Cryptanalysis of the New Multilinear Map over the Integers

Brice Minaud[1] and Pierre-Alain Fouque[1,2]

[1] Université de Rennes 1, France
[2] Institut Universitaire de France
brice.minaud@gmail.com, pierre-alain.fouque@ens.fr

Abstract. This note describes a polynomial attack on the new multilinear map over the integers presented by Coron, Lepoint and Tibouchi at Crypto 2015 (CLT15). This version is a fix of the first multilinear map over the integers presented by the same authors at Crypto 2013 (CLT13) and broken by Cheon *et al.* at Eurocrypt 2015. The attack essentially downgrades CLT15 to its original version CLT13, and leads to a full break of the multilinear map for virtually all applications. A more complete version of the paper will be made available in the coming weeks. Nevertheless the main attack is given in full details.

Keywords: Multilinear maps, graded encoding schemes.

1 Introduction

Cryptographic multilinear maps are a powerful and versatile tool to build cryptographic schemes, ranging from one-round multipartite Diffie-Hellman to witness encryption and general program obfuscation. The notion of cryptographic multilinear map was first introduced by Boneh and Silverberg in 2003, as a natural generalization of bilinear maps such as pairings on elliptic curves [BS03]. However it was not until 2013 that the first concrete instantiation over ideal lattices was realized by Garg, Gentry and Halevi [GGH13a], quickly inspiring another construction over the integers by Coron, Lepoint and Tibouchi [CLT13]. Alongside these first instantiations, a breakthrough result by Garg, Gentry, Halevi, Raykova, Sahai and Waters achieved (indistinguishability) obfuscation for all circuits from multilinear maps [GGH+13b]. From that point multilinear maps have garnered considerable interest in the cryptographic community, and a host of other applications have followed.

However this wealth of applications rests on the relatively fragile basis of only three actual constructions of multilinear maps to date: namely the original construction over ideal lattices [GGH13a], the construction over the integers [CLT13], and another recent construction over lattices [GGH15]. Moreover none of these constructions relies on standard hardness assumptions. In fact the first two constructions have since been broken for applications requiring low-level encodings of zero, including the "direct" application to one-round multipartite

Diffie-Hellman [HJ15, CHL+15]. Thus building candidate multilinear maps and assessing their security may be regarded as a work in progress, and research in this area has been very active in recent years.

Following the attack by Cheon *et al.* on the [CLT13] multilinear map over the integers, several attempts to repair the scheme were published on ePrint, which hinged on hiding encodings of zero in some way; however these attempts were quickly proven insecure [CGH+15]. At CRYPTO 2015, Coron, Lepoint and Tibouchi set out to repair their scheme by following a different route [CLT15]: they essentially retained the structure of encodings from [CLT13], but added a new type of noise designed to thwart Cheon *et al.*'s approach. Their construction was thus able to retain the attractive features of the original, namely conceptual simplicity, relative efficiency, and wide range of presumed hard problems on which applications could be built.

1.1 Our contribution

In this paper we propose a polynomial attack on the new multilinear map over the integers presented by Coron, Lepoint and Tibouchi at CRYPTO 2015 [CLT15]. The attack operates by computing the secret parameter x_0, and from there all other secret parameters can be recovered via (a close variant of) Cheon *et al.*'s attack [CHL+15]. In the optimized version of the scheme where an exact multiple of x_0 is provided in the public parameters, the attack recovers x_0 instantly. In the more general non-optimized version of the scheme, the practical complexity of our polynomial attack is very close to the security parameters for the concrete instances implemented in [CLT15], *e.g.* 2^{81} for the 80-bit instance.

Moreover the attack applies to virtually all possible applications of the CLT15 multilinear map. Indeed, while it does require low-level encodings of zero, these encodings are provided by the ladders given in the public parameters. In this respect CLT15 is weaker than CLT13.

Our attacks have been verified on the reference implementation of CLT15.

An upcoming complete version of this paper will also include a probabilistic variant of the attack, which avoids a costly determinant computation. Instead the attack relies on finding and exploiting divisors of the secret parameter v_0. While it is conceptually less simple than our main attack, the probabilistic variant offers a lower practical complexity.

1.2 Overview of the Attack

We begin by briefly recalling the CLT15 multilinear map (more precisely, graded encoding scheme). The message space is $\mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$ for some small primes $g_1, \ldots, g_n$, and $(m_1, \ldots, m_n)$ is encoded at some level $k \leq \kappa$ as:

$$\mathsf{CRT}_{(p_i)}\Big(\frac{r_i g_i + m_i}{z^k}\Big) + a x_0$$

where:

(p_i) is a sequence of n large primes.
$x_0 = \prod p_i$.
$\mathsf{CRT}_{(p_i)}(x_i)$ is the unique integer in $(-x_0/2, x_0/2]$ congruent to x_i modulo p_i.
z is a fixed secret integer modulo x_0.
r_i is a small noise.
a is another noise.

Encodings at the same level can be added together, and the resulting encoding encodes the sum of the messages. Similarly encodings at levels i and j can be multiplied to yield an encoding at level $i + j$ of the coordinate-wise product of the encoded messages. This behavior holds as long as the values $r_i g_i + m_i$ do not go over p_i, *i.e.* reduction modulo p_i does not interfere. In order to prevent the size of encodings from increasing as a result of additions and multiplications, a *ladder* of encodings of zero of increasing size is published at each level. Encodings can then be reduced by subtracting elements of the ladder at the same level.

The power of the multilinear map comes from the zero-testing procedure, which allows users to test whether an encoding at the maximal level κ encodes zero. This is achieved by publishing a so-called zero-testing parameter denoted $\boldsymbol{p}_{zt} \in \mathbb{Z}$, together with a large prime $N \gg x_0$. An encoding at the maximal level κ may be written as:

$$e = \sum (r_i + m_i g_i^{-1} \bmod p_i) u_i + a x_0$$
$$\text{where } u_i \triangleq \left(g_i z^{-\kappa} (p_i^*)^{-1} \bmod p_i\right) p_i^* \quad \text{with } p_i^* = \prod_{j \neq i} p_j.$$

That is, some constants independent of the encoding have been folded with the CRT coefficients into u_i. Now $\boldsymbol{p}_{zt}$ is chosen such that $v_i \triangleq u_i \boldsymbol{p}_{zt} \bmod N$ and $v_0 \triangleq x_0 \boldsymbol{p}_{zt} \bmod N$ satisfy $\mathrm{abs}(v_i) \ll N$ and $\mathrm{abs}(v_0) \ll N$. In this way, for any encoding e of zero at level κ, since $m_i = 0$, we have:

$$\mathrm{abs}(e \boldsymbol{p}_{zt} \bmod N) = \mathrm{abs}\Big(\sum r_i v_i + a v_0\Big) \ll N$$

provided the noises r_i and a are small enough. Thus, users can test whether e is an encoding of zero at level κ by checking whether $\mathrm{abs}(e \boldsymbol{p}_{zt} \bmod N) \ll N$.

