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] Department of Mathematics, 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 [3] 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 [20] 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 [12] 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 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 [28] 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 [19], full-domain hash [25], functional encryption [21, 22], witness encryption [23], and indistinguishability obfuscation [6, 21, 7]. 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.*

[⋆] This is the full version of a paper [29] presented at the CRYPTO 2014 conference.

Table 1. Algorithms for n-MPACD

Algorithm	Error Type	Computation ($\mathbb{Z}_{x_0}$ op.)	Space
(Corrected) CLT [12]	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.

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 [12] and the re-randomization parameter [12, 9]) from integer-based schemes such as multilinear maps [12] and Fully Homomorphic Encryptions (FHE) [9]. 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 $Q \overset{\$}{\leftarrow} \mathbb{Z}_{x_0}$, $r_{ij} \overset{\$}{\leftarrow} (-2^\rho, 2^\rho)$ 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 [8]; therefore, it appears to be harder than the Partial Approximate Common Divisors (PACD) problem [26]. In fact, the attack algorithm of Coron, Lepoint, and Tibouchi (CLT) [12], 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 [8]. 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} 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 [12]. 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 $\|\mathbf{H}^\top\|_\infty \le 2^\beta$ and $\|(\mathbf{H^{-1}})^\top\|_\infty \le 2^\beta$. In [12], 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.

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

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 [12] with a slight modification; the implementation in [12] 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.

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 between 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) [17, 4, 5] and have been used in various areas of cryptography; e.g., for efficiency improvement in protocols [10, 35, 24, 2] and in cryptanalysis [11, 8, 18]. 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 [36, 31].

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* [26]. 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 $\mathcal{D}_\rho^M(Q, q_0, p_1, \ldots, p_n)$ if*

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

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 $\mathcal{D}_\rho^M(Q, q_0, p_1, \ldots, p_n)$, 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 [26, 15, 16, 8] is also hard. The reduction is provided in Appendix B.

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 $\gamma = \omega(\eta^2 \log \lambda)$ [34, 15, 12] 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

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

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 [12], 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 Appendix C, 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 $x_j \equiv r_{ij} \bmod p_i$ for $1 \leq j \leq 2m$ where p_i's are η-bit primes, $r_{ij} \in (-2^\rho, 2^\rho)$ for $1 \leq j \leq 2m$, 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 $2^{2m} \geq m2^{\rho+3}$ (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[3]

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

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

[3] 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.

expect that it will have a collision in at least one modulus p_i, which is exactly what we want. In fact, we can 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 $2m \le n$ and $m2^{\rho+2} \le 2^{2m}$, we require two mild assumptions regarding samples.

Assumption 1. *$2m2^{\rho+1} \le p_i$ 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 $1 \le i_1 \le m, m+1 \le i_2 \le 2m$, 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,

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

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.

