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO*, volume 8616 of *LNCS*, pages 224–240. Springer, 2014.

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

[NS99] Phong Q. Nguyen and Jacques Stern. The hardness of the hidden subset sum problem and its cryptographic implications. In Michael J. Wiener, editor, *CRYPTO '99*, volume 1666 of *LNCS*, pages 31–46. Springer, 1999.

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

[PS14] David Pointcheval and Olivier Sanders. Forward secure non-interactive key exchange. In Michel Abdalla and Roberto De Prisco, editors, *SCN*, volume 8642 of *LNCS*, pages 21–39. Springer, 2014.

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

A Graded Encoding Schemes

In [BS03], Boneh and Silverberg proposed a generalization of pairings called *cryptographic multilinear maps*. A (symmetric) κ-multilinear map is a non-degenerate map $e\colon \mathbb{G}^\kappa \to \mathbb{G}_T$ (where $\mathbb{G}$ and $\mathbb{G}_T$ are groups of prime order p, denoted additively, and g is a generator of $\mathbb{G}$) such that, for all $a_1, \ldots, a_\kappa \in \mathbb{Z}_p$, $e(a_1 \cdot g, \ldots, a_\kappa \cdot g) = (a_1 \cdots a_\kappa) \cdot e(g, \ldots, g)$. In [GGH13a], Garg, Gentry and Halevi introduced a richer structure than cryptographic multilinear maps – as in [HSW13], we denote it leveled multilinear maps – and proposed an approximation of these maps called graded encoding schemes.

Definition 1 (Leveled Multilinear Map). *A κ-leveled multilinear map is a set of bilinear maps $\{e_{i,j} \colon \mathbb{G}_i \times \mathbb{G}_j \to \mathbb{G}_{i+j} \mid i,j \in [\kappa], i+j \leqslant \kappa\}$ where each $\mathbb{G}_i$ is a group of prime order p, denoted additively and generated by g_i for all $i \in [\kappa]$, and such that each map $e_{i,j}$ satisfies $e_{i,j}(a \cdot g_i, b \cdot g_j) = (a \cdot b) \cdot g_{i+j}$ for $a, b \in \mathbb{Z}_p$ and $i, j \in [\kappa]$ and $i + j \leqslant \kappa$.*

The integer κ is called the multilinearity level. The graded encoding schemes candidates are only approximate leveled multilinear maps [GGH13a,CLT13]. The main difference is that encodings are randomized (with some noise) instead of deterministic and one can only test whether an encoding in $\mathbb{G}_\kappa$ encodes 0 or not. We recall the definition of graded encoding system from [GGH13a]:

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

1. *For every $v \in \mathbb{N}$, the sets $\{S_v^{(\alpha)} : \alpha \in R\}$ are disjoint.*
2. *There are binary operations $+$ and $-$ (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every $v \in \mathbb{N}$, and every $u_1 \in S_v^{(a_1)}$ and $u_2 \in S_v^{(a_2)}$, it holds that $u_1 + u_2 \in S_v^{(\alpha_1+\alpha_2)}$ and $u_1 - u_2 \in S_v^{(\alpha_1-\alpha_2)}$ where $\alpha_1 + \alpha_2$ and $\alpha_1 - \alpha_2$ are addition and subtraction in R.*
3. *There is an associative binary operation $\times$ (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every v_1, v_2 with $0 \leqslant v_1 + v_2 \leqslant \kappa$, and every $u_1 \in S_{v_1}^{(\alpha_1)}$ and $u_2 \in S_{v_2}^{(\alpha_2)}$, it holds that $u_1 \times u_2 \in S_{v_1+v_2}^{(\alpha_1 \cdot \alpha_2)}$ where $\alpha_1 \cdot \alpha_2$ is multiplication in R.*

Definition 3 (Graded Encoding Schemes). *A symmetric κ-graded encoding scheme $\mathcal{G}$ with rerandomization and public sampling consists of the following (polynomial time) procedures $\mathcal{G} = \{\mathsf{instGen}, \mathsf{samp}, \mathsf{enc}, \mathsf{reRand}, \mathsf{neg}, \mathsf{add}, \mathsf{mult}, \mathsf{isZero}, \mathsf{ext}\}$:*

$\mathsf{instGen}(1^\lambda, 1^\kappa)$: *The randomized instance generation procedures takes as input the security parameter λ, the multilinearity level κ, and outputs the public parameters $(\mathsf{pp}, \boldsymbol{p}_{zt})$ where pp is a description of a κ-graded encoding system as above, and $\boldsymbol{p}_{zt}$ is a zero-test parameter;*

$\mathsf{samp}(\mathsf{pp})$: *The randomized sampling procedure takes as input the public parameters pp and outputs a "level-0" encoding $u \in S_0^{(\alpha)}$ for a nearly uniform $\alpha \in R$;*

$\mathsf{enc}(\mathsf{pp}, i, u)$: *The (possibly randomized) encoding procedure takes as inputs the parameters pp, an index $i \in [\kappa]$ and a "level-0" encoding $u \in S_0^{(\alpha)}$ for some $\alpha \in R$ and outputs a "level-i" encoding $u' \in S_i^{(\alpha)}$;*

$\mathsf{reRand}(\mathsf{pp}, i, u)$: *The randomized rerandomization procedures takes as inputs the parameters pp, an index $i \in [\kappa]$ and a "level-i" encoding $u \in S_i^{(\alpha)}$ for some $\alpha \in R$, and outputs another $u' \in S_i(\alpha)$ such that, for any $u_1, u_2 \in S_i^{(\alpha)}$, the output distributions of $\mathsf{reRand}(\mathsf{pp}, i, u_1)$ and $\mathsf{reRand}(\mathsf{pp}, i, u_2)$ are nearly the same;*

