An improved method for predicting truncated multiple recursive generators with unknown parameters

Han-Bing Yu[1], Qun-Xiong Zheng[1*], Yi-Jian Liu[2], Jing-Guo Bi[2*], Yu-Fei Duan[2], Jing-Wen Xue[2], You Wu[3], Yue Cao[3], Rong Cheng[3], Lin Wang[3*] and Bai-Shun Sun[4]

[1]PLA Strategic Support Force Information Engineering University, P.O. Box 407, 62 Kexue Road, Zhengzhou, Henan, 450001, China.
[2]School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing, 100876, China.
[3]Science and Technology on Communication Security Laboratory, Chengdu, Sichuan, 610041, China.
[4]School of Cyber Engineering, Xidian University, Xian, Shaanxi, 710126, China.

*Corresponding author(s). E-mail(s): qunxiong_zheng@163.com; jguobi@bupt.edu.cn; wanglin4math@outlook.com;
Contributing authors: hbing_yu@163.com; lordriotglacier@gmail.com; duanyufi@foxmail.com; nairwx@gmail.com; 497591946@qq.com; 2670505187@qq.com; xidian_chengrong@163.com; bssun@stu.xidian.edu.cn;

Abstract

Multiple recursive generators are an important class of pseudorandom number generators which are widely used in cryptography. The predictability of truncated sequences that predict the whole sequences by the truncated high-order bits of the sequences is not only a crucial aspect of evaluating the security of pseudorandom number generators but also serves an important role in the design of pseudorandom number generators. This paper improves the work of Sun et al [1] on the predictability of truncated multiple recursive generators with unknown parameters. Given

a few truncated digits of high-order bits output by a multiple recursive generator, we adopt the resultant, the Chinese Remainder Theorem and the idea of recovering p-adic coordinates of the coefficients layer by layer, and Kannan's embedding technique to recover the modulus, the coefficients and the initial state, respectively. Experimental results show that our new method is superior to that of [1], no matter in terms of the running time or the number of truncated digits required.

Keywords: Multiple recursive generators, Truncated prediction, Lattice reduction, The resultant

1 Introduction

Throughout the paper, for an integer $m > 1$, let $\mathbb{Z}/(m)$ denote the residue ring of integers modulo m. We choose $\{0, 1, \ldots, m-1\}$ as the set of representative elements of the ring $\mathbb{Z}/(m)$. Let log denote the base 2 logarithm.

Random numbers are widely used in cryptography, statistical sampling, Monte Carlo simulation, etc. There are two main approaches to generating random numbers: using high-speed physical devices to generate stochastic noise, or using deterministic algorithms to generate pseudorandom sequences, i.e., pseudorandom number generators (PRNGs).

In 1949, Lehmer [2] proposed an important class of PRNGs, linear congruence generators (LCGs). An LCG outputs a sequence $\underline{a} = (a_i)_{i\geq 0}$ of the form

$$a_{i+1} \equiv ba_i + c \bmod m, i = 0, 1, 2, \ldots,$$

where a_0 is the initial seed and b, c are constants over $\mathbb{Z}/(m)$. The quality of the generators is determined by the choices of the increment c, multiplier b, initial seed a_0, and modulus m. The largest possible period of an LCG is m when $c \neq 0$ and $m-1$ when $c = 0$, for details see [3]. LCGs were once very popular for their simplicity, high efficiency and clear theoretical properties. However, they are not recommended by today's standards because they have relatively short periods which are bounded by the modulus m and inadequate uniformity in dimensions higher than 1 [4]. So to increase the period and to obtain a denser lattice which covers the space more evenly, the last several decades have seen a growing trend towards the higher-order linear recurrence, that is, multiple recursive generators (MRGs) [5–8].

An MRG of order n outputs a sequence $\underline{a} = (a_i)_{i\geq 0}$ as follows

$$a_{i+n} \equiv c_{n-1}a_{i+n-1} + c_{n-2}a_{i+n-2} + \cdots + c_0 a_i \bmod m, i = 0, 1, 2, \ldots,$$

where $c_0, c_1, \ldots, c_{n-1} \in \mathbb{Z}/(m)$ are not all zero. We can choose any n not-all-zero $a_0, a_1, \ldots, a_{n-1} \in \mathbb{Z}/(m)$ as the initial state. If the modulus m is a prime number, say $m = p$, then the maximum period of the output sequences of an MRG of order n is $p^n - 1$ [9]. For a general m, let $m = p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s}$ be the canonical factorization of m. Then

the maximum period of the output sequences of an MRG of order n is $\mathrm{lcm}\left(p_1^{e_1-1}\left(p_1^n-1\right), p_2^{e_2-1}\left(p_2^n-1\right), \ldots, p_s^{e_s-1}\left(p_s^n-1\right)\right)$. An MRG reaches the maximum period if its characteristic polynomial

$$f(x) = x^n - c_{n-1}x^{n-1} - \cdots - c_0$$

is a primitive polynomial over $\mathbb{Z}/(m)$ [10, 11] (The definition of primitive polynomials over $\mathbb{Z}/(m)$ see Definition 3).

In reality, MRGs have been used to design a variety of cryptographic algorithms, for example, ZUC [12] (ISO/IEC 18033-4:2011/Amd1:2020), a stream cipher which is the core of the 3GPP mobile standards 128-EEA3 (for encryption) and 128-EIA3 (for message integrity). ZUC uses an MRG of order 16 and its characteristic polynomial is a primitive polynomial over $\mathbb{Z}/(2^{31}-1)$, that is

$$f(x) = x^{16} - (2^{15}x^{15} + 2^{17}x^{13} + 2^{21}x^{10} + 2^{20}x^4 + 2^8 + 1).$$

The predictability of PRNGs is a crucial aspect of evaluating the security of PRNGs, which also serves an important role in the design of PRNGs. A PRNG is considered cryptographically secure if a cryptanalyst is unable to predict with advantage any other segment of the generator's output within feasible time and memory complexity bounds, even after obtaining a long enough segment of this output.

Numerous studies have attempted to discuss the predictability of LCGs. When all bits of the sequence are output, Plumstead [13] showed that an LCG can be predicted by an initial segment of length $2+\log m$ even if b, c, m are all unknown. When partial bits of the sequence are output, Knuth [3] observed that when the modulus m is equal to the computer's word size, namely 2^e on an e-bit binary computer, the low-order bits of the sequence output by an LCG are much less random than the high-order bits. So, Knuth [3] proposed the truncated linear congruential pseudorandom number generators, which use the high-order bits of a linear congruential sequence, to introduce nonlinear operation into linear sequences. For the case that m is known while b, c are unknown, if $m = 2^k$ and the first N values of the h high-order bits of a_n are given, Knuth [14] proposed a method to determine the unknown constants in $O(k^2 \cdot 2^{2(k-h)}/N^2)$ steps when $h \geq 2$. Stern [15] also showed the predictability in this case but not restricted m to be a power of 2. For the case that m, b are known and the increment c is unknown, if a segment of truncated high-order bits of the linear congruential generator is given, Frieze et al [16] gave a polynomial-time algorithm based on lattice to predict the generator, which may fail on a small set of exceptional multipliers. For the case that m, b, c are all unknown, Stern [15, 17] gave a polynomial-time algorithm based on lattice to predict the generator, which relied on two heuristic assumptions. Boyar [18] presented a polynomial-time algorithm that correctly infers the sequence with at most a polynomial in $\log m$ errors, even if $O(\log \log m)$ of the low-order bits are unknown. Contini and Shparlinski [19] improved Stern's method to make their algorithm simpler, more robust, and require less data.

There are also some results about the predictability of MRGs. When m is a prime number and all bits of the sequence are outputted, an MRG can be predicted by the Berlekamp-Massey (BM) iterative algorithm [20, 21] in $O(N^2)$ operations, where N is the length of the output sequence. In addition, this problem can also be solved by the Sugiyama-Kasahara-Hirasawa-Namekawa algorithm [22] based on the Euclidean algorithm and the Mills algorithm [23] using continued fractions. When m is a general integer and all bits of the sequence are outputted, Reeds and Sloane [24] successfully extended the BM algorithm from fields to integer residue rings. For a more general model,

$$a_n \equiv \sum_{j=1}^{k} \alpha_j \phi_j(a_0, a_1, \ldots, a_{n-1}) \bmod m,$$

where functions ϕ_j are known and polynomial time computable, Boyar [25] pointed out that the MRG can be predicted even if the coefficients α_j and the modulus m are unknown. When only outputting the highest bit sequence, Huang et al [26] and Kuzmin et al [27] independently proved that the sequences consisting of the highest bit only are pairwise distinct provided that the primitive polynomial and the modulus 2^e are known. In other words the primitive sequences can be theoretically recovered by the highest bits of the sequences. This result is fairly meaningful and serves the theoretical foundation for subsequent studies. Huang et al [26] also proposed a practical prediction algorithm but it required a whole period of the highest bits of the sequences. Zhu et al [28] and Kuzmin et al [29, 30] gave some recovery algorithms for recovering the sequences from its highest bits. Because these algorithms are either demanding on data, or have huge time and space complexity, their applications are limited in practice. On the other hand, scholars began to wonder if there was an effective way to predict truncated sequences when the number of truncated bits were greater than 1. When all the parameters of MRGs are known, Yang et al [31] gave a method based on Coppersmith's method [32] to recover the initial state by the truncated sequences. When all the parameters of MRGs are unknown, Sun et al [1] extended Stern's algorithm to predict truncated MRGs. They first constructed proper lattice to recover the modulus m, then recovered the coefficients $c_0, c_1, \ldots, c_{n-1}$, and finally recovered the initial state $a_0, a_1, \ldots, a_{n-1}$.

This paper gives the latest achievements in the predictability of truncated sequences over integer residue rings during the sixth (2021) National Crypto-Math Challenge (http://www.cmsecc.com/). The problem was designed by the second author of this paper. From March 5 to July 21, 2021, three major teams have separately come up with some interesting new methods to solve this problem. It was further improved and perfected after the Challenge, and a lot of experiments were also performed to confirm the correctness and effectiveness. To better illustrate the current research progress, we organized them into this paper. In this paper, we improve the method in [1] to predict truncated MRGs with the parameters $m, c_0, c_1, \ldots, c_{n-1}$ all unknown. To be more specific,

- When searching linear relations, we instead to find vectors that are orthogonal to the segments of unknown sequence, rather than vectors that are both orthogonal to the segments of unknown sequence and the segments of truncated sequence. In addition, compared with [1], we construct the lattice without multiplying a big integer, which can accelerate single call to the lattice reduction algorithm by shrinking the data size of the lattice. As a result, both improvements reduce the size of the lattice and then reduce the number of truncated digits required.
- When recovering the modulus m, the method in [1] requires multiple calls to the SVP algorithm to get a large number of linear relations while we only need a single call to the SVP algorithm to obtain several linear relations, so our new method can greatly reduce the time complexity.
- When recovering the coefficients $c_0, c_1, \ldots, c_{n-1}$, the method in [1] recovers $c_0, c_1, \ldots, c_{n-1}$ item by item by reducing n lattices. In this paper, to recover the coefficients $c_0, c_1, \ldots, c_{n-1}$, we instead to recover the polynomial $f(x) = x^n - c_{n-1}x^{n-1} - \cdots - c_0$ over $\mathbb{Z}/(m)$. Let $m = p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s}$ be the canonical factorization of the modulus m. We first get $f(x) \bmod p_j$ by computing the greatest common divisor of some certain polynomials over $\mathbb{Z}/(p_j)$ for $1 \leq j \leq s$, and then get $f(x) \bmod p_j^{e_j}$ by the Extended Euclidean algorithm (or solving linear equations) over $\mathbb{Z}/(p_j)$ layer by layer, and finally get $f(x)$ by the Chinese Remainder Theorem. This change also has the benefit of shortening the running time required.
- When recovering the initial state, we adopt Kannan's embedding technique to transform the short integer solutions problem into the shortest vector problem and balance each coordinate of the target vector so that we can use less truncated digits to recover the initial state.

