Practical Multilinear Maps over the Integers

Jean-Sébastien Coron[1], Tancrède Lepoint[2,3], and Mehdi Tibouchi[4]

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

Abstract. Extending bilinear elliptic curve pairings to multilinear maps is a long-standing open problem. The first plausible construction of such multilinear maps has recently been described by Garg, Gentry and Halevi, based on ideal lattices. In this paper we describe a different construction that works over the integers instead of ideal lattices, similar to the DGHV fully homomorphic encryption scheme. We also describe a different technique for proving the full randomization of encodings: instead of Gaussian linear sums, we apply the classical leftover hash lemma over a quotient lattice. We show that our construction is relatively practical: for reasonable security parameters a one-round 7-party Diffie-Hellman key exchange requires less than 40 seconds per party. Moreover, in contrast with previous work, multilinear analogues of useful, base group assumptions like DLIN appear to hold in our setting.

1 Introduction

Multilinear Maps. Extending bilinear elliptic curve pairings to multilinear maps is a long-standing open problem. In 2003 Boneh and Silverberg showed two interesting applications of multilinear maps [BS03], namely multipartite Diffie-Hellman and very efficient broadcast encryption; however they were pessimistic about the existence of such maps from the realm of algebraic geometry.

The first plausible construction of multilinear maps has recently been described by Garg, Gentry and Halevi, based on ideal lattices [GGH13]. The main difference with bilinear pairings is that the encoding $a \cdot g$ of an element a is randomized (with some noise) instead of deterministic; only the computed multilinear map $e(a_1 \cdot g, \ldots, a_\kappa \cdot g)$ is a deterministic function of the a_i's only. The construction has bounded degree with a maximum degree κ at most polynomial in the security parameter. Indeed, the encoding noise grows linearly with the degree, and when the noise reaches a certain threshold the encoding can become incorrect, as for ciphertexts in a somewhat homomorphic encryption scheme. The security of the construction relies on new hardness assumptions which are natural extensions of the Decisional Diffie-Hellman (DDH) assumption. To gain more confidence in their scheme the authors provide an extensive cryptanalytic

R. Canetti and J.A. Garay (Eds.): CRYPTO 2013, Part I, LNCS 8042, pp. 476–493, 2013.

survey. The authors focus on one application: the multipartite Diffie-Hellman key exchange.

The construction from [GGH13] 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 $\boldsymbol{u} = [\boldsymbol{c}/\boldsymbol{z}^\kappa]$ one can compute $[\boldsymbol{p}_{zt} \cdot \boldsymbol{u}]_q = [\boldsymbol{h}\boldsymbol{c}/\boldsymbol{g}]_q$. When $\boldsymbol{c}$ is an encoding of zero we have $\boldsymbol{c}/\boldsymbol{g} \in R$, which implies that $\boldsymbol{h}\boldsymbol{c}/\boldsymbol{g}$ is small in R, and therefore $[\boldsymbol{h}\boldsymbol{c}/\boldsymbol{g}]_q$ is small; this provides a way to test whether a level-κ encoding $\boldsymbol{c}$ is an encoding of 0. For the same reason the high-order bits of $[\boldsymbol{p}_{zt} \cdot \boldsymbol{u}]_q = [\boldsymbol{h}\boldsymbol{c}/\boldsymbol{g}]_q$ only depend on the coset $\boldsymbol{e}+\mathcal{I}$ and not on the particular $\boldsymbol{c} \in \boldsymbol{e}+\mathcal{I}$; this makes it possible to extract a representation of cosets encoded at level κ, and eventually defines a degree-κ multilinear map for level-1 encodings.

Our Contributions. Our main contribution is to describe a different construction that works over the integers instead of ideal lattices, similar to the DGHV fully homomorphic encryption scheme [DGHV10] and its batch variant [CCK+13]. Our construction offers the same flexibility as the original from [GGH13]; in particular it can be modified to support the analogue of asymmetric maps and composite-order maps. Moreover, it does not seem vulnerable to the "zeroizing" attack that breaks base group hardness assumptions like the analogues of DLIN and subgroup membership for the multilinear maps of [GGH13]. Since those assumptions are believed necessary to adapt constructions of primitives like adaptively secure functional encryption and NIZK, our construction seems even more promising for applications than [GGH13].

As in [GGH13], the security of our construction relies on new assumptions; it cannot be derived from "classical" assumptions such as the Approximate-GCD assumption used in [DGHV10]. We describe various possible attacks against our scheme; this enables us to derive parameters for which our scheme remains secure against these attacks.

Our new construction works as follows: one first generates n secret primes p_i and publishes $x_0 = \prod_{i=1}^n p_i$ (where n is large enough to ensure correctness and security); one also generates n small secret primes g_i, and a random secret integer z modulo x_0. A level-k encoding of a vector $\boldsymbol{m} = (m_i) \in \mathbb{Z}^n$ 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. It is clear that such encodings can be both added and multiplied

modulo x_0, as long as the numerators remain smaller than the p_i's. In particular the product of κ encodings c_j at level 1 gives an encoding at level κ where the corresponding vectors $\boldsymbol{m}_j$ are multiplied componentwise. For such level-κ encodings one defines a zero-testing parameter p_{zt} with:

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

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

$$\omega = \sum_{i=1}^{n} h_i \cdot \left(r_i + m_i \cdot (g_i^{-1} \bmod p_i)\right) \cdot \prod_{i' \neq i} p_{i'} \bmod x_0 \ .$$

Then 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 as in [GGH13] a degree-κ multilinear map.[1]

Our second contribution is to describe a different technique for proving the full randomization of encodings. As in [GGH13] the randomization of encodings is obtained by adding a random subset-sum of encodings of $\mathbf{0}$ from the public parameters. However as in [GGH13] the Leftover Hash Lemma (LHL) cannot be directly applied since the encodings live in some infinite ring instead of a finite group. The solution in [GGH13] consists in using linear sums with Gaussian coefficients; it is shown in [AGHS12] that the resulting sum has a Gaussian distribution (over some lattice). As noted by the authors, this can be seen as a "leftover hash lemma over lattices". In this paper we describe a different technique that does not use Gaussian coefficients; instead it consists in working modulo some lattice $L \subset \mathbb{Z}^n$ and applying the leftover hash lemma over the quotient $\mathbb{Z}^n/L$, which is still a finite group. Such technique was already used to prove the security of the batch variant of the DGHV fully homomorphic encryption scheme [CCK+13, CLT13a]. Here we provide a more formal description: we clearly state our "LHL over lattices" so that it can later be applied as a black-box (as the corresponding Theorem 3 in [AGHS12]). Our quotient lattice technique can independently be applied to the original encoding scheme from [GGH13], while the Gaussian sum technique from [AGHS12] is also applicable to ours.

