Security Analysis of Multilinear Maps over the Integers

Hyung Tae Lee[1] and Jae Hong Seo[2]

[1] Division of Mathematical Sciences,
School of Physical and Mathematical Sciences,
Nanyang Technological University, Singapore
hyungtaelee@ntu.edu.sg
[2] Myongji University, Korea
jaehongseo@mju.ac.kr

Abstract. At Crypto 2013, Coron, Lepoint, and Tibouchi (CLT) proposed a practical Graded Encoding Scheme (GES) over the integers, which has very similar cryptographic features to ideal multilinear maps. In fact, the scheme of Coron *et al.* is the second proposal of a secure GES, and has advantages over the first scheme of Garg, Gentry, and Halevi (GGH). For example, unlike the GGH construction, the subgroup decision assumption holds in the CLT construction. Immediately following the elegant innovations of the GES, numerous GES-based cryptographic applications were proposed. Although these applications rely on the security of the underlying GES, the security of the GES has not been analyzed in detail, aside from the original papers produced by Garg *et al.* and Coron *et al.*

We present an attack algorithm against the system parameters of the CLT GES. The proposed algorithm's complexity $\tilde{\mathcal{O}}(2^{\rho/2})$ is exponentially smaller than $\tilde{\mathcal{O}}(2^{\rho})$ of the previous best attack of Coron *et al.*, where ρ is a function of the security parameter. Furthermore, we identify a flaw in the generation of the zero-testing parameter of the CLT GES, which drastically reduces the running time of the proposed algorithm. The experimental results demonstrate the practicality of our attack.

1 Introduction

In 2003, Boneh and Silverberg [2] introduced the concept of cryptographic multilinear maps by generalizing cryptographic bilinear maps. They proposed interesting applications based on the concept, such as the multipartite Diffie-Hellman key exchange and an efficient broadcast encryption. Until recently, it was an important, yet hard-to-achieve open problem to construct multilinear maps satisfying cryptographic requirements. At Eurocrypt 2013, Garg, Gentry, and Halevi [18] proposed the first candidate multilinear maps, called *Graded Encoding Scheme (GES)*, having very similar cryptographic features to ideal multilinear maps. At Crypto 2013, Coron, Lepoint, and Tibouchi [10] proposed the second GES over the integers. The CLT construction has an advantage over the GGH construction; specifically, it allows one to use a desirable assumption

J.A. Garay and R. Gennaro (Eds.): CRYPTO 2014, Part I, LNCS 8616, pp. 224–240, 2014.

such as the subgroup decision assumption, which does not hold with the GGH construction. Thus, the CLT construction has broader applications. Very recently, Langlois, Stehlé, and Steinfeld [24] improved the GGH construction in terms of the bit size of the public parameters. Immediately following the elegant inventions of the GES, they received significant attention from the cryptography community, and numerous cryptography applications based on the GES inventions were built; for example, programmable hash [17], full-domain hash [22], functional encryption [19,20], witness encryption [21], and indistinguishability obfuscation [5,19,6]. Although these applications rely on the security of the underlying GES, the security of the GES itself has not been analyzed in detail, aside from the original papers produced by Garg *et al.* and Coron *et al.*

1.1 Our Contributions

n-Masked Partial Approximate Common Divisors (n-MPACD). We begin by introducing a new number theoretic problem, called *n-Masked Partial Approximate Common Divisors (n-MPACD)*, which is a generalization of the system parameters (such as the zero-testing parameter [10] and the re-randomization parameter [10,8]) from integer-based schemes such as multilinear maps [10] and Fully Homomorphic Encryptions (FHE) [8]. Roughly speaking, a problem instance is a product of η-bit primes $x_0 = \prod_i p_i$ and polynomially-many samples x_j such that $x_j \equiv Q \cdot r_{ij} \pmod{p_i}$ where $Q \xleftarrow{\$} \mathbb{Z}_{x_0}$, $r_{ij} \xleftarrow{\$} (-2^\rho, 2^\rho)$ and $\rho \ll \eta$. Because of the unknown Q, it is unlikely to directly apply the meet-in-the-middle attack of Chen and Nguyen [7]; therefore, it appears to be harder than the Partial Approximate Common Divisors (PACD) problem [23]. In fact, the attack algorithm of Coron, Lepoint, and Tibouchi (CLT) [10], which is the most efficient currently known algorithm for n-MPACD, has $\tilde{\mathcal{O}}(2^\rho)$ complexity, although it employs the technique used in the Chen-Nguyen attack.

Exponentially Faster Attack for n-MPACD. We present an attack algorithm for n-MPACD, which is exponentially faster than the CLT attack. The proposed algorithm follows the basic flow of the strategy of the Chen-Nguyen attack [7]. However, several tricks are required to manage the unknown Q and several moduli. Our attack is based on the following observation for subset-sums of integers in the same interval $(-2^\rho, 2^\rho)$: given $2m$ integers, there are 2^{2m}

Table 1. Algorithms for n-MPACD

Algorithm	Error Type	Computation ($\mathbb{Z}_{x_0}$ op.)	Space
(Corrected) CLT [10]	arbitrary errors†	$\mathcal{O}(\rho^2 2^\rho)$	$\mathcal{O}(\rho^2 2^\rho)$
This paper	arbitrary errors†	$\mathcal{O}(\sqrt{\rho \log \rho} \cdot \rho^2 2^{\rho/2})$	$\mathcal{O}(\sqrt{\rho \log \rho} \cdot \rho^2 2^{\rho/2})$
	uniform errors	$\mathcal{O}(\sqrt{\frac{\rho \log \rho}{n}} \cdot \rho^2 2^{\rho/2})$	$\mathcal{O}(\sqrt{\frac{\rho \log \rho}{n}} \cdot \rho^2 2^{\rho/2})$

An instance of n-MPACD consists of x_0 (product of n primes) and polynomially many samples with errors chosen from $(-2^\rho, 2^\rho)$.

†: Mild assumptions are necessary, which are specified in the paper.

different subset-sums (ignoring duplications), but such subset-sums range from $(-2m2^{\rho}, 2m2^{\rho})$. That is, the number of subset-sums increases exponentially in m; however, those ranges increase only polynomially in m. Therefore, by slightly increasing m, we can find a collision among subset-sums. This observation is essential to our exponentially faster algorithm, as compared to the CLT attack. We summarize the comparison in Table 1.