So our improvements are comprehensive. No matter in what kind of cases, the modulus is known, the modulus and the coefficients are all known or all parameters are unknown, our new method shows more superiority than that of [1]. Experimental results have confirmed that our new method is more effective and data-saving compared with [1]. Take ZUC's driving sequence as an example, we only need 239 truncated digits of 17 high-order bits to recover all unknown parameters and the whole process can be executed in 10 minutes, while [1] needs 404 truncated digits and 40 hours, see Section 5 for more details.

The rest of the paper is organized as follows. Section 2 introduces the basic definitions and results used in later proof. Section 3 briefly reviews the work of [1]. Section 4 proposes our improved method for predicting truncated MRGs. Section 5 gives the experimental results to show the superiority of our method. Section 6 summarizes our work and lists an issue to be further discussed.

2 Preliminaries

In this section, we will introduce some basic definitions and results about sequences over integer residue rings, lattice reduction and Galois rings.

2.1 Characteristic polynomials, minimal polynomials and primitive polynomials

Definition 1 Let m be an integer greater than 1. If a sequence $\underline{a} = (a_i)_{i\geq 0}$ over $\mathbb{Z}/(m)$ satisfies the linear recurrence relation

$$a_{i+n} \equiv c_{n-1}a_{i+n-1} + c_{n-2}a_{i+n-2} + \cdots + c_0 a_i \bmod m, i = 0, 1, 2, \ldots,$$

where $c_0, c_1, \ldots, c_{n-1} \in \mathbb{Z}/(m)$ are not all zero and n is a positive integer, then the sequence $\underline{a}$ is called a linear recurring sequence of order n over $\mathbb{Z}/(m)$ generated by $f(x) = x^n - c_{n-1}x^{n-1} - \cdots - c_0$, and $f(x)$ is called a characteristic polynomial of $\underline{a}$. A characteristic polynomial of $\underline{a}$ with the minimal degree is called a minimal polynomial of $\underline{a}$. A polynomial $g(x) = d_s x^s + d_{s-1}x^{s-1} + \cdots + d_1 x + d_0$ over $\mathbb{Z}/(m)$ is said to annihilate $\underline{a}$ if

$$d_s a_{i+s} + d_{s-1}a_{i+s-1} + \cdots + d_1 a_{i+1} + d_0 a_i \equiv 0 \bmod m, i = 0, 1, 2, \ldots.$$

Let $G(f(x), m)$ denote the set of all linear recurring sequences generated by $f(x)$ over $\mathbb{Z}/(m)$, and let $I_m(\underline{a})$ denote the set of polynomials over $\mathbb{Z}/(m)$ that annihilate $\underline{a}$. It can be easily checked that $I_m(\underline{a})$ forms an ideal of $\mathbb{Z}/(m)[x]$.

For any integer a and any positive integer $b \geq 2$, the least nonnegative residue of a modulo b is denoted by $a \bmod b$ or $[a] \bmod b$. For a sequence $\underline{a} = (a_i)_{i\geq 0}$ over $\mathbb{Z}/(m)$, let $\underline{a} \bmod b = ([a_i] \bmod b)_{i\geq 0}$, where $[a_i] \bmod b$ means that first let a_i be the least nonnegative residue of a_i modulo m, then take the least nonnegative residue of a_i modulo b. Similarly, for a polynomial $f(x) = c_n x^n + \cdots + c_1 x + c_0$ over $\mathbb{Z}/(m)$, let $f(x) \bmod b = [c_n] \bmod b\, x^n + \cdots + [c_1] \bmod b\, x + [c_0] \bmod b$.

Let $f(x)$ be a monic polynomial of degree n over $\mathbb{Z}/(m)$. If $f(0)$ is an invertible element over $\mathbb{Z}/(m)$, then there exists a positive integer T such that $x^T - 1$ is divisible by $f(x)$ in $\mathbb{Z}/(m)[x]$. The minimum of such T is called the period of $f(x)$ over $\mathbb{Z}/(m)$ and denoted by $\mathrm{per}(f(x), m)$.

Definition 2 [33] Let p^e be a prime power and let $f(x)$ be a monic polynomial of degree n over $\mathbb{Z}/(p^e)$ with $\gcd(f(0), p) = 1$. If $\mathrm{per}(f(x), p^e) = p^{e-1}(p^n - 1)$, then $f(x)$ is called a primitive polynomial of degree n over $\mathbb{Z}/(p^e)$. A sequence $\underline{a}$ is called a primitive sequence of order n over $\mathbb{Z}/(p^e)$ if $\underline{a}$ is generated by a primitive polynomial of degree n over $\mathbb{Z}/(p^e)$ and $\underline{a} \bmod p \neq \underline{0}$.

If $f(x)$ is a primitive polynomial of degree n over $\mathbb{Z}/(p^e)$, then for every $i \in \{1, 2, \ldots, e-1\}$, $f(x) \bmod p^i$ is also a primitive polynomial of degree n over $\mathbb{Z}/(p^i)$. In particular, $f(x) \bmod p$ is a primitive polynomial of degree n over the finite field $\mathbb{Z}/(p)$.

Definition 3 [11] Let $m = p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s}$ be the canonical factorization of m. A monic polynomial $f(x)$ with $\gcd(f(0), m) = 1$ is called a primitive polynomial of degree n over $\mathbb{Z}/(m)$ if for every $i \in \{1, 2, \ldots, s\}$, $f(x) \bmod p_i^{e_i}$ is a primitive polynomial of degree n over $\mathbb{Z}/(p_i^{e_i})$. A sequence $\underline{a}$ is called a primitive sequence of order n

over $\mathbb{Z}/(m)$ if $\underline{a}$ is generated by a primitive polynomial of degree n over $\mathbb{Z}/(m)$ and $\underline{a} \bmod p_i \neq \underline{0}$ for every $i \in \{1, 2, \ldots, s\}$, that is, $\underline{a} \bmod p_i^{e_i}$ is a primitive sequence of order n over $\mathbb{Z}/(p_i^{e_i})$.

Let $G'(f(x), m)$ denote the set of all primitive sequences generated by the primitive polynomial $f(x)$ over $\mathbb{Z}/(m)$.

2.2 Lattices and lattice reduction algorithms

In this section, we will introduce some results about lattices and lattice reduction algorithms.

Definition 4 A lattice is a discrete additive subgroup of $\mathbb{R}^n$. Specifically speaking, given s linearly independent vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_s \in \mathbb{R}^n$, then the set

$$L = \{\mathbf{v} \mid \mathbf{v} = c_1\mathbf{b}_1 + c_2\mathbf{b}_2 + \cdots + c_s\mathbf{b}_s, c_1, c_2, \ldots, c_s \in \mathbb{Z}\}$$

is called an n-dimensional lattice. The set of linearly independent vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_s$ is called a basis of the lattice L and the integer s is called the rank of L. A lattice L is called a full-rank lattice if $s = n$.

Let B be the $s \times n$ matrix consisting of row vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_s$. Then the determinant of the lattice L is defined by

$$\det(L) = \sqrt{\det(BB^T)},$$

where B^T is the transpose of B. The i-th successive minima of L denoted by $\lambda_i(L)$ is the radius of the smallest zero-centered ball containing at least i linearly independent vectors of L. In particular, $\lambda_1(L)$ is the length of the shortest nonzero vector in L.

Definition 5 (Gaussian Heuristic) [34] Let L be a full-rank lattice in $\mathbb{R}^n$, and C a measurable subset of $\mathbb{R}^n$ with volume $\mathrm{vol}(C)$. The Gaussian Heuristic "predicts" that the number of points of $L \cap C$ is roughly $\mathrm{vol}(C)/\det(L)$.

A lattice is called a random lattice if it follows from Haar measures of classical groups [35], which give rise to a natural probability distribution on the set of lattices. Nguyen [36] suggested that random lattices have the following property: with overwhelming probability, the successive minima of a random n-dimensional lattice L are all asymptotically close to the Gaussian heuristic. In particular, $\lambda_1(L)$ can be estimated as follows.

Lemma 1 [37] *Let L be an n-dimensional lattice. If the Gaussian heuristic holds for L, then*

$$\lambda_1(L) \approx \sqrt{\frac{n}{2\pi e}} \left(\det L\right)^{1/n}.$$

For a vector $\mathbf{v}$, let $\|\mathbf{v}\|$ denote its Euclidean norm. Next we introduce two basic computational lattice problems which are very useful in cryptography: the shortest vector problem (SVP) and the closest vector problem (CVP).

Definition 6 (SVP) Given a basis of an n-dimensional lattice L, find a $\mathbf{v} \in L$ such that $\|\mathbf{v}\| = \lambda_1(L)$.

Definition 7 (CVP) Given a basis of an n-dimensional lattice L and a point $\mathbf{t} \in \mathbb{R}^n$, find a $\mathbf{v} \in L$ such that $\|\mathbf{v} - \mathbf{t}\| \leq \|\mathbf{u} - \mathbf{t}\|$ for any vector $\mathbf{u} \in L$.

The main algorithms for solving SVP can be divided into two types, one is the approximation algorithms, such as LLL [38] and BKZ [39], and the other is the exact algorithms, such as Kannan's enumeration [40, 41] and the sieve of AKS [42]. In [38], Lenstra et al put forward the famous LLL algorithm, which can solve an approximate-SVP in polynomial time. In [39], Schnorr and Euchner proposed the BKZ algorithm, which is a blockwise generalization of LLL with potentially superexponential complexity and behaves better practical performance.

The reduced basis of a lattice output by the LLL algorithm has the following properties.

Theorem 1 [38] *Let L be an n-dimensional lattice and let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ be a reduced basis of L output by the LLL algorithm. Then*

(1) $\|\mathbf{v}_i\| \leq \rho^{(n-1)/2}\lambda_i(L), 1 \leq i \leq n,$

(2) $\|\mathbf{v}_1\| \leq \rho^{(n-1)/4}\det(L)^{1/n},$

where $\rho = \frac{4}{4\delta-1}$ and δ is the reduction parameter of the LLL algorithm satisfying $1/4 < \delta < 1$.

Remark 1 The reduction parameter is usually set to $\delta = \frac{3}{4}$, and then $\rho = 2$.

2.3 Galois rings

In this section, we will introduce the basic concepts and properties related to Galois rings. For a more detailed introduction, please refer to [43].

Let R be a ring and let a, b be two nonzero elements of R. If $ab = 0$, then a, b are called zero divisors of R. A nonempty set I of R is called an ideal of R, if $a, b \in I$ and $r \in R$ imply $a + b \in I$ and $ra \in I$. The ideal $Ra = \{ra : r \in R\}$ of R is called a principal ideal generated by a and denoted by (a).

Definition 8 [43] A Galois ring is defined to be a finite ring with identity 1 such that the set of its zero divisors with 0 added forms a principal ideal $(p1)$ for some prime number p.

