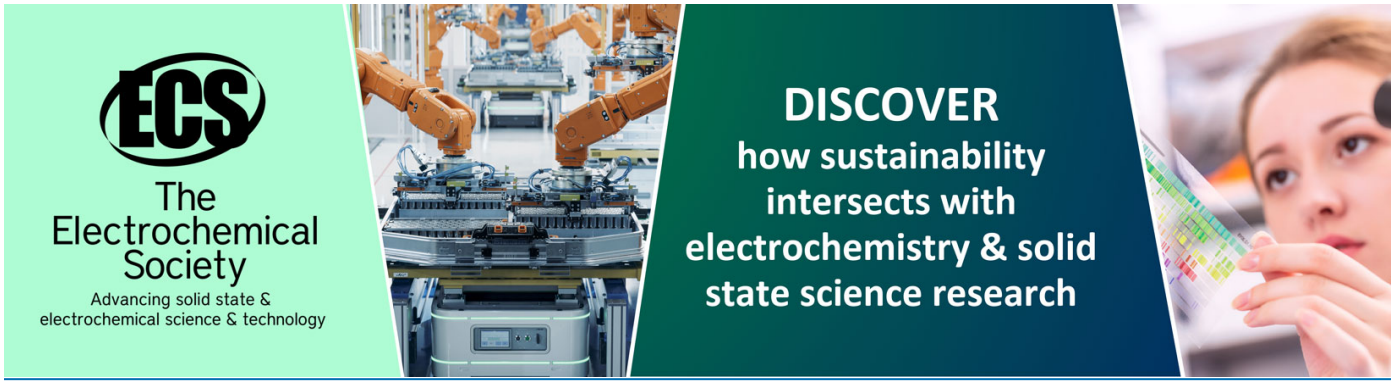
Inversion of two new circulant matrices over Z_m



Inversion of two new circulant matrices over Z_m

Yanpeng Zheng, Sugoog Shon, * Zunwei Fu

The University of Suwon, Bongdameup, Hwaseong-si, Gyeonggi-do, 445-743, Korea.

E-mail: zhengyanpeng0702@sina.com; sshon@suwon.ac.kr; fuzunwei@lyu.edu.cn

Abstract. In this paper we consider the problem of inverting an $n \times n$ RSFMLR circulant matrix with entries over Z_m. We present two different algorithms. Our algorithms require different degrees of knowledge of m and n, and their costs range, from $n \log n \log\log n$ to $n \log^2 n \log\log n \log m$ operations over Z_m. Moreover, for each algorithm we give the cost in terms of bit operations. Finally, the extended algorithms is used to solve the problem of inverting RSFMLR circulant matrices over Z_m.

1 Introduction

Circulant matrices have important applications in various disciplines including, image processing, communications, signal processing, encoding, computer vision and they have been put on firm basis with the work of P. Davis [1] and Z. L. Jiang [2].

The circulant matrices, long a fruitful subject of research [1,2], have in recent years been extended in many directions [3,4, 6–8,11]. The $f(x)$-circulant matrices are another natural extension of this well-studied class, and can be found in [9–14]. The $f(x)$-circulant matrix has a wide application, especially on the generalized cyclic codes [9], where is a monic polynomial with no repeated roots in its splitting field over a field. The properties and structures of the $x^n + x + 1$-circulant matrices, which are called RSFMLR circulant matrices, are better than those of the general $f(x)$-circulant matrices, so there are good algorithms for finding the inverse of the RSFMLR circulant matrices. In this paper we consider the problem of inverting RSFMLR circulant with entries over the ring Z_m.

In this paper we describe two algorithms for inverting an $n \times n$ RSFMLR circulant matrix over Z_m which transform the original problem into an equivalent problem over the ring $Z_m[x]$. Our first algorithm assumes the factorization of m is known and requires $n \log^2 n + n \log m$ multiplications and $n \log^2 n \log m$ additions over Z_m. This corresponds to the bit complexity bound $O(n \log^2 n + n \log m)\mu(\log m) + n \log^2 n \log\log n \log m$, where $\mu(d)$) denotes the bit complexity of multiplying d-bit integers. Our second algorithm does not require the factorization of *m* and its cost is greater, by a factor $\log m$; than in the previous case.