A Flaw in the Generation of the Zero-Testing Parameter. We apply the proposed attack algorithm to the system parameters of multilinear maps over the integers; in particular, the zero-testing parameter [10]. The complexity of both our attack algorithm and the CLT attack primarily depend on ρ, the size of errors r_{ij}; therefore, it is necessary to enlarge the size of errors. In the generation of the zero-testing parameter, the matrix $\mathbf{H} = (h_{ij}) \in \mathbb{Z}^{n \times n}$ plays the role of (r_{ij}) in n-MPACD, indicating that the size of h_{ij} is very important for the security of the CLT GES. For the functionality of the multilinear maps, the matrix $\mathbf{H}$ is defined to be unimodular, and to satisfy two bounds $\|\mathbf{H}^{\top}\|_{\infty} \leq 2^{\beta}$ and $\|(\mathbf{H}^{-\mathbf{1}})^{\top}\|_{\infty} \leq 2^{\beta}$. In [10], the authors provided a method for generating $\mathbf{H}$. However, we point out that the given method does not provide sufficient randomness in $\mathbf{H}$; that is, the average size of each entry h_{ij} in $\mathbf{H}$ is much less than expected. Eventually, this will weaken the security of multilinear maps over the integers.

Experimental Results. We provide several experimental results for our algorithm. In particular, we apply our attack algorithm to the implementation parameters on Small size for 52-bit security and Medium size for 62-bit security in [10] with a slight modification; the implementation in [10] used only a single zero-testing *integer*. However, we assume that a zero-testing *vector* is given, as in the original CLT GES. Our experimental results demonstrate that our algorithm requires less than $2^{34.84}$ and $2^{37.23}$ clock cycles on average for Small size and Medium size, respectively.

We remark that a part of this paper was made public through [28] and the missing details can be found in the full version of this paper [25].

1.2 Outline

In the following section, we provide some preliminary information that should be helpful for reading this paper. In Section 3, we define our new problem, and investigate a relation between it and the system parameters of multilinear maps. Section 4 provides our attack algorithm along with a detailed analysis. We describe how to speed our basic algorithm up and provide implementation results of our algorithm on the parameters of multilinear maps over the integers in Section 5. In Section 6, we discuss additional issues related to multilinear maps and our attack algorithms.

2 Preliminaries

Notation. Throughout the paper, λ is the security parameter, and we consider only discrete values; the interval notation $[a, b]$ indicates all integers be-

tween a and b, containing a and b. Similarly, (a, b) and $(a, b]$ notations also indicate respective sets of all integers contained in the corresponding continuous intervals. For integers a and p, the reduction of a modulo p is denoted by $a \pmod p \in (-p/2, p/2]$. Problem instances are defined by Chinese Remaindering with respect to n co-prime integers $p_1, \ldots, p_n$, making it convenient to use the notation $\mathsf{CRT}_{p_1,\ldots,p_n}(r_1, \ldots, r_n)$ (abbreviated as $\mathsf{CRT}_{(p_i)}(r_i)$) to denote the unique integer x in $(-\frac{1}{2}\prod_{i\in[1,n]} p_i, \frac{1}{2}\prod_{i\in[1,n]} p_i]$ with $x \equiv r_i \pmod{p_i}$ for all $i \in [1, n]$.

2.1 Fast Polynomial Algorithms

We consider polynomials with integer coefficients modulo x_0. There are classic algorithms for fast polynomial arithmetic, which use the Fast Fourier Transformation (FFT) [15,3,4] and have been used in various areas of cryptography, in particular, cryptanalysis [9,7,16]. In this paper, we use two fast polynomial arithmetic algorithms, each denoted by $\mathsf{Alg}^{FFT}_{Poly}$ and Alg^{FFT}_{MPE}, as subroutines; the algorithm $\mathsf{Alg}^{FFT}_{Poly}$ takes ℓ points as inputs and outputs a monic degree-ℓ polynomial over $\mathbb{Z}_{x_0}$ having ℓ input points as roots. The algorithm Alg^{FFT}_{MPE} takes a degree-ℓ polynomial $f(x)$ over $\mathbb{Z}_{x_0}$ and ℓ points as inputs, and then it evaluates $f(x)$ at ℓ input points and outputs the results. $\mathsf{Alg}^{FFT}_{Poly}$ (Alg^{FFT}_{MPE}, resp.) has quasi-linear complexity in the number of the input points (the degree of the input polynomial, resp.). We summarize the basic information regarding these fast polynomial algorithms in Table 2. We omit details of these classical algorithms; instead, we refer to [31,27].

Table 2. Fast polynomial algorithms using FFT

	$\mathsf{Alg}^{FFT}_{Poly}$	Alg^{FFT}_{MPE}
Input	x_0 and $\{a_0, \ldots, a_{\ell-1}\}$	x_0, $f(X)$ of ℓ-deg., and $\{\mathtt{pt}_i\}_{i\in[0,\ell-1]}$
Output	$f(X) = \prod_{i=0}^{\ell-1}(X - a_i) \pmod{x_0}$	$f(\mathtt{pt}_0), \ldots, f(\mathtt{pt}_{\ell-1}) \pmod{x_0}$
Comp. cost	$\mathcal{O}(\ell \log^2 \ell)$ operations modulo x_0	$\mathcal{O}(\ell \log^2 \ell)$ operations modulo x_0
Space cost	$\mathcal{O}(\ell \log^2 \ell)$ polynomially many bits	$\mathcal{O}(\ell \log^2 \ell)$ polynomially many bits

3 Masked Partial Approximate Common Divisors

Before providing our algorithm, we first generalize the problem instances for both the re-randomization parameter and the zero-testing parameter in the CLT GES. We believe that the following generalization will help readers to understand the security of the multilinear maps; specifically, both the hardness and weakness of the problem. We introduce a new number theoretic problem, which is a variant of *(Partial) Approximate Common Divisors* [23]. First, we describe the new hardness problem, then discuss its relationship with the system parameters of CLT GES in the following subsection.

Definition 1 (n-Masked Partial Approximate Common Divisors). *Given integers $Q, q_0, p_1, \ldots, p_n$, we state that x_j is sampled from the distribution $\mathcal{D}^M_\rho(Q, q_0, p_1, \ldots, p_n)$ if*

$$x_j = Q \cdot \mathsf{CRT}_{q_0,(p_i)}(q_j, r_{1j}, \ldots, r_{nj}) (\text{mod } q_0 \prod_{i\in[1,n]} p_i),$$

where $q_j \leftarrow [0, q_0)$ and $r_{ij} \leftarrow (-2^\rho, 2^\rho)$.

We define the (ρ, η, γ, n)-Masked Partial Approximate Common Divisors (abbreviated as n-MPACD) problem as follows. Choose η-bit random primes p_i for $i \in [1, n]$ and let π be their product. Set $x_0 := q_0 \cdot \pi$, where q_0 is a randomly chosen 2^{λ^2}-rough integer from $[0, 2^\gamma/\pi)$. Choose $Q \leftarrow [0, x_0)$. Given x_0 and polynomially many samples x_j from $\mathcal{D}^M_\rho(Q, q_0, p_1, \ldots, p_n)$, find a non-trivial factor of (x_0/q_0).

Note that we do not restrict the distribution of r_{ij}'s and Q in Definition 1 explicitly to cover various variants; in addition, our attack algorithm provided in the following section succeeds regardless of the distributions of Q and r_{ij}'s. We require only mild restrictions satisfied by both the zero-parameters and the re-randomization parameters of multilinear maps, which are the primary targets of our algorithm.