Let p^e be a prime power. It is well-known that the integer residue ring $\mathbb{Z}/(p^e)$ is a Galois ring with characteristic p^e and cardinality p^e. Let $f(x)$ be a monic basic irreducible polynomial of degree n over $\mathbb{Z}/(p^e)$, that is, $f(x) \bmod p$ is an irreducible polynomial of degree n over the finite field $\mathbb{Z}/(p)$. Then the residue class of ring $\mathbb{Z}/(p^e)[x]/(f(x))$ is also a Galois ring with characteristic p^e and cardinality p^{en}. It is also well-known that any two Galois rings of the same characteristic and the same cardinality are isomorphic. Therefore, a Galois ring of characteristic p^e and cardinality p^{en} is denoted by $GR(p^e, p^{en})$. In particular, we have $\mathbb{Z}/(p^e) \cong GR(p^e, p^e)$ and $\mathbb{Z}/(p^e)[x]/(f(x)) \cong GR(p^e, p^{en})$. If $R = GR(p^e, p^{en})$ is a Galois ring, then $R/(p)$ is isomorphic to $\mathbb{F}_{p^n}$, i.e., $R/(p) \cong \mathbb{F}_{p^n}$, where $\mathbb{F}_{p^n}$ is an n degree extension field of the prime field $\mathbb{Z}/(p)$.

The following lemma implies that a monic basic irreducible polynomial of degree n over $\mathbb{Z}/(p^e)$ splits in $GR(p^e, p^{en})$.

Lemma 2 [43] *Let p^e be a prime power and $f(x)$ a monic basic irreducible polynomial of degree n over $\mathbb{Z}/(p^e)$. Then there exists $\xi \in GR(p^e, p^{en})$ such that $f(x) = \prod_{i=0}^{n-1}(x - \xi^{p^i})$.*

Remark 2 Since a primitive polynomial over $\mathbb{Z}/(p)$ is necessarily irreducible, it follows immediately from Lemma 2 that a primitive polynomial of degree n over $\mathbb{Z}/(p^e)$ splits in $GR(p^e, p^{en})$.

The resultant is an important tool in algebra, widely used in determining whether two polynomials have common factors. It is defined as follows.

Definition 9 [44] Let R be an arbitrary commutative ring and $R[x]$ the polynomial ring over R. Let $g(x) = a_u x^u + a_{u-1}x^{u-1} + \cdots + a_0 \in R[x]$ and $h(x) = b_v x^v + b_{v-1}x^{v-1} + \cdots + b_0 \in R[x]$ be two polynomials of formal degree u resp. v with $u, v \in \mathbb{N}$. Then the resultant $R(g, h)$ of the two polynomials is defined by the determinant

$$R(g,h) = \begin{vmatrix} a_u & a_{u-1} & \cdots & a_0 & 0 & \cdots & 0 \\ 0 & a_u & a_{u-1} & \cdots & a_0 & \cdots & 0 \\ \vdots & & & & & & \vdots \\ 0 & \cdots & 0 & a_u & a_{u-1} & \cdots & a_0 \\ b_v & b_{v-1} & \cdots & & b_0 & \cdots & 0 \\ 0 & b_v & b_{v-1} & \cdots & & \cdots & 0 \\ \vdots & & & & & & \vdots \\ 0 & \cdots & 0 & b_v & b_{v-1} & \cdots & b_0 \end{vmatrix} \begin{matrix} \\ \\ \\ \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} v \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} u \end{matrix}$$

of order $u + v$.

Following the connection between the roots and the resultant of polynomials over fields, we give the following main theorem of this subsection. The proof mainly refers to [44].

Theorem 2 *Let p^e be a prime power. Let $g(x) = a_u x^u + a_{u-1}x^{u-1} + \cdots + a_0 \in \mathbb{Z}/(p^e)[x]$ and $h(x) = b_v x^v + b_{v-1}x^{v-1} + \cdots + b_0 \in \mathbb{Z}/(p^e)[x]$ be two polynomials of formal degree u resp. v with $u, v \in \mathbb{N}$. If the two polynomials have a primitive polynomial $f(x)$ of degree n over $\mathbb{Z}/(p^e)$ as a common divisor, then $p^{en} \mid R(g,h)$ over $\mathbb{Z}$.*

Proof See Appendix A. □

By the Chinese Remainder Theorem, the result of Theorem 2 can be generalized to the following Theorem 3.

Theorem 3 *Let m be an integer greater than 1. Let $g(x) = a_u x^u + a_{u-1}x^{u-1} + \cdots + a_0 \in \mathbb{Z}/(m)[x]$ and $h(x) = b_v x^v + b_{v-1}x^{v-1} + \cdots + b_0 \in \mathbb{Z}/(m)[x]$ be two polynomials of formal degree u resp. v with $u, v \in \mathbb{N}$. If the two polynomials have a primitive polynomial $f(x)$ of degree n over $\mathbb{Z}/(m)$ as a common divisor, then $m^n \mid R(g,h)$ over $\mathbb{Z}$.*

Proof See Appendix B. □

3 A review of Sun-Zhu-Zheng's method

In this section, we give a brief review of the method, which we abbreviate as Sun-Zhu-Zheng's method, in [1] for predicting truncated MRGs with unknown parameters. First, we introduce some necessary notations and the problem to be studied.

Let m be an integer greater than 1 and let $f(x) = x^n - c_{n-1}x^{n-1} - \cdots - c_0 \in \mathbb{Z}/(m)[x]$. Let $\underline{a} = (a_i)_{i\geq 0} \in G(f(x), m)$, that is, the sequence $\underline{a}$ satisfies

$$a_{i+n} \equiv c_{n-1}a_{i+n-1} + c_{n-2}a_{i+n-2} + \cdots + c_0 a_i \bmod m, i = 0, 1, 2, \ldots.$$

Let $k = \lceil \log m \rceil$ denote the number of bits of $m-1$. Assume that y_i consists of a proportion α of the high-order bits of a_i and z_i consists of the low-order bits of a_i, then a_i can be expressed as

$$a_i = 2^{\beta k} y_i + z_i, i = 0, 1, 2, ..., \tag{1}$$

where $0 < \alpha < 1$ and $\beta = 1 - \alpha$.

Let

$$Q = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 & c_0 \\ 1 & 0 & \cdots & 0 & 0 & c_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & c_{n-2} \\ 0 & 0 & \cdots & 0 & 1 & c_{n-1} \end{pmatrix}.$$

It is obvious from the linear recurrence relation of the sequence $\underline{a}$ that

$$(a_j, a_{j+1}, \ldots, a_{j+n-1}) \equiv (a_0, a_1, \ldots, a_{n-1})\, Q^j \bmod m,$$

and then

$$a_j \equiv \sum_{i=0}^{n-1} q_{j,i} a_i \bmod m, \tag{2}$$

where $j \geq 0$ and $(q_{j,0}, q_{j,1}, \ldots, q_{j,n-1})^T$ denotes the first column vector of Q^j. We note that each $q_{j,i}$ is a polynomial in $c_0, c_1, \ldots, c_{n-1}$ with integral coefficients. In particular, $q_{n,i} = c_i$ for $0 \leq i \leq n-1$.

The problem of predicting truncated MRGs with unknown parameters can be described as follows: given the first N truncated digits $y_0, y_1, \ldots, y_{N-1}$, how to recover the unknown modulus m, the unknown coefficients $c_0, c_1, \ldots, c_{n-1}$ and the unknown initial state $a_0, a_1, \ldots, a_{n-1}$.

Sun-Zhu-Zheng's method consists of 5 steps: searching linear relations, constructing congruence equations, recovering the modulus, recovering the coefficients and recovering the initial states.

3.1 Searching linear relations

The goal of searching linear relations is to find a set of linear relations for $\underline{a}^{(N)} = (a_0, a_1, \ldots, a_{N-1})$, a segment of length N of the sequence $\underline{a}$. However, since $\underline{a}^{(N)}$ is unknown, one can not directly find such linear relations. The core idea of Sun-Zhu-Zheng's method is to use lattice reduction algorithm to find linear relations with small coefficients for the known truncated sequence $\underline{y}^{(N)}$ and then discuss when such linear relations are also hold for $\underline{a}^{(N)}$.

The process consists of two steps. The first step is to search linear relations with small integer coefficients for $\underline{y}^{(N)}$, which is achieved by using the orthogonal lattice technique.

Choose suitable positive integers r and t satisfying $r > t > n$ and $r+t-1 \leq N$ (in practice, r and t are selected experimentally) and let

$$\mathbf{Y}_i = (y_i, y_{i+1}, \ldots, y_{i+t-1}) \in \mathbb{Z}^t, i = 0, 1, \ldots, r-1.$$

Then there exist (for details, see [17]) integer coefficients $\zeta_0, \zeta_1, \ldots, \zeta_{r-1}$ such that

$$\sum_{i=0}^{r-1} \zeta_i \mathbf{Y}_i = \mathbf{0},$$

where $|\zeta_i| \leq B$, with

$$\log B = t\frac{\alpha k + \log r + 1}{r - t}.$$

Let $K = \lceil \sqrt{r} 2^{(r-1)/2} B \rceil$ be a constant. Construct the following lattice

$$L = \begin{pmatrix} K\mathbf{Y}_0 & 1 & 0 & \cdots & 0 \\ K\mathbf{Y}_1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ K\mathbf{Y}_{r-1} & 0 & 0 & \cdots & 1 \end{pmatrix},$$

and let $\mathbf{W}_0, \mathbf{W}_1, \ldots, \mathbf{W}_{r-1}$ be a reduced basis of L output by the LLL algorithm. Then $\mathbf{W}_0$ has the form $(\underbrace{0, 0, \ldots, 0}_{t}, \eta_0, \eta_1, \ldots, \eta_{r-1})$ that satisfies

$$\sum_{i=0}^{r-1} \eta_i \mathbf{Y}_i = \mathbf{0}. \tag{3}$$

The second step of the process is to discuss in which case the equation

$$\mathbf{U} \triangleq \sum_{i=0}^{r-1} \eta_i \mathbf{A}_i = \mathbf{0} \tag{4}$$

holds, where $\mathbf{A}_i = (a_i, a_{i+1}, \ldots, a_{i+t-1}) \in \mathbb{Z}^t, i = 0, 1, \ldots, r-1$. The core idea is first to construct a lattice such that $\mathbf{U}$ belongs to it, and then (4) holds if $\mathbf{U}$ is shorter than the shortest non-zero vector of this lattice.

Let $\mathbf{Z}_i = (z_i, z_{i+1}, \ldots, z_{i+t-1}) \in \mathbb{Z}^t, i = 0, 1, \ldots, r-1$. Then by (1) and (3), we can get

$$\mathbf{U} = \sum_{i=0}^{r-1} \eta_i \mathbf{A}_i - 2^{\beta k} \sum_{i=0}^{r-1} \eta_i \mathbf{Y}_i = \sum_{i=0}^{r-1} \eta_i \mathbf{Z}_i$$

with

$$\|\mathbf{U}\| < r\sqrt{t} 2^{\beta k + (r-1)/2 + t(\alpha k + \log r + 1)/(r-t)}. \tag{5}$$

It follows from (2) that for each $i \in \{0, 1, \ldots, r-1\}$, the vector $\mathbf{A}_i$ belongs to the following lattice $L(m, t)$

$$L(m,t) = \begin{pmatrix} 1 & 0 & \cdots & 0 & q_{n,0} & q_{n+1,0} & \cdots & q_{t-1,0} \\ 0 & 1 & \cdots & 0 & q_{n,1} & q_{n+1,1} & \cdots & q_{t-1,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & q_{n,n-1} & q_{n+1,n-1} & \cdots & q_{t-1,n-1} \\ 0 & 0 & \cdots & 0 & m & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & m & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & m \end{pmatrix},$$

and so $\mathbf{U} \in L(m, t)$. Let $\lambda_1(L(m, t))$ be the norm of the shortest non-zero vector in $L(m, t)$. If

$$\|\mathbf{U}\| < \lambda_1(L(m, t)),$$

then $\mathbf{U} = \mathbf{0}$.