Integer Extraction. Our attack proceeds in two steps. As a first step, we define the integer extraction procedure $\phi : \mathbb{Z} \to \mathbb{Z}$. In short, ϕ computes $\sum_i r_i v_i + a v_0$ over the integers for any level-κ encoding e (of size up to the largest ladder element). Note that this value is viewed over the integers and not modulo N. If e is "small", then $\phi(e) = e \boldsymbol{p}_{zt} \bmod N$, *i.e.* ϕ matches the computation from the zero-testing procedure.

If e is "large" on the other hand, then e would need to be reduced by the ladder before zero-testing can be applied. However the crucial observation is that ϕ is $\mathbb{Z}$-linear as long as the values $r_i g_i + m_i$ associated with each encoding do not go over p_i. Thus e can be ladder-reduced into e', then $\phi(e') = e' \boldsymbol{p}_{zt} \bmod N$

is known, and $\phi(e)$ can be recovered from $\phi(e')$ by compensating the ladder reduction using $\mathbb{Z}$-linearity. In a nutshell, ϕ allows us to ignore ladder reductions in equations appearing in the rest of the attack.

Recovering $\boldsymbol{x_0}$. In the optimized variant of the scheme implemented in [CLT15], a small multiple qx_0 of x_0 is given in the public parameters. In that case qx_0 may be regarded as an encoding of zero at level κ, and $\phi(qx_0) = qv_0$. Since this holds over the integers, we can compute $q = \gcd(qx_0, qv_0)$ and then $x_0 = qx_0/q$.

In the general case where no exact multiple of x_0 is given in the public parameters, pick $n+1$ encodings a_i at some level t, and $n+1$ encodings of zero b_i at level $\kappa - t$. Note that ladder elements provide encodings of zero even if the scheme itself does not. Then compute:

$$\omega_{i,j} \triangleq \phi(a_i b_j).$$

If we write $a_i \bmod v_0 = \mathsf{CRT}_{(p_j)}(a_{i,j}/z^t)$ and $b_i \bmod v_0 = \mathsf{CRT}_{(p_j)}(r_{i,j}g_j/z^{\kappa-t})$, then we get:

$$\omega_{i,j} \bmod v_0 = \sum_k a_{i,k} r_{j,k} v_k \bmod v_0.$$

Similar to Cheon *et al.*'s attack on the CLT13 multilinear map, this equality can be viewed as a matrix product. Indeed, let Ω denote the $(n+1)\times(n+1)$ integer matrix with entries $\omega_{i,j}$, let A denote the $(n+1)\times n$ integer matrix with entries $a_{i,j}$, let R denote the $(n+1)\times n$ integer matrix with entries $r_{i,j}$, and finally let V denote the $n \times n$ diagonal matrix with diagonal entries v_i. If we embed everything into $\mathbb{Z}/v_0\mathbb{Z}$, then we have:

$$\Omega = A \cdot V \cdot R^{\mathrm{T}} \qquad \text{in } \mathbb{Z}/v_0\mathbb{Z}.$$

Since A and R are $(n+1)\times n$ matrices, this implies that Ω is not full-rank when embedded into $\mathbb{Z}/v_0\mathbb{Z}$. As a consequence v_0 divides $\det(\Omega)$. We can repeat this process with different choices of the families (a_i), (b_i) to build another matrix Ω' with the same property. Finally we recover v_0 as $v_0 = \gcd(\det(\Omega), \det(\Omega'))$, and $x_0 = v_0/\boldsymbol{p}_{zt} \bmod N$.

Recovering other secret parameters. Once x_0 is known, Cheon *et al.*'s attack can be applied by taking all values modulo v_0, and every remaining secret parameter is recovered, fully breaking the scheme.

1.3 Impact of the Attack

Two variants of the CLT15 multilinear map should be considered. Either a small multiple of x_0 is provided in the public parameters. In that case x_0 can be recovered instantly, and the scheme becomes equivalent to CLT13 in terms of security (cf. Section 5.1). In particular it falls victim to Cheon *et al.*'s attack when low-level encodings of zero are present, but it may still be secure for applications that do not require such encodings, such as obfuscation. It is interesting to note that Cheon *et al.*'s attack is very efficient since all computations can be performed modulo a small prime as the outputs are small integers. However the scheme is strictly less efficient than CLT13 by construction, so there is no point in using CLT15 for those applications.

Otherwise, if no small multiple of x_0 is given out in the public parameters, then ladders of encodings of zero must be provided at levels below the maximal level. Thus we have access to numerous encodings of zero below the maximal level, even if the particular application of multilinear maps under consideration does not require them. As a result our determinant-based attack is applicable (cf. Section 5.4), and we still recover x_0 in polynomial time, albeit less efficiently than the previous case. Moreover once x_0 is recovered, encodings of zero provided by the ladder enable Cheon *et al.*'s attack, and every secret parameter is recovered.

In summary, the optimized version of CLT15 providing a small multiple of x_0 is no more secure than CLT13, and less efficient. On the other hand in the general non-optimized case, the scheme is broken for virtually all possible applications due to encodings of zero provided by the ladder. Thus overall the CLT15 scheme can be considered fully broken.

1.4 Organization of the Paper

For the sake of being self-contained, in Section 3, we present multilinear maps, graded encoding schemes, as well as the CLT15 construction. In Section 4 we recall Cheon *et al.*'s attack on CLT13 since it serves as a follow-up to our attack once x_0 is recovered, and shares similar ideas. Readers already familiar with the CLT15 multilinear map can skip straight to Section 5 where we describe our main attack.

2 Notation

The symbol $\triangleq$ denotes an equality by definition. For n an integer, $|n|$ is the size of n in bits. To avoid confusion, we write $\mathrm{abs}(n)$ for the absolute value of n.

Modular arithmetic. The group of integers modulo n, $\mathbb{Z}/n\mathbb{Z}$, is denoted $\mathbb{Z}_n$. The notation "mod p" should be understood as having the lowest priority. For instance, in the expression $(a/b) \cdot c \bmod p$, the division a/b should be computed modulo p.

Moreover we always view $a \bmod p$ as an integer in $\mathbb{Z}$. The representative closest to zero is always chosen, positive in case of tie. In other words $-p/2 < a \bmod p \leq p/2$.

Chinese Remainder Theorem. Given n prime numbers (p_i), we define p_i^* as in [Hal15a]:

$$p_i^* = \prod_{j \neq i} p_j.$$

Moreover, for $(x_1, \ldots, x_n) \in \mathbb{Z}^n$ let:

$$\mathsf{CRT}_{(p_i)}(x_i) \triangleq \sum_i \left(x_i (p_i^*)^{-1} \bmod p_i\right) p_i^* \bmod \prod_i p_i.$$

That is, $\mathsf{CRT}_{(p_i)}(x_i)$ is (a representative of) the unique integer modulo $\prod p_i$ such that $\mathsf{CRT}_{(p_i)}(x_i) \bmod p_i = x_i \bmod p_i$, as per the Chinese Remainder Theorem.

It is useful to observe that for any $(x_1, \ldots, x_n) \in \mathbb{Z}^n$:

$$\mathsf{CRT}_{(p_i)}(x_i p_i^*) = \sum_i x_i p_i^* \bmod \prod_i p_i. \tag{1}$$

3 Presentation of the CLT15 Multilinear Map

3.1 Multilinear Maps and Graded Encoding Schemes

In this section we give a brief introduction to multilinear maps to make our article self-contained. In particular we only consider symmetric multilinear maps. We refer the interested reader to [GGH13a, Hal15b] for a more thorough presentation.