Our third contribution is to describe the first implementation of multilinear maps. It appears that the basic versions of both [GGH13] and our scheme are rather unpractical, because of the huge public parameter size required to randomize the encodings. Therefore we use a simple optimization that consists in storing only a small subset of the public elements and combining them pairwise to generate the full public-key. Such optimization was originally described in [GH11] for

[1] Technically for p_{zt} we use a *vector* of integers instead of a single integer (see Sec. 3).

reducing the size of the encryption of the secret-key bits in the implementation of Gentry's FHE scheme [Gen09]. It was also used in [CMNT11] to reduce the public-key size of the DGHV scheme; however, as opposed to the latter work our randomization of encodings is heuristic only, whereas in [CMNT11] the semantic security was still guaranteed. Thanks to this optimization our construction becomes relatively practical: for reasonable security parameters a multipartite Diffie-Hellman computation with 7 users requires less than 40 seconds, with a public parameter size of roughly 2.6 GBytes; a proof-of-concept implementation is available at [Lep].

2 Definition of Randomized Encodings and Multilinear Maps

In this section we recall the setting introduced in [GGH13] for the notion of randomized encodings and multilinear maps, which they call *graded encoding schemes*. There are essentially two main differences with classical bilinear pairings (and their generalization to cryptographic multilinear maps as considered in [BS03]):

1. In bilinear pairings (and more generally cryptographic multilinear maps) we have a map $e\colon G^\kappa \to G_T$ that is linear with respect to all its κ inputs:

$$e(a_1 \cdot g, \ldots, a_\kappa \cdot g) = \left(\prod_{i=1}^{\kappa} a_i\right) \cdot e(g, \ldots, g)\ . \tag{2}$$

   One can view $a \cdot g$ as an "encoding" of the integer $a \in \mathbb{Z}_p$ over the group G of order p generated by g. The main difference in our setting is that encodings are now randomized. This means that an element $a \in R$ (where R is a ring that plays the role of the exponent space $\mathbb{Z}_p$) has many possible encodings; only the final multilinear map $e(a_1 \cdot g, \ldots, a_\kappa \cdot g)$ is a deterministic function of the a_i's only, and not on the randomness used to encode a_i into $a_i \cdot g$.
2. The second main difference is that to every encoding is now associated a *level*. At level 0 we have the "plaintext" ring elements $a \in R$, at level 1 we have the encoding $a \cdot g$, and by combining k such encodings $a_i \cdot g$ at level 1 one obtains a level-k encoding where the underlying elements a_i are homomorphically multiplied in R. The difference with "classical" cryptographic multilinear maps is that we can now multiply any (bounded) subset of encodings, instead of strictly κ at a time as with (2). For encodings at level κ we have a special zero-testing parameter $\boldsymbol{p}_{zt}$ that can extract a deterministic function of the underlying ring elements. This enables to define a degree-κ multilinear map for encodings at level 1.

2.1 Graded Encoding System

We recall the formal definition of a κ-Graded Encoding System from [GGH13]. For simplicity we only consider the symmetric case throughout the paper; we

refer to [GGH13] for a more general framework that can handle the asymmetric case.

Definition 1 (κ-Graded Encoding System [GGH13]). *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.*

2.2 Multilinear Map Procedures

We also recall the definition of the procedures for manipulating encodings. As previously we consider only the symmetric case; we refer to [GGH13] for the general case.

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

Ring Sampler. The randomized $\mathsf{samp}(\mathsf{params})$ outputs a "level-zero encoding" $a \in S_0^{(\alpha)}$ for a nearly uniform element $\alpha \in_R R$. Note that the encoding a does not need to be uniform in $S_0^{(\alpha)}$.

Encoding. The (possibly randomized) $\mathsf{enc}(\mathsf{params}, a)$ takes as input a level-zero encoding $a \in S_0^{(\alpha)}$ for some $\alpha \in R$, and outputs the level-one encoding $u \in S_1^{(\alpha)}$ for the same α.

Re-randomization. The randomized $\mathsf{reRand}(\mathsf{params}, i, u)$ re-randomizes encodings relative to the same level i. Specifically, given an encoding $u \in S_v^{(\alpha)}$, it outputs another encoding $u' \in S_v^{(\alpha)}$. Moreover for any two $u_1,\ u_2 \in S_v^{(\alpha)}$, the output distributions of $\mathsf{reRand}(\mathsf{params}, i, u_1)$ and $\mathsf{reRand}(\mathsf{params}, i, u_2)$ are nearly the same.

Addition and Negation. Given params and two encodings relative to the same level, $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, we have $\mathsf{add}(\mathsf{params}, u_1, u_2) \in S_i^{(\alpha_1+\alpha_2)}$ and $\mathsf{neg}(\mathsf{params}, u_1) \in S_i^{(-\alpha_1)}$. Below we write $u_1 + u_2$ and $-u_1$ as a shorthand for applying these procedures.

Multiplication. For $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_j^{(\alpha_2)}$, we have $\mathsf{mul}(\mathsf{params}, u_1, u_2) = u_1 \times u_2 \in S_{i+j}^{(\alpha_1 \cdot \alpha_2)}$.

Zero-Test. The procedure $\mathsf{isZero}(\mathsf{params}, \boldsymbol{p}_{zt}, u)$ outputs 1 if $u \in S_\kappa^{(0)}$ and 0 otherwise.

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

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

This concludes the definition of the procedures. In [GGH13] the authors consider a slightly relaxed definition of isZero and ext, where isZero can still output 1 even for some non-zero encoding u with negligible probability, and ext can extract different outputs when applied to encodings of the same elements, also with negligible probability; see [GGH13] for the corresponding definitions.

2.3 Hardness Assumptions

Finally we recall the hardness assumptions for multilinear maps from [GGH13]; as previously we consider only the symmetric case and refer to [GGH13] for the general case. In this symmetric case given a set of $\kappa + 1$ level-one encodings of random elements, it should be unfeasible to distinguish a level-κ encoding of their product from random.