Remark 3 By (5) and the Gaussian heuristic assumption, it seems that one can give a heuristic analysis for the selection of r and t. However, the suggestion of r and t provided by the heuristic analysis often differ greatly from the actual selection of r and t because the upper bound of $\mathbf{U}$ in (5) is often overestimated. On the other hand, extensive experiments have shown that by trial and error one can indeed find the correct parameters r and t such that (4) holds. Moreover, the correct parameters r and t that make (4) holds are not unique.

3.2 Constructing congruence equations

As discussed in Subsection 3.1, by correctly choosing the parameters r and t, Equation (4) could hold, which implies that

$$\begin{pmatrix} a_0 & a_1 & \cdots & a_{n-1} \\ a_1 & a_2 & \cdots & a_n \\ \vdots & \vdots & \ddots & \vdots \\ a_{t-1} & a_t & \cdots & a_{t+n-2} \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ \vdots \\ g_{n-1} \end{pmatrix} \equiv \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \bmod m, \tag{6}$$

where

$$g_i = \eta_i + \sum_{j=n}^{r-1} \eta_j q_{j,i}, \quad i = 0, 1, \ldots, n-1, \tag{7}$$

are polynomials in $c_0, c_1, \ldots, c_{n-1}$ with integer coefficients. If $f(x)$ is a primitive polynomial over $\mathbb{Z}/(m)$, then it is proved in [1] that $g_i \equiv 0 \bmod m$ for $i = 0, 1, \ldots, n-1$.

Theorem 4 [1] *Let $f(x)$ be a primitive polynomial over $\mathbb{Z}/(m)$ and let $\underline{a} \in G'(f(x), m)$. Then (6) has only the trivial solution, that is $g_i \equiv 0 \bmod m$ for $i = 0, 1, \ldots, n-1$.*

3.3 Recovering the modulus m

By (7), if $g_i \equiv 0 \bmod m$ for $i = 0, 1, \ldots, n-1$, then there exists an integer u_i such that

$$\eta_i = u_i m - \sum_{j=n}^{r-1} \eta_j q_{j,i}, i = 0, 1, \ldots, n-1.$$

Construct the following two lattices

$$L(g_i) = \begin{pmatrix} m & 0 & 0 & \cdots & 0 \\ -q_{n,i} & 1 & 0 & \cdots & 0 \\ -q_{n+1,i} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -q_{r-1,i} & 0 & 0 & \cdots & 1 \end{pmatrix},$$

$$M(g_i) = \begin{pmatrix} \eta_i^{(0)} & \eta_n^{(0)} & \eta_{n+1}^{(0)} & \cdots & \eta_{r-1}^{(0)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \eta_i^{(d-1)} & \eta_n^{(d-1)} & \eta_{n+1}^{(d-1)} & \cdots & \eta_{r-1}^{(d-1)} \end{pmatrix},$$

where each row vector $\boldsymbol{\eta_i}^{(j)} = (\eta_i^{(j)}, \eta_n^{(j)}, \eta_{n+1}^{(j)}, \ldots, \eta_{r-1}^{(j)})$, $0 \le j \le d-1$, of $M(g_i)$ satisfies that $g_i \equiv 0 \bmod m$. It is clear that the determinant of $L(g_i)$ is m and $M(g_i)$ is a sublattice of $L(g_i)$, and so the determinant of $M(g_i)$ is a multiple of m. If we can collect enough such vectors $\boldsymbol{\eta_i}^{(j)}$, then $M(g_i)$ will become equal to $L(g_i)$, which implies that the unknown modulus m can be recovered by calculating the determinant of $M(g_i)$. This is just the core idea of Sun-Zhu-Zheng's method to recover the modulus m.

3.4 Recovering the coefficients $c_0, c_1, \ldots, c_{n-1}$

By Subsection 3.3, it is known that once the determinant of $M(g_i)$ reaches m, then $M(g_i)$ and $L(g_i)$ in fact are the same lattice. We can now assume, without loss of generality, that the determinant of $M(g_i)$ reaches m. Let $L(g_i)^*$ be the lattice generated by the LLL-reduced basis of $M(g_i)$. Let $L(g_i)^* = (\tau_{u,v})_{0 \le u,v \le r-n}$ and let

$$G_i = \begin{pmatrix} \tau_{0,0} & \tau_{0,1} & K\tau_{0,2} & \cdots & K\tau_{0,r-n} \\ \tau_{1,0} & \tau_{1,1} & K\tau_{1,2} & \cdots & K\tau_{1,r-n} \\ \tau_{2,0} & \tau_{2,1} & K\tau_{2,2} & \cdots & K\tau_{2,r-n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tau_{r-n,0} & \tau_{r-n,1} & K\tau_{r-n,2} & \cdots & K\tau_{r-n,r-n} \end{pmatrix},$$

where $K > m2^{(r-n)/2}$ is a constant. Reduce G_i through the LLL algorithm and denote by $H = (h_{u,v})_{0 \le u,v \le r-n}$ the reduced matrix.

Theorem 5 [1] *Let $L(g_i)^*$ and $L(g_i)$ be defined as above. If $L(g_i)^* = L(g_i)$, then the lattice generated by $(h_{0,0}, h_{0,1}, \ldots, h_{0,r-n})$ and $(h_{1,0}, h_{1,1}, \ldots, h_{1,r-n})$ is equal to the lattice generated by $(m, 0, \ldots, 0)$ and $(-c_i, 1, \ldots, 0)$.*

By Theorem 5, there exists integers u, v such that $uh_{0,1} + vh_{1,1} = 1$, and so

$$c_i = -(uh_{0,0} + vh_{1,0}) \bmod m.$$

Since the values of u and v can be determined by the extended Euclidean algorithm, the coefficient c_i is recovered.

3.5 Recovering the initial state $a_0, a_1, \ldots, a_{n-1}$

Since all parameters and $y_0, y_1, \ldots, y_{n-1}$ are known, recovering $a_0, a_1, \ldots, a_{n-1}$ is equivalent to recovering $z_0, z_1, \ldots, z_{n-1}$. Yang et al. [31] solved this problem by first transforming it into the problem of finding small integer solutions to

systems of linear congruences, and then used the method proposed by Frieze [16] to solve this problem.

Suppose the first d ($d > n$) consecutive truncated digits $y_0, y_1, \ldots, y_{d-1}$ are given. By (1) and (2),

$$\sum_{i=0}^{n-1} q_{j,i} z_i - z_j \equiv 2^{\beta k} \left(y_j - \sum_{i=0}^{n-1} q_{j,i} y_i \right) \bmod m, j = n, \ldots, d-1. \tag{8}$$

It can be seen that the unknown $z_0, z_1, \ldots, z_{d-1}$ is a relatively small integer solution of (8) with each $0 \leq z_i < 2^{\beta k}$. Let

$$L(m,d) = \begin{pmatrix} m & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & m & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m & 0 & 0 & \cdots & 0 \\ q_{n,0} & q_{n,1} & \cdots & q_{n,n-1} & -1 & 0 & \cdots & 0 \\ q_{n+1,0} & q_{n+1,1} & \cdots & q_{n+1,n-1} & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{d-1,0} & q_{d-1,1} & \cdots & q_{d-1,n-1} & 0 & 0 & \cdots & -1 \end{pmatrix}.$$

Denote by λ_d the d-th successive minima of the lattice $L(m,d)$. Frieze et al. [16] have showed that if

$$\alpha k \geq \frac{\log d}{2} + \log \lambda_d + \frac{d+1}{2},$$

then $z_0, z_1, \ldots, z_{d-1}$ can be recovered by solving the small integer solutions of (8) using the method in [16].

4 Main results

In this section, we propose a more effective and data-saving method for predicting truncated MRGs with unknown parameters compared to Sun-Zhu-Zheng's method [1]. Specifically, we adopt the resultant, the Chinese Remainder Theorem and the idea of recovering p-adic coordinates of the coefficients layer by layer, and Kannan's embedding technique [41] to recover the modulus, the coefficients and the initial state, respectively.

4.1 Searching linear relations

As a preparation for recovering the unknown parameters, we still need to find a set of linear relations for $\underline{a}^{(N)} = (a_0, a_1, \ldots, a_{N-1})$. However, different from [1], we do not need to find the vectors $\boldsymbol{\eta} = (\eta_0, \eta_1, \ldots, \eta_{r-1})$ such that (3) and (4) both hold. Actually, Equation (4) is our ultimate goal while Equation (3)

is unnecessary. Observe that if (4) holds, then

$$\mathbf{0} = \sum_{i=0}^{r-1} \eta_i \mathbf{A}_i = 2^{\beta k} \sum_{i=0}^{r-1} \eta_i \mathbf{Y}_i + \sum_{i=0}^{r-1} \eta_i \mathbf{Z}_i,$$

and so

$$\sum_{i=0}^{r-1} \eta_i \mathbf{Y}_i = -2^{-\beta k} \sum_{i=0}^{r-1} \eta_i \mathbf{Z}_i.$$

Since $|z_{i+j}| < 2^{\beta k}$, it follows that

$$\left\| \sum_{i=0}^{r-1} \eta_i \mathbf{Y}_i \right\| \leq \sum_{i=0}^{r-1} \left\| -2^{-\beta k} \eta_i \mathbf{Z}_i \right\| \leq \sqrt{t} \sum_{i=0}^{r-1} |\eta_i| .$$

This means $\sum_{i=0}^{r-1} \eta_i \mathbf{Y}_i$ is a short vector, which inspires us to search a short vector of the lattice rather than a vector that makes (3) holds.

Construct the following lattice

$$L' = \begin{pmatrix} \mathbf{Y}_0 & 1 & 0 & \cdots & 0 \\ \mathbf{Y}_1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{Y}_{r-1} & 0 & 0 & \cdots & 1 \end{pmatrix},$$

and obviously, the $(r+t)$-dimensional vector

$$\mathbf{W} = (w_0, \ldots, w_{t-1}, w_t, \ldots, w_{t+r-1}) = \left(\sum_{i=0}^{r-1} \eta_i \mathbf{Y}_i, \eta_0, \eta_1, \ldots, \eta_{r-1} \right) \in L'.$$

If $\boldsymbol{\eta}$ is a vector that makes (4) holds, then $\sum_{i=0}^{r-1} \eta_i \mathbf{Y}_i$ is a short vector, and so is $\mathbf{W}$. Therefore, the desired vectors $\boldsymbol{\eta} = (\eta_0, \eta_1, \ldots, \eta_{r-1})$ that make (4) holds can be obtained by applying lattice reduction algorithms to the lattice L'.

Compared with [1], we construct the lattice L' without multiplying $\mathbf{Y}_i$ by a big integer, this change can not only accelerate single call to lattice reduction algorithm by shrinking the data size of the lattice, but also greatly reduce the number of truncated digits required for recovering the unknown parameters.

It is obvious that the choices of r and t are of great importance. Unfortunately, as in [1], we can not provide a valid heuristic analysis for the choices of r and t. In actual experiments, we usually perform a lot of experiments to search stable values of r and t, that is, in every experiment, the vector $\boldsymbol{\eta} = (\eta_0, \eta_1, \ldots, \eta_{r-1})$ that makes (4) holds always can be found by lattice reduction algorithms with such values of r, t.