Cryptographic multilinear maps were introduced by Boneh and Silverberg [BS03], as a natural generalization of bilinear maps stemming from pairings on elliptic curves, which had found striking new applications in cryptography [Jou00, BF01, ...]. A (symmetric) multilinear map is defined as follows.

Definition 1 (Multilinear Map [BS03]). *Given two groups $\mathbb{G}, \mathbb{G}_T$ of the same prime order, a map $e : \mathbb{G}^\kappa \to \mathbb{G}_T$ is a κ-*multilinear map *iff it satisfies the following two properties:*

1. *for all $a_1, \ldots, a_\kappa \in \mathbb{Z}$ and $x_1, \ldots, x_\kappa \in \mathbb{G}$,*

$$e(x_1^{a_1}, \ldots, x_\kappa^{a_\kappa}) = e(x_1, \ldots, x_\kappa)^{a_1 \cdots a_\kappa}$$

2. *if g is a generator of $\mathbb{G}$, then $e(g, \ldots, g)$ is a generator of $\mathbb{G}_T$.*

A natural special case are *leveled* multilinear maps:

Definition 2 (Leveled Multilinear Map [HSW13]). *Given $\kappa + 1$ groups $\mathbb{G}_1, \ldots, \mathbb{G}_\kappa, \mathbb{G}_T$ of the same prime order, and for each $i \leq \kappa$, a generator $g_i \in \mathbb{G}_i$, a κ-*leveled multilinear map *is a set of bilinear maps $\{e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \to G_{i+j} | i, j, i + j \leq \kappa\}$ such that for all i, j with $i + j \leq \kappa$, and all $a, b \in \mathbb{Z}$:*

$$e_{i,j}(g_i^a, g_j^b) = g_{i,j}^{ab}.$$

Similar to public-key encryption [DH76] and identity-based cryptosystems [Sha85], multilinear maps were originally introduced as a compelling target for cryptographic research, without a concrete instantiation [BS03]. The first multilinear map was built ten years later in the breakthrough construction of Garg, Gentry and Halevi [GGH13a]. More accurately, what the authors proposed was a *graded encoding scheme*, and to this day all known cryptographic multilinear maps constructions are actually variants of graded encoding schemes [Hal15b]. For this reason, and because both constructions have similar expressive power, the term "multilinear map" is used in the literature in place of "graded encoding scheme", and we will follow suit in the rest of this article.

Graded encoding schemes are a relaxed definition of leveled multilinear map, where elements x_i^a for $x_i \in \mathbb{G}_i, a \in \mathbb{Z}$ are no longer required to lie in a group. Instead, they are regarded as "encodings" of a ring element a at level i, with no assumption about the underlying structure. Formally, encodings are thus defined as general binary strings in $\{0,1\}^*$. In the following definition, $S_i^{(\alpha)}$ should be regarded as the set of encodings of a ring element α at level i.

Definition 3 (Graded Encoding System [GGH13a]). *A* κ-graded encoding system *consists of a ring* R *and a system of sets* $\mathcal{S} = \{S_i^{(\alpha)} \subset \{0,1\}^* | \alpha \in R, 0 \leq i \leq \kappa\}$*, with the following properties:*

1. *For each fixed* i*, the sets* $S_i^{(\alpha)}$ *are pairwise disjoint as* α *spans* R*.*
2. *There is an associative binary operation '+' and a self-inverse unary operation '−' on* $\{0,1\}^*$ *such that for every* $\alpha_1, \alpha_2 \in R$*, every* $i \leq \kappa$*, and every* $u_1 \in S_i^{(\alpha_1)}, u_2 \in S_i^{(\alpha_2)}$*, it holds that:*
$$u_1 + u_2 \in S_i^{(\alpha_1+\alpha_2)} \quad \textit{and} \quad -u_1 \in S_i^{(-\alpha_1)}$$
where $\alpha_1 + \alpha_2$ *and* $-\alpha_1$ *are addition and negation in* R*.*
3. *There is an associative binary operation '×' on* $\{0,1\}^*$ *such that for every* $\alpha_1, \alpha_2 \in R$*, every* $i_1, i_2 \in \mathbb{N}$ *such that* $i_1 + i_2 \leq \kappa$*, and every* $u_1 \in S_{i_1}^{(\alpha_1)}, u_2 \in S_{i_2}^{(\alpha_2)}$*, it holds that* $u_1 \times u_2 \in S_{i_1+i_2}^{(\alpha_1 \cdot \alpha_2)}$*. Here* $\alpha_1 \cdot \alpha_2$ *is the multiplication in* R*, and* $i_1 + i_2$ *is the integer addition.*

Observe that a leveled multilinear map is a graded encoding system where $R = \mathbb{Z}$ and, with the notation from the definitions, $S_i^{(\alpha)}$ contains the single element g_i^α. Also note that the behavior of addition and multiplication of encodings with respect to the levels i is the same as that of a graded ring, hence the *graded* qualifier.

All known constructions of graded encoding schemes do not fully realize the previous definition, insofar as they are "noisy"[3]. That is, all encodings have a certain amount of noise; each operation, and especially multiplication, increases this noise; and the correctness of the scheme breaks down if the noise goes above a certain threshold. The situation in this regard is similar to somewhat homomorphic encryption schemes.

3.2 Multilinear Map Procedures

The exact interface offered by a multilinear map, and called upon when it is used as a primitive in a cryptographic scheme, varies depending on the scheme.

[3] In fact the question of achieving the functionality of multilinear maps without noise may be regarded as an important open problem [Zim15].

However the core elements are the same. Below we reproduce the procedures for manipulating encodings defined in [CLT15], which are a slight variation of [GGH13a].

In a nutshell, the scheme relies on a trusted third party that generates the instance (and is usually no longer needed afterwards). Users of the instance (that is, everyone but the generating trusted third party) cannot encode nor decode arbitrary encodings: they can only combine existing encodings using addition, negation and multiplication, and subject to the limitation that the level of an encoding cannot exceed κ. The power of the multilinear map comes from the zero-testing (resp. extraction) procedure, which allows users to test whether an encoding at level κ encodes zero (resp. roughly get a λ-bit "hash" of the value encoded by a level-κ encoding).

Here users are also given access to random level-0 encodings, and have the ability to re-randomize encodings, as well as promote any encoding to a higher-level encoding of the same element. These last functionalities are tailored towards the application of multilinear maps to one-round multi-party Diffie-Hellman. In general different applications of multilinear map require different subsets of the procedures below, and sometimes variants of them.

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

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

$\mathsf{enc}(\mathsf{pp}, i, u)$: the possibly randomized encoding procedure takes as input the public parameters pp, a level $i \leq \kappa$, and a level-0 encoding $u \in S_0^\alpha$ for some $\alpha \in R$, and outputs a level-i encoding $u' \in S_i^{(\alpha)}$.