Graded DDH (GDDH). Let $\mathcal{GE}$ be a graded encoding scheme consisting of all the routines above. For an adversary $\mathcal{A}$ and parameters λ, κ we consider the following process:

1. $(\mathsf{params}, \boldsymbol{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa)$
2. Choose $a_j \leftarrow \mathsf{samp}(\mathsf{params})$ for all $1 \leqslant j \leqslant \kappa + 1$
3. Set $u_j \leftarrow \mathsf{reRand}(\mathsf{params}, 1, \mathsf{enc}(\mathsf{params}, 1, a_j))$ for all $1 \leqslant j \leqslant \kappa + 1$ // encodings at level 1
4. Choose $b \leftarrow \mathsf{samp}(\mathsf{params})$ // encoding at level 0
5. Set $\tilde{u} = a_{\kappa+1} \times \prod_{i=1}^{\kappa} u_i$ // encoding of the right product at level κ
6. Set $\hat{u} = b \times \prod_{i=1}^{\kappa} u_i$ // encoding of a random product at level κ

The GDDH distinguisher is given as input the $\kappa + 1$ level-one encodings u_j and either $\tilde{u}$ (encoding the right product) or $\hat{u}$ (encoding a random product), and must decide which is the case. The GDDH assumption states that the advantage of any efficient adversary is negligible in the security parameter.

3 Our New Encoding Scheme

System Parameters. The main parameters are the security parameter λ and the required multilinearity level $\kappa \leqslant \mathsf{poly}(\lambda)$. Based on λ and κ, we choose the vector dimension n, the bit-size η of the primes p_i, the bit-size α of the primes g_i,

the maximum bit-size ρ 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 the next section. For integers z, p we denote the reduction of z modulo p by $(z \bmod p)$ or $[z]_p$ with $-p/2 < [z]_p \leqslant p/2$.

In our scheme a level-k encoding of a vector $\boldsymbol{m} = (m_i) \in \mathbb{Z}^n$ is an integer c such that for all $1 \leqslant i \leqslant n$:

$$c \equiv \frac{r_i \cdot g_i + m_i}{z^k} \pmod{p_i} \tag{3}$$

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 η-bit prime integers, the g_i's are 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, where x_0 is made public. Since the p_i's must remain secret, the user cannot encode the vectors $\boldsymbol{m} \in \mathbb{Z}^n$ by CRT directly from (3); instead one includes in the public parameters a set of ℓ level-0 encodings x'_j of random vectors $\boldsymbol{a}_j \in \mathbb{Z}^n$, and the user can generate a random level-0 encoding by computing a random subset-sum of those x'_j's.

Remark 1. From (3) each integer m_i is actually defined modulo g_i. Therefore our scheme encodes vectors $\boldsymbol{m}$ from the ring $R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$.

Instance Generation: $(\mathsf{params}, \boldsymbol{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa)$. We generate n secret random η-bit primes p_i and publish $x_0 = \prod_{i=1}^{n} p_i$. We generate a random invertible integer z modulo x_0. We generate n random α-bit prime integers g_i and a secret matrix $\boldsymbol{A} = (a_{ij}) \in \mathbb{Z}^{n \times \ell}$, where each component a_{ij} is randomly generated in $[0, g_i) \cap \mathbb{Z}$. We generate an integer y, three sets of integers $\{x_j\}_{j=1}^{\tau}$, $\{x'_j\}_{j=1}^{\ell}$ and $\{\Pi_j\}_{j=1}^{n}$, a zero-testing vector $\boldsymbol{p}_{zt}$, and a seed s for a strong randomness extractor, as described later. We publish the parameters $\mathsf{params} = \big(n, \eta, \alpha, \rho, \beta, \tau, \ell, y, \{x_j\}_{j=1}^{\tau}, \{x'_j\}_{j=1}^{\ell}, \{\Pi_j\}_{j=1}^{n}, s\big)$ and $\boldsymbol{p}_{zt}$.

Sampling Level-Zero Encodings: $c \leftarrow \mathsf{samp}(\mathsf{params})$. 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 the secret matrix $\boldsymbol{A} = (a_{ij}) \in \mathbb{Z}^{n \times \ell}$. More precisely, using the CRT modulo x_0 we generate integers x'_j such that:

$$1 \leqslant j \leqslant \ell, \qquad x'_j \equiv r'_{ij} \cdot g_i + a_{ij} \pmod{p_i} \tag{4}$$

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

Our randomized sampling algorithm $\mathsf{samp}(\mathsf{params})$ works as follows: we generate a random binary vector $\boldsymbol{b} = (b_j) \in \{0,1\}^\ell$ and output the level-0 encoding

$$c = \sum_{j=1}^{\ell} b_j \cdot x'_j \bmod x_0 \ .$$

From Equation (4), this gives $c \equiv \left(\sum_{j=1}^{\ell} r'_{ij} b_j\right) \cdot g_i + \sum_{j=1}^{\ell} a_{ij} b_j \pmod{p_i}$; as required the output c is a level-0 encoding:

$$c \equiv r_i \cdot g_i + m_i \pmod{p_i} \tag{5}$$

of some vector $\boldsymbol{m} = \boldsymbol{A} \cdot \boldsymbol{b} \in \mathbb{Z}^n$ 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.

The following Lemma states that, as required, the distribution of $\boldsymbol{m}$ can be made statistically close to uniform over $R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$; the proof is based on applying the LHL over the set R (see the full version of this paper [CLT13b]).

Lemma 1. *Let* $c \leftarrow \mathsf{samp}(\mathsf{params})$ *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* $(\mathsf{params}, \boldsymbol{m})$ *is statistically close to the distribution of* $(\mathsf{params}, \boldsymbol{m}')$ *where* $\boldsymbol{m}' \leftarrow R$.

Encodings at Higher Levels: $c_k \leftarrow \mathsf{enc}(\mathsf{params}, k, c)$. 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 integers $r_i \in (-2^{\rho}, 2^{\rho}) \cap \mathbb{Z}$; as previously the integer y is computed by CRT modulo x_0.