Hardness of n-MPACD: This paper mainly proposes attack algorithms against n-MPACD; however, it would be interesting to precisely understand the hardness of n-MPACD as well. To this end, we prove that n-MPACD is hard if PACD [23,13,14,7] is also hard. The reduction is provided in the full version.

Asymptotic Parameters: When we consider algorithms for n-MPACD, we basically assume that parameters are set to thwart various lattice-based attacks and factoring algorithms; that is, γ (x_0's bit size) must be large enough to prevent lattice-based attacks, so that $\gamma = \omega(\eta^2 \log \lambda)$ [30,13,10] and $\eta = \omega(\lambda^2)$, to prevent an efficient factorization algorithm such as ECM from having sub-exponential complexity in the size of factors. In this paper, we focus on the size of errors $r_{ij} \in (-2^\rho, 2^\rho)$ and the complexities of all algorithms associated with ρ.

3.1 Parameters as an Instance of the MPACD Problem

We demonstrate that the system parameters in the CLT GES can be considered as instances of n-MPACD.

Zero-testing Parameter: The zero-testing parameters (x_0, $(\boldsymbol{p}_{zt})_j$ for $j \in [1, n]$) are of form

$$\begin{aligned}(\boldsymbol{p}_{zt})_j &= \textstyle\sum_{i=1}^n h_{ij} \cdot (z^\kappa \cdot g_i^{-1} \mod p_i) \cdot \prod_{i'\neq i} p_{i'} \ (\text{mod } x_0) \\ &= Q \cdot \mathsf{CRT}_{(p_i)}(h_{ij}) \ (\text{mod } x_0)\end{aligned}$$

where $Q = \mathsf{CRT}_{(p_i)}(z^\kappa \cdot g_i^{-1} \cdot \prod_{i'\neq i} p_{i'})$. Here, h_{ij} is distributed in a small bounded set $(-2^\beta, 2^\beta)$, where $2^\beta \ll p_i$. Therefore, we can regard the zero-testing parameters as an instance of n-MPACD.

Re-randomization Parameter: The re-randomization parameters are of form

$$\Pi_j = \mathsf{CRT}_{(p_i)}(\tfrac{\varpi_{ij} \cdot g_i}{z}) \equiv Q \cdot \mathsf{CRT}_{(p_i)}(\varpi_{ij}) \bmod x_0,$$

where $Q = \mathsf{CRT}_{(p_i)}(\frac{g_i}{z})$. Note that the ϖ_{ij}'s of the errors are not chosen from the same set, unlike those in n-MPACD; non-diagonal entries are chosen from

$(-2^\rho, 2^\rho)$, while the diagonal entries are chosen from $(n2^\rho, n2^\rho + 2^\rho)$. Although errors are chosen from two different sets, the sizes of both sets are almost equal to 2^ρ. This is sufficient for our attack algorithm provided in Section 4.

Remark 1. In fact, by excluding some parts that have entries chosen from $(n2^\rho, n2^\rho + 2^\rho)$, the re-randomization parameters generated by n primes may be considered as an instance of $(n-k)$-MPACD as well for $k < n$. That is, $\{\Pi_j\}_{j\in[1,k]}$ for $k \in [1, n]$ can be re-written by

$$\Pi_j \equiv \mathsf{CRT}_{(p_i)}(\frac{g_i}{z}) \cdot \mathsf{CRT}_{(p_i)}(\varpi_{ij}) \equiv \mathsf{CRT}_{q_0,(p_i)_{i\in[k+1,n]}}(q', \frac{\varpi_{ij} \cdot g_i}{z}) \bmod x_0,$$

where $q_0 = \prod_{i=1}^{k} p_i$ and $q' = \mathsf{CRT}_{p_1,\ldots,p_k}(\varpi_{1j}, \ldots, \varpi_{kj})$. Subsequently, all errors ϖ_{ij} for $i \in [k+1, n]$ and $j \in [1, k]$ are chosen from $(-2^\rho, 2^\rho)$, so that $(x_0, \{\Pi_j\}_{j\in[1,k]})$ is an instance of $(n-k)$-MPACD.

4 Our Algorithms for the n-MPACD Problem

We present an exponentially faster algorithm for solving n-MPACD problems; our (basic) algorithm requires $\mathcal{O}((\log \rho)^{0.5}\rho^{2.5}2^{\rho/2})$ $\mathbb{Z}_{x_0}$ operations. In [10], the attack algorithm for n-MPACD is roughly sketched and details are omitted. We present the detailed description of the CLT attack based on our speculation in the full version, which achieves the complexity Coron *et al.* claimed. Our analysis of the CLT algorithm for n-MPACD requires two mild assumptions about the distribution of samples. Similarly, the proposed algorithm also requires two mild assumptions about samples satisfied by our main application, multilinear maps over the integers.

4.1 Overview

We provide an overview of our algorithm for solving n-MPACD problems. Our strategy follows the basic flow of the Chen-Nguyen attack; however, we require several additional ideas to manage the unknown masking Q and several moduli in the n-MPACD problem, in contrast to the Chen-Nguyen attack for the PACD problem.

Consider an instance of an n-MPACD problem: $x_0 = q_0 \prod_{i=1}^{n} p_i$ and $x_j \equiv r_{ij} \bmod p_i$ where p_i's are η-bit primes, $r_{ij} \in (-2^\rho, 2^\rho)$ for $1 \le j \le 2m$, and $2^\rho \ll p_i$. For randomly chosen bits b'_j's, if m is sufficiently large, then for each p_i there is a high probability that

$$p_i \;\Big|\; \gcd\Big(x_0, \prod_{\substack{(b_1,\ldots,b_{2m})\in\{0,1\}^{2m} \\ (b_1,\ldots,b_{2m})\neq(b'_1,\ldots,b'_{2m})}} (\sum_{j=1}^{2m} b_j x_j - \sum_{j=1}^{2m} b'_j x_j) \pmod{x_0}\Big). \tag{1}$$

For each p_i, there are 2^{2m} possible sums $\sum_{j=1}^{2m} b_j x_j$ such that $\sum_{j=1}^{2m} b_j x_j \pmod{p_i}$ is contained in the relatively small range $(-2m2^\rho, 2m2^\rho)$, which is contained in $(-p_i/2, p_i/2]$. If the number of samples m satisfies an inequality $2^{2m} \ge m2^{\rho+3}$ (e.g., $2m = \rho + \log\rho + \log\log\rho$ for sufficiently large ρ), then there are many

collisions in the range. In fact, at least a half of all possible elements have a collision in the range $(-2m2^{\rho}, 2m2^{\rho})$ according to the pigeonhole principle. Therefore, for such an m, we have $\prod_{\substack{(b_1,\ldots,b_{2m})\in\{0,1\}^{2m} \\ (b_1,\ldots,b_{2m})\neq(b'_1,\ldots,b'_{2m})}} (\sum_{j=1}^{2m} b_j x_j - \sum_{j=1}^{2m} b'_j x_j) \equiv 0 \pmod{p_i}$ with at least $1/2$ probability, depending on the choice of b'_j's.