Definition 1 *A row skew first-minus-last right(RSFMLR) circulant matrix* with the first row $(a_0, a_1, \cdots, a_{n-1})$ over Z_m, denoted by RSFMLRcircfr$(a_0, a_1, \cdots, a_{n-1})$, is meant a square matrix of the form:

$$\begin{pmatrix} a_0 & a_1 & \cdots & a_{n-2} & a_{n-1} \\ -a_{n-1} & a_0 - a_{n-1} & a_1 & \cdots & a_{n-2} \\ \vdots & -a_{n-1} - a_{n-2} & \ddots & \ddots & \vdots \\ -a_2 & \vdots & \ddots & a_0 - a_{n-1} & a_1 \\ -a_1 & -a_2 - a_1 & \cdots & -a_{n-1} - a_{n-2} & a_0 - a_{n-1} \end{pmatrix}_{n\times n} \quad (1)$$

It can be seen that the matrix over Z_m with an arbitrary first row and the following rule for obtaining any other row from the previous one: Get the $i+1$st row by minus the last element of the ith row to the first element of the ith row,and -1 times the last element of the ith row, and then shifting the elements of the ith row (cyclically) one position to the right.

Obviously, the RSFMLR circulant matrix over Z_m is a $x^n + x + 1$-circulant matrix [9], and that is neither the extension of circulant matrix over Z_m [3] nor its special case and they are two different families of patterned matrices.

We define $\Theta_{(-1,-1)}$ as the basic RSFMLR circulant matrix over Z_m, that is

$$\Theta_{(-1,-1)} = \text{RSFMLRcircfr}(0,1,0,\cdots,0). \quad (2)$$

It is easily verified that $g(x) = x^n + x + 1$has no repeated roots over Z_m and $g(x) = x^n + x + 1$ is both the minimal polynomial and the characteristic polynomial of the matrix $\Theta_{(-1,-1)}$. In addition, $\Theta_{(-1,-1)}$ is nonderogatory and satisfies $\Theta^{j}_{(-1,-1)} =$ RSFMLRcircfr $(\underbrace{0,\cdots,0}_{j},1,\underbrace{0,\cdots,0}_{n-j-1})$ and $\Theta^{n}_{(-1,-1)} = -I_n - \Theta_{(-1,-1)}$. In view of the structure of the powers of the basic RSFMLR circulant matrix $\Theta_{(-1,-1)}$ over Z_m, it is clear that

$$A = RSFMLRcircfr(a_0, a_1, \cdots, a_{n-1}) = \sum_{i=0}^{n-1} a_i \Theta^i_{(-1,-1)} \quad (3)$$

Thus, A is a RSFMLR circulant matrix over Z_m if and only if $A = f(\Theta_{(-1,-1)})$ for some polynomial $f(x)$ over Z_m. The polynomial $f(x) = \sum_{i=0}^{n-1} a_i x^i$ will be called the representer of the RSFMLR circulant matrix A over Z_m. By Definition 1 and Equation (3), it is clear that A is a RSFMLR circulant matrix over Z_m if and only if A commutes with $\Theta_{(-1,-1)}$, that is, $A\Theta_{(-1,-1)} = \Theta_{(-1,-1)}A$. In addition to the algebraic properties that can be easily derived from the representation (3), we mention that RSFMLR circulant matrices have very nice structure. The product of two RSFMLR circulant matrices is a RSFMLR circulant matrix and A^{-1} is a RSFMLR circulant matrix, too. Further more, let $Z_m[\Theta_{(-1,-1)}] = \{A \mid A = f(\Theta_{(-1,-1)}), f(x) \in Z_m[x]\}$

It is a routine to prove that $Z_m[\Theta_{(-1,-1)}]$is a commutative ring with the matrix addition and multiplication.

Definition 2 A *row skew last-minus-first left* (RSLMFL) circulant matrix with the first row $(a_0, a_1, \cdots, a_{n-1})$ over Z_m, denoted by RSLMFLcircfr$(a_0, a_1, \cdots, a_{n-1})$, is meant a square matrix of the form:

$$\begin{pmatrix} a_0 & a_1 & \cdots & a_{n-2} & a_{n-1} \\ a_1 & a_2 & \cdots & a_{n-1}-a_0 & -a_0 \\ \vdots & \vdots & \vdots & -a_0-a_1 & \vdots \\ a_{n-2} & a_{n-1}-a_0 & \cdots & \vdots & -a_{n-3} \\ a_{n-1}-a_0 & -a_0-a_1 & \cdots & -a_{n-3}-a_{n-2} & -a_{n-2} \end{pmatrix}_{n\times n} \tag{4}$$

Lemma 1 Let $\hat{I} = \begin{pmatrix} 0 & \cdots & 0 & 1 \\ \vdots & & \iddots & 0 \\ 0 & \iddots & & \vdots \\ 1 & 0 & \cdots & 0 \end{pmatrix}$ be the $n \times n$ matrix of the counter identity.