Given a level-0 encoding c of $\boldsymbol{m} \in \mathbb{Z}^n$ as given by (5), we can then compute a level-1 encoding of the same $\boldsymbol{m}$ by computing $c_1 = c \cdot y \bmod x_0$. Namely we obtain as required:

$$c_1 \equiv \frac{r_i^{(1)} \cdot g_i + m_i}{z} \pmod{p_i} \tag{6}$$

for some integers $r_i^{(1)}$, and we get $|r_i^{(1)} \cdot g_i + m_i| \leqslant \ell \cdot 2^{2(\rho+\alpha)}$ for all i. More generally to generate a level-k encoding we compute $c_k = c_0 \cdot y^k \bmod x_0$.

In multipartite Diffie-Hellman key-exchange every user keeps a private level-0 encoding c and publishes a level-1 encoding of the same underlying (unknown) $\boldsymbol{m}$; however one cannot publish $c_1 = c \cdot y \bmod x_0$ directly since the private level-0 encoding c could be recovered immediately from $c = c_1/y \bmod x_0$. Instead the level-1 encoding c_1 must first be re-randomized into a new level-1 encoding c'_1 whose distribution does not depend on the original c as long as it encodes the same $\boldsymbol{m}$.

Re-randomization: $c' \leftarrow \mathsf{reRand}(\mathsf{params}, i, c)$. To allow re-randomization of encodings at level one, we publish as part of our instance-generation a set of n integers Π_j which are all level-1 random encodings of zero:

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

The matrix $\boldsymbol{\Pi} = (\varpi_{ij}) \in \mathbb{Z}^{n\times n}$ 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 $(n2^\rho, n2^\rho + 2^\rho) \cap \mathbb{Z}$.

We also publish as part of our instance-generation a set of τ integers x_j, where 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 $(r_{ij}) \in \mathbb{Z}^{n\times\tau}$ are randomly and independently generated in the half-open parallelepiped spanned by the columns of the previous matrix $\boldsymbol{\Pi}$; see the full version of this paper [CLT13b] for a concrete algorithm.

Given as input a level-1 encoding c_1 as given by (6), we randomize c_1 with a random subset-sum of the x_j's and a linear combination of the Π_j's:

$$c_1' = c_1 + \sum_{j=1}^{\tau} b_j \cdot x_j + \sum_{j=1}^{n} b_j' \cdot \Pi_j \bmod x_0 \tag{7}$$

where $b_j \leftarrow \{0,1\}$, and $b_j' \leftarrow [0, 2^\mu) \cap \mathbb{Z}$. 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 follows essentially from our "leftover hash lemma over lattices"; see Section 4.

Lemma 2. *Let the encodings* $c \leftarrow \mathsf{samp}(\mathsf{params})$, $c_1 \leftarrow \mathsf{enc}(\mathsf{params}, 1, c)$, *and* $c_1' \leftarrow \mathsf{reRand}(\mathsf{params}, 1, c_1)$. *Write* $c_1' \equiv (r_i \cdot g_i + m_i)/z \pmod{p_i}$ *for all* $1 \leqslant i \leqslant n$, *and* $\boldsymbol{r} = (r_1, \ldots, r_n)^T$. *Let* $\tau \geqslant n \cdot (\rho + \log_2(2n)) + 2\lambda$ *and* $\mu \geqslant \rho + \alpha + \lambda$. *The distribution of* $(\mathsf{params}, \boldsymbol{r})$ *is statistically close to that of* $(\mathsf{params}, \boldsymbol{r'})$, *where* $\boldsymbol{r'} \in \mathbb{Z}^n$ *is randomly generated in the half-open parallelepiped spanned by the column vectors of* $2^\mu \boldsymbol{\Pi}$.

Writing $c_1' \equiv (r_i' \cdot g_i + m_i)/z \pmod{p_i}$, and using $|r_{ij} \cdot g_i| \leqslant 2n2^{\rho+\alpha}$ for all i, j, we obtain $|r_i' \cdot g_i + m_i| \leqslant \ell 2^{2(\rho+\alpha)} + \tau \cdot 2n2^{\rho+\alpha} + n \cdot 2n2^{\mu+\rho+\alpha}$. Using $\mu \geqslant \rho+\alpha+\lambda$ this gives $|r_i' \cdot g_i + m_i| \leqslant 4n^2 2^{\mu+\rho+\alpha}$.

Adding and Multiplying Encodings. It is clear that one can homomorphically add encodings. Moreover the product of κ level-1 encodings u_i can be interpreted as an encoding of the product. Namely, given level-one encodings u_j of vectors $\boldsymbol{m}_j \in \mathbb{Z}^n$ for $1 \leqslant j \leqslant \kappa$, with $u_j \equiv (r_{ij} \cdot g_i + m_{ij})/z \pmod{p_i}$, we simply let:

$$u = \prod_{j=1}^{\kappa} u_j \bmod x_0 \ .$$

This gives:

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

for some $r_i \in \mathbb{Z}$. This is a level-κ encoding of the vector $\boldsymbol{m}$ obtained by componentwise product of the vectors $\boldsymbol{m}_j$, as long as $\prod_{j=1}^{\kappa}(r_{ij} \cdot g_i + m_{ij}) < p_i$ for all i. When computing the product of κ level-1 encodings from reRand and one level-0 encoding from samp (as in multipartite Diffie-Hellman key exchange), we obtain from previous bounds $|r_i| \leqslant (4n^2 2^{\mu+\rho+\alpha})^{\kappa} \cdot \ell \cdot 2^{\rho+1}$ for all i.

Zero Testing. $\mathsf{isZero}(\mathsf{params}, \boldsymbol{p}_{zt}, u_\kappa) \stackrel{?}{=} 0/1$. We can test equality between encodings by subtracting them and testing for zero. To enable zero testing we randomly generate 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 the full version of this paper [CLT13b]. We then publish as part of our instance generation the following zero-testing vector $\boldsymbol{p}_{zt} \in \mathbb{Z}^n$:

$$(\boldsymbol{p}_{zt})_j = \sum_{i=1}^{n} h_{ij} \cdot \left(z^\kappa \cdot g_i^{-1} \bmod p_i\right) \cdot \prod_{i' \neq i} p_{i'} \bmod x_0 \,. \tag{8}$$

To determine whether a level-κ encoding c is an encoding of zero or not, we compute the vector $\boldsymbol{\omega} = c \cdot \boldsymbol{p}_{zt} \bmod x_0$ and test whether $\|\boldsymbol{\omega}\|_\infty$ is small:

$$\mathsf{isZero}(\mathsf{params}, \boldsymbol{p}_{zt}, c) = \begin{cases} 1 & \text{if } \|c \cdot \boldsymbol{p}_{zt} \bmod x_0\|_\infty < x_0 \cdot 2^{-\nu} \\ 0 & \text{otherwise} \end{cases}$$

for some parameter ν specified later.