To solve an n-MPACD problem using the relation (1), two remaining issues must be considered, in terms of efficiency and correctness. First, $2^{2m} > 2^{\rho}$ modulus multiplications, which are quite large, are required for naive computation of the above product. To reduce the complexity, we follow the concept of the meet-in-the-middle approach, similar to the Chen-Nguyen attack. Second, it is likely that the result of the gcd computation in (1) is not a non-trivial factor of x_0, but just x_0. To overcome this obstacle, we additionally equip our algorithm with the concept of the *divide-and-conquer* technique.

Let us address the efficiency issue first. We define the 2^d-degree polynomial $f_{d,(b'_j)}(X)$ over $\mathbb{Z}_{x_0}$ as follows:

$$f_{d,(b'_j)}(X) = \prod_{(b_1,\ldots,b_d)\in\{0,1\}^d} ((X + \sum_{j=1}^{d} b_j x_j) - \sum_{j=1}^{2m} b'_j x_j) \pmod{x_0} \tag{2}$$

Using this new notation, we can rewrite (1) as[1]

$$p_i \;\Big|\; \gcd\Big(x_0, \prod_{(b_{m+1},\ldots,b_{2m})\in\{0,1\}^m} f_{m,(b'_j)}(\sum_{k=m+1}^{2m} b_k x_k) \pmod{x_0}\Big). \tag{3}$$

We can compute the 2^m-degree polynomial $f_{m,(b'_j)}(X)$ via $\mathsf{Alg}^{FFT}_{Poly}$ and evaluate $f_{m,(b'_j)}(X)$ at 2^m points $\{\sum_{k=m+1}^{2m} b_k x_k\}_{(b_{m+1},\ldots,b_{2m})\in\{0,1\}^m}$ via Alg^{FFT}_{MPE} so that we can solve the n-MPACD problem with $\mathcal{O}(2^m m^2)$ complexity. If we set $2m = \rho + \log\rho + \log\log\rho$, then we determine that the complexity is $\mathcal{O}((\log\rho)^{0.5}\rho^{2.5}2^{\rho/2})$.

For the second issue regarding the gcd computation result, we can apply the divide-and-conquer method. It is clear that the result should be x_0 or its divisor. If the output of the gcd computation is x_0, then we divide the product $\prod_{(b_{m+1},\ldots,b_{2m})\in\{0,1\}^m} f_{m,(b'_j)}(\sum_{k=m+1}^{2m} b_k x_k) \pmod{x_0}$ into four factors and compute all factors. If there is a non-trivial factor among four factors, then the algorithm succeeds. Otherwise, we select a factor that is a multiple of x_0, and repeat the same process until a non-trivial factor is found. We can demonstrate that this process will find a non-trivial factor with overwhelming probability, and the recursive process's asymptotic complexity is still $\mathcal{O}((\log\rho)^{0.5}\rho^{2.5}2^{\rho/2})$. We provide a clear description and analysis of our algorithm in the following subsections.

If the errors r_{ij}'s are distributed (almost) uniformly, then we can reduce the complexity further by scrunching the domain of the product up; if the domain size is decreasing, we cannot expect that $(\sum_{j=1}^{2m} b'_j x_j)$ will have a collision in each modulus p_i with high probability; however, we can expect that it will have a collision in at least one modulus p_i, which is exactly what we want. In fact, we can

[1] Strictly speaking, (3) is not equal to (1) because (3) contains the case $(b_1,\ldots,b_{2m}) = (b'_1,\ldots,b'_{2m})$ therefore the product is trivially 0. We can easily change (3) to not contain the case $(b_1,\ldots,b_{2m}) = (b'_1,\ldots,b'_{2m})$. Because such a modification is technical, we omit it in this overview and relegate the details to the next subsection.

reduce the $\sqrt{n}$ factor further from the complexity. In Section 5, we discuss the method we used to increase the speed of our basic algorithm.

4.2 Basic Algorithm for n-MPACD

Given $2m$ samples x_j's when $2m \leq n$ and $m2^{\rho+2} \leq 2^{2m}$, we require two mild assumptions regarding samples.

Assumption 1. $2m2^{\rho+1} \leq p_i$ *for each* p_i.
Assumption 2. The rank of the integer matrix $(r_{ij})_{\substack{i\in[1,n]\\ j\in[1,2m]}} \in \mathbb{Z}^{n\times 2m}$ *is* $2m$*, where* $x_j \equiv r_{ij} \pmod{p_i}$.

Note that both the zero-testing parameter and the re-randomization parameter of multilinear maps over the integers satisfy both Assumption 1 & 2; Assumption 1 is trivial. In the zero-testing parameter, the matrix (h_{ij}) is invertible, so it can satisfy Assumption 2. For the re-randomization parameter, r_{ij}'s are distributed uniformly and independently; thus, the $rank(r_{ij})$ will be equal to $2m$ with overwhelming probability because r_{ij}'s are chosen from the exponentially large set in the security parameter.

Our n-MPACD Algorithm: We present our basic algorithm for n-MPACD in Algorithm 1. Our algorithm consists of two steps. First, the algorithm computes a product A that is a multiple of some prime factor of x_0. Second, if A is not a multiple of x_0, then the algorithm stops and outputs it. Otherwise, the algorithm runs the *while loop* to extract a non-trivial factor from the multiple of x_0; that is, we repeatedly split multiples of x_0 into four factors, until a non-trivial factor is found.

Because A is a product, we can compute A's four factors denoted by A_{00}, A_{01}, A_{10}, and A_{11} via the same process used for computing A such that $A = A_{00}A_{01}A_{10}A_{11}$, and then check if there is a non-trivial factor of x_0 among them. If not, repeat the same process until a non-trivial factor of x_0 is found. To optimize efficiency, we divide A into four factors *evenly*, that is, each A_i is also a product with the same size domain. Furthermore, we should set each domain of A_i to take full advantage of $\mathsf{Alg}^{FFT}_{Poly}$ and Alg^{FFT}_{MPE}. To this end, we define A_{00}, A_{01}, A_{10}, and A_{11} as follows: In the while loop, $A \in \mathbb{Z}_{x_0}$ is of the form

$$\prod_{\substack{\forall (b_{i_1},\ldots,b_m),\forall (b_{i_2},\ldots,b_{2m})\\ (b_1,\ldots,b_{2m})\neq(b'_1,\ldots,b'_{2m})}} \left(\sum_{j=i_1}^{m} b_j x_j + \sum_{j=i_2}^{2m} b_j x_j + C\right) \pmod{x_0},$$

where $b_1, \ldots, b_{i_1-1}, b_{m+1}, \ldots, b_{i_2-1}$ are fixed for some $1 \leq i_1 \leq m, m+1 \leq i_2 \leq 2m$, and so $C = \sum_{j=1}^{i_1-1} b_j x_j + \sum_{j=m+1}^{i_2-1} b_j x_j - \sum_{j=1}^{2m} b'_j x_j$ is a constant. Then,