Remark 4 If the values of r and t are chosen correctly, a single call to lattice reduction algorithm can output multiple sets of $\eta_0, \eta_1, \ldots, \eta_{r-1}$ such that (4) holds.

As discussed above, by correctly choosing the values of r and t, we can get $\eta_0, \eta_1, \ldots, \eta_{r-1}$ output by the lattice reduction algorithm such that (4) holds, that is,

$$\begin{cases} \eta_0 a_0 + \eta_1 a_1 + \cdots + \eta_{r-1} a_{r-1} = 0 \\ \eta_0 a_1 + \eta_1 a_2 + \cdots + \eta_{r-1} a_r = 0 \\ \qquad\qquad \vdots \\ \eta_0 a_{t-1} + \eta_1 a_t + \cdots + \eta_{r-1} a_{r+t-2} = 0 \end{cases}.$$

Obviously, it means that we have got a polynomial

$$F(x) = \eta_{r-1} x^{r-1} + \cdots + \eta_1 x + \eta_0,$$

which annihilates the sequence $\underline{a}^{(r+t-1)} = (a_0, a_1, \ldots, a_{r+t-2})$. If we can obtain enough such polynomials, then the modulus and the coefficients can be recovered by the methods in Section 4.2 and 4.3, respectively.

4.2 Recovering the modulus m

In this subsection, we will prove that the modulus m can be recovered by computing the resultants of the polynomials in $I_m(\underline{a})$ obtained in Section 4.1, where $I_m(\underline{a})$ denotes the set of polynomials over $\mathbb{Z}/(m)$ that annihilate $\underline{a}$. First, we will prove that this method is feasible when the modulus m is a prime power. Then we will extend the method to the case that the modulus m is a general positive integer by the Chinese Remainder Theorem.

Theorem 6 [45] *Let p^e be a prime power and let $\underline{a}$ be a linear recurring sequence over $\mathbb{Z}/(p^e)$. If for every $i \in \{1, 2, \ldots, e\}$, the monic polynomial $f_i(x) \in \mathbb{Z}/(p^e)[x]$ satisfies that $f_i(x) \bmod p^i$ is a minimal polynomial of $\underline{a} \bmod p^i$, then*

$$I_{p^e}(\underline{a}) = (p^{e-1} \cdot f_1(x), \ldots, p \cdot f_{e-1}(x), f_e(x)),$$

where $(p^{e-1} \cdot f_1(x), \ldots, p \cdot f_{e-1}(x), f_e(x))$ denotes the ideal generated by $p^{e-1} \cdot f_1(x), \ldots, p \cdot f_{e-1}(x), f_e(x)$.

By Theorem 6, we can get the following Corollary 1.

Corollary 1 *Let p^e be a prime power. If $f(x)$ is a primitive polynomial over $\mathbb{Z}/(p^e)$ and $\underline{a}$ is a primitive sequence generated by $f(x)$, then $I_{p^e}(\underline{a}) = (f(x))$. That is, for any $F(x) \in I_{p^e}(\underline{a})$, $f(x)$ is a divisor of $F(x)$ over $\mathbb{Z}/(p^e)$.*

Proof Since $f(x)$ is a primitive polynomial over $\mathbb{Z}/(p^e)$ and $\underline{a}$ is generated by $f(x)$, it follows that, for every $i \in \{1, 2, \ldots, e-1\}$, $f(x) \bmod p^i$ is a primitive polynomial over $\mathbb{Z}/(p^i)$ and $\underline{a} \bmod p^i$ is a primitive sequence generated by $f(x) \bmod p^i$. So by Theorem 6, we have

$$I_{p^e}(\underline{a}) = (p^{e-1} \cdot f(x), \ldots, p \cdot f(x), f(x)) = (f(x)).$$

That is, for any $F(x) \in I_{p^e}(\underline{a})$, $f(x)$ is a divisor of $F(x)$ over $\mathbb{Z}/(p^e)$. □

Remark 5 In Corollary 1, the condition that $\underline{a}$ is a primitive sequence over $\mathbb{Z}/(p^e)$ can be relaxed to the condition that the minimal polynomial of $\underline{a}$ over $\mathbb{Z}/(p^e)$ is unique.

By Theorem 2 and Corollary 1, we can easily get the result we want when m is a prime power.

Theorem 7 *Let p^e be a prime power. If $f(x)$ is a primitive polynomial of degree n over $\mathbb{Z}/(p^e)$ and $\underline{a}$ is a primitive sequence generated by $f(x)$ over $\mathbb{Z}/(p^e)$, then for any two polynomials $F^{(1)}(x), F^{(2)}(x) \in I_{p^e}(\underline{a})$, we have*

$$p^{en} \mid R(F^{(1)}(x), F^{(2)}(x))$$

over $\mathbb{Z}$.

Proof By Corollary 1, the primitive polynomial $f(x)$ is a common divisor of $F^{(1)}(x)$ and $F^{(2)}(x)$. So the result follows directly from Theorem 2. □

Next we will generalize Theorem 7 to the general case that the modulus is an integer greater than 1 by the Chinese Remainder Theorem. We first generalize Corollary 1 to the general case.

Lemma 3 *Let m be an integer greater than 1. If $f(x)$ is a primitive polynomial over $\mathbb{Z}/(m)$ and $\underline{a}$ is a primitive sequence generated by $f(x)$ over $\mathbb{Z}/(m)$, then for any $F(x) \in I_m(\underline{a})$, $f(x)$ is a divisor of $F(x)$ over $\mathbb{Z}/(m)$.*

Proof Let $m = p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s}$ be the canonical factorization of m. By Definition 3, for every $i \in \{1, 2, \ldots, s\}$, $f(x) \bmod p_i^{e_i}$ is a primitive polynomial over $\mathbb{Z}/(p_i^{e_i})$ and $\underline{a} \bmod p_i^{e_i}$ is a primitive sequence generated by $f(x) \bmod p_i^{e_i}$. For any $F(x) \in I_m(\underline{a})$, we have $F(x) \bmod p_i^{e_i} \in I_{p_i^{e_i}}(\underline{a})$, where $I_{p_i^{e_i}}(\underline{a})$ denotes the set of polynomials over $\mathbb{Z}/(p_i^{e_i})$ that annihilate $\underline{a} \bmod p_i^{e_i}$. Then by Corollary 1, $f(x) \bmod p_i^{e_i}$ is a divisor of $F(x) \bmod p_i^{e_i}$ over $\mathbb{Z}/(p_i^{e_i})$, i.e.,

$$F(x) \equiv 0 \bmod (f(x), p_i^{e_i}),$$

which means that $f(x)$ is a divisor of $F(x)$ over $\mathbb{Z}/(p_i^{e_i})$. According to Chinese Remainder Theorem, we have

$$F(x) \equiv 0 \bmod (f(x), m).$$

So, for every $F(x) \in I_m(\underline{a})$, $f(x)$ is a divisor of $F(x)$ over $\mathbb{Z}/(m)$. □

By Theorem 3 and Lemma 3, we immediately get the following Theorem 8, which is the main result of this subsection.

Theorem 8 *Let m be an integer greater than 1. If $f(x)$ is a primitive polynomial of degree n over $\mathbb{Z}/(m)$ and $\underline{a}$ is a primitive sequence generated by $f(x)$ over $\mathbb{Z}/(m)$, then for any two polynomials $F^{(1)}(x), F^{(2)}(x) \in I_m(\underline{a})$, we have*

$$m^n \mid R(F^{(1)}(x), F^{(2)}(x))$$

over $\mathbb{Z}$.

Theorem 8 implies that we can recover the modulus m by computing the greatest common divisor of resultants of polynomials in $I_m(\underline{a})$. Experiments have shown that the modulus m can be recovered in most cases if three pairs of nonzero resultants are given. Notice that if $R(F^{(1)}(x), F^{(2)}(x)) = 0$, we cannot get any information about the modulus, so next we need to discuss the proportion of this case.

We have carried out 500 experiments. In each experiment, we first randomly choose a 31-bit integer m and a primitive polynomial $f(x)$ of degree $n = 16$ over $\mathbb{Z}/(m)$, next we randomly choose the initial state and generate the sequence $\underline{a}$ by the primitive polynomial $f(x)$. Let y_i be the 17 high-order bits of a_i. We choose $r = 175, t = 65$, then by Section 4.1, we can obtain 110 polynomials of $I_m(\underline{a})$. Take these polynomials in pairs and compute their resultants, experiments show that the proportion of $R(F^{(1)}(x), F^{(2)}(x)) = 0$ in the total 5995 resultants is about 0.1409%, which is so small that even if the computed resultant is equal to 0, we still have enough options to choose another couple of polynomials. So our method works effectively in recovering the modulus [1].

4.3 Recovering the coefficients $c_0, c_1, \ldots, c_{n-1}$

In this subsection, we show how to recover the coefficients $c_0, c_1, \ldots, c_{n-1}$, which is equivalent to recover the primitive polynomial $f(x) = x^n - c_{n-1}x^{n-1} - \cdots - c_0$.

Let $m = p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s}$ be the canonical factorization of the modulus m. Then, by the Chinese Remainder Theorem, to recover $f(x)$, it suffices to recover $f(x) \bmod p_1^{e_1}, \ldots, f(x) \bmod p_s^{e_s}$. For the sake of brevity, we first assume that $f(x)$ is a primitive polynomial over $\mathbb{Z}/(p^e)$ and $\underline{a}$ is a primitive sequence generated by $f(x)$ over $\mathbb{Z}/(p^e)$, where p^e is a prime power.

Let $F^{(i)}(x) = \eta_{r-1}^{(i)} x^{r-1} + \cdots + \eta_1^{(i)} x + \eta_0^{(i)} \in \mathbb{Z}/(p^e)[x]$, $i = 1, 2, \ldots, w$, denote the polynomials that annihilate $\underline{a}$ obtained in Section 4.1. It follows from Corollary 1 that $f(x)$ is a common divisor of $F^{(i)}(x)$, that is, for every $i \in \{1, 2, \ldots, w\}$, there exists $G^{(i)}(x) \in \mathbb{Z}/(p^e)[x]$ whose leading coefficient is equal to $\eta_{r-1}^{(i)}$ such that

$$F^{(i)}(x) = f(x)G^{(i)}(x) \tag{9}$$

[1] During the sixth (2021) National Crypto-Math Challenge, Jing-Hui Wang, Ming-Hao Xu and Xiao-Yue Hu (mentor: Zheng-Fu Lu) from Yunnan University also independently pointed that the modulus m can be recovered by computing the resultants of polynomials in $I_m(\underline{a})$.