$\mathsf{reRand}(\mathsf{pp}, i, u)$: the randomized rerandomization procedure takes as input the public parameters pp, a level $i \leq \kappa$, and a level-i encoding $u \in S_i^\alpha$ for some $\alpha \in R$, and outputs another level-i encoding $u' \in S_i^{(\alpha)}$ of the same α, such that for any $u_1, u_2 \in S_i^{(\alpha)}$, the output distributions of $\mathsf{reRand}(\mathsf{pp}, i, u_1)$ and $\mathsf{reRand}(\mathsf{pp}, i, u_2)$ are nearly the same.

$\mathsf{neg}(\mathsf{pp}, u)$: the negation procedure is deterministic and that takes as input the public parameters pp, and a level-i encoding $u \in S_i^{(\alpha)}$ for some $\alpha \in R$, and outputs a level-i encoding $u' \in S_i^{(-\alpha)}$.

$\mathsf{add}(\mathsf{pp}, u_1, u_2)$: the addition procedure is deterministic and takes as input the public parameters pp, two level-i encodings $u_1 \in S_i^{(\alpha_1)}, u_2 \in S_i^{(\alpha_2)}$ for some $\alpha_1, \alpha_2 \in R$, and outputs a level-i encoding $u' \in S_i^{(\alpha_1+\alpha_2)}$.

$\mathsf{mult}(\mathsf{pp}, u_1, u_2)$: the multiplication procedure is deterministic and takes as input the public parameters pp, two encodings $u_1 \in S_i^{(\alpha_1)}, u_2 \in S_j^{(\alpha_2)}$ of some $\alpha_1, \alpha_2 \in R$ at levels i and j such that $i + j \leq \kappa$, and outputs a level-$(i + j)$ encoding $u' \in S_{i+j}^{(\alpha_1 \cdot \alpha_2)}$.

$\mathsf{isZero}(\mathsf{pp}, u)$: the zero-testing procedure is deterministic and takes as input the public parameters pp, and an encoding $u \in S_\kappa^{(\alpha)}$ of some $\alpha \in R$ at the maximum level κ, and outputs 1 if $\alpha = 0$, 0 otherwise, with negligible probability of error (over the choice of $u \in S_\kappa^{(\alpha)}$).

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

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

3.3 The CLT15 Multilinear Map over the Integers

Shortly after the multilinear map over ideal lattices by Garg, Gentry and Halevi [GGH13a], another construction over the integers was proposed by Coron, Lepoint and Tibouchi [CLT13]. However a devastating attack was published by Cheon, Han, Lee, Ryu and Stehlé at Eurocrypt 2015 (on ePrint in late 2014). In the wake of this attack, a revised version of their multilinear map over the integers was presented by Coron, Lepoint and Tibouchi at Crypto 2015 [CLT15]. In the remainder of this article, we will refer to the original construction over the integers as the CLT13 multilinear map, and to the new version from Crypto 2015 as the CLT15 multilinear map.

In this section we recall the CLT15 construction. Once again we omit aspects of the construction that are not relevant to our attack, and refer the reader to [CLT15] for more details. The message space is $R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$, for some (relatively small) primes $g_i \in \mathbb{N}$. An encoding of a message $(m_1, \ldots, m_n) \in \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$ at level $k \leq \kappa$ has the following form:

$$e = \mathsf{CRT}_{(p_i)}\Big(\frac{r_i g_i + m_i}{z^k} \bmod p_i\Big) + a x_0 \tag{2}$$

where:

- The p_i's are n large secret primes.
- The r_i's are random noise such that $\mathrm{abs}(r_i g_i + m_i) \ll p_i$.
- $x_0 = \prod_{i \leq n} p_i$.
- z is a fixed secret integer modulo x_0.
- a is random noise.

The scheme relies on the following parameters:

$$\begin{aligned}
\lambda &: \text{the security parameter.}\\
\kappa &: \text{the multilinearity level.}\\
n &: \text{the number of primes } p_i.\\
\eta &: \text{the bit length of secret primes } p_i.\\
\gamma = n\eta &: \text{the bit length of } x_0.\\
\rho &: \text{the bit length of the } g_i\text{'s and initial } r_i\text{'s.}
\end{aligned}$$

Addition, negation and multiplication of encodings is exactly addition, negation and multiplication over the integers. Indeed, m_i is recovered from e as $m_i = (e \bmod p_i) \bmod g_i$, and as long as $r_i g_i + m_i$ does not go over p_i, addition and multiplication will go through both moduli. Thus we have defined encodings and how to operate on them.

Regarding the sampling procedure from Section 3.2, for our purpose, it suffices to know that it is realized by publishing a large number of level-0 encodings of random elements. Users can then sample a new random element as a subset sum of published elements. Likewise, the rerandomization procedure is achieved by publishing a large number of encodings of zero at each level, and an element is re-randomized by adding a random subset sum of encodings of zero at the same level. The encoding procedure is realized by publishing a single level-1 encoding y of 1 (by which we mean $(1, \ldots, 1) \in \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$): any encoding can then be promoted to an encoding of the same element at a higher level by multiplying by y.

Zero-testing in CLT13. We now move on to the crucial zero-testing procedure. This is where CLT13 and CLT15 differ. We begin by briefly recalling the CLT13 approach.

In CLT13, the product x_0 of the p_i's is public. In particular, every encoding can be reduced modulo x_0, and every value below should be regarded as being modulo x_0. Let $p_i^* = \prod_{j \neq i} p_j$. Using (1), define:

$$\boldsymbol{p}_{zt} \triangleq \sum_{i \leq n} \Big(\frac{h_i z^\kappa}{g_i} \bmod p_i \Big) p_i^* = \mathsf{CRT}_{(p_i)} \Big(\frac{h_i z^\kappa}{g_i} p_i^* \bmod p_i \Big) \qquad \bmod x_0.$$

where the h_i's are some relatively small numbers such that $\operatorname{abs}(h_i) \ll p_i$. Now take a level-$\kappa$ encoding of zero:

$$e = \mathsf{CRT}_{(p_i)} \Big(\frac{r_i g_i}{z^\kappa} \bmod p_i \Big) \qquad \bmod x_0.$$

Since multiplication acts coordinate-wise on the CRT components, using (1) again, we have:

$$\omega \triangleq e \boldsymbol{p}_{zt} = \mathsf{CRT}_{(p_i)}(h_i r_i p_i^*) = \sum_i h_i r_i p_i^* \qquad \bmod x_0.$$

Since $p_i^* = x_0 / p_i$, as long as we set our parameters so that $\operatorname{abs}(h_i r_i) \ll p_i$, we have $\operatorname{abs}(\omega) \ll x_0$.

Thus the zero-testing procedure is as follows: for a level-κ encoding e, compute $\omega = e \boldsymbol{p}_{zt} \bmod x_0$. Output 1, meaning we expect e to encode zero, iff the ν most significant bits of ω are zero, for an appropriately chosen ν. In [CLT13], multiple $\boldsymbol{p}_{zt}$'s can be defined in order to avoid false positives; we restrict our attention to a single $\boldsymbol{p}_{zt}$.