$\mathsf{neg}(\mathsf{pp}, u)$ *The arithmetic negation is deterministic, takes as input the public parameters pp and a "level-i" encoding $u \in S_i^{(\alpha)}$ for some α in R and outputs a "level-i" encoding $u' \in S_i^{(-\alpha)}$;*

$\mathsf{add}(\mathsf{pp}, u_1, u_2)$ *The arithmetic addition is deterministic, takes as input the public parameters pp and "level-i" encodings $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$ for some α_1, α_2 in R and outputs a "level-i" encoding $u' \in S_i^{(\alpha_1+\alpha_2)}$;*

$\mathsf{mult}(\mathsf{pp}, u_1, u_2)$ *The arithmetic multiplication is deterministic, takes as input the public parameters* pp*, a "level-i" encoding $u_1 \in S_i^{(\alpha_1)}$ and a level-j encoding $u_2 \in S_j^{(\alpha_2)}$ for some α_1, α_2 in R such that $i + j \leqslant \kappa$ and outputs a "level-$(i+j)$" encoding $u' \in S_{i+j}^{(\alpha_1 \cdot \alpha_2)}$;*

$\mathsf{isZero}(\mathsf{pp}, u)$ *The zero-testing procedure is deterministic, takes as input the public parameters* pp *and a "level-κ" encoding $u \in S_\kappa^{(\alpha)}$ and outputs 1 if $\alpha = 0$ and 0 otherwise;*

$\mathsf{ext}(\mathsf{pp}, \boldsymbol{p}_{zt}, u)$ *The extraction procedure is deterministic, takes as input the public parameters* pp*, the zero-test parameter $\boldsymbol{p}_{zt}$ and a "level-κ" encoding $u \in S_\kappa^{(\alpha)}$ and outputs a λ-bit string s such that:*

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

The graded encoding schemes proposed in [GGH13a,CLT13] consider slightly relaxed definitions of isZero and ext, where isZero can output 1 for some non-zero encoding with negligible probability and ext can extract to different outputs for encodings of the same ring element with negligible probability.

Hardness assumption. In [GGH13a] is introduced an analogue of the multilinear decisional Diffie-Hellman assumption (see [BS03]) for graded encoding schemes: the Graded Decisional Diffie-Hellman assumption (GDDH).

Definition 4 (κ-GDDH). *Let $\mathcal{G}$ denote a GES. For any security parameter $\lambda \in \mathbb{N}$, the κ-Graded Decisional Diffie-Hellman assumption is to distinguish between the following distributions $\mathcal{D}_0[(\mathsf{pp}, \boldsymbol{p}_{zt})]$ and $\mathcal{D}_1[(\mathsf{pp}, \boldsymbol{p}_{zt})]$, where $(\mathsf{pp}, \boldsymbol{p}_{zt}) \leftarrow \mathcal{G}.\mathsf{instGen}(1^\lambda, 1^\kappa)$:*

$$\mathcal{D}_0[(\mathsf{pp}, \boldsymbol{p}_{zt})] = \left\{ \left(\{u'_j\}_{j\in[\kappa+1]}, \gamma\right) \;\middle|\; \begin{array}{r} \forall j \in [\kappa+1], a_j \leftarrow \mathcal{G}.\mathsf{samp}(\mathsf{pp}), \\ u_j \leftarrow \mathcal{G}.\mathsf{enc}(\mathsf{pp}, 1, a_j) \\ u'_j \leftarrow \mathcal{G}.\mathsf{reRand}(\mathsf{pp}, 1, u_j) \\ \gamma \leftarrow \mathcal{G}.\mathsf{ext}(\mathsf{pp}, \boldsymbol{p}_{zt}, a_{\kappa+1} \cdot \prod_{j\in[\kappa]} u'_j) \end{array} \right\}$$

and

$$\mathcal{D}_1[(\mathsf{pp}, \boldsymbol{p}_{zt})] = \left\{ \left(\{u'_j\}_{j\in[\kappa+1]}, \gamma\right) \;\middle|\; \begin{array}{r} \forall j \in [\kappa+1], a_j \leftarrow \mathcal{G}.\mathsf{samp}(\mathsf{pp}), \\ u_j \leftarrow \mathcal{G}.\mathsf{enc}(\mathsf{pp}, 1, a_j) \\ u'_j \leftarrow \mathcal{G}.\mathsf{reRand}(\mathsf{pp}, 1, u_j) \\ \gamma \leftarrow \{0,1\}^\lambda \end{array} \right\}$$

B Proof of Lemma 2

We have by definition of c'_1:

$$c'_1 = \hat{c}_1 + \sum_{j=1}^{\tau} b_j \cdot x_j + \sum_{j=1}^{n+1} b'_j \cdot \Pi_j \tag{25}$$

where $\hat{c}_1$ is given as input of the reRand algorithm. We have:

$$\hat{c}_1 = \sum_{i=1}^{n} (\hat{r}_i \cdot g_i + m_i) \cdot u_i + \hat{r}_{n+1} \cdot x_0 \tag{26}$$

where we write $\hat{c}_1/z \bmod p_i = \hat{r}_i \cdot g_i + m_i$, where $|\hat{r}_i \cdot g_i + m_i| \leqslant \rho_{\mathsf{enc}}$ for all $1 \leqslant i \leqslant n$, where $\rho_{\mathsf{enc}} = 2\ell \cdot 2^{2\rho+2\alpha}$; since $0 \leqslant \hat{c}_1 < x_0$, we must also have $|\hat{r}_{n+1}| \leqslant 1 + n \cdot \rho_{\mathsf{enc}}$. We obtain from (25):

$$c'_1/z \equiv \hat{r}_i \cdot g_i + m_i + \sum_{j=1}^{\tau} b_j \cdot r_{ij} \cdot g_i + \sum_{j=1}^{n+1} b'_j \cdot \varpi_{ij} \cdot g_i \pmod{p_i}$$