over $\mathbb{Z}/(p^e)$. It is worth noting that the coefficients of $F^{(i)}(x)$ may not be coprime with p, which causes the Euclidean algorithm for directly computing the greatest common divisor over $\mathbb{Z}/(p^e)$ to fail in some cases. To overcome this situation, we instead to recover $f(x)$ by recovering its p-adic coordinates layer by layer. In details, let the p-adic expansion of $F^{(i)}(x), f(x)$ and $G^{(i)}(x)$ be as follows

$$\begin{aligned} F^{(i)}(x) &= [F^{(i)}(x)]_0 + [F^{(i)}(x)]_1 p + \cdots + [F^{(i)}(x)]_{e-1} p^{e-1} \\ &= F^{(i)}(x) \bmod p + [F^{(i)}(x)]_1 p + \cdots + [F^{(i)}(x)]_{e-1} p^{e-1}, \\ f(x) &= [f(x)]_0 + [f(x)]_1 p + \cdots + [f(x)]_{e-1} p^{e-1} \\ &= f(x) \bmod p + [f(x)]_1 p + \cdots + [f(x)]_{e-1} p^{e-1}, \\ G^{(i)}(x) &= [G^{(i)}(x)]_0 + [G^{(i)}(x)]_1 p \cdots + [G^{(i)}(x)]_{e-1} p^{e-1} \\ &= G^{(i)}(x) \bmod p + [G^{(i)}(x)]_1 p \cdots + [G^{(i)}(x)]_{e-1} p^{e-1}, \end{aligned} \tag{10}$$

where $[F^{(i)}(x)]_j, [f(x)]_j, [G^{(i)}(x)]_j \in \mathbb{Z}/(p)[x]$ for $0 \le j \le e-1$. Since $\mathbb{Z}/(p)$ is a finite field, $f(x) \bmod p$ can be firstly recovered by computing the greatest common divisor of $F^{(1)}(x) \bmod p, \ldots, F^{(w)}(x) \bmod p$ with the Euclidean algorithm over $\mathbb{Z}/(p)$. If $e = 1$, then $f(x)$ has been recovered. Otherwise, by (9) and (10), we have

$$\begin{aligned} F^{(i)}(x) \bmod p^l &\equiv f(x) \bmod p^l \cdot G^{(i)}(x) \bmod p^l \\ &\equiv (f(x) \bmod p^{l-1} + [f(x)]_{l-1} p^{l-1}) \\ &\quad \cdot (G^{(i)}(x) \bmod p^{l-1} + [G^{(i)}(x)]_{l-1} p^{l-1}) \bmod p^l, \end{aligned} \tag{11}$$

where $2 \le l \le e$. Since now $f(x) \bmod p$ has be recovered and $F^{(i)}(x) \bmod p$ is known, it follows immediately that $G^{(i)}(x) \bmod p = F^{(i)}(x) \bmod p / f(x) \bmod p$. By (11), when $f(x) \bmod p^{l-1}$ and $G^{(i)}(x) \bmod p^{l-1}$ are known, recovering $f(x) \bmod p^l$ and $G^{(i)}(x) \bmod p^l$ means recovering $[f(x)]_{l-1}$ and $[G^{(i)}(x)]_{l-1}$. In this case we have

$$\begin{aligned} &\frac{F^{(i)}(x) \bmod p^l - f(x) \bmod p^{l-1} \cdot G^{(i)}(x) \bmod p^{l-1}}{p^{l-1}} \\ &\equiv f(x) \bmod p^{l-1} \cdot [G^{(i)}(x)]_{l-1} + G^{(i)}(x) \bmod p^{l-1} \cdot [f(x)]_{l-1} \\ &\equiv [f(x)]_0 \cdot [G^{(i)}(x)]_{l-1} + [G^{(i)}(x)]_0 \cdot [f(x)]_{l-1} \bmod p. \end{aligned}$$

When $\gcd([f(x)]_0, [G^{(i)}(x)]_0) = 1$, we can compute the inverse of $[G^{(i)}(x)]_0$ modulo $[f(x)]_0$ by the Extended Euclidean algorithm over $\mathbb{Z}/(p)$, which we denote by $[G^{(i)}(x)]_0^{-1}$. Then $[f(x)]_{l-1}$ can be computed by

$$[G^{(i)}(x)]_0^{-1} \cdot \frac{F^{(i)}(x) \bmod p^l - f(x) \bmod p^{l-1} \cdot G^{(i)}(x) \bmod p^{l-1}}{p^{l-1}} \bmod (p, [f(x)]_0). \tag{12}$$

Similarly, we can also compute $[G^{(i)}(x)]_{l-1}$ by

$$[f(x)]_0^{-1} \cdot \frac{F^{(i)}(x) \bmod p^l - f(x) \bmod p^{l-1} \cdot G^{(i)}(x) \bmod p^{l-1}}{p^{l-1}} \bmod (p, [G^{(i)}(x)]_0), \tag{13}$$

where $[f(x)]_0^{-1}$ denotes the inverse of $[f(x)]_0$ modulo $[G^{(i)}(x)]_0$ over $\mathbb{Z}/(p)$. Repeat the above process until $f(x)$ is totally recovered. It is worth noting that $[f(x)]_0$ is irreducible, so $\gcd([f(x)]_0, [G^{(i)}(x)]_0) = 1$ will hold with great probability. In other words, this method works in the vast majority of cases.

To sum up, for $m = p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s}$, we can get $f(x) \bmod p_j$ by computing the greatest common divisor of $F^{(i)}(x) \bmod p_j$, then get $f(x) \bmod p_j^{e_j}$ by the Extended Euclidean algorithm over finite fields layer by layer, finally get $f(x)$ by the Chinese Remainder Theorem. The detailed procedures are as follows.

Step 1: For every $j \in \{1, 2, \ldots, s\}$, compute the greatest common divisor of $F^{(i)}(x) \bmod p_j, i = 1, 2, \ldots, w$, by the Euclidean algorithm until the degree of this divisor reaches n. Next multiply this divisor by the inverse of its leading coefficient modulo p_j to make it become a monic polynomial. Then we get $f(x)$ modulo p_j, i.e., $f(x) \bmod p_j$.

Step 2: For every $j \in \{1, 2, \ldots, s\}$, compute $f(x)^{-1}_{\bmod p_j}$ over the finite field $\mathbb{Z}/(p_j)$ by the Extended Euclidean algorithm and then compute $[f(x)]_{l-1}$ by (12) from $l = 2$ to $l = e_j$. After this, we can get all components of the p_j-adic expansion of $f(x) \bmod p_j^{e_j}$ and then get $f(x) \bmod p_j^{e_j}$.

Step 3: Since $f(x) \bmod p_1^{e_1}, \ldots, f(x) \bmod p_s^{e_s}$ are all known, then by the Chinese Remainder Theorem,

$$f(x) \equiv f(x) \bmod p_1^{e_1} P_1 P_1^{-1} + \cdots + f(x) \bmod p_s^{e_s} P_s P_s^{-1} \bmod m,$$

where $P_i = m/p_i^{e_i}$ and $P_i P_i^{-1} \equiv 1 \bmod p_i^{e_i}$. Thus the coefficients $c_0, c_1, \ldots, c_{n-1}$ can be successfully recovered.

Remark 6 In Step 2 above, we can also get $f(x) \bmod p_j^{e_j}$ by solving linear equations over finite fields layer by layer. However, the complexity of the most efficient algorithm currently, Optimized CW-like algorithm, for solving $O(n)$ linear equations is about $O(n^{2.373})$ [46]. While the complexity of the Extended Euclidean algorithm is only $O(n(\log n)^2 \log\log n)$ (See Chapter 11.1 of [47] and [48] for the detailed analysis), which behaves better than Optimized CW-like algorithm.

4.4 Recovering the initial state $a_0, a_1, \ldots, a_{n-1}$

We have recovered the modulus and the coefficients, and then we need to recover the initial state $a_0, a_1, \ldots, a_{n-1}$. Since the truncated digits $y_0, y_1, \ldots, y_{n-1}$ are known, by (1), we only need to recover $z_0, z_1, \ldots, z_{n-1}$, which is proved to be equivalent to finding small integer solutions to systems of linear congruences, i.e., the Short Integer Solutions problem (SIS) [31]. In

this subsection, we use Kannan's embedding technique to transform this SIS problem into SVP.

Suppose the first d ($d > n$) truncated digits $y_0, y_1, \ldots, y_{d-1}$ are given, then by (8), we have

$$z_j = \sum_{i=0}^{n-1} q_{j,i} z_i - b_j + k_j m, j = n, \ldots, d-1, \tag{14}$$

where $b_j = 2^{\beta k}\left(y_j - \sum_{i=0}^{n-1} q_{j,i} y_i\right)$ and $k_j \in \mathbb{Z}$. Let $\widehat{z}_j = z_j - 2^{\beta k-1}$. Then

$$-2^{\beta k-1} \leq \widehat{z}_j < 2^{\beta k-1}.$$

By Kannan's embedding technique [41], we construct the following lattice

$$L'(m,d) = \begin{pmatrix} 2^{\beta k-1} & 2^{\beta k-1} & \cdots & 2^{\beta k-1} & b_n + 2^{\beta k-1} & \cdots & b_{d-1} + 2^{\beta k-1} \\ 0 & 1 & \cdots & 0 & q_{n,0} & \cdots & q_{d-1,0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & q_{n,n-1} & \cdots & q_{d-1,n-1} \\ 0 & 0 & \cdots & 0 & m & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & m \end{pmatrix}.$$

The target vector $\mathbf{v} = (-2^{\beta k-1}, \widehat{z}_0, \ldots, \widehat{z}_{n-1}, \widehat{z}_n, \ldots, \widehat{z}_{d-1}) \in L'(m,d)$. Such construction balances each coordinate of the target vector[2] so that the upper bound for the absolute values of all coordinates is $2^{\beta k-1}$, and so

$$\|\mathbf{v}\| \leq 2^{\beta k-1}\sqrt{d+1}.$$

By Lemma 1,

$$\lambda_1(L'(m,d)) \approx \sqrt{\frac{d+1}{2\pi e}}\,(\det L)^{1/(d+1)} = \sqrt{\frac{d+1}{2\pi e}}\,2^{\frac{\beta k-1}{d+1}} m^{\frac{d-n}{d+1}}.$$

Heuristically, if

$$2^{\beta k-1}\sqrt{d+1} \leq \sqrt{\frac{d+1}{2\pi e}}\,2^{\frac{\beta k-1}{d+1}} m^{\frac{d-n}{d+1}},$$

that is,

$$d \geq \frac{n\log m + 0.5 \cdot \log 2\pi e}{\log m - (\beta k - 1) - 0.5 \cdot \log 2\pi e}, \tag{15}$$

then $\mathbf{v} = (-2^{\beta k-1}, \widehat{z}_0, \ldots, \widehat{z}_{n-1}, \widehat{z}_n, \ldots, \widehat{z}_{d-1})$ could be recovered by reducing $L'(m,d)$ through the lattice reduction algorithm. If this is done, we can

[2] During the sixth (2021) National Crypto-Math Challenge, Jia-Le Fang (mentor: Hua Zhong) from Hangzhou Dianzi University also independently proposed the strategy of balancing each coordinate of the target vector. They also obtained an inequality similar to (15).

recover $z_0, z_1, \ldots, z_{n-1}$ and then the initial state $a_0, a_1, \ldots, a_{n-1}$ by simple computation.

5 Experimental results

We have performed a lot of experiments to test the correctness and effectiveness of our new method. In order to show our method is superior to [1], here we only list the experimental results of the same sequence, ZUC's driving sequence. All the experiments are performed on our personal computer (Windows 10, Processor Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz, 8 Core(s), RAM: 32.0 GB (31.8 GB usable)). The lattice reduction algorithm used is the BKZ algorithm in the NTL 11.4.3 library [49]. The reduction parameter of the BKZ algorithm is set to $\delta = 0.99$ and the blocksize is set to 20.

The characteristic polynomial of ZUC's driving sequence is a primitive polynomial of degree $n = 16$ over $\mathbb{Z}/(2^{31} - 1)$, that is

$$f(x) = x^{16} - (2^{15}x^{15} + 2^{17}x^{13} + 2^{21}x^{10} + 2^{20}x^4 + 2^8 + 1).$$

We choose the same initial state as [1] that

$$\begin{aligned} a_0 &= 5113033, a_1 = 101513454, a_2 = 342517303, a_3 = 602278014, \\ a_4 &= 545182365, a_5 = 2109832480, a_6 = 83899935, a_7 = 1353806595, \\ a_8 &= 56274808, a_9 = 220365776, a_{10} = 1900047307, a_{11} = 1287703311, \\ a_{12} &= 754976298, a_{13} = 344592060, a_{14} = 2072123851, a_{15} = 1906585150, \end{aligned}$$

and then generate the sequence $\underline{a} = (a_i)_{i \geq 0} \in G(f(x), 2^{31} - 1)$.