Zero-testing in CLT15. In CLT13, an encoding at some fixed level is entirely defined by its vector of associated values $c_i = r_i g_i + m_i$. Moreover, addition and multiplication of encodings act coordinate-wise on these values, and the value of the encoding itself is $\mathbb{Z}_{x_0}$-linear as a function of these values. Likewise, ω is $\mathbb{Z}_{x_0}$-linear as a function of the r_i's. This nice structure is an essential part of what makes the devastating attack by Cheon *et al.* [CHL+15] possible. In CLT15, the authors set out to break this structure by introducing a new noise component a.

For this purpose, the public parameters include a new prime number $N \gg x_0$, with $|N| = \gamma + 2\eta + 1$. Meanwhile x_0 is kept secret, and no longer part of the public parameters. Encodings are thus no longer reduced modulo x_0, and take the general form given in (3), including a new noise value a. Equivalently, we can write an encoding e of (m_i) at level k as:

$$\begin{aligned} e = \sum_i \left(r_i + m_i(g_i^{-1} \bmod p_i)\right) u_i + a x_0 \\ \text{with } u_i \triangleq \left(g_i z^{-k} (p_i^*)^{-1} \bmod p_i\right) p_i^*. \end{aligned} \tag{3}$$

That is, we fold the $g_i z^{-k}$ multiplier of r_i with the CRT coefficient into u_i.

The zero-testing parameter $\boldsymbol{p}_{zt}$ is now defined modulo N in such a way that:

$$\begin{array}{lll} & v_0 \triangleq x_0 \boldsymbol{p}_{zt} \bmod N & \forall i, v_i \triangleq u_i \boldsymbol{p}_{zt} \bmod N \\ \text{satisfy:} & \mathrm{abs}(v_0) \ll N & \mathrm{abs}(v_i) \ll N \end{array} \tag{4}$$

To give an idea of the sizes involved, $|v_0| \approx \gamma$ and $|v_i| \approx \gamma + \eta$ for $i > 0$. We refer the reader to [CLT15] for how to build such a $\boldsymbol{p}_{zt}$. The point is that if e is an encoding of zero at level κ, then we have:

$$\omega = e\boldsymbol{p}_{zt} \bmod N = \sum r_i v_i + a v_0 \bmod N.$$

In order for this quantity to be smaller than N, the size of a must be somehow controlled. Conversely as long as a is small enough and the noise satisfies $\mathrm{abs}(r_i) \ll p_i$ then $\mathrm{abs}(\omega) \ll N$. We refer the reader to [CLT15] for an exact choice of parameters.

Thus the size of a must be controlled. The term $a x_0$ will be dominant in (3) in terms of size, so decreasing a is the same as decreasing the size of the encoding as a whole. The scheme requires a way to achieve this without altering the encoded value (and without publishing x_0).

For this purpose, inspired by [VDGHV10], a *ladder* $(X_i^{(k)})_{i \le \ell}$ of encodings of zero of increasing size is published for each level $k \le \kappa$. The size of an encoding e at level k can then be reduced without altering the encoded value by recursively subtracting from e the largest ladder element smaller than e, until e is smaller than X_0. More precisely we can choose X_0 small enough that the previous zero-testing procedure goes through, and then choose X_ℓ twice the size of X_0, so that the product of any two encodings smaller than X_0 can be reduced to an encoding smaller than X_0. After each addition and multiplication, the size of the resulting encoding is reduced via the ladder.

In the end, the zero-testing procedure is very similar to CLT13: given a (ladder-reduced) level-κ encoding e, compute $\omega = e\boldsymbol{p}_{zt} \bmod N$. Then output 1, meaning we expect e to encode zero, iff the ν high-order bits of ω are zero.

Extraction. The extraction procedure simply outputs the ν high-order bits of ω, computed as above. For both CLT13 and CLT15, it can be checked that they only depend on the m_i's (as opposed to the noises a and the r_i's).

4 Cheon *et al.*'s Attack on CLT13

In this section we provide a short description of Cheon *et al.*'s attack on CLT13 [CHL+15], as elements of this attack appear in our own. We actually present (a close variant of) the slightly simpler version in [CGH+15].

Assume we have access to a level-0 encoding a of some random value, n level-1 encodings (b_i) of zero, and a level-1 encoding y of 1. This is the case for one-round multi-party Diffie-Hellman (see previous section). Let $a_i = a \bmod p_i$, *i.e.* a_i is the i-th value "$r_i g_i + m_i$" associated with a. For $i \le n$, define $r_{i,j} = b_i z/g_j \bmod p_j$, *i.e.* $r_{i,j}$ is the j-th value "r_j" associated with b_i (recall that b_i is an encoding of zero, so $m_j = 0$). Finally let $y_k = yz \bmod p_k$.

Now compute:

$$\begin{aligned} e_{i,j} &= a \cdot b_i \cdot b_j \cdot y^{\kappa-2} \bmod x_0 & \qquad \omega_{i,j} &= e_{i,j}\boldsymbol{p}_{zt} \bmod x_0 \\ e'_{i,j} &= \phantom{a \cdot{}} b_i \cdot b_j \cdot y^{\kappa-2} \bmod x_0 & \qquad \omega'_{i,j} &= e'_{i,j}\boldsymbol{p}_{zt} \bmod x_0 \end{aligned}$$

Note that:

$$\begin{aligned} \omega_{i,j} &= \sum_k \Big(a_k \frac{r_{i,k} g_k}{z} \frac{r_{j,k} g_k}{z} \frac{y_k^{\kappa-2}}{z^{\kappa-2}} \frac{h_k z^\kappa}{g_k} \bmod p_k \Big) p_k^* \\ &= \sum_k a_k r_{i,k} r_{j,k} c_k \qquad \text{with } c_k = g_k y_k^{\kappa-2} h_k p_k^*. \end{aligned} \tag{5}$$

Crucially, in the second line, the modulo p_k disappears and the equation holds over the integers, because $e_{i,j}$ is a valid encoding of zero, so the correctness of the scheme requires $\mathrm{abs}(e_{i,j} z^\kappa / g_k \bmod p_k) \ll p_k$.

Equation (5) may be seen as a matrix multiplication. Indeed, define Ω, resp. Ω', as the $n \times n$ matrix with entries $\omega_{i,j}$, resp. $\omega'_{i,j}$, and likewise R with entries $r_{i,j}$. Moreover let A, resp. C, be the diagonal matrix with diagonal entries a_i, resp. c_i. Then (5) may be rewritten:

$$\begin{aligned} \Omega &= R \cdot A \cdot C \cdot R^{\mathrm{T}} \\ \Omega' &= R \cdot C \cdot R^{\mathrm{T}} \\ \Omega \cdot (\Omega')^{-1} &= R \cdot A \cdot R^{-1}. \end{aligned}$$

Here matrices are viewed over $\mathbb{Q}$ for inversion (they are invertible whp).

Once $\Omega \cdot (\Omega')^{-1}$ has been computed, the (diagonal) entries of A can be recovered as its eigenvalues. In practice this can be achieved by computing the characteristic polynomial, and all computations can be performed modulo some prime p larger than the a_i's (which are size 2ρ).

Thus we recover the a_i's, and by definition $a_i = a \bmod p_i$, so p_i can be recovered as $p_i = \gcd(a - a_i, x_0)$. From there it is trivial to recover all other secret parameters of the scheme.