$$\begin{aligned}
A_{00} &:= \textstyle\prod\left(\sum_{j=i_1+1}^{m} b_j x_j + \sum_{j=i_2+1}^{2m} b_j x_j + C\right) \pmod{x_0},\\
A_{01} &:= \textstyle\prod\left(\sum_{j=i_1+1}^{m} b_j x_j + \sum_{j=i_2+1}^{2m} b_j x_j + C + x_{i_2}\right) \pmod{x_0},\\
A_{10} &:= \textstyle\prod\left(\sum_{j=i_1+1}^{m} b_j x_j + \sum_{j=i_2+1}^{2m} b_j x_j + C + x_{i_1}\right) \pmod{x_0},\\
A_{11} &:= \textstyle\prod\left(\sum_{j=i_1+1}^{m} b_j x_j + \sum_{j=i_2+1}^{2m} b_j x_j + C + x_{i_1} + x_{i_2}\right) \pmod{x_0},
\end{aligned}$$

Algorithm 1. n-MPACD algorithm: arbitrary distribution

Input: $(x_0, x_1, \ldots, x_{2m})$
Output: a non-trivial factor of x_0 or $\perp$

1: Choose $b'_j \xleftarrow{\$} \{0,1\}$ for $1 \le j \le 2m$.
2: Compute $A = \prod_{\substack{(b_1,\ldots,b_{2m})\in\{0,1\}^{2m} \\ (b_1,\ldots,b_{2m})\neq(b'_1,\ldots,b'_{2m})}} (\sum_{j=1}^{2m} b_j x_j - \sum_{j=1}^{2m} b'_j x_j) \pmod{x_0}$
▷ by using Alg. 2
3: **if** $A \not\equiv 0 \pmod{x_0}$ **then return** $\gcd(x_0, A)$.
4: **else** Set $k \leftarrow 1$.
5: **while** $k \le m$ **do**
6: Compute $\gcd(x_0, A_i)$ for $i \in \{00, 01, 10, 11\}$.
▷ by using (a variant of) Alg. 2
7: **if** $\gcd(x_0, A_i) \in (1, x_0)$ for some i **then return** $\gcd(x_0, A_i)$.
8: **else** Choose an A_i s.t. $A_i \equiv 0 \pmod{x_0}$ and set $A \leftarrow A_i$, and $k \leftarrow k+1$.
9: **end if**
10: **end while return** $\perp$.
11: **end if**

where C is defined as before and each product is defined over all $b_{i_1+1}, \ldots, b_m$, $b_{i_2+1}, \ldots, b_{2m} \in \{0,1\}$ such that $(b_1, \ldots, b_{2m}) \neq (b'_1, \ldots, b'_{2m})$. It is clear that $A = A_{00}A_{01}A_{10}A_{11}$ and each A_i has the same form as A with a different domain for the product.

Subroutine for Computing A and Its Factors: We describe how to compute $A = \prod_{\substack{(b_1,\ldots,b_{2m})\in\{0,1\}^{2m} \\ (b_1,\ldots,b_{2m})\neq(b'_1,\ldots,b'_{2m})}} (\sum_{j=1}^{2m} b_j x_j - \sum_{j=1}^{2m} b'_j x_j) \pmod{x_0}$.
Using the notation in (2), A can be rewritten as

$$\prod_{\substack{(b_{m+1},\ldots,b_{2m})\\ \in\{0,1\}^m, \\ (b_{m+1},\ldots,b_{2m})\\ \neq(b'_{m+1},\ldots,b'_{2m})}} f_{m,(b'_j)}\Big(\sum_{k=m+1}^{2m} b_k x_k\Big) \cdot \prod_{\substack{(b_1,\ldots,b_m)\\ \in\{0,1\}^m, \\ (b_1,\ldots,b_m)\\ \neq(b'_1,\ldots,b'_m)}} \Big(\sum_{j=1}^{m}(b_j - b'_j)x_j\Big) \tag{4}$$

The left term is for the case $(b_{m+1}, \ldots, b_{2m}) \neq (b'_{m+1}, \ldots, b'_{2m})$ and the right term is for the case $(b_{m+1}, \ldots, b_{2m}) = (b'_{m+1}, \ldots, b'_{2m})$ with $(b_1, \ldots, b_m) \neq (b'_1, \ldots, b'_m)$. Therefore, (4) covers all $(b_1, \ldots, b_{2m})$'s except $(b'_1, \ldots, b'_{2m})$, so that it is equal to A. We describe an algorithm for (4) in Algorithm 2. Factors A_{00}, A_{01}, A_{10} and A_{11} of A have approximately the same form as A, and hence we can compute it similarly to Algorithm 2.

Algorithm 2. Subroutine for solving n-MPACD

Input: $(x_0, x_1, \ldots, x_{2m})$ and $(b'_1, \ldots, b'_{2m})$.

Output: $A = \prod_{\substack{(b_1,\ldots,b_{2m})\in\{0,1\}^{2m} \\ (b_1,\ldots,b_{2m})\neq(b'_1,\ldots,b'_{2m})}} (\sum_{j=1}^{2m} b_j x_j - \sum_{j=1}^{2m} b'_j x_j) \pmod{x_0}$

1: Compute a polynomial $f_{m,(b'_j)}(X)$ over $\mathbb{Z}_{x_0}$ as follows.
$\prod_{(b_1,\ldots,b_m)\in\{0,1\}^m}((X + \sum_{j=1}^{m} b_j x_j) - \sum_{j=1}^{2m} b'_j x_j) \pmod{x_0}$.
▷ by $\mathsf{Alg}^{FFT}_{Poly}$ with x_0 and $\{(\sum_{j=1}^{2m} b'_j x_j - \sum_{j=1}^{m} b_j x_j)\}_{(b_1,\ldots,b_m)\in\{0,1\}^m}$ as inputs.

2: Perform multi-points evaluation of $f_{m,(b'_j)}(X)$ at $\{\sum_{k=m+1}^{2m} b_k x_k\}_{\forall b_k\in\{0,1\}}$ ▷ by Alg^{FFT}_{MPE}.