In this experiment, suppose that the modulus m, the coefficients $c_0, c_1, \ldots, c_{15}$ and the initial state $a_0, a_1, \ldots, a_{15}$ are all unknown and let y_i be the 17 high-order bits of a_i, then given the first N truncated digits $y_0, y_1, \ldots, y_{N-1}$, we need to recover the modulus m, the coefficients $c_0, c_1, \ldots, c_{15}$ and the initial state $a_0, a_1, \ldots, a_{15}$. The detailed experimental processes are as follows.

Step 1: After massive repeated experiments, we choose $r = 175, t = 65$ and then use the method in Section 4.1 to get a couple of polynomials in $I_m(\underline{a})$, that is, the vector $\boldsymbol{\eta} = (\eta_0, \eta_1, \ldots, \eta_{r-1})$ that makes (4) hold. In this step, we construct a 175×240 lattice and call the BKZ algorithm to reduce this lattice in 9.4 minutes. Actually, a single call to the BKZ algorithm can provide 110 such vectors $\boldsymbol{\eta}$ and it is sufficiently enough to recover the modulus and the coefficients.

Step 2: By Section 4.2, we can recover the modulus m by computing the resultant of the polynomials $F^{(i)}(x)$ obtained in Step 1. In this step, a small number of such polynomials are enough to recover the modulus. In our experiment, we only need to use 3 polynomials $F^{(1)}(x), F^{(2)}(x), F^{(3)}(x)$ to

compute the resultants of these polynomials in pairs first, that is

$$\begin{aligned} res_1 &= R(F^{(1)}(x), F^{(2)}(x)), \\ res_2 &= R(F^{(2)}(x), F^{(3)}(x)), \\ res_3 &= R(F^{(1)}(x), F^{(3)}(x)), \end{aligned}$$

and then to compute the greatest common divisor of res_1, res_2 and res_3. After this we get m^n and then m by taking the n-th root of m^n. In our experiment, we can also factor m^n to get m directly because m^n is not too huge. The whole computation can be finished in $2.625s$.

Step 3: Since the prime modulus m is known, we can calculate the inverse of $\eta_{r-1}^{(i)}$ modulo m and then get these characteristic polynomials as follows

$$F_*^{(i)}(x) = x^{r-1} + (\eta_{r-1}^{(i)})^{-1} \cdot (\eta_{r-2}^{(i)} x^{r-2} + \cdots + \eta_1^{(i)} x + \eta_0^{(i)}), i = 1, 2, \ldots, 110.$$

In our experiment, the degree of the greatest common divisor of $F_*^{(1)}(x)$ and $F_*^{(2)}(x)$ reaches n. So we compute the greatest common divisor of $F_*^{(1)}(x)$ and $F_*^{(2)}(x)$, then we get $f(x)$, that is, we recover all the coefficients. This step costs about $0.075s$.

Step 4: When the modulus m and the coefficients $c_0, c_1, \ldots, c_{15}$ are known, we use the method in Section 4.4 to recover the initial state $a_0, a_1, \ldots, a_{15}$. By (15), the number of truncated digits d needed in this experiment satisfies $d > 31.22$. Because lattice reduction algorithms usually perform better in practice, so such estimation is not entirely accurate. But this estimation still offers useful advice that we can choose d around this estimation to test its correctness. For example, in the practical experiment, we can successfully recover the initial state by using 30 truncated digits but fail when only using 29 truncated digits, which illustrates that (15) offers useful instructions on the number of truncated digits. To be specific, we use the first 30 truncated digits to construct the 31×31 lattice $L'(m, d)$, by reducing this lattice through the BKZ algorithm, we can recover the initial state in $0.26s$.

In table 1, we compare our method with Sun-Zhu-Zheng's method [1]. As can be seen from this table that, our method only uses 239 truncated digits to recover all unknown parameters and the whole process can be executed in 10 minutes while Sun-Zhu-Zheng's method needs 404 truncated digits and 40 hours to perform the experiment. In addition, no matter in what kind of cases, the modulus is known, the modulus and the coefficients are all known or all parameters are unknown, our method shows more superiority than Sun-Zhu-Zheng's method. To be specific, to find the polynomials annihilating $\underline{a}$ in this experiment, Sun-Zhu-Zheng's method needs to call the BKZ algorithm 71 times to reduce 265×335 lattices and cost 36 hours while our method only needs a single call to the BKZ algorithm to reduce a 175×240 lattice which takes no more than 10 minutes. For recovering the modulus and the coefficients, Sun-Zhu-Zheng's method spends about 3.5 hours in running the

Table 1 The comparison of our method with Sun-Zhu-Zheng's method

		Sun-Zhu-Zheng's method	Our method
Search linear relations	Size of lattice	265×335	175×240
	Calls to BKZ	71	1
	Time of single call	30 min	9.4 min
	Total time	$36h$	9.4 min
Recover the modulus	Size of lattice	252×250	–
	Calls to BKZ	16	–
	Time of single call	11 min	–
	Total time	$3h$	$2.625s$
Recover the coefficients	Size of lattice	250×250	–
	Calls to BKZ	16	–
	Time of single call	2 min	–
	Total time	30 min	$0.075s$
Recover the initial state	Size of lattice	35×35	31×31
	Calls to BKZ	1	1
	Time of single call	$3s$	$0.26s$
	Total time	$3s$	$0.26s$
Total time		$40h$	10 min
Number of truncated digits		404	239

BKZ algorithm, and our new method costs only $3s$ to compute the parameters without any lattice reduction. In the last step, Sun-Zhu-Zheng's method requires at least 35 truncated digits while our method only need 30 truncated digits. Moreover, Sun-Zhu-Zheng's method fails to recover the initial state if the number of bits of y_i is 4, while our method can still sucessfully recover the initial state. Therefore, our improvements are comprehensive in terms of the running time or the number of truncated digits required.

We provide the source code in https://pastebin.com/nAQBjrTB.

6 Conclusion

In this paper, we improve Sun-Zhu-Zheng's method on the predictability of truncated MRGs

$$a_{i+n} \equiv c_{n-1}a_{i+n-1} + c_{n-2}a_{i+n-2} + \cdots + c_0 a_i \bmod m, i = 0, 1, 2, \ldots.$$

Assume that the modulus m, the coefficients $c_0, c_1, \ldots, c_{n-1}$ and the initial state $a_0, a_1, \ldots, a_{n-1}$ are all unknown. If the first N high-order-truncated digits $y_0, y_1, \ldots, y_{N-1}$ of $a_0, a_1, \ldots, a_{N-1}$ are known, we adopt the resultant, the Chinese Remainder Theorem and the idea of recovering p-adic coordinates of the coefficients layer by layer, and Kannan's embedding technique to recover the modulus, the coefficients and the initial state, respectively. Compared with [1], no matter in what kind of cases, our method shows more superiority in

terms of the running time or the number of truncated digits required. Extensive experiments have confirmed the correctness and effectiveness of our new method. Unfortunately, as in [1], we can not provide a valid heuristic analysis for the choices of r and t. As a result, it will waste a lot of time before we find the correct r and t by trial and error in practice. Whether there is a valid heuristic analysis for the choices of r and t is still an open problem worthy of further study.

Acknowledgments. We thank all the members of the organizing committee of National Crypto-Math Challenge and Topsec Technologies Group Inc. for providing a platform for communication and discussion, which finally results in the born of this paper. This work was supported by NSF of China (No. 61872383) and by the Fundamental Research Funds for the Central Universities (Grant No.2021RC29) and by National Key R&D Program of China (Grant No. 2021YFB3100200).

Appendix A

Proof of Theorem 2 Let $R = GR(p^e, p^{en})$. Then by Remark 2 we can let $f(x) = \prod_{i=1}^{n}(x - \xi_i)$, where $\xi_1, \xi_2, \ldots, \xi_n \in R$, and so $g(x)$ and $h(x)$ can be represented as follows

$$g(x) = g_1(x) \cdot \prod_{i=1}^{n}(x - \xi_i),$$

$$h(x) = h_1(x) \cdot \prod_{i=1}^{n}(x - \xi_i),$$

where $g_1(x), h_1(x) \in \mathbb{Z}/(p^e)[x]$.

Consider the polynomial ring $R[x, y_1, \ldots, y_n, z_1, \ldots, z_n]$ and the polynomials in this ring

$$\begin{aligned}\widetilde{g}(x, y_1, \ldots, y_n) &= g_1(x) \cdot \prod_{i=1}^{n}(x - y_i) \\ &= a_u x^u + a_{u-1}(y_1, \ldots, y_n)x^{u-1} + \cdots + a_0(y_1, \ldots, y_n), \\ \widetilde{h}(x, z_1, \ldots, z_n) &= h_1(x) \cdot \prod_{i=1}^{n}(x - z_i) \\ &= b_v x^v + b_{v-1}(z_1, \ldots, z_n)x^{v-1} + \cdots + b_0(z_1, \ldots, z_n),\end{aligned}$$

where $a_0(y_1, \ldots, y_n), \ldots, a_{u-1}(y_1, \ldots, y_n)$ are polynomials in $y_1, \ldots, y_n$ over R and $b_0(z_1, \ldots, z_n), \ldots, b_{v-1}(z_1, \ldots, z_n)$ are polynomials in $z_1, \ldots, z_n$ over R. Substitute x by y_j for $j = 1, 2, \ldots, n$, then

$$\widetilde{g}(y_j, y_1, \ldots, y_n) = g_1(y_j) \cdot \prod_{i=1}^{n}(y_j - y_i) = 0,$$

$$\widetilde{h}(y_j, z_1, \ldots, z_n) = h_1(y_j) \cdot \prod_{i=1}^{n}(y_j - z_i).$$

The resultant of $\widetilde{g}$ and $\widetilde{h}$ with respect to the variable x is denoted by $R_x(\widetilde{g},\widetilde{h})$, which is defined by

$$\left|\begin{array}{ccccccc} a_u & \cdots & a_0(y_1,\ldots,y_n) & 0 & & \cdots & 0 \\ 0 & a_u & \cdots & a_0(y_1,\ldots,y_n) & 0 & \cdots & 0 \\ \vdots & & & & & & \vdots \\ 0 & \cdots & a_u & & \cdots & & a_0(y_1,\ldots,y_n) \\ b_v & \cdots & & b_0(z_1,\ldots,z_n) & 0 & \cdots & 0 \\ 0 & b_v & \cdots & & b_0(z_1,\ldots,z_n) & \cdots & 0 \\ \vdots & & & & & & \vdots \\ 0 & \cdots & b_v & & \cdots & & b_0(z_1,\ldots,z_n) \end{array}\right| \begin{array}{l} \left.\begin{array}{l}\\\\\\\end{array}\right\} v \\ \left.\begin{array}{l}\\\\\\\end{array}\right\} u \end{array}.$$

Multiple the first column of $R_x(\widetilde{g},\widetilde{h})$ by y_j^{u+v-1}, the second column by y_j^{u+v-2} until the penultimate column by y_j, and then add all of them to the last column. Now we can get a new determinant which is equal to $R_x(\widetilde{g},\widetilde{h})$ and its last column is

$$\begin{pmatrix} y_j^{v-1}0 \\ y_j^{v-2}0 \\ \vdots \\ 0 \\ y_j^{u-1}\widetilde{h}(y_j,z_1,\ldots,z_n) \\ y_j^{u-2}\widetilde{h}(y_j,z_1,\ldots,z_n) \\ \vdots \\ \widetilde{h}(y_j,z_1,\ldots,z_n) \end{pmatrix},$$

so $\widetilde{h}(y_j,z_1,\ldots,z_n) \mid R_x(\widetilde{g},\widetilde{h})$ for $1 \le j \le n$, which implies that

$$\prod_{i=1}^{n}(y_j - z_i) \mid R_x(\widetilde{g},\widetilde{h}) \text{ for } 1 \le j \le n.$$

Since there is no common linear divisor in polynomials $\widetilde{h}(y_i,z_1,\ldots,z_n)$ and $\widetilde{h}(y_j,z_1,\ldots,z_n)$ if $i \ne j$, it follows that

$$\prod_{j=1}^{n}\prod_{i=1}^{n}(y_j - z_i) \mid R_x(\widetilde{g},\widetilde{h})$$

over R.