5 Main Attack

5.1 On the Impact of Recovering x_0

If x_0 is known, CLT15 essentially collapses to CLT13. In particular, all encodings can be reduced modulo x_0 so ladders are no longer needed. What is more, all $\omega_{i,j}$'s from Cheon *et al.*'s attack can be reduced modulo $v_0 = x_0 \boldsymbol{p}_{zt} \bmod N$, which effectively removes the new noise a. As a direct consequence Cheon *et al.*'s attack goes through and all secret parameters are recovered (cf. [CLT15, Section 3.3]). Moreover ladder elements reduced by x_0 provide low-level encodings of zero even if the scheme itself does not.

Our attack recovers x_0. As a first step, we introduce *integer extraction*.

5.2 Integer Extraction

Integer extraction essentially removes the extra noise induced by ladder reductions when performing computations on encodings. In addition, as we shall see in Section 5.3, this step is enough to recover x_0 when an exact multiple is known, as is the case in the optimized variant proposed and implemented in [CLT15].

Integer Extraction of Level-$\boldsymbol{\kappa}$ Encodings of Zero. In the remainder we say that an encoding at level k is small iff it is less than $X_0^{(k)}$ in absolute value. In particular, any ladder-reduced encoding is small.

Definition 4 (integer extraction of an encoding). *Let $e \in \mathbb{Z}$, and write:*

$$e = \sum_{i=1}^{n} r_i u_i + a x_0$$
$$\textit{with: } u_i = \left(g_i z^{-k} (p_i^*)^{-1} \bmod p_i\right) p_i^* \ \textit{as in (3)}$$
$$r_i \in \mathbb{Z} \cap (-p_i/2, p_i/2].$$

Note that r_i is uniquely defined as $r_i = e g_i^{-1} z^k \bmod p_i$, and $a = (e - \sum r_i u_i)/x_0$. Hence the following map is well-defined over $\mathbb{Z}$:

$$\phi : e \mapsto \sum_i r_i v_i + a v_0$$
$$\textit{with: } v_0 = x_0 \boldsymbol{p}_{zt} \bmod N, \ \textit{and } \forall i > 0, v_i = u_i \boldsymbol{p}_{zt} \bmod N \ \textit{as in (4)}.$$

We call $\phi(e)$ the integer extraction *of e.*

Remark. ϕ is defined over the integers, and not modulo N. Indeed the v_i's are seen as integers: recall from Section 2 that throughout this paper $x \bmod N$ denotes an integer in $\mathbb{Z} \cap (-N/2, N/2]$.

The point is that if e is a small encoding of zero at level κ, then $\phi(e) = e\boldsymbol{p}_{zt} \bmod N$. In that case $\phi(e)$ matches the extraction in the sense of the `ext` procedure of Section 3.2 (more precisely `ext` returns the high-order bits of $\phi(e)$).

However we want to compute $\phi(e)$ even when e is larger. For this purpose, the crucial point is that ϕ is actually $\mathbb{Z}$-linear as long as for all encodings involved, the associated r_i's do not go over $p_i/2$, *i.e.* reduction modulo p_i does not interfere. More formally:

Lemma 1. *Let $e, a, r_1, \ldots, r_n \in \mathbb{Z}$ with $-p_i/2 < r_i \leq p_i/2$ such that $e = \sum r_i u_i + a x_0$ as in Definition 4. Define $e' = \sum r'_i u_i + a' x_0$ in the same manner. Let $k \in \mathbb{Z}$.*

1. *If $\forall i, -p_i/2 < r_i + r'_i \leq p_i/2$, then:* $\qquad \phi(e + e') = \phi(e) + \phi(e')$
2. *If $\forall i, -p_i/2 < kr_i \leq p_i/2$, then:* $\qquad \phi(ke) = k\phi(e)$

An important remark is that the conditions on the r_i's above are also required for the correctness of the scheme to hold. In other words, as long as we perform valid computations from the point of view of the multilinear map (*i.e.* there is no reduction of the r_i's modulo p_i, and correctness holds), then the $\mathbb{Z}$-linearity of ϕ also holds.

Using this observation, we can recursively compute the integer extraction of every ladder element $X_i^{(\kappa)}$. Indeed $\phi(X_0^{(\kappa)}) = X_0^{(\kappa)} \boldsymbol{p}_{zt} \bmod N$. Then assume we know $\phi(X_0^{(\kappa)}), \ldots, \phi(X_i^{(\kappa)})$ for some $i < \ell$. Reduce X_{i+1} by the previous elements of the ladder. We get:

$$
\begin{aligned}
& & Y_{i+1} &\triangleq X_{i+1}^{(\kappa)} - \alpha_i X_i^{(\kappa)} - \cdots - \alpha_0 X_0^{(\kappa)} \\
&\text{with:} & \operatorname{abs}(Y_{i+1}) &< \operatorname{abs}(X_0^{(\kappa)}) \\
&\text{whence:} & \phi(X_{i+1}^{(\kappa)}) &= \phi(Y_{i+1}) + \sum_{j \leq i} \alpha_j \phi(X_j^{(\kappa)})
\end{aligned}
$$

Since $\operatorname{abs}(Y_{i+1}) < \operatorname{abs}(X_0)$ we can compute $\phi(Y_{i+1}) = Y_{i+1}\boldsymbol{p}_{zt} \bmod N$, and deduce $\phi(X_{i+1}^{(\kappa)})$.

In exactly the same manner, we can compute $\phi(e)$ for any valid level-κ encoding of zero, by first reducing via the ladder and then compensating using $\mathbb{Z}$-linearity. Here, by valid we mean of size up to X_ℓ, and such that the corresponding r_i's are within the limit imposed by the correctness of the multilinear map.

In Appendix A, we show how to also compensate ladder reductions at intermediate levels for any computation on encodings, e.g. compute $\phi(abc)$ for a three-way product abc. However this will not be needed for our attack, as the previous technique will suffice.

5.3 Recovering x_0 when an Exact Multiple is Known

The authors of [CLT15] propose an optimized version of their scheme, where a multiple qx_0 of x_0 is provided in the public parameters. The size of q is chosen such that qx_0 is about the same size as N. Ladders at levels below κ are no longer necessary: every encoding can be reduced modulo qx_0 without altering encoded values or increasing any noise. The ladder at level κ is still needed as a preliminary to zero-testing, however it does not need to go beyond qx_0, which makes it much smaller. In the end this optimization greatly reduces the size of the public key and speeds up computations.

In this scenario, note that qx_0 may be regarded as an encoding of 0 at level κ (and indeed every level). Moreover by construction it is small enough to be reduced by the ladder at level κ with a valid computation (*i.e.* with low enough noise for every intermediate encoding involved that the scheme operates as desired and zero-extraction is correct). As a direct consequence we have:

$$\phi(qx_0) = qv_0$$

and so we can recover q as $q = \gcd(qx_0, \phi(qx_0))$, and get $x_0 = qx_0/q$. This attack has been verified on the reference implementation, and recovers x_0 instantly.

Remark. qv_0 is larger than N by design, so that it cannot be computed simply as $qx_0\boldsymbol{p}_{zt} \bmod N$ due to modular reductions (cf. [CLT15, Section 3.4]). The point is that our computation of ϕ is over the integers and not modulo N.