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 $b'_j \overset{\$}{\leftarrow} \{0,1\}$ 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
```

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

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

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.

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.[4]

Algorithm 1 begins by selecting $b'_j \in \{0,1\}$ for $1 \le j \le 2m$. 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 $(b_1, \ldots, b_{2m}) \in \{0,1\}^{2m}$

[4] 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.

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

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 $i \in [1, n]$,*

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

We provide the proof of Lemma 1 in Appendix D.

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.

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

We provide the proof of Lemma 2 in Appendix D.

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 $2m \le n$ and $2m2^{\rho+1} < 2^{2m}$, where $\rho \ge 4$. 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* $(b'_1,\dots,b'_{2m})$ for each p_i, we select m with $2m2^{\rho+1} \le 2^{2m}$. 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 $(b'_1,\dots,b'_{2m})$ for some p_i at the beginning. It is likely that $(b'_1,\dots,b'_{2m})$ 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 $(b'_1,\dots,b'_{2m})$ 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

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

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.

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.

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 [12], 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[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 implementation parameters in [12]; 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 [12, 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 [12] could not increase enough entropy in Section 6.

[5] Note that experimental results about bit sizes on Table 5 in [29] were entered 1 smaller by the authors' mistake.

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	2.33	9	18.91
				$42(=\lambda)$	5.66	17	34.91
				80	12.80	26	72.91
				$84(=2\lambda)$	14.99	30	76.91
				$126(=3\lambda)$	24.73	42	118.91
				$168(=4\lambda)$	33.67	52	160.91
Small	52	540	41	41	3.84	15	31.92
				$52(=\lambda)$	5.14	18	42.92
				80	10.70	30	70.92
				$104(=2\lambda)$	17.17	35	94.92
				$156(=3\lambda)$	30.07	48	146.92
				$208(=4\lambda)$	42.69	67	198.92
Medium	62	2085	56	56	6.59	18	44.97
				$62(=\lambda)$	6.63	18	50.97
				80	12.73	28	68.97

5.2 Implementation

We have implemented Algorithm 3 with various parameters in C++, using the Gnu MP library [1] and NTL library [33], 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 [12, Section 6.4] and generated the zero-testing parameter normally by using the method described in [13, 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	10.70	8	16	8.42 sec	$\leq 2^{34.84}$ clock cycles
Medium	62	2085	2043	80	12.73	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 [14], which is conservatively determined according to the complexity of the Chen-Nguyen attack [8]. 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.

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

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.

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

Generating The Matrix H: First, we look at the method of generating a matrix $\mathbf{H}$ in the zero-testing parameters, which satisfies the bound $\|\mathbf{H}^\top\|_\infty \le 2^\beta$ and $\|(\mathbf{H}^{-1})^\top\|_\infty \le 2^\beta$, proposed by Coron *et al.* [13].

For any matrix $A \in Mat_{\lfloor n/2\rfloor\times\lceil n/2\rceil}(\{-1,0,1\})$, define $\mathbf{H}_A \in \mathbb{Z}^{n\times n}$ as

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

Let $\beta' = \left\lfloor \frac{\beta}{\lceil \log(1+\lceil n/2\rceil)\rceil} \right\rfloor$ and randomly choose $A_i \xleftarrow{\$} Mat_{\lfloor n/2\rfloor\times\lceil n/2\rceil}(\{-1,0,1\})$. Pick $\mathbf{H}_i$ randomly as either $\mathbf{H}_{A_i}$ or its transpose for each $i \in [1,\beta']$, and compute $\mathbf{H}$ as $\prod_{i=1}^{\beta'} \mathbf{H}_i$.

We can easily prove that the above resulting matrix $\mathbf{H}$ satisfies the bounds $\|\mathbf{H}^\top\|_\infty \le 2^\beta$ and $\|(\mathbf{H}^{-1})^\top\|_\infty \le 2^\beta$; for each $i \in [1,\beta']$, $\|\mathbf{H}_i\|_\infty \le 1 + \lceil n/2\rceil$ and $\|\mathbf{H}_i^\top\|_\infty \le 1 + \lceil n/2\rceil$ and hence

$$\|\mathbf{H}^\top\|_\infty \le \prod_{i\in[1,\beta']} \|\mathbf{H}_i^\top\|_\infty \le (1+\lceil n/2\rceil)^{\beta'} \le 2^\beta,$$

and similarly $(\mathbf{H}^{-1})^\top$ satisfies the same bound because $\mathbf{H}_A^{-1} = \mathbf{H}_{(-A)}$.

Although the above approach enables the resulting matrix $\mathbf{H}$ to satisfy the upper bound of the operator norms, and the set of resulting matrices generated in this manner could be exponentially large, this approach does not guarantee the *average* and *expectation* of the operation norms of resulting matrices sufficiently large. We briefly explain the reason why this approach slowly increases the expected bit size of the absolute value of h_{ij}'s, in comparison with its upper bound $\beta - \log n$. For brevity, we assume that n is even number, that is, $n = 2t$. The maximum operator norm of $\mathbf{H}$ can be achieved when the operator norm of each matrix $\mathbf{H}_i$ has the maximum value and $\mathbf{H}$'s operator norm is the product of $\mathbf{H}_{A_i}$'s operator norms. $\mathbf{H}$ is a product of $\mathbf{H}_{A_i}$'s and $\mathbf{H}_{A_i}^\top$'s, there would be many continuous product of $\mathbf{H}_{A_i}$ and $\mathbf{H}_{A_j}$, or $\mathbf{H}_{A_i}^\top$ and $\mathbf{H}_{A_j}^\top$ in computing $\mathbf{H}$. (We state that $\mathbf{H}_{A_i}$ and $\mathbf{H}_{A_j}$ are the same type, and $\mathbf{H}_{A_i}^\top$ and $\mathbf{H}_{A_i}^\top$ are also the same type.) Considering a product of $\mathbf{H}_{A_i}$ and $\mathbf{H}_{A_j}$,

$$\mathbf{H}_{A_i}\mathbf{H}_{A_j} = \begin{pmatrix} I_t & A_i \\ 0 & I_t \end{pmatrix}\begin{pmatrix} I_t & A_j \\ 0 & I_t \end{pmatrix} = \begin{pmatrix} I_t & A_i + A_j \\ 0 & I_t \end{pmatrix}.$$

Thus, $\|\mathbf{H}_{A_i}\mathbf{H}_{A_j}\|_\infty \le 1 + n$. Although the multiplicative bound ($\le (1+\lceil n/2\rceil)^2$) is utilized to calculate the operator norm of two matrices in [12], the additive bound is enough. Furthermore, we can generalize for products of many continuous matrices of the same type. In this case, each entry of a sum of A_i's is a sum of random elements in $\{-1, 0, 1\}$, and hence its distribution will be bell-shaped and it has a value nearby zero with a high probability. Therefore, the expected operator norm of the product matrix of the same types will be much smaller than even the additive bound.

From the above, we know $\mathbf{H}$ is of the form $\mathbf{H}_{B_1}\mathbf{H}_{B_2}^\top\mathbf{H}_{B_3}\mathbf{H}_{B_4}^\top\cdots$ or its transpose, where B_i's are sums of A_j's. Considering $\mathbf{H}_{B_1}\mathbf{H}_{B_2}^\top$,

$$\mathbf{H}_{B_1}\mathbf{H}_{B_2}^\top = \begin{pmatrix} I_t & B_1 \\ 0 & I_t \end{pmatrix}\begin{pmatrix} I_t & 0 \\ B_2 & I_t \end{pmatrix} = \begin{pmatrix} I_t + B_1B_2 & B_1 \\ B_2 & I_t \end{pmatrix},$$