3: **return**
$$\prod_{\substack{(b_{m+1},\ldots,b_{2m})\\ \in\{0,1\}^m, \\ (b_{m+1},\ldots,b_{2m})\\ \neq(b'_{m+1},\ldots,b'_{2m})}} f_{m,(b'_j)}(\sum_{k=m+1}^{2m} b_k x_k) \cdot \prod_{\substack{(b_1,\ldots,b_m)\\ \in\{0,1\}^m, \\ (b_1,\ldots,b_m)\\ \neq(b'_1,\ldots,b'_m)}} (\sum_{j=1}^{m}(b_j - b'_j)x_j) \pmod{x_0}$$

4.3 Analysis

Success Probability: We demonstrate that Algorithm 1 correctly finds a non-trivial factor of x_0 with at least $1/2$ probability, where the probability goes over only the algorithm's random tape.[2]

Algorithm 1 begins by selecting $b'_j \in \{0,1\}$ for $1 \leq j \leq 2m$. Given an n-MPACD instance x_0 and x_j's, we state that $(b'_1, \ldots, b'_{2m}) \in \{0,1\}^{2m}$ is '*good for* p_i' if there exists $(b_1, \ldots, b_{2m}) \in \{0,1\}^{2m}$ such that $(b_1, \ldots, b_{2m}) \neq (b'_1, \ldots, b'_{2m})$ and $\sum_{j=1}^{2m} b_j x_j = \sum_{j=1}^{2m} b'_j x_j \pmod{p_i}$. We can prove that if we select b'_j's uniformly and independently, then with high probability $(b'_1, \ldots, b'_{2m})$ is 'good for p_i' for each p_i. See Lemma 1 for details.

Lemma 1. *Given an n-MPACD instance x_0 and x_j's, we have that for each $i \in [1, n]$,*

$$\Pr_{b'_j \xleftarrow{\$} \{0,1\}} [(b'_1, \ldots, b'_{2m}) \text{ is good for } p_i] > 1/2 \quad \textit{under Assumption 1.}$$

Once the algorithm has a good $(b'_1, \ldots, b'_{2m})$ for p_1, then we can demonstrate that the algorithm eventually outputs a non-trivial factor of x_0. If the while loop arrives at the end before finding a non-trivial factor of x_0 (that is, it is repeated m times), then ultimately we should have an integer $\sum_{j=1}^{2m} b_j x_j - \sum_{j=1}^{2m} b'_j x_j \equiv 0 \pmod{x_0}$ for some $(b_1, \ldots, b_{2m}) \neq (b'_1, \ldots, b'_{2m})$; that is, we are not able to divide A any further. Therefore, it is sufficient to demonstrate that such a tuple $(b_1, \ldots, b_{2m})$ cannot exist, and Lemma 2 guarantees it.

[2] Because the success probability of our algorithm is constant, we can make that the probability of success is overwhelming by running the algorithm linear in the security parameter, with a fresh random tape.

Lemma 2. *Under Assumption 1 and 2, if* $(b_1, \ldots, b_{2m}) \neq (b'_1, \ldots, b'_{2m})$*, then there is an index* $i' \in [1, n]$ *such that*

$$\sum_{j=1}^{2m} b_j x_j \neq \sum_{j=1}^{2m} b'_j x_j \pmod{p_{i'}}$$

so that $\sum_{j=1}^{2m} b_j x_j \neq \sum_{j=1}^{2m} b'_j x_j \pmod{x_0}$.

Algorithm 1 uses the randomness only in the 1st and 8th steps. Because any A_i with correct conditions will suffice in the 8th step, it does not affect the success probability of the algorithm. Only a selection of $(b'_1, \ldots, b'_{2m})$ will determine the success of the algorithm, and we have a probability of greater than $1/2$ for a good $(b'_1, \ldots, b'_{2m})$ for p_1. Therefore, the proposed algorithm has at least a $1/2$ probability for success.

Complexity: The complexity of Algorithm 1 is dominated by computing A and its factors. The complexity of Algorithm 2 mainly depends on the domain size in the product; we require $\mathcal{O}(m^2 2^m)$ operations modulo x_0 (from $\mathsf{Alg}_{Poly}^{FFT}$ and Alg_{MPE}^{FFT}'s complexity). Similarly, for each of A's four factors, we must perform $\mathcal{O}((m-1)^2 2^{m-1})$ operations modulo x_0 because each factor of A uses a half-size degree polynomial and number of points in Algorithm 2 in comparison with A. Similarly, we require $\mathcal{O}((m-2)^2 2^{m-2})$ operations modulo x_0 for each of A_i's four factors, and so on. Overall, the computational complexity for A and all its factors is bounded by $\mathcal{O}(m^2 2^m) + 4\mathcal{O}((m-1)^2 2^{m-1}) + \cdots + 4\tilde{\mathcal{O}}(2^1) = \mathcal{O}(5m^2 2^m) = \mathcal{O}(m^2 2^m)$ operations modulo x_0. Therefore, the overall computational cost is $\mathcal{O}(m^2 2^m) + \mathcal{O}(m 2^m) = \mathcal{O}(m^2 2^m)$ $\mathbb{Z}_{x_0}$ operations. Similarly, we can demonstrate that the space complexity is bounded by $\mathcal{O}(m^2 2^m)$ polynomially many bits from the storage complexity of Alg_{MPE}^{FFT} and $\mathsf{Alg}_{Poly}^{FFT}$.

If we set $m = \frac{\rho + \log \rho + \log\log \rho}{2}$, then it asymptotically satisfies the requirement $2m \leq n$ and $2m 2^{\rho+1} < 2^{2m}$, where $\rho \geq 4$. Therefore, for $m = \frac{\rho + \log \rho + \log\log \rho}{2}$, the computational cost is $\mathcal{O}((\frac{\rho + \log \rho + \log\log \rho}{2})^2 \, 2^{\frac{\rho + \log \rho + \log\log \rho}{2}}) = \mathcal{O}((\log \rho)^{0.5} \rho^{2.5} 2^{\rho/2})$ $\mathbb{Z}_{x_0}$ operations and the space complexity is $\mathcal{O}((\log \rho)^{0.5} \rho^{2.5} 2^{\rho/2})$ polynomially many bits.

5 Attack on System Parameters of Multilinear Maps over the Integers

5.1 Speed Increase for Multilinear Maps Parameters

We introduce several techniques to increase the speed of Algorithm 1, where all of our techniques are applicable to the parameters of multilinear maps. If r_{ij}'s are uniformly distributed, we can increase the speed of the attack algorithm. For example, ϖ_{ij}'s in the re-randomization parameter are uniformly distributed in the corresponding domains. Furthermore, we know the distribution of h_{ij}'s in the zero-testing parameter. Although it is not a uniform distribution, we can consider it as a quasi-uniform distribution in an appropriate bound.

Using Shorter m: To guarantee exponentially many *good* $(b'_1, \ldots, b'_{2m})$ for each p_i, we select m with $2m2^{\rho+1} \leq 2^{2m}$. The sum of uniform variables follows the *bell-shaped* distribution, so that $\sum_{j=1}^{2m} b_j x_j$ has a shorter image size than its range. Furthermore, the bell-shaped distribution has more collisions around a center than uniform distributions. This fact allows us to select a shorter m, and our experimental results provided in Table 3 support our expectation.