Namely for a level-κ ciphertext c we have $c \equiv (r_i \cdot g_i + m_i)/z^\kappa \pmod{p_i}$ for some $r_i \in \mathbb{Z}$; therefore for all $1 \leqslant i \leqslant n$ we can write:

$$c = q_i \cdot p_i + (r_i \cdot g_i + m_i) \cdot \left(z^{-\kappa} \bmod p_i\right) \tag{9}$$

for some $q_i \in \mathbb{Z}$. Therefore combining (8) and (9), we get:

$$(\boldsymbol{\omega})_j = (c \cdot \boldsymbol{p}_{zt} \bmod x_0)_j = \sum_{i=1}^{n} h_{ij} \cdot \left(r_i + m_i \cdot (g_i^{-1} \bmod p_i)\right) \cdot \prod_{i' \neq i} p_{i'} \bmod x_0 \,. \tag{10}$$

Therefore if $m_i = 0$ for all $1 \leqslant i \leqslant n$, then $\|\boldsymbol{\omega}\|_\infty = \|c \cdot \boldsymbol{p}_{zt} \bmod x_0\|_\infty$ is small compared to x_0 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. More precisely we prove the following lemma in the full version of this paper [CLT13b].

Lemma 3. *Let n, η, α and β be as in our parameter setting. Let ρ_f be such that $\rho_f + \lambda + \alpha + 2\beta \leqslant \eta - 8$, and let $\nu = \eta - \beta - \rho_f - \lambda - 3 \geqslant \alpha + \beta + 5$. 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 $\|\boldsymbol{\omega}\|_\infty < x_0 \cdot 2^{-\nu-\lambda-2}$. Conversely if $\boldsymbol{m} \neq 0$ then $\|\boldsymbol{\omega}\|_\infty \geqslant x_0 \cdot 2^{-\nu+2}$.*

Extraction. $sk \leftarrow \mathsf{ext}(\mathsf{params}, \boldsymbol{p}_{zt}, u_\kappa)$. This part is essentially the same as in [GGH13]. 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 x_0, 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 params):

$$\mathsf{ext}(\mathsf{params}, \boldsymbol{p}_{zt}, c) = \mathsf{Extract}_s\big(\mathsf{msbs}_\nu(c \cdot \boldsymbol{p}_{zt} \bmod x_0)\big)$$

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 $\|(c - c') \cdot \boldsymbol{p}_{zt} \bmod x_0\|_\infty < x_0 \cdot 2^{-\nu-\lambda-2}$, and therefore we expect that $\boldsymbol{\omega} = c \cdot \boldsymbol{p}_{zt} \bmod x_0$ and $\boldsymbol{\omega'} = c' \cdot \boldsymbol{p}_{zt} \bmod x_0$ agree on their ν most significant bits, and therefore extract to the same value.[2]

Conversely if c and c' encode different vectors then by Lemma 3 we must have $\|(c - c') \cdot \boldsymbol{p}_{zt} \bmod x_0\|_\infty > x_0 \cdot 2^{-\nu+2}$, 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 $\mathsf{msbs}_\nu(c \cdot \boldsymbol{p}_{zt} \bmod x_0)$ 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 $\lfloor \log_2 |R| \rfloor - \lambda$.

3.1 Setting the Parameters

The system parameters must satisfy the following constraints:

- The bit-size ρ of the randomness used for encodings must satisfy $\rho = \Omega(\lambda)$ to avoid brute force attack on the noise, including the improved attack from [CN12] with complexity $\tilde{\mathcal{O}}(2^{\rho/2})$.
- 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; this is required to prove the security of the multipartite Diffie-Hellman Key Exchange (see the full version of this paper [CLT13b]). 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 5.1.
- 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 $\tau \geqslant n \cdot (\rho + \log_2(2n)) + 2\lambda$ in order to apply our "leftover hash lemma over lattices". For the same reason the bit-size μ of the linear sum coefficients must satisfy $\mu \geqslant \alpha + \rho + \lambda$; see Lemma 2.

[2] Two coefficients ω and ω' from $\boldsymbol{\omega}$ and $\boldsymbol{\omega'}$ could still be on the opposite sides of a boundary, with $\lfloor \omega / 2^k \rfloor = v$ and $\lfloor \omega' / 2^k \rfloor = v + 1$, so that ω and ω' would extract to different MSBs v and $v+1$. Heuristically this happens with probability $\mathcal{O}(2^{-\lambda})$. The argument can be made rigorous by generating a public random integer W modulo x_0 in the parameters, and extracting the MSBs of $\omega + W \bmod x_0$ instead of $\omega \bmod x_0$ for all coefficients ω of the vector $\boldsymbol{\omega}$.

- The bitsize β of the matrix $\boldsymbol{H}$ entries must satisfy $\beta = \Omega(\lambda)$ in order to avoid the GCD attack from Section 5.2. One can take $\beta = \lambda$.
- The bit-size η of the primes p_i must satisfy $\eta \geqslant \rho_f + \alpha + 2\beta + \lambda + 8$, 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 $N = \kappa + 1$ users), one obtains $\rho_f = \kappa \cdot (\mu + \rho + \alpha + 2\log_2 n + 2) + \rho + \log_2 \ell + 1$ (see previous Section).
- The number ν of most significant bits to extract can then be set to $\nu = \eta - \beta - \rho_f - \lambda - 3$ (see Lemma 3).

3.2 Security of Our Construction

As in [GGH13] the security of our construction relies on new assumptions that do not seem to be reducible to more classical assumptions. Namely, as in [GGH13] one can make the assumption that solving the Graded DDH problem (GDDH) recalled in Section 2.3 is hard in our scheme. This enables to prove the security of the one-round N-way Diffie-Hellman key exchange protocol [GGH13]. Ideally one would like to reduce such assumption to a more classical assumption, such as the Approximate-GCD assumption, but that does not seem feasible. Therefore to gain more confidence in our scheme we describe various attacks in Section 5.