Since the column vectors of the matrix $\boldsymbol{X}$ are randomly generated in the half-open parallelepiped spanned by the columns of $\boldsymbol{\Pi}$, we have $|r_{ij} \cdot g_i| \leqslant 3n \cdot 2^{\rho+\alpha}$ for all i, j. Therefore the right-hand side of the previous equation is upper bounded in absolute value by:

$$1 + n \cdot \rho_{\mathsf{enc}} + \tau \cdot 3n \cdot 2^{\rho+\alpha} + 2n^2 \cdot 2^{\mu+\alpha+\rho} \leqslant 4n^2 \cdot 2^{\mu+\alpha+\rho}$$

using $\mu = \rho + \alpha + \lambda$. Therefore under the condition:

$$\mu + \alpha + \rho + \lambda \leqslant \eta$$

the right-hand side does not wrap modulo p_i.

We define the function $g \colon \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_n} \times \mathbb{Z} \to \mathbb{Z}$ with:

$$g(a_1, \ldots, a_n, a_{n+1}) = \sum_{i=1}^{n} a_i \cdot u_i + a_{n+1} \cdot x_0$$

where the CRT coefficients u_i are defined by (9). We have that g a bijection; namely its inverse f is defined as:

$$f(a) = \Big([a \cdot z]_{p_1}, \ldots, [a \cdot z]_{p_n}, \big(a - \sum_{i=1}^{n} [a \cdot z]_{p_i} \cdot u_i\big)/x_0\Big).$$

Let $a, b \in \mathbb{Z}$; if for all $1 \leqslant i \leqslant n$, the i-th coefficient a_i of $f(a)$ and the i-th coefficient b_i of $f(b)$ are such that $|a_i + b_i| < p_i/2$, we have:

$$f(a+b) = f(a) + f(b)\,.$$

Since f is a bijection, we can work with the $(n+1)$-vector $f(a)$ instead of $a \in \mathbb{Z}$; moreover if the CRT components are small enough, we can work with these vectors additively.

We can write from (26):

$$f(\hat{c}_1) = (\hat{r}_1 \cdot g_1 + m_1, \ldots, \hat{r}_n \cdot g_n + m_n, \hat{r}_{n+1}) = \mathsf{diag}(g_1, \ldots, g_n, 1) \cdot \hat{\boldsymbol{r}} + \boldsymbol{m}$$

where $\hat{\boldsymbol{r}} = (\hat{r}_i)_{1 \leqslant i \leqslant n}$ and $\boldsymbol{m} = (m_1, \ldots, m_n, 0)$. Since the CRT components do not wrap modulo p_i, Equation (25) can be rewritten

$$\begin{aligned} f(c'_1) &= f\Big(\hat{c}_1 + \sum_{j=1}^{\tau} b_j \cdot x_j + \sum_{j=1}^{n+1} b'_j \cdot \Pi_j\Big) \\ &= \boldsymbol{m} + \mathsf{diag}(g_1, \ldots, g_n, 1) \cdot \Big(\hat{\boldsymbol{r}} + \boldsymbol{X} \cdot \boldsymbol{b} + \boldsymbol{\Pi} \cdot \boldsymbol{b}'\Big)\,, \end{aligned}$$

where $\boldsymbol{b} = (b_i)_i$ and $\boldsymbol{b}' = (b'_i)_i$. Now, using the notations introduced in Lemma 2 we can also write:

$$f(c'_1) = \boldsymbol{m} + \mathsf{diag}(g_1, \ldots, g_n, 1) \cdot \boldsymbol{r}\,,$$

which gives:

$$\boldsymbol{r} = \hat{\boldsymbol{r}} + \boldsymbol{X} \cdot \boldsymbol{b} + \boldsymbol{\Pi} \cdot \boldsymbol{b}' \,.$$

The latter equation is the same equation as in [CLT13, Sec. 4.3], except that $\boldsymbol{r}$ has $n+1$ components instead of n, because of the additional multiple of x_0; the rest of the proof is therefore similar; it is based on the Leftover-Hash-Lemma over lattices and some facts on diagonally dominant matrices [CLT13, Sec. 4].

We first recall the Leftover-Hash-Lemma over lattices. Let $L \subset \mathbb{Z}^n$ be a lattice of rank n of basis $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$. We denote by $\mathcal{D}_{\boldsymbol{B}}$ the distribution obtained by generating a random element in the quotient $\mathbb{Z}^n/L$ and taking its unique representative in the half-open parallelepiped generated by the basis $\boldsymbol{B}$. For any $\mu \in \mathbb{Z}^*$, we denote by $\mu\boldsymbol{B}$ the basis $(\mu\boldsymbol{b}_1, \ldots, \mu\boldsymbol{b}_n)$.

Lemma 4 (Leftover-Hash-Lemma over Lattices [CLT13]). *Let $L \subset \mathbb{Z}^n$ be a lattice of rank n of basis $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$. Let $\boldsymbol{x}_i$ for $1 \leqslant i \leqslant m$ be generated independently according to the distribution $\mathcal{D}_{\boldsymbol{B}}$. Set $s_1, \ldots, s_m \leftarrow \{0,1\}$ and $t_1, \ldots, t_n \leftarrow \mathbb{Z} \cap [0, 2^\mu)$. Let $\boldsymbol{y} = \sum_{i=1}^m s_i \boldsymbol{x}_i + \sum_{i=1}^n t_i \boldsymbol{b}_i$ and $\boldsymbol{y'} \leftarrow \mathcal{D}_{2^\mu \boldsymbol{B}}$. Then the distributions $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m, \boldsymbol{y})$ and $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m, \boldsymbol{y'})$ are ε-statistically close, with $\varepsilon = mn \cdot 2^{-\mu} + 1/2\sqrt{|\det L|/2^m}$.*