Table 3. Shorter domains (Experimental results on average of 100 instances)

ρ	14	16	18	20
m	8	9	10	11
|domain|/|range|	0.25	0.22	0.20	0.18
|domain|/|image|	1.49	1.51	1.48	1.44

Shorter Domain in Products: Basically, Algorithm 1 becomes a brute-force attack once we select a good $(b'_1, \ldots, b'_{2m})$ for some p_i at the beginning. It is likely that $(b'_1, \ldots, b'_{2m})$ is good for several moduli p_i's. (That is the exact reason why we must have the while loop in Algorithm 1.) However, our goal is to select a vector

Algorithm 3. n-MPACD algorithm: speedup for the uniform distribution

Input: $(x_0, x_1, \ldots, x_{2m})$, $d = 2^\delta$ for $\delta \geq 1$
Output: a non-trivial factor of x_0 or $\perp$

1: Choose $(b'_1, \ldots, b'_{2m}) \xleftarrow{\$} \{0,1\}^{2m}$.
2: Choose $b_1, \ldots, b_\delta, b_{m+1}, \ldots, b_{m+\delta} \xleftarrow{\$} \{0,1\}$.
3: Compute $A = \prod_{\substack{(b_{\delta+1}, \ldots, b_m) \in \mathbb{Z}_{x_0}^{m-\delta} \\ (b_{m+\delta+1}, b_{2m}) \in \mathbb{Z}_{x_0}^{m-\delta} \\ (b_1, \ldots, b_{2m}) \neq (b'_1, \ldots, b'_{2m})}} (\sum_{j=1}^{2m} b_j x_j - \sum_{j=1}^{2m} b'_j x_j) \pmod{x_0}$
▷ by using Alg. 2
4: The remaining process is the same as Step $3-11$ of Algorithm 1.

Table 4. Speedup with shorter interval (Experimental results on average of 100 instances)

Instantiation	λ	n	η	ρ	m	d	(Average) trials
Micro	≥ 34	64	1528	22	12	8	1.81 times

Parameters are set the average ratio between the domain and the image (modulus p_i) of $\sum_{1 \leq j \leq 2m} b_j x_j$ for 100 problem instances to be 1.44 for each p_i.

$(b'_1, \ldots, b'_{2m})$ that is good for only one (or a few) p_i and is not good for any others. We compute a product A' that is roughly $1/n$ of a random portion of the product A in Algorithm 1. Then, we can expect the probability, $\Pr_i$, that p_i divides A' is roughly equal to $1/2n$. Furthermore, r_{ij}'s are independent, and thus we can also expect that the probabilities $\Pr_i$'s are nearly independent. Therefore, the probability that A' is a multiple of at least one of p_i is significant, from the birthday

paradox; e.g., $1-1/\sqrt{e}$. Applying this technique, we present an improved attack in Algorithm 3. The analysis above is heuristic, and thus to support our expectations and the heuristic analysis, we provide experimental results in Table 4.

Insufficient Entropy in Zero-testing Parameters: The matrix $\mathbf{H} = (h_{ij}) \in \mathbb{Z}^{n \times n}$ in the zero-testing parameters is selected to satisfy $\|\mathbf{H}^\top\|_\infty \leq 2^\beta$ and $\|(\mathbf{H^{-1}})^\top\|_\infty \leq 2^\beta$ where $\|\cdot\|_\infty$ is the operator norm of $n \times n$ matrices with respect to the ℓ^∞ norm on $\mathbb{R}^n$. In [10], Coron *et al.* proposed an algorithm to generate such a matrix $\mathbf{H}$, with sufficient entropy. However, their approach does not rapidly increase the entropy of $\mathbf{H}$, though it satisfies the above two bounds. We will demonstrate this by providing some experimental results in this section.

Table 5. Bit-size of entries of a matrix $\mathbf{H}$ (Experimental results on average of 100 matrices for Toy and Small and 10 matrices for Medium)

	λ	n	ρ	β	Average Bit Size	Maximum Bit Size	$\beta - \log n$
Toy	42	136	26	26	1.33	8	18.91
				$42(=\lambda)$	4.66	16	34.91
				80	11.80	25	72.91
				$84(=2\lambda)$	13.99	29	76.91
				$126(=3\lambda)$	23.73	41	118.91
				$168(=4\lambda)$	32.67	51	160.91
Small	52	540	41	41	2.84	14	31.92
				$52(=\lambda)$	4.14	17	42.92
				80	9.70	29	70.92
				$104(=2\lambda)$	16.17	34	94.92
				$156(=3\lambda)$	29.07	47	146.92
				$208(=4\lambda)$	41.69	66	198.92
Medium	62	2085	56	56	5.59	17	44.97
				$62(=\lambda)$	5.63	17	50.97
				80	11.73	27	68.97

Table 5 lists the average bit size of entries in $\mathbf{H}$ generated by the algorithm of Coron *et al.* on various parameters β and n. From the last three columns of Table 5, one can observe that average bit sizes are approximately 10 when $\beta = 80$ as in the impelmentation parameters in [10]; moreover, the maximum bit sizes are lower than 30, and they are much smaller than the best $\beta - \log n$, which is obtained from the bound $\|\mathbf{H}^\top\|_\infty \leq 2^\beta$.

In [10, Section 3.1], the authors stated that *"One can take $\beta = \lambda$"*; however, our analysis and experimental results indicate that β should be much larger than λ. In Table 5, when $\beta \leq 3\lambda$, the expected average bit-size of $|h_{ij}|$ is still smaller than ρ, and for Small security, when $\beta \approx 4\lambda$, the expectation of the average bit-size of $|h_{ij}|$ is equal to ρ; thus, $\beta \geq 4\lambda$ would be safe for the security of the multilinear maps. We investigate the reason why the $\mathbf{H}$-generation in [10] could not increase enough entropy in the full version.

5.2 Implementation

We have implemented Algorithm 3 with various parameters in C++, using the Gnu MP library [1] and NTL library [29], on an Intel(R) Core(TM) i7-2600 CPU at 3.4 GHz with 16 GB RAM.

Attack on Zero-testing Parameter: We have implemented Algorithm 3 to attack on the zero-testing parameters; we set n, η, and ρ as in the implementation parameters for Small (52-bit) and Medium (62-bit) security [10, Section 6.4] and generated the zero-testing parameter normally by using the method described in [11, Appendix F]. We summarize the result in Table 6, and it displays that Algorithm 3 finds a non-trivial factor very quickly on the parameters for Small and Medium security levels.

Table 6. Attack on zero-testing parameter

Inst.	λ	n	η	β	Exp($\lvert h_{ij}\rvert$)	m	d	Time*	Security against Alg. 3
Small	52	540	1838	80	10	8	16	8.42 sec	$\leq 2^{34.84}$ clock cycles
Medium	62	2085	2043	80	12	9	32	47.28 sec	$\leq 2^{37.23}$ clock cycles

* The average running time for solving 50 problem instances

Attack on Re-randomization Parameter: We first define Toy parameters for 42-bit security. To this end, we benchmark the parameter of FHEs in [12], which is conservatively determined according to the complexity of the Chen-Nguyen attack [7]. In Table 7, we provide the average running time to solve 50 problems for Toy parameters, and the experimental result demonstrates that the expected security level is tight.