5.4 Recovering x_0 in the General Case

We now return to the non-optimized version of the scheme, where no exact multiple of x_0 is provided in the public parameters.

The second step of our attack recovers x_0 using a matrix product similar to Cheon *et al.*'s (cf. Section 4), except we start with families of $n+1$ encodings rather than n. That is, assume that for some t we have $n+1$ level-t small encodings (a_i) of any value, and $n+1$ level-$(\kappa - t)$ small encodings (b_i) of zero. This is easily achievable for one-round multi-party Diffie-Hellman (cf. Section 3.2), e.g. choose $t = 1$, then pick $(n+1)$ level-1 encodings (a_i) of zero from the public parameters, and let $b_i = a'_i y^{\kappa-2}$ for a'_i another family of $(n+1)$ level-1 encodings of zero and y any level-1 encoding, where the product is ladder-reduced at each level. In other applications of the multilinear map, observe that ladder elements provide plenty of small encodings of zero, as each ladder element can be reduced by the elements below it to form a small encoding of zero. Thus the necessary conditions to perform both our attack to recover x_0, and the follow-up attack by Cheon *et al.* to recover other secret parameters once x_0 is known, are very lax. In this respect [CLT15] is weaker than [CLT13].

Let $a_{i,j} = a_i z \bmod p_j$, *i.e.* $a_{i,j}$ is the j-th value "$r_j g_j + m_j$" associated with a_i. Likewise for $i \le n$, let $r_{i,j} = b_i z^{\kappa-1}/g_j \bmod p_j$, *i.e.* $r_{i,j}$ is the j-th value "r_j" associated with b_i (recall that b_i is an encoding of zero, so $m_j = 0$). Now compute:

$$\omega_{i,j} \triangleq \phi(a_i b_j).$$

If we look at the $\omega_{i,j}$'s modulo v_0 (which is unknown for now), everything behaves as in CLT13 since the new noise term av_0 disappears, and the ladder reduction at level κ is negated by the integer extraction procedure. Hence, similar to Section 4, we have:

$$\omega_{i,j} \bmod v_0 = \sum_k a_{i,k} r_{j,k} v_k \bmod v_0. \tag{6}$$

Again, equation (6) may be seen as a matrix product. Indeed, define Ω as the $(n+1) \times (n+1)$ integer matrix with entries $\omega_{i,j}$, let A be the $(n+1) \times n$ matrix with entries $a_{i,j}$, let R be the $(n+1) \times n$ matrix with entries $r_{i,j}$, and finally let V be the $n \times n$ diagonal matrix with diagonal entries v_i. Then (6) may be rewritten modulo v_0:

$$\Omega = A \cdot V \cdot R^{\mathrm{T}} \qquad \text{in } \mathbb{Z}_{v_0}.$$

Since A and R are $(n+1) \times n$ matrices, this implies that Ω is not full-rank when embedded into $\mathbb{Z}_{v_0}$. As a consequence v_0 divides $\det(\Omega)$, where the determinant is computed over the integers. Now we can build a new matrix Ω' in the same way using a different choice of b_i's, and recover v_0 as $v_0 = \gcd(\det(\Omega), \det(\Omega'))$. Finally we get $x_0 = v_0 / \boldsymbol{p}_{zt} \bmod N$ (note that $N \gg x_0$ by construction).

The attack has been verified on the reference implementation with reduced parameters.

Remark. As pointed out above, Ω cannot be full-rank when embedded into $\mathbb{Z}_{v_0}$. Our attack also requires that it *is* full-rank over $\mathbb{Q}$ (whp). This holds because while Ω can be nicely decomposed as a product when viewed modulo v_0, the "remaining" part of Ω, that is $\Omega - (\Omega \bmod v_0)$ is the matrix of the terms av_0 for each $\omega_{i,j}$, and the value a does have the nice structure of $\omega_{i,j} \bmod v_0$. This is by design, since the noise a was precisely added in CLT15 in order to defeat the matrix product structure in Cheon *et al.*'s attack.

5.5 Attack Complexity

It is clear that the attack is polynomial, and asymptotically breaks the scheme. In this section we provide a closer look at its practical complexity. When an exact multiple of x_0 is known, the attack is instant as mentioned in Section 5.3, so we focus on the general case from Section 5.4. There are two steps worth considering from a complexity point of view: computing Ω and computing its determinant. Computing the final gcd is negligible in comparison using a sub-quadratic algorithm [Möl08], which is practical for our parameter size.

Computing $\boldsymbol{\Omega}$. Computing Ω requires $(n+1)^2$ integer extractions of a single product. Each integer extraction requires 1 multiplication, and 2ℓ additions (as well as ℓ multiplications by small scalars). For comparison, using the multilinear scheme for one user requires 1 multiplication and ℓ additions on integers of similar size. Thus overall computing Ω costs about n^2 times as much as simply *using* the multilinear scheme. For the 52-bit instance proposed in [CLT15] for instance, this means that if it is practical to use the scheme about a million times, then it is practical to compute Ω. Thus we will focus on the determinant computation as the main bottleneck.

Computing the Determinant. Let n denote the size of a matrix Ω (it is $(n+1)$ in our case but we will disregard this), and β the number of bits of its largest entry. When computing the determinant of an integer matrix, one has to carefully control the size of the integers appearing in intermediate computations. It is generally possible to ensure that these integers do not grow past the size of the determinant. Using Hadamard's bound this size can be upper bounded as $\log(\det(\Omega)) \leq n(\beta + \frac{1}{2}\log n)$, which can be approximated to $n\beta$ in our case, since β is much larger than n.[4]

As a result, computing the determinant using "naive" methods requires $\mathcal{O}(n^3)$ operations on integers of size up to $n\beta$, which results in a complexity $\tilde{\mathcal{O}}(n^4\beta)$ using fast integer multiplication (but slow matrix multiplication). The asymptotic complexity is known to be strictly less than $\tilde{\mathcal{O}}(n^3\beta)$ [KV04]; however we are interested in the complexity of practical algorithms. Computing the determinant can be reduced to solving the linear system associated with Ω with a random target vector: indeed the determinant can then be recovered as the least common denominator of the (rational) solution vector. In this context the fastest algorithms use p-adic lifting [Dix82], and an up-to-date analysis using fast arithmetic in [MS04] gives a complexity $\mathcal{O}(n^3\beta\log^2\beta\log\log\beta)$ (with $\log n = o(\beta)$).[5]

For the concrete instantiations of one-round multipartite Diffie-Hellman implemented in [CLT15], this yields the following complexities:

Security parameter:	52	62	72	80
Determinant complexity:	2^{57}	2^{66}	2^{74}	2^{81}

Thus, beside being polynomial, the attack is actually coming very close to the security parameter as it increases to 80 bits.[6]

Acknowledgement. We would like to thank the authors of CLT13 and CLT15 Jean-Sébastien Coron, Tancrède Lepoint and Mehdi Tibouchi for fruitful discussions and remarks.

[4] This situation is fairly unusual, and in the literature the opposite is commonly assumed; algorithms are often optimized for large n rather than large β.

[5] This assumes a multitape Turing machine model, which is somewhat less powerful than a real computer.