4 Another Leftover Hash Lemma over Lattices

As mentioned in the introduction, to prove the full randomization of encodings (Lemma 2) one cannot apply the classical Leftover Hash Lemma (LHL) because the noise in the encodings live in some infinite ring instead of a finite group. In [GGH13] the issue was solved by using linear sums with Gaussian coefficients. Namely the analysis in [AGHS12] shows that the resulting sum has a Gaussian distribution (over some lattice). As noted by the authors this technique can be seen as a "leftover hash lemma over lattices". Such a technique would be applicable to our scheme as well.

In this section we describe an alternative technique (without Gaussian coefficients) that can also be seen as a "leftover hash lemma over lattices". It consists in working modulo a lattice $L \subset \mathbb{Z}^n$ and applying the classical leftover hash lemma over the finite group $\mathbb{Z}^n/L$. This technique was already used in [CCK+13, CLT13a] to prove the security of a batch extension of the DGHV scheme. In this paper we provide a more formal description; namely we clearly state our "LHL over lattices" so that it can later be applied as a black-box (as the corresponding Theorem 3 in [AGHS12]). Namely our quotient lattice technique can independently be applied to the original encoding scheme from [GGH13].

4.1 Classical Leftover Hash Lemma

We first recall the classical Leftover Hash Lemma. We say that the distributions D_1, D_2 over a finite domain X are ε-statistically close if the statistical distance

$\Delta(D_1, D_2) = \frac{1}{2}\sum_{x\in X} |D_1(x) - D_2(x)|$ is smaller than ε. A distribution D is ε-uniform if its statistical distance from the uniform distribution is at most ε. A family $\mathcal{H}$ of hash functions from X to Y, both finite sets, is said to be pairwise-independent if for all distinct $x, x' \in X$, $\Pr_{h\leftarrow H}[h(x) = h(x')] = 1/|Y|$.

Lemma 4 (Leftover Hash Lemma [HILL99]). *Let $\mathcal{H}$ be a family of pairwise hash functions from X to Y. Suppose that $h \leftarrow \mathcal{H}$ and $x \leftarrow X$ are chosen uniformly and independently. Then, $(h, h(x))$ is $\frac{1}{2}\sqrt{|Y|/|X|}$-uniform over $\mathcal{H}\times Y$.*

One can then deduce the following Lemma for random subset sums over a finite abelian group.

Lemma 5. *Let G be a finite abelian group. Set $x_1, \ldots, x_m \leftarrow G$ uniformly and independently, set $s_1, \ldots, s_m \leftarrow \{0, 1\}$, and set $y = \sum_{i=1}^m s_i x_i \in G$. Then $(x_1, \ldots, x_m, y)$ is $1/2\sqrt{|G|/2^m}$-uniform over G^{m+1}.*

Proof. We consider the following hash function family $\mathcal{H}$ from $\{0, 1\}^m$ to G. Each member $h \in \mathcal{H}$ is parameterized by the elements $(x_1, \ldots, x_m) \in G^m$. Given $\boldsymbol{s} \in \{0, 1\}^m$, we define $h(\boldsymbol{s}) = \sum_{i=1}^m s_i \cdot x_i \in G$. The hash function family is clearly pairwise independent. Therefore by Lemma 4, $(h, h(x))$ is $\frac{1}{2}\sqrt{|G|/2^m}$-uniform over G^{m+1}. □

4.2 Leftover Hash Lemma over Lattices

Let $L \subset \mathbb{Z}^n$ be a lattice of rank n of basis $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$. Then every $\boldsymbol{x} \subset \mathbb{Z}^n$ can be uniquely written as:

$$\boldsymbol{x} = \xi_1\boldsymbol{b}_1 + \ldots + \xi_n\boldsymbol{b}_n$$

for some real numbers ξ_i. Moreover, for every vector $\boldsymbol{x} \in \mathbb{Z}^n$ there is a unique $\boldsymbol{a} \in L$ such that:

$$\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{a} = \xi'_1\boldsymbol{b}_1 + \ldots + \xi'_n\boldsymbol{b}_n$$

where $0 \leqslant \xi'_i < 1$; we write $\boldsymbol{y} = \boldsymbol{x} \bmod \boldsymbol{B}$. Therefore each vector of $\mathbb{Z}^n/L$ has a unique representative in the half-open parallelepiped defined by the previous equation.

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 $\boldsymbol{B}$. Given a basis $\boldsymbol{B} = (\boldsymbol{b_1}, \ldots, \boldsymbol{b}_n)$ and $\mu \in \mathbb{Z}^*$ we denote by $\mu\boldsymbol{B}$ the basis $(\mu\boldsymbol{b}_1, \ldots, \mu\boldsymbol{b}_n)$. We are now ready to state our "Leftover Hash Lemma over Lattices".

Lemma 6. *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 $\varepsilon = mn \cdot 2^{-\mu} + 1/2\sqrt{|\det L|/2^m}$.*

Proof. We consider the intermediate variable:

$$\boldsymbol{y''} = \left(\sum_{i=1}^{m} s_i \boldsymbol{x}_i \bmod \boldsymbol{B} \right) + \sum_{i=1}^{n} t_i \boldsymbol{b_i} \ . \tag{11}$$

Firstly by applying the leftover hash lemma over the finite abelian group $G = \mathbb{Z}^n/L$, we obtain that the distributions $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m, \sum_{i=1}^{m} s_i \boldsymbol{x}_i \bmod \boldsymbol{B})$ and $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m, \boldsymbol{\psi})$ are ε_1-statistically close, where $\boldsymbol{\psi} \leftarrow \mathcal{D}_{\boldsymbol{B}}$ and

$$\varepsilon_1 = 1/2\sqrt{|G|/2^m} = 1/2\sqrt{|\det(L)|/2^m} \ .$$

This implies that the distributions $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m, \boldsymbol{y''})$ and $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m, \boldsymbol{y'})$ are also ε_1-statistically close.