Since at instance generation the columns of $\boldsymbol{X}$ are generated uniformly and independently in the parallelepiped spanned by the columns of $\boldsymbol{\Pi}$, Lemma 4 shows that the distribution of $(\mathsf{pp}, \boldsymbol{r})$ is ε_1-close to the distribution of $(\mathsf{pp}, \hat{\boldsymbol{r}} + \mathcal{D}_{2^\mu \boldsymbol{\Pi}})$, with

$$\varepsilon_1 = \tau(n+1)2^{-\mu} + 1/2\sqrt{|\det \boldsymbol{\Pi}|/2^\tau} \,.$$

Moreover it is proved in [CLT13] that $\mathcal{D}_{2^\mu \boldsymbol{B}}$ is not significantly modified when a small vector $\boldsymbol{z} \in \mathbb{Z}^n$ is added, if the operator norm of the corresponding matrix B^{-1} is upper-bounded.

Lemma 5 ([CLT13]). *Let $L \subset \mathbb{Z}^n$ be a full-rank lattice of basis $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$, and let $B \in \mathbb{Z}^{n \times n}$ be the matrix whose column vectors are the $\boldsymbol{b}_i$'s. For any $\boldsymbol{z} \in \mathbb{Z}^n$, the distribution of $\boldsymbol{z} + \mathcal{D}_{2^\mu \boldsymbol{B}}$ is ε-statistically close to that of $\mathcal{D}_{2^\mu \boldsymbol{B}}$, where $\varepsilon = 2^{-\mu} \cdot (\|\boldsymbol{z}\|_\infty \cdot \|B^{-1}\|_\infty + 1)$.*

Using Lemma 5, we obtain that the distribution of $(\mathsf{pp}, \boldsymbol{r})$ is $(\varepsilon_1 + \varepsilon_2)$-close to that of $(\mathsf{pp}, \mathcal{D}_{2^\mu \boldsymbol{\Pi}})$ for

$$\varepsilon_2 = 2^{-\mu} \cdot (\|\hat{\boldsymbol{r}}\|_\infty \cdot \|\boldsymbol{\Pi}^{-1}\|_\infty + 1) \,.$$

We recall the following facts for diagonally dominant matrices [Var75,Pri51]. Given a matrix $B = (b_{ij}) \in \mathbb{R}^{n \times n}$, we let $\Lambda_i(B) = \sum_{k \neq i} |b_{ik}|$. A matrix $B = (b_{ij}) \in \mathbb{R}^{n \times n}$ is said to be *diagonally dominant* if $|b_{ii}| > \Lambda_i(B)$ for all i.

Fact 1 *Let $B = (b_{ij}) \in \mathbb{R}^{n \times n}$ be a diagonally dominant matrix. Then the matrix B is invertible and $\|B^{-1}\|_\infty \leqslant \max_{i=1,\ldots,n} (|b_{ii}| - \Lambda_i(B))^{-1}$ where $\| \cdot \|_\infty$ is the operator norm on $n \times n$ matrices with respect to the ℓ^∞ norm on $\mathbb{R}^n$.*

Fact 2 *Let $B = (b_{ij}) \in \mathbb{R}^{n \times n}$ be a diagonally dominant matrix. Then*

$$\prod_{i=1}^n (|b_{ii}| - \Lambda_i(B)) \leqslant |\det B| \leqslant \prod_{i=1}^n (|b_{ii}| + \Lambda_i(B)) \,.$$

Using Fact 2 we obtain using $|\varpi_{ii}| \leqslant (n+2) \cdot 2^{\rho}$ and $\Delta_i(\boldsymbol{\Pi}) \leqslant n \cdot 2^{\rho}$ for all i:

$$\varepsilon_1 \leqslant \tau(n+1)2^{-\mu} + 2^{((n+1)(\rho+\log_2(3n))-\tau)/2} .$$

Given the constraint $\tau \geqslant (n+2) \cdot \rho + 2\lambda$, we obtain $\varepsilon_1 = \mathsf{negl}(\lambda)$. Moreover using Fact 1, we obtain $\|\boldsymbol{\Pi}^{-1}\|_{\infty} \leqslant 2^{-\rho}$ and therefore:

$$\varepsilon_2 \leqslant 2^{-\mu} \cdot ((1 + n \cdot \rho_{\mathsf{enc}}) \cdot 2^{-\alpha+1} \cdot 2^{-\rho} + 1) \leqslant 2^{-\mu} \cdot n \cdot 3\ell \cdot 2^{2\rho+2\alpha} \cdot 2^{-\rho-\alpha+1} \leqslant 6\ell n \cdot 2^{\rho+\alpha-\mu} .$$

With $\mu = \rho + \alpha + \lambda$, we obtain $\varepsilon_2 = \mathsf{negl}(\lambda)$. This proves Lemma 2.

C Generation of the Zero-Testing Matrix

For the construction of zero-testing parameters, it is necessary to generate, with sufficient entropy, an invertible matrix $\boldsymbol{H} \in \mathbb{Z}^{n\times n}$ such that the operator norm of both $\boldsymbol{H}$ and its inverse is suitably bounded: $\|\boldsymbol{H}\|_{\infty} \leqslant 2^{\beta}$ and $\|\boldsymbol{H}^{-1}\|_{\infty} \leqslant 2^{\beta}$. The following approach to do so was described in [CLT13]: simply choose $\boldsymbol{H}$ as a product of random matrices of the form:

$$\boldsymbol{H}_A = \begin{pmatrix} I_{\lfloor n/2 \rfloor} & A \\ 0 & I_{\lceil n/2 \rceil} \end{pmatrix} .$$

and transposes of such matrices, with A of size $\lfloor n/2 \rfloor \times \lceil n/2 \rceil$ and coefficients in $\{-1, 0, 1\}$.