and $\|\mathbf{H}_{B_1}\mathbf{H}_{B_2}^\top\|_\infty$ is significantly effected by $\|B_1B_2\|_\infty$. Each entry of B_i is distributed in $[-\ell, \ell]$, where B_i is a sum of ℓ matrices. Thus, the distribution of each entry of B_1B_2 will be bell-shaped and each entry has a value nearby zero with a high probability. Thus, we expect that $\|\mathbf{H}_{B_1}\mathbf{H}_{B_2}^\top\|_\infty$'s expectation is much smaller than its upper bound, and so $\|\mathbf{H}\|_\infty$'s expectation is much smaller than 2^β. Therefore, we expect the expected bit size of h_{ij} is much smaller than its possible upper bound $(\beta - \log n)$.

Next, we present a candidate for $\mathbf{H}$ generation, which is expected to become immune to our attack rather than the original one, by slightly modifying Coron *et al.*'s suggestion.[6] According to the above analysis on the original $\mathbf{H}$ generation, the most problematic situation making low entropy is continuous product of $\mathbf{H}_{A_i}$ and $\mathbf{H}_{A_j}$, or $\mathbf{H}_{A_i}^\top$ and $\mathbf{H}_{A_j}^\top$. In our modification, we multiply $\mathbf{H}_{A_i}$ when i is odd and $\mathbf{H}_{A_i}^\top$ when i is even, instead of randomly choosing between $\mathbf{H}_{A_i}$ and $\mathbf{H}_{A_i}^\top$. Then, we always multiply different types, so that the norm of $\mathbf{H}$ and bit sizes of entries of $\mathbf{H}$ are increased relatively quickly. Table 8 shows that the modified $\mathbf{H}$ generation quickly increases bit-size of entries of $\mathbf{H}$ in comparison with the original one (Table 5). However, it seems necessary to increase β to be larger than λ, in contrast to the original suggestion in [12]. For Toy parameter (with $\beta = 80$), our implementation shows that the actual security is less than or equal to 42.59 against our attack algorithm, which is almost the same as the expected security level $\lambda = 42$.

Although we suggest a candidate for $\mathbf{H}$ generation, it is still unclear how large β should be. According to our experimental result, it would be safe to set β to be larger than 3λ for practical parameters. However, it is open to find asymptotic bound of β for secure $\mathbf{H}$ generation, or to show asymptotic weakness of our modification.

Encoding-validity Test: Zero-testing Vector vs. Zero-testing Integer: In [12], Coron *et al.* implemented a one-round N-way Diffie-Hellman key exchange protocol [3], 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.* [20] 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 Appendix E,

[6] In fact, this is proposed by one of the anonymous reviewers of CRYPTO 2014. We give the credit for this modification to him/her.

Table 8. Bit-size of entries of $\mathbf{H}$ generated by our suggestion (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	4.28	9	18.91
				$42(=\lambda)$	12.51	17	34.91
				80	26.28	32	72.91
				$84(=2\lambda)$	29.01	34	76.91
				$126(=3\lambda)$	45.48	51	118.91
				$168(=4\lambda)$	61.96	68	160.91
Small	52	540	41	41	9.82	15	31.92
				$52(=\lambda)$	13.57	19	42.92
				80	24.80	30	70.92
				$104(=2\lambda)$	36.04	41	94.92
				$130(=2.5\lambda)$	47.27	53	120.92
				$156(=3\lambda)$	58.51	64	146.92
				$208(=4\lambda)$	80.97	86	198.92
Medium	62	2085	56	$62(=\lambda)$	17.36	22	50.97
				80	26.81	32	68.97
				$124(=2\lambda)$	45.69	51	112.97
				160	59.85	65	148.97
				$186(=3\lambda)$	69.29	75	174.97

Table 9. Attack on zero-testing parameters generated by our suggestion

Inst.	λ	n	η	β	Exp($\lvert h_{ij}\rvert$)	m	d	Time*	Security against Alg. 3
Toy	42	136	1628	80	26.28	14	16	1816.45 sec	$2^{42.59}$ clock cycles

* The average running time for solving 25 problem instances

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 [30] 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 [9].

In [9] two batch FHE schemes are proposed. Attacking the first scheme, called KLYC-FHE [27], Algorithm 3 with $d = \lceil \sqrt{n} \rceil$ is asymptotically faster than the previous best Chen-Nguyen attack on the error size [8]. The public parameter in the KLYC-FHE contains

$$x_0 = q_0 \prod_{i \in [1,n]} p_i, \ \{x_j = \mathsf{CRT}_{q_0,p_1,\dots,p_n}(e_{0j}, e_{1j}Q_1, \dots, e_{nj}Q_n)\}_{j=1}^{\tau},$$

where $e_{ij} \xleftarrow{\$} (-2^\rho, 2^\rho)$ for $i \in [1,n]$. x_j can be considered as $Q \cdot \mathsf{CRT}_{q_0,p_1,\dots,p_n}(e_{j0}, e_{j1}, \dots, e_{jn})$, where $Q = \mathsf{CRT}_{q_0,p_1,\dots,p_n}(1, Q_1, \dots, Q_n)$ is unknown. Therefore, we can consider these parameters as an instance of n-MPACD. In this case, the parameter τ is very large; therefore, we have a sufficiently large number of samples for applying Algorithm 3. As mentioned previously, Algorithm 3 is faster than the Chen-Nguyen attack, with the difference of $\mathcal{O}(\sqrt{\frac{\rho \log \rho}{n}})$. In fact, the parameter n in the KLYC-FHE could be from $\mathcal{O}(1)$ to $\mathcal{O}(\lambda^3)$ and $\rho = \mathcal{O}(\lambda)$; thus, for $n = \tilde{\omega}(\lambda)$, that is, a case with a large message space, Algorithm 3 is faster than the Chen-Nguyen attack.

7 Conclusion

We introduce a new hard problem, called n-MPACD, by generalizing system parameters of integer-based schemes. Furthermore, we present an attack algorithm for n-MPACD with speeding up techniques and implement our attack algorithm for particular system parameters of multilinear maps over integers [12]. We also point out a flaw, which makes the cost of our attack algorithm for the zero-testing parameters drastically reduce, in the generation of the zero-testing parameter in [12]. Although we present a simple modification of the original method for generating zero-testing parameter and it seems to mitigate the flaw for practical parameters, we do not have complete figures about theoretical and asymptotic analysis. We leave it as an interesting open problem.

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. Part of this paper was made public through [32].

References

1. GMP: The GNU multiple precision arithmetic library ver. 5.1.3. http://gmplib.org, 2013.
2. S. Bayer and J. Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280. Springer, 2012.
3. D. Boneh and A. Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324(1):71–90, 2003.
4. A. Bostan and É. Schost. On the complexities of multipoint evaluation and interpolation. *Theoretical Computer Sciences*, 329(1-3):223–235, 2004.
5. A. Bostan and É. Schost. Polynomial evaluation and interpolation on special sets of points. *Journal of Complexity*, 21(4):420–446, 2005.
6. Z. Brakerski and G. N. Rothblum. Obfuscating conjunctions. In *CRYPTO (2) 2013*, volume 8043 of *LNCS*, pages 416–434. Springer, 2013.
7. Z. Brakerski and G. N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *TCC 2013*, volume 8349 of *LNCS*, pages 1–25. Springer, 2013.
8. Y. Chen and P. Q. Nguyen. Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. In *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 502–519. Springer, 2012.
9. J. H. Cheon, J.-S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, and A. Yun. Batch fully homomorphic encryption over the integers. In *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 315–335. Springer, 2013.
10. J. H. Cheon, S. Jarecki, and J. H. Seo. Multi-party privacy-preserving set intersection with quasi-linear complexity. http://eprint.iacr.org/2010/512, 2010.
11. J.-S. Coron, A. Joux, A. Mandal, and D. Naccache. Cryptanalysis of the RSA subgroup assumption from TCC 2005. In *PKC 2011*, volume 6571 of *LNCS*, pages 147–155. Springer, 2011.
12. J.-S. Coron, T. Lepoint, and M. Tibouchi. Practical multilinear maps over the integers. In *CRYPTO (1) 2013*, volume 8042 of *LNCS*, pages 476–493. Springer, 2013.
13. J.-S. Coron, T. Lepoint, and M. Tibouchi. Practical multilinear maps over the integers. *http://eprint.iacr.org/2013/183*, 2013 (Full version of [12]).
14. J.-S. Coron, T. Lepoint, and M. Tibouchi. Batch fully homomorphic encryption over the integers. *http://eprint.iacr.org/2013/036*, 2013 (Full version of the second part of [9]).