[6] We may note in passing that in a random-access or log-RAM computing model [Für14], which is more realistic than the multitape model, the estimated complexity would already be slightly lower than the security parameter.

References

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology–CRYPTO 2001*, pages 213–229. Springer, 2001.

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324(1):71–90, 2003.

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New attacks on multilinear maps and their limitations. In *Advances in Cryptology–CRYPTO 2015*, pages 247–266. Springer, 2015.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *Advances in Cryptology–EUROCRYPT 2015*, pages 3–12. Springer, 2015.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology–CRYPTO 2013*, pages 476–493. Springer, 2013.

[CLT15] Jean-Sebastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In *Advances in Cryptology–CRYPTO 2015*, pages 267–286. Springer, 2015.

[DH76] Whitfield Diffie and Martin E Hellman. Multiuser cryptographic techniques. In *Proceedings of the June 7-10, 1976, national computer conference and exposition*, pages 109–112. ACM, 1976.

[Dix82] John D. Dixon. Exact solution of linear equations using P-adic expansions. *Nümerische Mathematik*, 40(1):137–141, 1982.

[Für14] Martin Fürer. How fast can we multiply large integers on an actual computer? In *LATIN 2014: Theoretical Informatics*, pages 660–670. Springer, 2014.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Eurocrypt*, volume 7881, pages 1–17. Springer, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 40–49. IEEE, 2013.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *Theory of Cryptography*, pages 498–527. Springer, 2015.

[Hal15a] Shai Halevi. Cryptographic graded-encoding schemes: Recent developments. TCS+ online seminar, available at https://sites.google.com/site/plustcs/past-talks/20150318shaihaleviibmtjwatson, 2015.

[Hal15b] Shai Halevi. Graded encoding, variations on a scheme. Technical report, Cryptology ePrint Archive, Report 2015/866, 2015. http://eprint. iacr. org, 2015.

[HJ15] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. Technical report, Cryptology ePrint Archive, Report 2015/301, 2015.

[HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In *Advances in Cryptology–CRYPTO 2013*, pages 494–512. Springer, 2013.

[Jou00] Antoine Joux. A one round protocol for tripartite Diffie–Hellman. In *Algorithmic number theory*, pages 385–393. Springer, 2000.

[KV04] Erich Kaltofen and Gilles Villard. Computing the sign or the value of the determinant of an integer matrix, a complexity survey. *Journal of Computational and Applied Mathematics*, 162(1):133–146, 2004.

[Möl08] Niels Möller. On Schönhage's algorithm and subquadratic integer GCD computation. *Mathematics of Computation*, 77(261):589–607, 2008.

[MS04] Thom Mulders and Arne Storjohann. Certified dense linear system solving. *Journal of Symbolic Computation*, 37(4):485–510, 2004.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in cryptology*, pages 47–53. Springer, 1985.

[VDGHV10] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in cryptology–EUROCRYPT 2010*, pages 24–43. Springer, 2010.

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In *Advances in Cryptology-EUROCRYPT 2015*, pages 439–467. Springer, 2015.

A Integer Extraction of Products

Using Section 5.2, if a, b are two small encodings at levels s and $\kappa - s$ respectively, and b encodes zero, we know how to compute $\phi(ab)$, because the size of ab is at most that of X_ℓ.

We now consider larger products. Let $a_1, \ldots, a_m$, be small encodings at level $s_1, \ldots, s_m$, with $t_j \triangleq \sum_{i \le j} s_i$, $t_m = \kappa$, and with a_m an encoding of zero. We would like to compute $\phi(a_1 \cdots a_m)$. Note that $a_1 \cdots a_m$ may be much larger than $X_\ell^{(\kappa)}$ in the absence of ladder reduction, so our previous technique is not enough.

Instead, a valid computation is to compute the product $\pi \triangleq a_1 \cdots a_m$ pairwise from the left, and reduce at each step. That is, let $\pi_1 \triangleq a_1$, and recursively define the ladder-reduced partial product $\pi_{i+1} \triangleq \pi_i a_{i+1} - \sum_j \alpha_j^{i+1} X_i^{(t_{i+1})} < X_0^{(t_{i+1})}$ for $i < m$. Thus $\pi_m < X_0^{(\kappa)}$ encodes the same element as π, and $\phi(\pi_m) = \pi_m \boldsymbol{p}_{zt} \bmod N$. In order to compute $\phi(\pi)$, observe:

$$\pi = \Big(\big((a_1 a_2 - \sum \alpha_i^{(2)} X_i^{(t_2)}) \ldots \big) a_{m-1} - \sum \alpha_i^{m-1} X_i^{(t_{m-1})} \Big) a_m - \sum \alpha_i^{(m)} X_i^{(\kappa)}$$
$$+ \sum_{2 \le k \le m} \sum_i \alpha_i^{(k)} X_i^{(t_k)} a_{k+1} \cdots a_m$$

Hence:

$$\phi(a_1 \cdots a_m) = \phi(\pi_m) + \sum_{2 \le k \le m} \sum_i \alpha_i^{(k)} \phi(X_i^{(t_k)} a_{k+1} \cdots a_m)$$

In the above equation, $\phi(\pi_m)$ is known since π_m is small, so we are reducing the computation of a product π of m elements to a sum of products of $m - 1$

elements, of the form $X_i^{(t_k)} a_{k+1} \cdots a_m$. As mentioned earlier we already know how to compute ϕ for products of 2 small elements, so by induction we are done.

To be more precise, the induction is carried out on the hypothesis: we know how to compute ϕ for products of up to m small encodings (with the last being an encoding of zero so that the overall product encodes zero). In order to apply the induction hypothesis on $X_i^{(t_k)} a_{k+1} \cdots a_m$, the term $X_i^{(t_k)}$ would need to be small, which is not the case. However it can be reduced by previous ladder elements, *i.e.* first compute $X_0^{(t_k)} a_{k+1} \cdots a_m$, then define $Y_1 = X_1^{(t_k)} - \alpha_0 X_0^{t_k} < X_0^{(t_k)}$, whence $\phi(X_1^{(t_k)} a_{k+1} \cdots a_m) = \phi(Y_1 a_{k+1} \cdots a_m) + \alpha_0 \phi(X_0^{t_k} a_{k+1} \cdots a_m)$, and so forth as in the previous section. Thus the induction goes through and we know how to compute $\phi(\pi)$.

All in all, while the above formalism may obfuscate the process somewhat, the idea is simple: ϕ is ($\mathbb{Z}$-)linear as long as we are performing valid computations from the point of view of the scheme. As a consequence every ladder reduction involved during a computation can be compensated for at its last stage, when the level-κ encoding is multiplied by the zero-testing parameter. The payback is that we will be able to ignore ladder reductions in the rest of the attack.

A note on complexity. It may seem that computing $\phi(a_1 \cdots a_m)$ using the previous approach has a huge complexity, but actually most of the computation overlaps. In fact we only ever need to compute the $\phi(X_i^{(t_k)} a_{k+1} \cdots a_m)$'s for each i, k. Memorizing intermediate results yields a complexity in ℓm, where ℓ is the size of the longest ladder. The time required for each term is quite close to merely *using* the multi-party Diffie-Hellman scheme.