Secondly we write:

$$\sum_{i=1}^{m} s_i \boldsymbol{x}_i \bmod \boldsymbol{B} = \sum_{i=1}^{m} s_i \boldsymbol{x}_i - \sum_{j=1}^{n} \chi_j \boldsymbol{b}_j \tag{12}$$

where $\chi_j \in \mathbb{Z}$ for all j. We also write $\boldsymbol{x_i} = \sum_j \xi_{ij} \boldsymbol{b}_j$ where by definition $0 \leqslant \xi_{ij} < 1$ for all i, j. This gives:

$$\sum_{i=1}^{m} s_i \boldsymbol{x}_i \bmod \boldsymbol{B} = \sum_{i=1}^{m} s_i \sum_{j=1}^{n} \xi_{ij} \boldsymbol{b}_j - \sum_{j=1}^{n} \chi_j \boldsymbol{b}_j = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} s_i \xi_{ij} - \chi_j \right) \boldsymbol{b}_j \ ,$$

which implies $0 \leqslant \sum_{i=1}^{m} s_i \xi_{ij} - \chi_j < 1$ for all j, and therefore $0 \leqslant \chi_j \leqslant m$ for all j. Combining Equations (11) and (12) we have:

$$\boldsymbol{y''} = \sum_{i=1}^{m} s_i \boldsymbol{x}_i + \sum_{i=1}^{n} (t_i - \chi_i) \boldsymbol{b_i} \ ,$$

where as shown above $0 \leqslant \chi_i \leqslant m$ for all i. This implies that the distributions $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m, \boldsymbol{y})$ and $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m, \boldsymbol{y''})$ are ε_2-statistically close, with $\varepsilon_2 = mn2^{-\mu}$. Therefore the distributions $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m, \boldsymbol{y})$ and $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m, \boldsymbol{y'})$ are $(\varepsilon_1 + \varepsilon_2)$-statistically close, which proves the Lemma. □

We also show that the previous distribution $\mathcal{D}_{2^\mu \boldsymbol{B}}$ is not significantly modified when a small vector $\boldsymbol{z} \in \mathbb{Z}^n$ is added and the operator norm of the corresponding matrix B^{-1} is upper-bounded; see the full version of this paper [CLT13b] for the proof.

Lemma 7. *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 $\varepsilon = 2^{-\mu} \cdot (\|\boldsymbol{z}\|_\infty \cdot \|B^{-1}\|_\infty + 1)$.*

4.3 Re-randomization of Encodings: Proof of Lemma 2

We are now ready to apply our "LHL over lattices" to prove the full randomization of encodings as stated in Lemma 2. Namely the re-randomization equation (7) can be rewritten in vector form as:

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

where $\boldsymbol{b} \leftarrow \{0,1\}^\tau$ and $\boldsymbol{b'} \leftarrow ([0, 2^\mu) \cap \mathbb{Z})^n$, and the columns of the matrix $\boldsymbol{X} \in \mathbb{Z}^{n \times \tau}$ are uniformly and independently generated in the parallelepiped spanned by the columns of the matrix $\boldsymbol{\Pi} \in \mathbb{Z}^{n \times n}$. To conclude, it therefore suffices to apply Lemma 6 and Lemma 7, using additionally an upper bound on $\|\boldsymbol{\Pi}^{-1}\|_\infty$. For this we use the fact that $\boldsymbol{\Pi}$ has been generated as a diagonally dominant matrix. We refer to the full version of this paper [CLT13b] for the full proof of Lemma 2.

5 Attacks against Our Multilinear Scheme

5.1 Lattice Attack on the Encodings

We first describe a lattice attack against level-0 encodings. We consider an element $x_0 = \prod_{i=1}^n p_i$ and a set of τ integers $x_j \in \mathbb{Z}_{x_0}$ such that:

$$x_j \bmod p_i = r_{ij}$$

where $r_{ij} \in (-2^\rho, 2^\rho) \cap \mathbb{Z}$. We want to estimate the complexity of the classical orthogonal lattice attack for recovering (some of) the noise values r_{ij}.

This attack works by considering the integer vector formed by a subset of the x_j's, say $\boldsymbol{x} = (x_j)_{1 \leqslant j \leqslant t}$ for some $n < t \leqslant \tau$, and relating the lattice of vectors orthogonal to $\boldsymbol{x} \bmod x_0$ to the lattice of vectors orthogonal to each of the corresponding noise value vectors $\boldsymbol{r}_i = (r_{ij})_{1 \leqslant j \leqslant t}$.

More precisely, let $L \subset \mathbb{Z}^t$ the lattice of vectors $\boldsymbol{u}$ such that:

$$\boldsymbol{u} \cdot \boldsymbol{x} \equiv 0 \pmod{x_0}.$$

Clearly, L contains $x_0\mathbb{Z}^t$ so it is of full rank t. Moreover, we have

$$\mathrm{vol}(L) = [\mathbb{Z}^t : L] = x_0 / \gcd(x_0, x_1, \ldots, x_t) = x_0 \ .$$

As a result, we heuristically expect the successive minima of L to be around $\mathrm{vol}(L)^{1/t} \approx 2^{n \cdot \eta / t}$, and hence applying lattice reduction should yield a reduced basis $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_t)$ with vectors of length $\|\boldsymbol{u}_k\| \approx 2^{n \cdot \eta / t + \alpha t}$ for some constant $\alpha > 0$ depending on the lattice reduction algorithm ($2^{\alpha t}$ is the Hermite factor).

Now, a vector $\boldsymbol{u} \in L$ satisfies $\boldsymbol{u} \cdot \boldsymbol{x} \equiv 0 \pmod{x_0}$, so for each $i \in \{1, \ldots, n\}$, reducing modulo p_i gives:

$$\boldsymbol{u} \cdot \boldsymbol{r}_i \equiv 0 \pmod{p_i}.$$

In particular, if $\boldsymbol{u}$ is short enough to satisfy $\|\boldsymbol{u}\| \cdot \|\boldsymbol{r}_i\| < p_i$, this implies $\boldsymbol{u} \cdot \boldsymbol{r}_i = 0$ in $\mathbb{Z}$. As a result, if we have:

$$2^{n \cdot \eta / t + \alpha t} \cdot 2^{\rho} < 2^{\eta}, \tag{13}$$

we expect the vectors $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{t-n})$ from the previous lattice reduction step to be orthogonal to the $\boldsymbol{r}_i$'s, and hence computing the rank n orthogonal lattice to the lattice spanned by those vectors should reveal the $\boldsymbol{r}_i$'s.