Substitute $y_1,\ldots,y_n$ and $z_1,\ldots,z_n$ by $\xi_1,\ldots,\xi_n$, then

$$\prod_{j=1}^{n}\prod_{i=1}^{n}(y_j - z_i) = \prod_{j=1}^{n} f(\xi_j)$$

and $R_x(\widetilde{g},\widetilde{h}) = R(g,h)$. Because $f(\xi_j) = 0$ over $\mathbb{Z}/(p^e)$ for $1 \le j \le n$, $R(g,h)$ has at least n p^es as factors when $R(g,h) \ne 0$ over $\mathbb{Z}$. When $R(g,h) = 0$ over $\mathbb{Z}$, the conclusion is obvious, which completes the proof.

□

Appendix B

Proof of Theorem 3 Let $m = p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s}$ be the canonical factorization of m. By Definition 3, for every $i \in \{1, 2, \ldots, s\}$, $f(x) \bmod p_i^{e_i}$ is a primitive polynomial of degree n over $\mathbb{Z}/(p_i^{e_i})$. Let $R = \mathbb{Z}/(m)[x]/(f(x))$. For every $i \in \{1, 2, \ldots, s\}$, let $R_i = \mathbb{Z}/(p_i^{e_i})[x]/(f(x) \bmod p_i^{e_i})$, a Galois ring with characteristic $p_i^{e_i}$ and cardinality $p_i^{e_i n}$.

According to the Chinese Remainder Theorem (CRT), the ring homomorphism

$$\phi : \begin{cases} R \longrightarrow R_1 \times \cdots \times R_s \\ \alpha \longmapsto (\alpha \bmod p_1^{e_1}, \ldots, \alpha \bmod p_s^{e_s}) \end{cases}$$

is an isomorphism. Since by Lemma 2, for every $i \in \{1, 2, \ldots, s\}$, $f(x) \bmod p_i^{e_i}$ splits in the Galois ring R_i, by CRT there exist $\xi_1, \ldots, \xi_n \in R$ such that $f(x) = \prod_{i=1}^{n}(x - \xi_i)$. Then $g(x)$ and $h(x)$ can be represented as follows

$$g(x) = g_1(x) \cdot \prod_{i=1}^{n}(x - \xi_i),$$

$$h(x) = h_1(x) \cdot \prod_{i=1}^{n}(x - \xi_i),$$

where $\xi_1, \ldots, \xi_n \in R$.

Similar with the proof of Theorem 2, it can be proved that $R(g, h)$ has at least n ms as factors when $R(g, h) \neq 0$ over $\mathbb{Z}$. When $R(g, h) = 0$ over $\mathbb{Z}$, the conclusion is obvious, which completes the proof. □

References

[1] Sun, H.Y., Zhu, X.Y., Zheng, Q.X.: Predicting truncated multiple recursive generators with unknown parameters. Designs, Codes and Cryptography **88**, 1083–1102 (2020)

[2] Lehmer, D.H.: Mathematical methods in large-scale computing units. The Annals of the Computation Laboratory of Harvard University **26**, 141–146 (1951)

[3] Knuth, D.E.: Seminumerical algorithms. In: The Art of Computer Programming, vol. 2. Addison-Wesley, Canada (1981)

[4] Coveyou, R.R., MacPherson, R.D.: Fourier analysis of uniform random number generators. Journal of the ACM **14**, 100–119 (1967)

[5] L'Ecuyer, P., Touzin, R.: Fast combined multiple recursive generators with multipliers of the form $a = \pm 2^q \pm 2^r$. In: Proceedings of the 32nd Conference on Winter Simulation, pp. 683–689. Society for Computer Simulation International, San Diego, CA, USA (2000)

[6] Deng, L., Xu, H.Q.: A system of high-dimensional, efficient, long-cycle and portable uniform random number generators. ACM Transactions on Modeling and Computer Simulation **13**(4), 299–309 (2003)

[7] Deng, L.: Efficient and portable multiple recursive generators of large order. ACM Transactions on Modeling and Computer Simulation **15**(1), 1–13 (2005)

[8] Deng, L., Shiau, J.H., Lu, H.H., Bowman, D.: Secure and fast encryption (safe) with classical random number generators. ACM Transactions on Mathematical Software **44**(4), 1–17 (2018)

[9] Zierler, N.: Linear recurring sequences. Journal of the Society for Industrial and Applied Mathematics **7**(1), 31–48 (1959)

[10] L'Ecuyer, P.: Good parameters and implementations for combined multiple recursive random number generators. Operations Research **47**(1), 159–164 (1999)

[11] Chen, H.J., Qi, W.F.: On the distinctness of maximal length sequences over $\mathbb{Z}/(pq)$ modulo 2. Finite Fields and Their Applications **15**(2), 23–39 (2009)

[12] ETSI/SAGE: Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 2: ZUC Specification (2011)

[13] Plumstead, J.B.: Inferring a sequence generated by a linear congruence. In: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982), pp. 153–159. IEEE, Chicago, IL, USA (1982)

[14] Knuth, D.E.: Deciphering a linear congruential encryption. IEEE Transactions on Information Theory **31**(1), 49–52 (1985)

[15] Stern, J.: Secret linear congruential generators are not cryptographically secure. In: Proceedings of the 28th Annual Symposium on Foundations of Computer Science (sfcs 1987), pp. 421–426. IEEE, Los Angeles (1987)

[16] Frieze, A.M., Hastad, J., Kannan, R., Lagarias, J.C., Shamir, A.: Reconstructing truncated integer variables satisfying linear congruences. SIAM Journal on Computing **17**(2), 262–280 (1988)

[17] Joux, A., Stern, J.: Lattice reduction: a toolbox for the cryptanalyst. Journal of Cryptology **11**(3), 161–185 (1998)

[18] Boyar, J.: Inferring sequences produced by a linear congruential generator missing low-order bits. Journal of Cryptology **1**(3), 177–184 (1989)

[19] Contini, S., Shparlinski, I.E.: On stern's attack against secret truncated linear congruential generators. In: Information Security and Privacy, pp.

52–60. Springer, Berlin, Heidelberg (2005)

[20] Massey, J.L.: Shift register synthesis and bch decoding. IEEE Transactions on Information Theory **15**(1), 122–127 (1969)

[21] Berlekamp, E.R.: Algebraic Coding Theory. McGraw-Hill, New York (1968)

[22] Sugiyama, Y., Kasahara, M., Hirasawa, S., Namekawa, T.: A method for solving key equation for decoding goppa codes. Information and Control **27**(1), 87–99 (1975)

[23] Mills, W.H.: Continued fractions and linear recurrences. Mathematics of computation **29**(129), 173–180 (1975)

[24] Reeds, A., Sloane, N.J.A.: Shift-register synthesis (mod m). SIAM Journal of Computing **14**, 505–513 (1985)

[25] Boyar, J.: Inferring sequences produced by pseudo-random number generators. Journal of the ACM **36**(1), 129–141 (1989)

[26] Huang, M.Q.: Analysis and Cryptologic Evaluation of Primitive Sequences over an Integer Residue Ring. Doctoral Dissertation of Graduate School of USTC, Academia Sinica (1988)

[27] Kuzmin, A.S., Nechaev, A.A.: Linear recurring sequences over galois ring. Russian Mathmatical Surveys **48**, 171–172 (1993)

[28] Zhu, X.Y.: Some Results on Injective Mappings of Primitive Sequences Modulo Prime Powers. PLA Information Engineering University, Zhengzhou (2004)

[29] Kuzmin, A.S., Marchalko, G.B., Nechaev, A.A.: Reconstruction of a linear recurrence over a primary residue ring. Memoires in Discrete Mathematics **12**, 155–194 (2009)

[30] Kuzmin, A.S., Nechaev, A.A.: Reconstruction of a linear recurrence of maximal period over a galois ring from its highest coordinate sequence. Discrete Mathematics and Applications **21**(2), 145–178 (2011)

[31] Yang, J.B.: Reconstructing Truncated Sequences Derived from Primitive Sequences over Inter Residue Rings. PLA Information Engineering University, Zhengzhou (2017)

[32] Coppersmith, D.: Finding a small root of a univariate modular equation. In: Advances in Cryptology – EUROCRYPT'96, pp. 155–165. Springer, Berlin, Heidelberg (1996)

[33] Ward, M.: The arithmetical theory of linear recurring series. Transactions of the American Mathematical Society **35**, 600–628 (1933)

[34] Nguyen, P.Q.: Hermite's constant and lattice algorithms. In: Nguyen, P.Q., Vallée, B. (eds.) The LLL Algorithm, pp. 19–69. Springer, Berlin, Heidelberg (2010)

[35] Ajtai, M.: Generating random lattices according to the invariant distribution (2006). Draft of March

[36] Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N. (ed.) Advances in Cryptology – EUROCRYPT 2008, pp. 31–51. Springer, Berlin, Heidelberg (2008)

[37] Nguyen, P.Q., Stehlé, D.: LLL on the average. In: Hess, F., Pauli, S., Pohst, M. (eds.) Algorithmic Number Theory, pp. 238–256. Springer, Berlin, Heidelberg (2006)

[38] Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**(4), 515–534 (1982)

[39] Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical Programming **66**(2), 181–199 (1994)

[40] Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing, pp. 193–206. Association for Computing Machinery, New York, NY, USA (1983)

[41] Kannan, R.: Minkowski's convex body theorem and integer programming. Mathematics of Operations Research **12**(3), 415–440 (1987)

[42] Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing, pp. 601–610. Association for Computing Machinery, New York, NY, USA (2001)

[43] Wan, Z.X.: Lectures on Finite Fields and Galois Rings. World Scientific, Singapore (2003)

[44] William, C.B.: Matrices over Commutative Rings. Marcel Dekker, New York (1993)

[45] Zhou, J.J., Qi, W.F.: On some properties of linear recurring sequences over $\mathbb{Z}/(m)$. Chinese Quarterly Journal of Mathmatics **5**(1–2), 166–171 (1990)

[46] Josh, A., Virginia, V.W.: A refined laser method and faster matrix multiplication. In: Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 522–539. Society for Industrial and Applied Mathematics, USA (2021)

[47] Von zur Gathen, J., Gerhard, J.: Modern Computer Algebra. Cambridge University Press, Cambridge (2013)

[48] David, G.C., Erich, K.: On fast multiplication of polynomials over arbitrary algebras. Acta Informatica **28**, 693–701 (1991)

[49] Shoup, V.: Number Theory C++ Library (NTL), version 11.4.3. http://www.shoup.net/ntl/