Table 7. Attack on re-randomization parameter

Inst.	λ	n	η	ρ	m	d	(Average) trials	Running time	Sec. ag. Alg 3†
Toy	42	136	1628	26	14	16	3.7 times	1979.55 sec	$2^{42.72}$

† This counts the number of clock cycles.

In fact, the complexity difference between Algorithm 3 and the Chen-Nguyen attack is $\mathcal{O}(\sqrt{\frac{\rho \log \rho}{n}})$ and $\sqrt{\frac{\rho \log \rho}{n}} \approx 1$ for 42-bit security. For Large and Extra security level parameters, $\sqrt{\frac{\rho \log \rho}{n}}$ is less than 1; therefore, Algorithm 3 will be slightly faster than the Chen-Nguyen attack algorithm. We extrapolate Algorithm 3 to be at least $2^{1.38}$ ($2^{2.12}$, resp.) times faster than the Chen-Nguyen attack for Large security (Extra security, resp.), with a similar storage advantage. Therefore, when one selects secure ρ size for large security level integer-based multilinear maps, we suggest that the performance of Algorithm 3 should be considered.

6 Discussions

Encoding-Validity Test: Zero-testing Vector vs. Zero-testing Integer: In [10], Coron *et al.* implemented a one-round N-way Diffie-Hellman key exchange protocol [2], based on their multilinear maps. They used heuristic optimizations for implementation, in particular the zero-testing *integer*, instead of the zero-testing *vector* as in the original construction. Note that both the CLT attack algorithm and our attack algorithm for n-MPACD require more than one sample; therefore, both are inapplicable to their optimized version of multilinear maps over the integers.

Garg *et al.* [18] pointed out a plausible threat when using a single zero-testing element. In applications that require resilience of the zero test, including against invalid encodings, several zero-testing elements can be utilized to prevent the use of invalid encodings. In cryptographic applications such as the Diffie-Hellman key exchange, it is important to test whether a given encoding is a group element. Because GES is a substitute for ideal multilinear groups, it is also important to test whether a given encoding is valid, and has an appropriate level. In the full version, we present a (polynomial-time) key recovery attack on the multipartite Diffie-Hellman key exchange protocol, based on the CLT GES with a single integer zero-testing parameter. The basic idea of the attack is analogous to the Lim-Lee [26] key recovery attack of using invalid encodings on two-party Diffie-Hellman key exchange based on group structures.

Applications Beyond Multilinear Maps: We note that Algorithm 3 is applicable to the public parameters of a FHE scheme in [8]. Due to the space limitation, we relegate the detail to the full version.

Acknowledgement. This work was supported by IT R&D program of MSIP/ KEIT [No. 10047212]. Hyung Tae Lee was also supported in part by the Singapore Ministry of Education under Research Grant MOE2013-T2-1-041. Part of work was done while Hyung Tae Lee was with Seoul National University, Korea. Jae Hong Seo is the corresponding author for this paper. The authors also would like to thank Jung Hee Cheon and the anonymous reviewers of CRYPTO 2014 for their helpful comments.

References

1. GMP: The GNU multiple precision arithmetic library ver. 5.1.3 (2013), `http://gmplib.org`
2. Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. Contemporary Mathematics 324(1), 71–90 (2003)
3. Bostan, A., Schost, É.: On the complexities of multipoint evaluation and interpolation. Theoretical Computer Sciences 329(1-3), 223–235 (2004)
4. Bostan, A., Schost, É.: Polynomial evaluation and interpolation on special sets of points. Journal of Complexity 21(4), 420–446 (2005)
5. Brakerski, Z., Rothblum, G.N.: Obfuscating conjunctions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 416–434. Springer, Heidelberg (2013)

6. Brakerski, Z., Rothblum, G.N.: Virtual black-box obfuscation for all circuits via generic graded encoding. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 1–25. Springer, Heidelberg (2014)
7. 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)
8. 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)
9. Coron, J.-S., Joux, A., Mandal, A., Naccache, D., Tibouchi, M.: Cryptanalysis of the RSA subgroup assumption from TCC 2005. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 147–155. Springer, Heidelberg (2011)
10. Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 476–493. Springer, Heidelberg (2013)
11. Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers (2013), `http://eprint.iacr.org/2013/183` (Full version of [10])
12. Coron, J.-S., Lepoint, T., Tibouchi, M.: Batch fully homomorphic encryption over the integers (2013), `http://eprint.iacr.org/2013/036` (Full version of the second part of [8])
13. 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)
14. Coron, J.-S., Naccache, D., Tibouchi, M.: Public key compression and modulus switching for fully homomorphic encryption over the integers. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 446–464. Springer, Heidelberg (2012)
15. Fiduccia, A.M.: Polynomial evaluation via the division algorithm: the fast Fourier transform revisited. In: STOC 1972, pp. 88–93 (1972)
16. Fouque, P.-A., Tibouchi, M., Zapalowicz, J.-C.: Recovering private keys generated with weak PRNGs. In: Stam, M. (ed.) IMACC 2013. LNCS, vol. 8308, pp. 158–172. Springer, Heidelberg (2013)
17. Freire, E.S.V., Hofheinz, D., Paterson, K.G., Striecks, C.: Programmable hash functions in the multilinear setting. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 513–530. Springer, Heidelberg (2013)
18. 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)
19. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS 2013, pp. 40–49. IEEE Computer Society (2013)
20. Garg, S., Gentry, C., Halevi, S., Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 479–499. Springer, Heidelberg (2013)
21. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: STOC 2013, pp. 467–476. ACM (2013)

22. Hohenberger, S., Sahai, A., Waters, B.: Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 494–512. Springer, Heidelberg (2013)
23. Howgrave-Graham, N.: Approximate integer common divisors. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 51–66. Springer, Heidelberg (2001)
24. Langlois, A., Stehlé, D., Steinfeld, R.: GGHLite: More efficient multilinear maps from ideal lattices. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 239–256. Springer, Heidelberg (2014)
25. Lee, H.T., Seo, J.H.: Security analysis of multilinear maps over the integers. IACR Cryptology ePrint Archive (2014), http://eprint.iacr.org
26. Lim, C.H., Lee, P.J.: A key recovery attack on discrete log-based schemes using a prime order subgroup. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 249–263. Springer, Heidelberg (1997)
27. Mateer, T.: Fast Fourier transform algorithms with applications. PhD thesis, Clemson University (2008)
28. Seo, J.H.: Faster algorithms for variants of approximate common divisors problem: Application to multilinear maps over the integers. In: Memoirs of the 7th Cryptology Paper Contest, arranged by Korea government organization (2013)
29. Shoup, V.: NTL: A library for doing number theory ver. 6.0.0 (2013), http://www.shoup.net/ntl/
30. 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)
31. von zur Gathen, J., Gerhard, J.: Modern computer algebra, 2nd edn. Cambridge University Press (2003)