15. J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *CRYPTO 2011*, volume 6841 of *LNCS*, pages 487–504. Springer, 2011.
16. J.-S. Coron, D. Naccache, and M. Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 446–464. Springer, 2012.

17. A. M. Fiduccia. Polynomial evaluation via the division algorithm: the fast fourier transform revisited. *STOC 1972*, pages 88–93, 1972.
18. P.-A. Fouque, M. Tibouchi, and J.-S. Coron. Recovering private keys genrated with weak PRNGs. In *IMACC 2013*, volume 8308 of *LNCS*, pages 158–172. Springer, 2013.
19. E. S. V. Freire, D. Hofheinz, K. G. Paterson, and C. Striecks. Programmable hash functions in the multilinear setting. In *CRYPTO (1) 2013*, volume 8042 of *LNCS*, pages 513–530. Springer, 2013.
20. S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, 2013.
21. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS 2013*, pages 40–49. IEEE Computer Society, 2013.
22. S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO (2) 2013*, volume 8043 of *LNCS*, pages 479–499. Springer, 2013.
23. S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness encryption and its applications. In *STOC 2013*, pages 467–476. ACM, 2013.
24. J. Groth. Efficient zero-knowledge arguments from two-tiered homomorphic commitments. In *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 431–448. Springer, 2011.
25. S. Hohenberger, A. Sahai, and B. Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In *CRYPTO (1) 2013*, volume 8042 of *LNCS*, pages 494–512. Springer, 2013.
26. N. Howgrave-Graham. Approximate integer common divisors. In J. H. Silverman, editor, *CaLC*, volume 2146 of *LNCS*, pages 51–66. Springer, 2001.
27. J. Kim, M. S. Lee, A. Yun, and J. H. Cheon. CRT-based fully homomorphic encryption over the integers. *http://eprint.iacr.org/2013/057*, 2013 (Full version of the first part of [9]).
28. A. Langlois, D. Stehlé, and R. Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 239–256. Springer, 2014.
29. H. T. Lee and J. H. Seo. Security analysis of multilinear maps over the integers. In J. A. Garay and R. Gennaro, editors, *CRYPTO (1) 2014*, volume 8616 of *LNCS*, pages 224–240. Springer, 2014.
30. C. H. Lim and P. J. Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In *CRYPTO 1997*, pages 249–263, 1997.
31. T. Mateer. *Fast Fourier Transform Algorithms with Applications.* PhD thesis, Clemson University, 2008.
32. J. H. Seo. Faster algorithms for variants of approximate common divisors problem: Application to multilinear maps over the integers. *Memoirs of the 7th Cryptology Paper Contest, arranged by Korea government organization*, 2013.
33. V. Shoup. NTL: A library for doing number theory ver. 6.0.0. Available at http://www.shoup.net/ntl/, 2013.
34. M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 24–43. Springer, 2010.
35. D. Vergnaud. Efficient and secure generalized pattern matching via fast fourier transform. In *AFRICACRYPT 2011*, volume 6737 of *LNCS*, pages 41–58. Springer, 2011.
36. J. von zur Gathen and J. Gerhard. *Modern computer algebra (2nd ed.).* Cambridge University Press, 2003.

A Coron *et al.*'s Multilinear Maps Construction

Here, we recall the description of the CLT GES construction only. For parameter selections and the design rationale, we refer to the original paper [12].

Multilinear Maps over the Integers. The GES consists of six algorithms InstGen, samp, enc, reRand, isZero and ext, except for adding and multilinear maps. The InstGen takes the security parameter λ and the required mulitilinearity level κ as inputs and outputs a zero testing parameter $\boldsymbol{p}_{zt}$ and other system parameters params. The samp algorithm takes as an input the system parameter params and then outputs a level-zero encoding of a random message. The enc algorithm is used to encode at higher levels; that is, it takes params and a level-0 encoding as inputs, and then it outputs a level-1 encoding of the same message. The reRand algorithm is for re-randomization of level-1 encodings. The isZero algorithm takes $\mathsf{params}, \boldsymbol{p}_{zt}, u_\kappa$ as inputs, and then it determines whether u_κ is an encoding of zero or not. Lastly, the ext algorithm with $\mathsf{params}, \boldsymbol{p}_{zt}, u_\kappa$ as inputs extracts a nearly uniform bit-string deterministically.

Instance Generation: $\mathsf{InstGen}(1^\lambda, 1^\kappa) \to (\mathsf{params}, \boldsymbol{p}_{zt})$. For $i \in [1, n]$, choose $p_i \overset{\$}{\leftarrow} \{\eta\text{-bit primes}\}$ and $g_i \overset{\$}{\leftarrow} \{\alpha\text{-bit primes}\}$. For $i \in [1, n]$ and $j \in [1, \ell]$, select $a_{ij} \overset{\$}{\leftarrow} [0, g_i)$. Set $x_0 = \prod_{i=1}^n p_i$ and pick $z \overset{\$}{\leftarrow} [0, x_0)$ with $\gcd(z, x_0) = 1$. Then, define parameters $y, \{x_j\}_{j=1}^{\tau}, \{x'_j\}_{j=1}^{\ell}, \{\Pi_j\}_{j=1}^{n}$ and zero-testing parameters $\{(\boldsymbol{p}_{zt})_j\}_{j=1}^{n}$ as follows:

$$\begin{array}{rll}
 & y & = \mathsf{CRT}_{(p_i)}(\frac{r_i \cdot g_i + 1}{z}), \text{ where } r_i \overset{\$}{\leftarrow} (-2^\rho, 2^\rho). \\
\text{For } 1 \le j \le \ell, & x'_j & = \mathsf{CRT}_{(p_i)}(r'_{ij} \cdot g_i + a_{ij}), \text{ where } r'_{ij} \overset{\$}{\leftarrow} (-2^\rho, 2^\rho). \\
\text{For } 1 \le j \le n, & \Pi_j & = \mathsf{CRT}_{(p_i)}(\frac{\varpi_{ij} \cdot g_i}{z}), \text{ where } \Pi = (\varpi_{ij}) \in \mathbb{Z}^{n \times n} \text{ is defined below.} \\
\text{For } 1 \le j \le \tau, & x_j & = \mathsf{CRT}_{(p_i)}(\frac{r_{ij} \cdot g_i}{z}), \text{ where } r_{ij}\text{'s are defined below.} \\
\text{For } j \in [1, n], & (\boldsymbol{p}_{zt})_j & = \sum_{i=1}^n h_{ij} \cdot (z^\kappa \cdot g_i^{-1} \mod p_i) \cdot \prod_{i' \ne i} p_{i'} \pmod{x_0},
\end{array}$$

where z, g_i, and h_{ij} are secret information and κ is the maximum allowed level. In particular, $\mathbf{H} = (h_{ij}) \in \mathbb{Z}^{n \times n}$ is invertible in $\mathbb{Z}$ and both $\|\mathbf{H}^\top\|_\infty \le \mathbf{2}^{\boldsymbol{\beta}}$ and $\|(\mathbf{H}^{-1})^\top\|_\infty \le \mathbf{2}^{\boldsymbol{\beta}}$. Π's non diagonal entries are randomly and independently chosen from $(-2^\rho, 2^\rho)$, while the diagonal entries are randomly chosen from $(n2^\rho, n2^\rho + 2^\rho)$. The column vector of the matrix $(r_{ij}) \in \mathbb{Z}^{n \times \tau}$ are randomly and independently chosen from the half-open parallelepiped spanned by the columns of Π.

Finally, the $\mathsf{InstGen}$ algorithm chooses a seed s for a strong randomness extractor, and then outputs the zero-testing vector $\boldsymbol{p}_{zt} \in \mathbb{Z}^n$ and the other system parameters

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

Sampling Level-zero Encodings: $\mathsf{samp}(\mathsf{params}) \to c$. This algorithm chooses a random binary vector $\boldsymbol{b} = (b_j) \in \{0, 1\}^\ell$ and outputs the level-0 encoding

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

Encoding at Higher Levels[7]**:** $\mathsf{enc}(\mathsf{params}, c) \to c_k$. Given a level-0 encoding c, this algorithm outputs a level-1 encoding of the same message by just computing the product $c \cdot y \bmod x_0$.

Re-randomization[8]**:** $\mathsf{reRand}(\mathsf{params}, c) \to c'$. Given as an input a level-1 encoding c_1, the algorithm randomizes c_1 as follows:

$$c'_1 = c_1 + \sum_{j=1}^{\tau} b_j \cdot x_j + \sum_{j=1}^{n} b'_j \cdot \Pi_j \bmod x_0,$$

where $b_j \overset{\$}{\leftarrow} \{0, 1\}$ and $b'_j \overset{\$}{\leftarrow} [0, 2^\mu)$.

Adding and Multilinear Map: We can add encoded messages by adding encodings modulus x_0. For multilinear maps, given level-1 encodings u_j for $j \in [1, \kappa]$, we can simply compute

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