Indeed, since the matrix norm $\|\boldsymbol{M}\|_{\infty}$ is simply the maximum absolute row sum of $\boldsymbol{M}$, both $\|\boldsymbol{H}_A\|_{\infty}$ and $\|\boldsymbol{H}_A^T\|_{\infty}$ are bounded by $1 + \lceil n/2 \rceil$, and the same holds for their inverses because $\boldsymbol{H}_A^{-1} = \boldsymbol{H}_{(-A)}$. Therefore, if we generate $\boldsymbol{H}$ as a product $\boldsymbol{H}_1 \cdots \boldsymbol{H}_{\beta'}$ where each $\boldsymbol{H}_i$ is randomly chosen as either $\boldsymbol{H}_A$ or $\boldsymbol{H}_A^T$ and $\beta' \leqslant \beta/\lceil \log_2(1 + \lceil n/2 \rceil)\rceil$, the required bounds on $\|\boldsymbol{H}\|_{\infty}$ and $\|\boldsymbol{H}^{-1}\|_{\infty}$ hold:

$$\|\boldsymbol{H}\|_{\infty} \leqslant \prod_{i=1}^{\beta'} \|\boldsymbol{H}_i\|_{\infty} \leqslant (1 + \lceil n/2 \rceil)^{\beta'} \leqslant 2^{\beta},$$

and similarly for $\boldsymbol{H}^{-1}$.

However, it was pointed out by Lee and Seo [LS14] that the norms of matrices $\boldsymbol{H}$ generated with this method is typically much smaller than the conservative upper bound by 2^{β}. This may be a security concern (it definitely was in the original CLT scheme, but not necessarily in ours: see Section 4.3), and is mainly due to the fact that $\boldsymbol{H}_A \cdot \boldsymbol{H}_{A'} = \boldsymbol{H}_{(A+A')}$, and thus the matrix norm increases additively rather than multiplicatively in products that involve only matrices of the form $\boldsymbol{H}_A$ (and not their transposes). To alleviate this problem, Lee and Seo suggested to always choose $\boldsymbol{H}_i$ of the form $\boldsymbol{H}_A$ when i is even and of the form $\boldsymbol{H}_A^T$ when i is odd; they found that the resulting distribution of $\boldsymbol{H}$ contains substantially larger matrices.

It is easy to improve the algorithm even further by not fixing the number β' of matrices $\boldsymbol{H}_A$ in the product a priori: we can simply keep multiplying alternatingly by random matrices of the form $\boldsymbol{H}_A$ and $\boldsymbol{H}_A^T$ until either the norm of the product or its inverse exceeds $2^{\beta}/(1 + \lceil n/2 \rceil)$, say. Since we can always keep track of the inverse as we go, the overhead of this improvement is relatively small, and although it is not trivial to bound the number of required iterations, we find that the modified algorithm is only a few times slower than the one from [CLT13] in practice, and returns matrices with much larger coefficients.

D Proof of Lemma 3

We consider a level-κ encoding c, with $0 \leqslant c < x_0$, which we can write:

$$c = \sum_{i=1}^{n} \left(r_i + m_i \cdot g_i^{-1} \bmod p_i\right) \cdot u'_i - a \cdot x_0 \tag{27}$$

where

$$u'_i = \left(g_i \cdot z^{-k} \cdot \left(\frac{x_0}{p_i}\right)^{-1} \bmod p_i\right) \cdot \frac{x_0}{p_i}$$

From (27), using $0 \leqslant c < x_0$ and $0 \leqslant u'_i < x_0$ for all $1 \leqslant i \leqslant n$, we obtain:

$$|a| \leqslant 1 + n \cdot 2^{\eta}$$

Consider the row vectors $\boldsymbol{s}, \boldsymbol{t} \in \mathbb{Z}^n$ given by:

$$\begin{aligned} s_i &= \left(r_i + m_i \cdot g_i^{-1} \bmod p_i\right) \cdot \beta_i \mod N \\ t_i &= \alpha_i \cdot \sum_{k=1,\, k\neq i}^{n} \left(r_k + m_k \cdot g_k^{-1} \bmod p_k\right) \cdot \frac{u'_k}{p_i} - a \cdot \alpha_i \cdot \frac{x_0}{p_i} \end{aligned}$$

for all i (where the mod operator denotes centered modular reduction, so that $|x \bmod N| \leqslant |x|$ for all $x \in \mathbb{Z}$). Then, from (14) we obtain:

$$\boldsymbol{\omega}^T = \boldsymbol{H}^T \left(\boldsymbol{s} + \boldsymbol{t}\right)^T \bmod N. \tag{28}$$

In all cases, we have for all i, using $N > 2^{\gamma+2\eta}$:

$$\begin{aligned} |t_i| &\leqslant 2^{\eta-1} \sum_{k=1,\, k\neq i}^{n} \frac{p_k}{2} \cdot \frac{|u'_k|}{p_i} + 2^{\eta-1}|a| \cdot \frac{x_0}{p_i} \\ &\leqslant (n-1)2^{\eta+\gamma} + 2^{\gamma} \cdot (1 + n \cdot 2^{\eta}) \leqslant 2n2^{\eta+\gamma} \leqslant \frac{n}{2^{\eta-1}} \cdot N \end{aligned}$$

and hence $\|\boldsymbol{t}\|_\infty \leqslant n2^{-\eta+1} \cdot N$.