Since t must be greater than n for the attack to apply, condition (13) implies in particular that:

$$\alpha < \frac{\eta - \rho}{n}.$$

Since a Hermite factor of $2^{\alpha t}$ is achieved in time $2^{\Omega(1/\alpha)}$ (usually by carrying out BKZ reduction with block size $\beta = \omega(1/\alpha)$, in which each block is BKZ-reduced in time exponential in β, see e.g. [HPS11]), we obtain that this orthogonal lattice attack has a complexity exponential in n. In fact, with $\gamma = \eta \cdot n$, we get a time complexity of $2^{\Omega(\gamma/\eta^2)}$, similar to [DGHV10, §5.2] (see also [CH12]).

5.2 GCD Attack on the Zero-Testing Parameter

We consider the ratio modulo x_0 of two coefficients from the zero-testing vector $\boldsymbol{p}_{zt}$, namely $u := (\boldsymbol{p}_{zt})_1 / (\boldsymbol{p}_{zt})_2 \bmod x_0$. From (8) we obtain for all $1 \leqslant i \leqslant \ell$:

$$u \equiv h_{i1}/h_{i2} \pmod{p_i}$$

We can therefore recover p_i by computing $\gcd(h_{i2} \cdot u - h_{i1}, x_0)$ for all possible h_{i1}, h_{i2}. Since the h_{ij}'s are upper bounded by 2^{β} in absolute value, the attack has complexity $\mathcal{O}(2^{2\beta})$. By using a technique similar to [CN12], the attack complexity can be reduced to $\tilde{\mathcal{O}}(2^{\beta})$.

We describe more attacks in the full version of this paper [CLT13b].

6 Optimizations and Implementation

In this section we describe an implementation of our scheme in the one-round N-way Diffie-Hellman key exchange protocol; we recall the protocol in the full version of this paper [CLT13b], as described in [BS03, GGH13].

We note that without optimizations the size of the public parameters makes our scheme completely unpractical; this is also the case in [GGH13]. Namely, for sampling we need to store at least $n \cdot \alpha$ encodings (resp. $n \cdot \rho$ encodings for re-randomization), each of size $n \cdot \eta$ bits; the public-key size is then at least $n^2 \cdot \eta \cdot \alpha$ bits. With $n \simeq 10^4$, $\eta \simeq 10^3$ and $\alpha \simeq 80$, the public-key size would be at least 1 TB. Therefore we use three heuristic optimizations to reduce the memory requirement; we refer to the full version of this paper [CLT13b] for a detailed description.

1. Non-uniform sampling: for the sampling algorithm we use a small number of encodings ℓ only; this implies that the sampling cannot be proved uniform anymore.
2. Quadratic re-randomization: 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.
3. Integer p_{zt}: we use a single integer p_{zt} instead of a vector $\boldsymbol{p}_{zt}$ with n components. An encoding c of zero still gives a small integer $\omega = p_{zt} \cdot c \bmod x_0$, but the converse does not necessarily hold anymore.

We have implemented a one-round N-way Diffie-Hellman key exchange protocol with $N = 7$ users, in C++ using the Gnu MP library [Gt13] to perform operations on large integers. We refer to the full version of this paper [CLT13b] 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.

Table 1. Parameters and timings to instantiate a *one-round 7-way Diffie-Hellman key exchange protocol* with $\ell = 160$, $\beta = 80$, $\alpha = 80$, $N = 7$ (*i.e.* $\kappa = 6$) and $\nu = 160$ on a 16-core computer (Intel(R) Xeon(R) CPU E7-8837 at 2.67GHz) using GMP 5.1.1. Note that the Setup was parallelized on the 16 cores to speed-up the process while the other steps ran on a single core.

Instantiation	λ	n	η	Δ	ρ	pk size
Small	52	540	1838	23	41	24 MB
Medium	62	2085	2043	45	56	129 MB
Large	72	8250	2261	90	72	709 MB
Extra	80	26115	2438	161	85	2.6 GB

Instantiation	Setup (once)	Publish (per party)	KeyGen (per party)
Small	6 s	0.23 s	0.20 s
Medium	38 s	1.0 s	1.2 s
Large	1700 s	5.1 s	5.9 s
Extra	29000 s	18 s	20 s

The timings above show that our scheme is relatively practical, as the KeyGen phase of the multipartite Diffie-Hellman protocol requires only a few seconds per user; however the parameter size is still very large even with our optimizations.

Acknowledgments. We would like to thank the Crypto referees for their helpful comments.

References

[AGHS12] Agrawal, S., Gentry, C., Halevi, S., Sahai, A.: Discrete Gaussian Leftover Hash Lemma over infinite domains. Cryptology ePrint Archive, Report 2012/714 (2012), http://eprint.iacr.org/

[BS03] Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. Contemporary Mathematics 324, 71–90 (2003)

[CCK+13] Cheon, J.H., Coron, J.-S., Kim, J., Lee, M.S., Lepoint, T., Tibouchi, M., Yun, A.: Batch fully homomorphic encryption over the integers. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 315–335. Springer, Heidelberg (2013)

[CH12] Cohn, H., Heninger, N.: Approximate common divisors via lattices. In: ANTS X (2012)

[CLT13a] Coron, J.-S., Lepoint, T., Tibouchi, M.: Batch fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2013/036 (2013), http://eprint.iacr.org/

[CLT13b] Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. Cryptology ePrint Archive, Report 2013/183 (2013), http://eprint.iacr.org/

[CMNT11] Coron, J.-S., Mandal, A., Naccache, D., Tibouchi, M.: Fully homomorphic encryption over the integers with shorter public keys. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 487–504. Springer, Heidelberg (2011)

[CN12] Chen, Y., Nguyen, P.Q.: Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 502–519. Springer, Heidelberg (2012)

[DGHV10] van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)

[Gen09] Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009), http://crypto.stanford.edu/craig

[GGH13] Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013)

[GH11] Gentry, C., Halevi, S.: Implementing Gentry's fully-homomorphic encryption scheme. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 129–148. Springer, Heidelberg (2011)

[Gt13] Granlund, T., and the GMP development team: GNU MP: The GNU Multiple Precision Arithmetic Library, 5.1.1 edition (2013), http://gmplib.org/

[HILL99] Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing 28, 12–24 (1999)

[HPS11] Hanrot, G., Pujol, X., Stehlé, D.: Analyzing blockwise lattice algorithms using dynamical systems. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 447–464. Springer, Heidelberg (2011)

[Lep] Lepoint, T.: An Implementation of Multilinear Maps over the Integers. Available under the Creative Commons License BY-NC-SA at https://github.com/tlepoint/multimap