[7] In [12], this algorithm takes one more value k as an input, but in the algorithm description of [12] k is not used, and hence we omit it here.

[8] We omit an input variable i that is in [12] because it is not used in the algorithm description in [12].

Zero Testing:

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

Given u_κ as an input, this algorithm determines whether u_κ is a level-κ encoding of 0 or not by computing $\|u_\kappa \cdot \boldsymbol{p}_{zt} \bmod x_0\|_\infty$ with the zero-testing parameter $\boldsymbol{p}_{zt}$.

Extraction: $\mathsf{ext}(\mathsf{params},\boldsymbol{p}_{zt}, u_\kappa) = \mathsf{Extract}_s(\mathsf{msbs}_\nu(u_\kappa \cdot \boldsymbol{p}_{zt} \bmod x_0))$, where msbs_ν extracts the ν most significant bits of the result and $\mathsf{Extract}_s$ is a strong randomness extractor with the seed s from the system parameter params.

B Approximate Common Divisors and Variants

We first give the definitions of the *approximate common divisors* (ACD) problem and its weaker version, partial ACD, which were originally presented by Howgrave-Graham [26].

Definition 2 (Approximate Common Divisors) *We first define the following distribution over γ-bit integers for an integer p:*

$$\mathcal{D}_{\gamma,\rho}(p) = \{\mathsf{Choose}\ q \xleftarrow{\$} [0, 2^\gamma/p),\ r \xleftarrow{\$} (-2^\rho, 2^\rho)\ :\ \mathsf{Output}\ x = pq + r\}.$$

The (ρ, η, γ)-Approximate Common Divisors (abbreviated as ACD) problem is defined as follows. Given polynomially many samples from $\mathcal{D}_{\gamma,\rho}(p)$ for a randomly chosen η-bit odd integer p, find p.

Definition 3 (Partial Approximate Common Divisors) *We use the following distribution over γ-bit integers for an integer p:*

$$\mathcal{D}_{\gamma,\rho}(p) = \{\mathsf{Choose}\ q \xleftarrow{\$} [0, 2^\gamma/p),\ r \xleftarrow{\$} (-2^\rho, 2^\rho)\ :\ \mathsf{Output}\ x = pq + r\}.$$

The (ρ, η, γ)-Partial Approximate Common Divisors (abbreviated as PACD) problem is defined as follows. Pick a random η-bit prime p and a random 2^{λ^2}-rough integer q_0 from $[0, 2^\gamma/p)$.[9] *Given $x_0 = pq_0$ and polynomially many samples x_j from $\mathcal{D}_{\gamma,\rho}(p)$, find p.*

Coron *et al.* introduced a variant of ACD, which was utilized to argue the security of their FHE scheme [14]. Here, we give its definition because it will be used as an intermediate problem to show relations between PACD and new variants.

Definition 4 (n-decisional Partial Approximate Common Divisors) *We first define the following oracle $\mathcal{O}_{q_0,(p_i)}$, given integers $q_0, p_1, \ldots, p_n$. $\mathcal{O}_{q_0,(p_i)}$ takes as inputs a vector $(v_1, \ldots, v_n) \in \mathbb{Z}^n$ and outputs x with*

$$x = \mathsf{CRT}_{q_0,(p_i)}(q, v_1 + 2r_1, \ldots, v_n + 2r_n)$$

where $q \xleftarrow{\$} [0, q_0)$ and $r_i \xleftarrow{\$} (-2^\rho, 2^\rho)$.

[9] We state that an integer is a-rough when it does not contain prime factors smaller than a. There are literatures defining the PACD problem by choosing q_0 uniformly without any requirement about the size of factors like ours. However, one can find small factors by using integer factorization algorithms such as ECM, whose computational complexity is sub-exponential in the size of factors. Hence such the general PACD problem can be reduced to our definition.

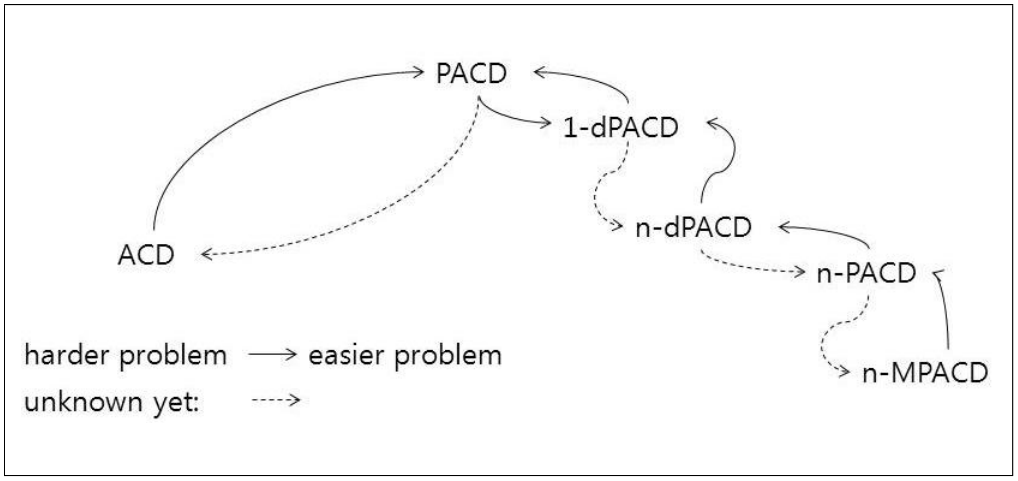

Fig. 1. Relations among ACD variants

The (ρ, η, γ, n)-decisional Partial Approximate Common Divisors (abbreviated as n-dPACD) problem is as follows. Pick random η-bit primes p_i for $i \in [1, n]$ and let π be the their product. Set $x_0 := q_0 \cdot \pi$, where q_0 is a randomly chosen 2^{λ^2}-rough integer from $[0, 2^\gamma/\pi)$, and let $\boldsymbol{w}_0 = (0, \ldots, 0)$ and choose $\boldsymbol{w}_1 \overset{\$}{\leftarrow} \{0,1\}^n$. Given x_0, $z = \mathcal{O}_{q_0,(p_i)}(\boldsymbol{w}_b)$ and oracle access to $\mathcal{O}_{q_o,(p_i)}$ for a random bit b, guess b.

To link the n-MPACD with other well-known problems, we introduce another number-theoretic problem, *n-Partial Approximate Common Divisors.*

Definition 5 (n-Partial Approximate Common Divisors) *Given integers q_0, $p_1, \ldots, p_n$, we state that x_j is sampled from the distribution $\mathcal{D}_\rho(q_0, p_1, \ldots, p_n)$ if $x_j = \mathsf{CRT}_{q_0,(p_i)}(q_j, r_{1j}, \ldots, r_{nj})$ where $q_j \overset{\$}{\leftarrow} [0, q_0)$ and $r_{ij} \overset{\$}{\leftarrow} (-2^\rho, 2^\rho)$ for all $i \in [1, n]$.*

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

Let us clarify differences between n-PACD and n-MPACD; in the distribution $\mathcal{D}_\rho^M$, we generalize by allowing non-uniform r_{ij} in contrast to $\mathcal{D}_\rho$. Furthermore, (more importantly) the unknown masking Q is multiplied in $\mathcal{D}_\rho^M$ in contrast to $\mathcal{D}_\rho$. We do not restrict Q to be chosen uniformly, but it is required to have enough entropy so that it is hard to guess Q. It looks like that these two differences make n-MPACD much harder than n-PACD.

B.1 Relations

In this subsection, we demonstrate that relations among ACD variants, in particular, a relation between PACD and new variants. We prove, in particular, that both n-PACD and n-MPACD are hard if PACD, which has already been utilized in several literatures [15, 16, 8], is hard. Relations we have are summarized in Figure 1. In the figure, a relation between problems A and B is indicated by a line or a dotted line; $\mathsf{A} \to \mathsf{B}$ means that A is a hard problem if B so. If the other side relation is unknown yet, we use $\mathsf{A} \dashleftarrow \mathsf{B}$.

By definition, it is straightforward that PACD is easier than or equal to ACD. Moreover, there exists a specialized algorithm for PACD [8] that has asymptotically lower cost than that of the known best algorithms for ACD [8, 16]. We can also easily check that n-MPACD is harder than or equal to n-PACD from their definitions, as we briefly discussed in the previous section.[10] Furthermore, one side relations between PACD and 1-dPACD and between 1-dPACD and n-dPACD are proven by Coron *et al.* [14]. We recall their results;

Theorem 1 *[14, Lemma 11] The n-dPACD problem is hard if the 1-dPACD problem is hard.*

Theorem 2 *[14, Lemma 12] The 1-dPACD problem is hard if the PACD problem is hard.*

Now we address the remaining relations between n-PACD and n-dPACD and between PACD and 1-dPACD (the other direction of Theorem 2).

Theorem 3 *The n-PACD problem is hard if the n-dPACD problem is hard.*

Proof. We consider all parameters n, ρ, and m (the number of samples) used for generating problem instances as being polynomially bounded in the security parameter. Suppose that there is an algorithm $\mathcal{A}$ for solving the n-PACD problem whose errors are chosen from $(-2^{\rho+1}, 2^{\rho+1})$. By using $\mathcal{A}$, we construct an algorithm $\mathcal{B}$ for the n-dPACD problem whose errors are distributed in $(-2^{\rho}, 2^{\rho})$. Subsequently, we show that if $\mathcal{A}$ outputs a prime factor with a non-negligible probability, then $\mathcal{B}$'s advantage for distinguishing the n-dPACD instance is also non-negligible.

$\mathcal{B}$ starts with receiving an instance of the n-dPACD problem (x_0, z) along with oracle access to $\mathcal{O}_{q_0,(p_i)}$. $\mathcal{B}$ chooses and queries polynomially many $\boldsymbol{v}_i \overset{\$}{\leftarrow} \{0,1\}^n$ to the oracle, and then receives $\mathcal{O}_{q_0,(p_i)}(\boldsymbol{v}_i)$'s. $\mathcal{B}$ sends $\mathcal{O}_{q_0,(p_i)}(\boldsymbol{v}_i)$'s along with x_0 to $\mathcal{A}$. Then, $\mathcal{A}$ outputs p. $\mathcal{B}$ checks whether p is a prime factor of x_0/q_0 or not; it can be done by computing $\gcd(p, x_0) \in (1, x_0)$ and by checking if the size of p is the same with the predetermined size. If yes, $\mathcal{B}$ outputs 1 if and only if $(z \bmod p)$ is odd. Otherwise, $\mathcal{B}$ outputs a random bit.