On the other hand, the size of $\boldsymbol{s}$ depends on whether $\boldsymbol{m}$ is the zero vector or not. Indeed, suppose first that $\boldsymbol{m} = 0$. For all i we then have:

$$|s_i| = \left|(r_i \bmod p_i) \cdot \beta_i \bmod N\right| \leqslant |r_i| \cdot |\beta_i| < 2^{\rho_f - \eta + 2} \cdot N$$

and hence $\|\boldsymbol{s} + \boldsymbol{t}\|_\infty < 2^{\rho_f - \eta + 3} \cdot N$. Now, in view of (28), we obtain with $\nu = \eta - \rho_f - \beta - \lambda - 3$:

$$\begin{aligned} \|\boldsymbol{\omega}\|_\infty = \left\|\boldsymbol{H}^T \left(\boldsymbol{s} + \boldsymbol{t}\right)^T \bmod N\right\|_\infty &\leqslant \left\|\boldsymbol{H}^T \left(\boldsymbol{s} + \boldsymbol{t}\right)^T\right\|_\infty \leqslant \|\boldsymbol{H}^T\|_\infty \|\boldsymbol{s} + \boldsymbol{t}\|_\infty \\ &< 2^{\beta + \rho_f - \eta + 3} \cdot N = 2^{-\nu - \lambda} \cdot N \end{aligned}$$

as required.

Conversely, suppose that $\boldsymbol{m} \neq 0$, so that $m_i \neq 0$ for at least one index i. Recall that

$$\beta_i = \alpha_i \cdot (u'_i / p_i) \bmod N.$$

In view of Lemma 6 below, we must have $|\beta_i| \geqslant N/(2p_i)$, which gives:

$$2^{-\eta-1} \cdot N < |\beta_i| \leqslant 2^{-\eta+2} \cdot N.$$

Lemma 6. *For any integers a, b, m such that $|b| < m/2$, $\gcd(a, m) = 1$ and $a \neq 0$ does not divide b, if $x = b/a \bmod m$, then $|x| \geqslant \frac{m}{2|a|}$.*

Proof. Indeed, if we had $|a| \cdot |x| < m/2$, then the congruence $ax \equiv b \pmod{m}$ would actually be an equality over $\mathbb{Z}$, implying that a divides b: a contradiction. $\square$

Let:

$$f_i = (g_i^{-1} \bmod p_i) \cdot \beta_i \bmod N. \tag{29}$$

We have:

$$f_i = \frac{(g_i^{-1} \bmod p_i) \cdot \alpha_i \cdot u_i'}{p_i} \bmod N$$

and since p_i does not divide the numerator, Lemma 6 implies

$$|f_i| \geqslant N/(2p_i) > 2^{-\eta-1} \cdot N$$

Similarly, there exists $|\mu| < g_i/2$ such that $g_i \cdot (g_i^{-1} \bmod p_i) = 1 + \mu p_i$, and we can then write from (29):

$$f_i = \frac{(1 + \mu p_i) \cdot \beta_i}{g_i} \bmod N = \frac{\beta_i + \mu \alpha_i u_i'}{g_i} \bmod N. \tag{30}$$

If g_i divided the numerator, we would thus get using $N > 2^{\gamma+2\eta}$:

$$|f_i| = \left| \frac{\beta_i + \mu \cdot \alpha_i \cdot u_i'}{g_i} \right| \leqslant 2^{-\eta-\alpha+3} \cdot N + 2^{\gamma+\eta-2} \leqslant 2^{-\eta-1} \cdot N$$

which would contradict the previous bound. Therefore, Lemma 6 applies and ensures that

$$|f_i| \geqslant N/(2g_i) \geqslant 2^{-\alpha-1} \cdot N.$$

Now, since $m_i \neq 0$ and g_i is prime, m_i has an inverse $m_i' = m_i^{-1} \bmod g_i$ in $\mathbb{Z}_{g_i}$, and we can write:

$$s_i = (m_i'^{-1} \bmod g_i) f_i + r_i \beta_i \bmod N.$$

If we define $|\mu'| < g_i/2$ such that $m_i m_i' = 1 + \mu' g_i$, we obtain using (30):

$$m_i' s_i \equiv (1 + \mu' g_i) f_i + m_i' r_i \beta_i \equiv f_i + \mu'(\beta_i + \mu \alpha_i u_i') + m_i' r_i \beta_i \pmod{N}.$$

Therefore:

$$f_i = \left[m_i' s_i - (m_i' r_i + \mu')\beta_i - \mu\mu'\alpha_i u_i' \right] \bmod N,$$

and since modular reductions are centered, it follows that:

$$\begin{aligned} |f_i| &\leqslant 2^{\alpha}|s_i| + 2^{\alpha+\rho_f+2-\eta} \cdot N + 2^{2\alpha+\eta-1+\gamma} \\ &\leqslant 2^{\alpha} \cdot (|s_i| + 2^{\rho_f-\eta+2} \cdot N + 2^{\alpha-\eta-1} \cdot N) \leqslant 2^{\alpha}\big(|s_i| + 2^{3+\rho_f-\eta} \cdot N\big), \end{aligned}$$

and as a result:

$$|s_i| \geqslant 2^{-\alpha}|f_i| - 2^{3+\rho_f-\eta} \cdot N \geqslant 2^{-2\alpha-1} \cdot N - 2^{3+\rho_f-\eta} \cdot N$$

and using $\rho_f \leqslant \eta - 2\alpha - 5$, we get:

$$|s_i| \geqslant 2^{-2\alpha-2} \cdot N.$$

It follows that $\|\boldsymbol{s}\|_\infty \geqslant 2^{-2\alpha-2} \cdot N$, and hence:

$$\|\boldsymbol{s}+\boldsymbol{t}\|_\infty \geqslant \|\boldsymbol{s}\|_\infty - \|\boldsymbol{t}\|_\infty > 2^{-2\alpha-3} \cdot N.$$

Finally, using (28) again, we get:

$$\|\boldsymbol{s}+\boldsymbol{t}\|_\infty = \left\|(\boldsymbol{H}^T)^{-1}\boldsymbol{\omega}^T \bmod N\right\|_\infty \leqslant \left\|(\boldsymbol{H}^T)^{-1}\boldsymbol{\omega}^T\right\|_\infty \leqslant \left\|(\boldsymbol{H}^T)^{-1}\right\|_\infty \cdot \|\boldsymbol{\omega}\|_\infty$$
$$\|\boldsymbol{\omega}\|_\infty \geqslant \frac{\|\boldsymbol{s}+\boldsymbol{t}\|_\infty}{\left\|(\boldsymbol{H}^T)^{-1}\right\|_\infty} > 2^{-2\alpha-\beta-3} \cdot N \geqslant 2^{-\nu+2} \cdot N$$

which concludes the proof.

E One-Round N-Way Diffie-Hellman Key Exchange Protocol

A straightforward application of multilinear maps is multipartite Diffie-Hellman key exchange, with $N = \kappa + 1$ users, where κ is the maximum level of the multilinear map scheme. It was first described in [BS03], and was adapted to graded encoding schemes in [GGH13a]. As in [CLT13], our scheme can be used to instantiate a one-round N-way Diffie-Hellman key exchange protocol in the common reference string model, under the GDDH assumption, with $N = \kappa + 1$ users. We recall the construction from [CLT13].

$\mathsf{Setup}(1^\lambda, 1^N)$. Output $(\mathsf{pp}, \boldsymbol{p}_{zt}) \leftarrow \mathsf{instGen}(1^\lambda, 1^\kappa)$ as the public parameter, with $\kappa = N - 1$.

$\mathsf{Publish}(\mathsf{pp}, i)$. Each party i samples a random $u_i \leftarrow \mathsf{samp}(\mathsf{pp})$ as a secret value, and publishes as the public value the corresponding level-1 encoding

$$u_i' \leftarrow \mathsf{reRand}(\mathsf{pp}, 1, \mathsf{enc}(\mathsf{pp}, 1, u_i)) .$$

$\mathsf{KeyGen}(\mathsf{pp}, \boldsymbol{p}_{zt}, i, u_i, \{u_j'\}_{j\neq i})$. Each party i computes $\tilde{u}_i = u_i \cdot \prod_{j\neq i} u_j'$, and uses the extraction routine to locally compute the key $s \leftarrow \mathsf{ext}(\mathsf{pp}, \boldsymbol{p}_{zt}, \tilde{u}_i)$.

The protocol is correct (i.e. the N parties generate the same shared key s with high probability) because all parties get valid encodings of the same vector when using the constraints of Section 2.2. The security of the protocol follows from the randomness property of the extraction procedure, and the GDDH hardness assumption. The proof of the following theorem is the same as in [CLT13].

Theorem 3 (Th. 3 of [CLT13]). *The protocol described above is a secure one-round N-way Diffie-Hellman key exchange if the GDDH assumption holds for the underlying encoding scheme.*

F Public Sampling Domain

The sampling procedure of graded encoding schemes (namely samp) has been defined as a procedure outputting a nearly uniform (unknown) element in the domain $R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$ in [CLT13]. However, some applications building upon graded encoding schemes might require to *select* elements from the domain R (see e.g. [PS14, Sec. 4.4]). In [BWZ14, Sec. 4.3], a procedure to publicly encode values up to $\lfloor \log_2(g_1 \cdots g_n) \rfloor$ bits was described for a CLT-based scheme. In this section, we show that their technique can be applied to our new scheme.

First, we publish as part of our instance generation $\theta = \lfloor \log_2(g_1 \times \cdots \times g_n) \rfloor + 1$ elements $y_0, \ldots, y_{\theta-1}$ such that

$$y_j \equiv r_{ij} \cdot g_i + (2^j \bmod g_i) \pmod{p_i},$$

for random $r_{ij} \in (-2^\rho, 2^\rho) \cap \mathbb{Z}$; the integers y_j's are computed by CRT modulo x_0.

Then, we define the procedure $\mathsf{publicSamp}(\mathsf{pp}, m)$ which takes as input the public parameters pp and an integer $M \in \mathbb{Z}$, $M < 2^{\theta-1}$. Denote $M = M_0 \cdots M_{\theta-1}$ the binary decomposition of M. Compute the level-0 encoding

$$c = \sum_{i=0}^{\theta-1} M_i \cdot y_i .$$

As required, $c \equiv (r_i \cdot g_i + m_i) \pmod{p_i}$ encodes the value $\boldsymbol{m} = (m_i)_i = (M \bmod g_1, \ldots, M \bmod g_n) \in R$, and we get $|r_i \cdot g_i + m_i| \leqslant \theta \cdot 2^{\rho+\alpha}$ for all i.

G Preventing SubM and DLIN Attacks for GGH

In this section we show how to adapt our technique to the GGH scheme [GGH13a]. The goal is to prevent attacks against base group problems such as SubM and DLIN, which are known to be easy in GGH. However, a thorough security analysis of that GGH variant is beyond the scope of this paper.

The GGH multilinear maps scheme. The construction from [GGH13a] works in the polynomial ring $R = \mathbb{Z}[X]/(X^n + 1)$, where n is large enough to ensure security. One generates a secret short ring element $\boldsymbol{g} \in R$, generating a principal ideal $\mathcal{I} = \langle \boldsymbol{g} \rangle \subset R$. One also generates an integer parameter q and another random secret $\boldsymbol{z} \in R/qR$. One encodes elements of the quotient ring $R/\mathcal{I}$, namely elements of the form $\boldsymbol{e} + \mathcal{I}$ for some $\boldsymbol{e}$, as follows: a level-i encoding of the coset $\boldsymbol{e} + \mathcal{I}$ is an element of the form $u_k = [\boldsymbol{c}/\boldsymbol{z}^i]_q$, where $\boldsymbol{c} \in \boldsymbol{e} + \mathcal{I}$ is short. Such encodings can be both added and multiplied, as long as the norm of the numerators remain shorter than q; in particular the product of κ encodings at level 1 gives an encoding at level κ. For such level-κ encodings one can then define a zero-testing parameter

$$\boldsymbol{p}_{zt} = [\boldsymbol{h}\boldsymbol{z}^\kappa/\boldsymbol{g}]_q$$

for some small $\boldsymbol{h} \in R$. Then given a level-κ encoding $\boldsymbol{u} = [\boldsymbol{c}/\boldsymbol{z}^\kappa]_q$ one can compute:

$$\boldsymbol{\omega} = [\boldsymbol{p}_{zt} \cdot \boldsymbol{u}]_q = [\boldsymbol{h}\boldsymbol{c}/\boldsymbol{g}]_q$$