Let us first consider the distribution of $\mathcal{O}_{q_0,(p_i)}(\boldsymbol{v}_i)$;

$$\mathcal{O}_{q_0,(p_i)}(\boldsymbol{v}_i) = \mathsf{CRT}_{q_0,(p_i)}(q, v_1 + 2r_1, \ldots, v_n + 2r_n),$$

where $\boldsymbol{v}_i = (v_1, \ldots, v_n)$ and $r_1, \ldots, r_n \overset{\$}{\leftarrow} (-2^{\rho}, 2^{\rho})$. Because v_i is uniformly chosen from $\{0,1\}$ and r_i is uniformly chosen from $(-2^{\rho}, 2^{\rho})$, $v_i + 2r_i$ is uniformly distributed in $(-2^{\rho+1}+1, 2^{\rho+1})$ for each $i \in [1, n]$. Then, the statistical distance between the simulated distribution of $\{\mathcal{O}_{q_0,(p_i)}(\boldsymbol{v}_i)\}_{i\in[1,m]}$ and the real distribution of the instance of n-PACD is at most $\frac{nm}{2^{\rho+1}}$, which is negligible in the security parameter under our selection of polynomial n, m, and ρ.

Finally, we analyze the advantage of $\mathcal{B}$ for distinguishing b. Let E be the event that $\mathcal{A}$ outputs a prime factor of x_0/q_0. Recall $z = \mathcal{O}_{q_0,(p_i)}(\boldsymbol{w}_b)$, where $b \overset{\$}{\leftarrow} \{0,1\}$, $\boldsymbol{w}_0 = (0, \ldots, 0)$ and $\boldsymbol{w}_1 \overset{\$}{\leftarrow} \{0,1\}^n$. Note that b is completely hidden in $\mathcal{A}$'s view because z is not given to $\mathcal{A}$. The advantage of $\mathcal{B}$ is as

[10] For n-PACD, we are required to output a prime factor in contrast to n-MPACD. We note that there is a trivial reduction from finding a non-trivial factor to finding a prime factor (by repeating finding algorithm).

follows:

$$\begin{aligned}
&\Pr[\mathcal{B} \text{ outputs } 0|b=0] - \Pr[\mathcal{B} \text{ outputs } 0|b=1]\\
&= \Pr[\mathcal{B} \text{ outputs } 0 \wedge \mathsf{E}|b=0] + \Pr[\mathcal{B} \text{ outputs } 0 \wedge \neg\mathsf{E}|b=0]\\
&\quad - \Big(\Pr[\mathcal{B} \text{ outputs } 0 \wedge \mathsf{E}|b=1] + \Pr[\mathcal{B} \text{ outputs } 0 \wedge \neg\mathsf{E}|b=1]\Big)\\
&= \Pr[\mathcal{B} \text{ outputs } 0|\mathsf{E} \wedge (b=0)] \cdot \Pr[\mathsf{E}|b=0] + \Pr[\mathcal{B} \text{ outputs } 0|\neg\mathsf{E} \wedge (b=0)] \cdot \Pr[\neg\mathsf{E}|b=0]\\
&\quad - \Big(\Pr[\mathcal{B} \text{ outputs } 0|\mathsf{E} \wedge (b=1)] \cdot \Pr[\mathsf{E}|b=1] + \Pr[\mathcal{B} \text{ outputs } 0|\neg\mathsf{E} \wedge (b=1)] \cdot \Pr[\neg\mathsf{E}|b=1]\Big)\\
&= \Pr[\mathcal{B} \text{ outputs } 0|\mathsf{E} \wedge (b=0)] \cdot \Pr[\mathsf{E}] + \Pr[\mathcal{B} \text{ outputs } 0|\neg\mathsf{E} \wedge (b=0)] \cdot \Pr[\neg\mathsf{E}]\\
&\quad - \Big(\Pr[\mathcal{B} \text{ outputs } 0|\mathsf{E} \wedge (b=1)] \cdot \Pr[\mathsf{E}] + \Pr[\mathcal{B} \text{ outputs } 0|\neg\mathsf{E} \wedge (b=1)] \cdot \Pr[\neg\mathsf{E}]\Big)\\
&\geq \Pr[\mathcal{B} \text{ outputs } 0|\mathsf{E} \wedge (b=0)] \cdot (\epsilon - \tfrac{nm}{2^{\rho+1}}) + \Pr[\mathcal{B} \text{ outputs } 0|\neg\mathsf{E} \wedge (b=0)] \cdot (1-\epsilon - \tfrac{nm}{2^{\rho+1}})\\
&\quad - \Big(\Pr[\mathcal{B} \text{ outputs } 0|\mathsf{E} \wedge (b=1)] \cdot (\epsilon + \tfrac{nm}{2^{\rho+1}}) + \Pr[\mathcal{B} \text{ outputs } 0|\neg\mathsf{E} \wedge (b=1)] \cdot (1-\epsilon + \tfrac{nm}{2^{\rho+1}})\Big)\\
&= 1 \cdot (\epsilon - \tfrac{nm}{2^{\rho+1}}) + \tfrac{1}{2} \cdot (1-\epsilon - \tfrac{nm}{2^{\rho+1}}) \;-\; \Big(\tfrac{1}{2} \cdot (\epsilon + \tfrac{nm}{2^{\rho+1}}) + \tfrac{1}{2} \cdot (1-\epsilon + \tfrac{nm}{2^{\rho+1}})\Big)\\
&= \tfrac{\epsilon}{2} - \tfrac{5}{2} \cdot \tfrac{nm}{2^{\rho+1}},
\end{aligned}$$

where ϵ is $\mathcal{A}$'s success probability. Without loss of generality, we assume that $\mathcal{A}$ outputs a prime factor p_1. The third equality holds since b is completely hidden in the view of $\mathcal{A}$. When $b = 1$, w_1 could be 0 with 1/2 probability, where w_1 is the first component of $\boldsymbol{w}_1$, and hence $\Pr[\mathcal{B}$ outputs $0|\mathsf{E} \wedge (b = 1)] = 1/2$ in the fifth equation. Therefore, $\mathcal{B}$'s advantage is non-negligible if $\mathcal{A}$'s success probability is non-negligible and n, m, ρ are polynomial in the security parameter. □

Because the 1-PACD problem is the exactly same with the PACD problem, Theorem 3 implies the following corollary.

Corollary 1 *The PACD problem is hard if the* 1*-dPACD problem is hard.*

From the above results, we can conclude both n-PACD and n-MPACD are hard problems if PACD is hard.

C (Corrected) Coron-Lepoint-Tibouchi Algorithm for n-MPACD

The CLT attack for n-MPACD takes x_0 and two samples x_1 and x_2 as inputs and computes $u = x_1/x_2 \pmod{x_0}$. Then, it computes $\gcd(i_2u - i_1, x_0)$ for all possible $i_1, i_2 \in (-2^\rho, 2^\rho)$. While the naive gcd computation for all i_1 and i_2 costs $\mathcal{O}(2^{2\rho})$ operations, their algorithm reduces the attack complexity to $\mathcal{O}(\rho^2 2^\rho)$ using the meet-in-the-middle attack like the Chen-Nguyen's attack [8] with the fast polynomial algorithm provided in Table 2. More precisely, the CLT algorithm takes x_0, x_1 and x_2 and then computes

$$\gcd\Big(x_0, \prod_{i_1,i_2 \in (-2^\rho, 2^\rho)} (i_2u - i_1) \pmod{x_0}\Big)$$

by using the fast polynomial algorithms.

However, this approach is insufficient to find a non-trivial factor of x_0. The approach of computing common divisors between x_0 and a product of all candidates of multiples of a prime factor was successful to attack PACD. But, if we naively apply the same approach to n-MPACD, there is an issue we have to carefully address; because the product $\prod_{i_1,i_2 \in (-2^\rho, 2^\rho)}(i_2u - i_1) \pmod{x_0}$ consists of $(i_2u - i_1)$'s for all possible i_1 and i_2, it is equal to zero modulus p_i for all $i \in [1, n]$. Hence, the result gcd $\left(x_0, \prod_{i_1,i_2\in(-2^\rho,2^\rho)}(i_2u - i_1) \pmod{x_0}\right)$ will be a zero, not a non-trivial factor of x_0. To overcome this obstacle, we apply the divide-and-conquer approach used in our attack algorithms under the following assumption.

Assumption 3. $2^{2\rho+1} \le p_i$; *that is, the* r_{ij}*'s are sufficiently smaller than* p_i *for all* $i \in [1,n]$.
Assumption 4. The integer matrix $(r_{ij})_{\substack{i\in[1,n]\\ j\in[1,2]}} \in \mathbb{Z}^{n\times 2}$*'s rank is* 2.

Note that the matrix $(h_{ij})_{\substack{i\in[1,n]\\ j\in[1,2]}}$ in the zero-testing parameter satisfies Assumption 4. We provide the full description of the (corrected) CLT algorithm for n-MPACD in Algorithm 4. The following lemma shows that each integer in the product cannot be zero modulus x_0, so that we conclude our procedure should find a non-trivial factor before arriving at the final step.

Lemma 3 *For all* $i_1, i_2 \in (-2^\rho, 2^\rho) \setminus \{0\}$, $i_2u - i_1 \not\equiv 0 \bmod x_0$ *under Assumption 3 and 4.*

We provide the proof of Lemma 3 in Appendix D.

Complexity: The complexity of computing A mainly depends on the interval size used in the product; that is, we require $\mathcal{O}((\rho+1)^2 2^{\rho+1})$ operations modulo x_0 from $\mathsf{Alg}^{FFT}_{Poly}$ and Alg^{FFT}_{MPE}'s complexity. Similarly, for each of A's four factors, we must perform $\mathcal{O}(\rho^2 2^\rho)$ operations modulo x_0 because each A's factor uses a half size interval of I. Similarly again, we require $\mathcal{O}((\rho-1)^2 2^{\rho-1})$ 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

$$\mathcal{O}((\rho+1)^2 2^{\rho+1}) + 4\mathcal{O}(\rho^2 2^\rho) + \cdots + 4\tilde{\mathcal{O}}(2^1) = \mathcal{O}(5(\rho+1)^2 2^{\rho+1}) = \mathcal{O}(\rho^2 2^\rho) \text{ operations modulo } x_0.$$

We can also demonstrate that the space complexity is bounded by $\mathcal{O}(\rho^2 2^\rho)$ polynomially many bits from the storage complexity of Alg^{FFT}_{MPE} and $\mathsf{Alg}^{FFT}_{Poly}$.

Algorithm 4 (Corrected) CLT algorithm

```
Input: $(x_0, x_1, x_2)$
Output: a non-trivial factor of $x_0$ or $\perp$
 1: Compute $A := \prod_{i_1,i_2\in I}(i_2u - i_1) \bmod x_0$, where $I = (-2^\rho, 2^\rho)$;
    First, compute a polynomial $f(X) = \prod_{i_1\in I}(Xu - i_1) \bmod x_0$ by using $\mathsf{Alg}^{FFT}_{Poly}$.
    Second, evaluate on multipoints $\{f(i_2) \bmod x_0\}_{i_2\in I}$ by using $\mathsf{Alg}^{FFT}_{MPE}$.
    Finally, compute a product $\prod_{i_2\in I} f(i_2) \bmod x_0$, which is equal to $\prod_{i_1,i_2\in I}(i_2u - i_1) \bmod x_0$.
 2: Set $k \leftarrow 1$.
 3: while $k \le \rho + 1$ do
 4:     Compute $A_i$ for $i \in \{00, 01, 10, 11\}$.     ▷ Similarly to Step 1, where $A_i$ is defined in this paper.
 5:     Compute $\gcd(x_0, A_i)$ for $i \in \{00, 01, 10, 11\}$.
 6:     if $\gcd(x_0, A_i) \in (1, x_0)$ for some $i$ then
 7:         return $\gcd(x_0, A_i)$.
 8:     else
 9:         Choose an $A_i$ such that $\gcd(x_0, A_i) = x_0$, and then set $A \leftarrow A_i$ and $k \leftarrow k + 1$.
10:     end if
11: end while
12: return $\perp$.
```

D Proof of Lemmas

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

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

under Assumption 1.

Proof. For each p_i,

$$\sum_{j=1}^{2m} b_j x_j = \sum_{j=1}^{2m} b_j r_{ij} \pmod{p_i}$$

and the right-hand side is equal to $\sum_{j=1}^{2m} b_j r_{ij}$ as an integer because it is a sum of at most $2m$ samples and $2m2^{\rho+1} \leq p_i$ by Assumption 1. Therefore, $\sum_{j=1}^{2m} b_j x_j \pmod{p_i}$ is contained in $(-2m2^\rho, 2m2^\rho)$. In addition, we demonstrate that the range size is less than $2m2^\rho$; for the set $\{r_{ij}\}_{j\in[1,2m]}$, we set $\{pos_j\}_{j\in[1,\Delta]}$ to be the set of non-negative integers and $\{-neg_j\}_{j\in[\Delta+1,2m]}$ to be the set of negative integers, that is $\{r_{ij}\}_j = \{pos_j\}_j \cup \{-neg_j\}_j$. Then, $\sum_{j=1}^{2m} b_j x_j$ belongs to $[-\sum_{j\in[\Delta+1,2m]} neg_j, \sum_{j\in[1,\Delta]} pos_j]$, hence the range size is $\sum_{j\in[\Delta+1,2m]} neg_j + \sum_{j\in[1,\Delta]} pos_j < 2m2^\rho$. Therefore, the domain size 2^{2m} is larger than twice of the range size modulus p_i. By the pigeonhole principle, at least half of elements in the domain have a collision; that is, for each $i \in [1, n]$, there are at least $2^{2m-1} + 1$ tuples $(b'_1, \ldots, b'_{2m}) \in \{0, 1\}^{2m}$ such that there exists $(b_1, \ldots, b_{2m})$ satisfying $(b'_1, \ldots, b'_{2m}) \neq (b_1, \ldots, b_{2m})$ and $\sum_{j=1}^{2m} b_j x_j = \sum_{j=1}^{2m} b'_j x_j \bmod p_i$, hence the probability in the lemma is larger than 1/2. □

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

Proof. Suppose that $(b_1, \ldots, b_{2m}) \neq (b'_1, \ldots, b'_{2m})$. Consider

$$\sum_{j=1}^{2m} b_j x_j - \sum_{j=1}^{2m} b'_j x_j \pmod{p_i} \text{ for } i \in [1, n].$$

Then, we have the following equalities:

$$\sum_{j=1}^{2m} b_j x_j - \sum_{j=1}^{2m} b'_j x_j \pmod{p_i} = \sum_{j=1}^{2m} (b_j - b'_j) x_j \pmod{p_i} = \sum_{j=1}^{2m} (b_j - b'_j) r_{ij} \pmod{p_i} = \sum_{j=1}^{2m} (b_j - b'_j) r_{ij}$$

Let $a_j = b_j - b'_j$ for all j. Because we assume that $(b_1, \ldots, b_{2m}) \neq (b'_1, \ldots, b'_{2m})$, $(a_1, \ldots, a_{2m})$ is a non-zero vector. The last equality holds "as integers" because $2m2^{\rho+1} < p_i$ by Assumption 1. The above equalities hold for all $i \in [1, n]$ and the same vector $(a_1, \ldots, a_{2m})$; thus, we can consider the product between the $n \times 2m$ matrix $R = (r_{ij})$ and the vector $(a_1, \ldots, a_{2m})$. Because $rank(r_{ij}) = 2m$ by Assumption 2 and $(a_1, \ldots, a_{2m})$ is a non-zero vector, $R \cdot (a_1, \ldots, a_{2m})^\top$ is also a non-zero vector. Therefore, there is $i' \in [1, n]$ such that $(r_{i'1}, \ldots, r_{i'2m}) \cdot (a_1, \ldots, a_{2m})^\top \neq 0$, hence for such i', $\sum_{j=1}^{2m} b_j x_j \neq \sum_{j=1}^{2m} b'_j x_j \pmod{p_{i'}}$. □

Lemma 3 *For all* $i_1, i_2 \in (-2^\rho, 2^\rho) \setminus \{0\}$*,* $i_2 u - i_1 \not\equiv 0 \bmod x_0$ *under Assumption 3 and 4.*

Proof. Suppose that $i_2 u - i_1 \equiv 0 \bmod x_0$ for some i_1 and $i_2 \in (-2^\rho, 2^\rho)$. Then, we obtain that

$$i_2 r_{11} \equiv i_1 r_{12} \bmod p_1 \text{ and } i_2 r_{21} \equiv i_1 r_{22} \bmod p_2.$$

In fact, the above equalities hold as integers by Assumption 3; hence these can be re-written by

$$\begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix} \begin{pmatrix} i_2 \\ -i_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

but this contradicts Assumption 4. Therefore, we conclude $i_2 u \not\equiv i_1 \bmod x_0$ for all non-zero i_1, i_2. □

E Key Recovery Attack on the Multipartite Diffie-Hellman Protocol using GES

We recall the one-round N-party Diffie-Hellman key exchange protocol using the GES construction, which is secure in the common reference model under the GDDH assumption with $N = \kappa + 1$ [20, 12]; here, N parties want to share a secret key sk using a one-round protocol.

$\mathsf{Setup}(1^\lambda, 1^N)$. Run $\mathsf{InstGen}(1^\lambda, 1^\kappa) \to (\mathsf{params}, \boldsymbol{p}_{zt})$ with $\kappa = N - 1$.

$\mathsf{Publish}(\mathsf{params}, i)$. Each party i runs $\mathsf{samp}(\mathsf{params}) \to c_i$, keeps the level-0 encoding c_i as his secret key, and publishes the corresponding level-1 encoding $\mathsf{reRand}(\mathsf{params}, \mathsf{enc}(\mathsf{params}, c_i)) \to c_i'$.

$\mathsf{KeyGen}(\mathsf{params}, \boldsymbol{p}_{zt}, i, c_i, \{c_j'\}_{j \neq i})$. Each party i computes $\tilde{c}_i = c_i \cdot \prod_{j \neq i} c_j'$ and extracts the shared secret key by $\mathsf{ext}(\mathsf{params}, p_{zt}, \tilde{c}_i) \to sk$.

E.1 A Key Recovery Attack on a Single Zero-testing Parameter

In key recovery attacks, we assume that $N - 1$ parties collude to recover the secret key of one remaining party; that is, without loss of generality, we assume that the adversary controls parties $\mathcal{P}_2, \ldots, \mathcal{P}_N$ to recover $\mathcal{P}_1$'s secret key in the N-party Diffie-Hellman key exchange protocol using GES with a single integer zero-testing parameter p_{zt}. Furthermore, we assume that the adversary can participate the key exchange protocol several times (polynomially many times) with the honest user $\mathcal{P}_1$ using a fixed secret key.