We can write $\boldsymbol{c} = \boldsymbol{r}\boldsymbol{g} + \boldsymbol{e}$ for some small $\boldsymbol{r}, \boldsymbol{e} \in R$, which gives:

$$\omega \equiv \boldsymbol{h} \cdot (\boldsymbol{r} + \boldsymbol{e} \cdot \boldsymbol{g}^{-1}) \pmod{q}$$

Therefore when $\boldsymbol{u}$ is an encoding of 0, *i.e.* when $\boldsymbol{e} = 0$, we obtain $\boldsymbol{\omega} = [\boldsymbol{h} \cdot \boldsymbol{r}]_q$ and therefore $\boldsymbol{\omega}$ is small. Otherwise when $\boldsymbol{e} \neq 0$ we have that $\boldsymbol{\omega}$ is not small with high probability. This gives a zero-testing procedure.

Our GGH Variant. We show how to adapt our techique to GGH. Let N be a prime integer of the same size as q, and let $\boldsymbol{p}_{zt} \in R/NR$. For zero-testing a level-κ encoding $\boldsymbol{u}$, we compute:

$$\boldsymbol{\omega} = [\boldsymbol{p}_{zt} \cdot \boldsymbol{u}]_N$$

where $\boldsymbol{u}$ is viewed as an element of R, instead of R/qR. As previously we must ensure that $\boldsymbol{\omega}$ is small when $\boldsymbol{u}$ is an encoding of 0. We show how to generate $\boldsymbol{p}_{zt}$ and $\boldsymbol{z}$ to ensure this property.

Let $\boldsymbol{u}$ be a level-κ encoding, with $\boldsymbol{u} = [\boldsymbol{c}/\boldsymbol{z}^\kappa]_q$, where $\boldsymbol{c} = \boldsymbol{r}\boldsymbol{g} + \boldsymbol{e}$. We can write over R:

$$\boldsymbol{u} = [\boldsymbol{r} + \boldsymbol{e} \cdot \boldsymbol{g}^{-1}]_q \cdot [\boldsymbol{g}/\boldsymbol{z}^\kappa]_q - \boldsymbol{k} \cdot q \tag{31}$$

for some $\boldsymbol{k} \in R$. We obtain:

$$\boldsymbol{\omega} \equiv \boldsymbol{p}_{zt} \cdot \boldsymbol{u} \equiv [\boldsymbol{r} + \boldsymbol{e} \cdot \boldsymbol{g}^{-1}]_q \cdot [\boldsymbol{g}/\boldsymbol{z}^\kappa]_q \cdot \boldsymbol{p}_{zt} - \boldsymbol{k} \cdot q \cdot \boldsymbol{p}_{zt} \pmod N \tag{32}$$

We wish to obtain a small $\boldsymbol{\omega}$ when $\boldsymbol{e} = 0$. For this we generate $\boldsymbol{p}_{zt}$ and $\boldsymbol{z}$ such that both $[\boldsymbol{g}/\boldsymbol{z}^\kappa]_q \cdot \boldsymbol{p}_{zt}$ and $q \cdot \boldsymbol{p}_{zt}$ are small modulo N. We proceed as follows. We first generate a small $\boldsymbol{h} \in R$, and we let:

$$\boldsymbol{p}_{zt} = q^{-1} \cdot \boldsymbol{h} \bmod N \tag{33}$$

This ensures that $q \cdot \boldsymbol{p}_{zt}$ is small modulo N. We also generate a small $\boldsymbol{s} \in R$, and we let $\boldsymbol{z} \in R/qR$ such that:

$$[\boldsymbol{g}/\boldsymbol{z}^\kappa]_q \cdot q^{-1} \cdot \boldsymbol{h} \equiv \boldsymbol{s} \pmod N \tag{34}$$

For this we let $\boldsymbol{t} \in R$ such that $\boldsymbol{t} \equiv q \cdot \boldsymbol{s} \cdot \boldsymbol{h}^{-1} \pmod N$, and it suffices to solve:

$$[\boldsymbol{g}/\boldsymbol{z}^\kappa]_q = \boldsymbol{t}$$

Therefore we let over R/qR:

$$\boldsymbol{z} = (\boldsymbol{g}/\boldsymbol{t})^{1/\kappa}$$

Eventually combining (32), (33) and (34) we obtain:

$$\boldsymbol{\omega} \equiv [\boldsymbol{r} + \boldsymbol{e} \cdot \boldsymbol{g}^{-1}]_q \cdot \boldsymbol{s} - \boldsymbol{k} \cdot \boldsymbol{h} \pmod N$$

By construction both $\boldsymbol{s}$ and $\boldsymbol{h}$ are small; moreover $\boldsymbol{r}$ is small, and when $\boldsymbol{e} = 0$, from (31) we have that $\boldsymbol{k}$ is also small; therefore when $\boldsymbol{e} = 0$ we obtain that $\boldsymbol{\omega}$ is small; conversely when $\boldsymbol{e} \neq 0$ we have that $\boldsymbol{\omega}$ is not small with high probability. Hence we have a zero-testing procedure.