The adversary begins by generating public keys c_i' of $\mathcal{P}_i$ for $i = 2, \ldots, N-1$ honestly. For the public key of $\mathcal{P}_N$, the adversary first executes $\mathsf{Publish}$ normally by running $\mathsf{samp}(\mathsf{params}) \to c_N$ and $\mathsf{reRand}(\mathsf{params}, \mathsf{enc}(\mathsf{params}, c_N)) \to c_N'$. Then, the adversary generates an invalid encoding $c_N'' = c_N' + 2^k X \bmod x_0$ and publishes c_N'' as his public key, where k is a small integer less than ν and X is a level-1 encoding of zero.[11] After publishing public keys, $\mathcal{P}_1$ computes $\tilde{c}_1 = c_1 \cdot c_N'' \cdot \prod_{j \neq 1, N} c_j'$ and extracts a shared secret key by $\mathsf{ext}(\mathsf{params}, p_{zt}, \tilde{c}_1) \to sk$. When $\mathcal{P}_1$ sends a message m to other parties using the shared secret key sk, the adversary obtains $CT = Encryption_{sk}(m)$, where *Encryption* is an encryption scheme predetermined by all parties.

[11] A level-1 encoding of zero can be obtained from the system parameters provided in reRand because the re-randomization process is just generating and adding level-1 encodings of zeros.

Then, the adversary can extract the $\nu+k$ most significant bits of $c_1X(\prod_{j\neq 1,N} c'_j)p_{zt} \bmod x_0$ by the exhaustive search; first of all, we observe the following equalities:

$$\begin{aligned} sk &= \mathsf{Extract}(\mathsf{msbs}_\nu(\tilde{c}_1 \cdot p_{zt} \bmod x_0)) \\ &= \mathsf{Extract}(\mathsf{msbs}_\nu(c_1 \cdot c''_N \cdot \prod_{j\neq 1,N} c'_j \cdot p_{zt} \bmod x_0)) \\ &= \mathsf{Extract}(\mathsf{msbs}_\nu(c_1 \cdot c'_N \cdot \prod_{j\neq 1,N} c'_j \cdot p_{zt} + c_1 \cdot 2^k X \cdot \prod_{j\neq 1,N} c'_j \cdot p_{zt} \bmod x_0)). \end{aligned}$$

In the last expression, $\mathsf{msbs}_\nu(c_1\cdot c'_N\cdot\prod_{j\neq 1,N} c'_j\cdot p_{zt} \bmod x_0) = \mathsf{msbs}_\nu(c'_1\cdot c_N\cdot\prod_{j\neq 1,N} c'_j\cdot p_{zt} \bmod x_0)$, so that this value can be computed by the adversary. For the other part, we know that $\mathsf{msbs}_\nu(c_1X\cdot(\prod_{j\neq 1,N} c'_j)\cdot p_{zt} \bmod x_0) = 0$ because $c_1X\cdot(\prod_{j\neq 1,N} c'_j) \bmod x_0$ is a valid encoding of zero. Hence, $c_1\cdot 2^kX\cdot\prod_{j\neq 1,N} c'_j\cdot p_{zt} \bmod x_0$ is a k-left shift so that its ν most significant bits has at most 2^k different values. Note that a product of a zero-encoding and p_{zt} has zeros in the ν most significant bits. Therefore, with 2^{k+1} trials (containing carry-guessing), the adversary can choose a candidate sk' of the shared secret and check whether $Decryption_{sk'}(CT)$ is a meaningful message or not. When the adversary gets a meaningful message, he can see the $\nu+k$ most significant bits of $c_1X(\prod_{j\neq 1,N} c'_j)p_{zt} \bmod x_0$, where the ν most significant bits are zeros.

Next, the adversary can update $\mathcal{P}_N$'s public key with another invalid encoding $c'_N+2^{2k}X \bmod x_0$ and, with 2^{k+1} trials, he can extract the next k bits of $c_1X(\prod_{j\neq 1,N} c'_j)p_{zt} \bmod x_0$ similarly. The adversary can repeat the same procedure so that he can extract at most the $\nu+\nu$ most significant bits of $c_1X(\prod_{j\neq 1,N} c'_j)p_{zt} \bmod x_0$, where the maximum is achieved when ν is a multiple of k. This is because the ν most significant bits of $c_1X(\prod_{j\neq 1,N} c'_j)p_{zt} \bmod x_0$ are zeros.

To obtain the remaining bits of $c_1X(\prod_{j\neq 1,N} c'_j)p_{zt} \bmod x_0$ (that is, from the $(2\nu+1)$-th most significant bit to the last significant bit of $c_1X(\prod_{j\neq 1,N} c'_j)p_{zt} \bmod x_0$), the adversary does somewhat differently; we assume that the adversary knows the 2ν most significant bits of $c_1X(\prod_{j\neq 1,N} c'_j)p_{zt} \bmod x_0$. To extract from the $(2\nu+1)$-th to the $(2\nu+k)$-th most significant bits of $c_1X(\prod_{j\neq 1,N} c'_j)p_{zt} \bmod x_0$, the adversary updates $\mathcal{P}_N$'s public key with $c'_N+2^{\nu+k}X \bmod x_0$ so that the shared secret key is

$$sk = \mathsf{Extract}(\mathsf{msbs}_\nu(c_1\cdot c'_N\cdot\prod_{j\neq 1,N} c'_j\cdot p_{zt} + c_1\cdot 2^{\nu+k}X\cdot\prod_{j\neq 1,N} c'_j\cdot p_{zt} \bmod x_0)).$$

We can easily check the following:

$$2^{\nu+k}c_1X\cdot\prod_{j\neq 1,N} c'_j\cdot p_{zt} \equiv (2^\nu c_1X\cdot\prod_{j\neq 1,N} c'_j\cdot p_{zt} \bmod x_0)\cdot 2^k = S\cdot 2^{|x_0|}+T$$

for some k-bit integer S and some $|x_0|$-bit integer T, where '$\equiv$' means the congruence modulus x_0 and ' $=$' means the equality as integers. With this notation, we know that from the $(2\nu+1)$-th to the $(2\nu+k)$-th most significant bits of $c_1X(\prod_{j\neq 1,N} c'_j)p_{zt} \bmod x_0$ are exactly the same as from the $(\nu-k+1)$-th to the ν-th most significant bits of T. Because we assume that the adversary knows the 2ν most significant bits of $c_1X(\prod_{j\neq 1,N} c'_j)p_{zt} \bmod x_0$, he knows S and the $(\nu-k)$ most significant bits of T. Therefore, the adversary guesses from the $(\nu-k+1)$-th to the ν-th most significant bits of T, and hence with considering carries, the adversary has 2^{k+2} candidates for $\mathsf{msbs}_\nu(c'_1c_N(\prod_{j\neq 1,N} c'_j)p_{zt}+S2^{|x_0|}+T \bmod x_0)$, which is equal to

$$\mathsf{msbs}_\nu(c_1c'_N(\prod_{j\neq 1,N} c'_j)p_{zt}+c_12^{\nu+k}X(\prod_{j\neq 1,N} c'_j)p_{zt} \bmod x_0).$$

Then, the adversary can verify his guess from an encryption under sk, similarly to before.

Overall, the adversary can obtain the whole bits of $c_1 X(\prod_{j\neq 1,N} c'_j)p_{zt} \bmod x_0$ with $O(|x_0|)$ public key updates, $O(|x_0|)$ $\mathbb{Z}_{x_0}$ operations and additional encryptions and decryptions, which are polynomial in the security parameter.

Finally, the adversary can compute c_1, which is a secret key of $\mathcal{P}_1$, by multiplying $(X(\prod_{j\neq 1,N} c'_j)p_{zt})^{-1}$ mod x_0 to the result.[12]

[12] x_0 is hard to factor and so we can compute the inverse of $X(\prod_{j\neq 1,N} c'_j)p_{zt}$ mod x_0. Otherwise, we can factor x_0.