Then

(i) RSLMFLcircfr$(a_0, a_1, \cdots, a_{n-1})$
= RSFMLRcircfr$(a_{n-1}, \cdots, a_1, a_0)\hat{I}_n$;

(ii) RSLMFLcircfr$(a_0, a_1, \cdots, a_{n-1})\hat{I}_n$
=RSFMLRcircfr$(a_{n-1}, \cdots, a_1, a_0)$.

Assuming A is inveible over Z_m, we consider the problem of computing a RSFMLR circulant matrix $B = \sum_{i=0}^{n-1} b_i \Theta^i{}_{(-1,-1)}$, such that $AB = I$.

It is natural to representer with a RSFMLR circulant matrix $A = \sum_{i=0}^{n-1} a_i \Theta^i_{(-1,-1)}$ the polynomial (over the ring $Z_m[x]$), $f(x) = \sum_{i=0}^{n-1} a_i x^i$. Computing the inverse of A is clearly equivalent to finding a polynomial $g(x) = \sum_{i=0}^{n-1} b_i x^i$ in $Z_m[x]$ such that

$$f(x)g(x) \equiv 1 (\text{mod } x^n + x + 1). \tag{5}$$

The congruence modulo $x^n + x + 1$ follows from the equality $\Theta^n_{(-1,-1)} = -\Theta_{(-1,-1)} - I_n$. Hence, the problem of inverting a RSFMLR circulant matrix is equivalent to inversion in the ring $Z_m[x]/\langle x^n + x + 1\rangle$.

The following theorem states a necessary and sufficient condition for the invertibility of a RSFMLR circulant matrix over Z_m.

Theorem 1 Let $m = p_1^{k_1} p_2^{k_2} \cdots p_h^{k_h}$ denote the prime powers factorization of m and let $f(x)$ denote the polynomial over Z_m representer to a RSFMLR circulant matrix A. The matrix A is invertible if and only if, for $i = 1, \cdots, h$, we have $\gcd(f(x), x^n + x + 1) = 1$ in $Z_{p_i}[x]$.

Proof If A is invertible, by (5) we have that there exists $t(x)$ such that for $i = 1, \cdots, h$ $f(x)g(x) + t(x)(x^n + x + 1) = 1$ in $Z_{p_i}[x]$.

Hence, $\gcd(f(x), x^n + x + 1) = 1$ in $Z_{p_i}[x]$ as claimed. The proof that the above condition is sufficient for invertibility is constructive and will be given in Section 2 (Lemmas 2 and 3).

Review of bit complexity results [3]. In the following we will give the cost of each algorithm in terms of number of bit operations. In our analysis we use the following well-known results (see for example [15] or [16]). Additions and subtractions in Z_m take $O\log m$ bit operations. We denote by $\mu(d = O(d\log d\log\log d)$ the number of bit operations required by the Schönhage-Strassen algorithm [18] for multiplication of integers modulo 2^d+1. Hence, multiplication between elements of Z_m takes $\mu\log m = O(\log m\log\log m\log\log m)$ bit operations. Computing the inverse of an element $x \in Z_m$ takes $\mu(\log m)\log\log m$ bit operations using a modified extended Euclidean algorithm (see [15], Theorem 8.20). The same algorithm returns $\gcd(x,m)$ when x is not invertible.

The sum of two polynomials $Z_m[x]$ in $f(x)g(x) \equiv 1(\bmod x^n + x + 1)$. of degree at most n can be trivially computed in $O(n\log m)$ bit operations. The product of two such polynomials can be computed in $O(n\log n)$ multiplications and $O(n\log n\log\log n)$ additions/ subtractions in Z_m (see [16], Theorem 1.7.1). Therefore, the asymptotic cost of polynomial multiplication is $O(\prod(m,n))$ bit operations, where

$$\prod(m,n) = n\log n\mu(\log m) + n\log n\log\log n\log m. \tag{6}$$

Given two polynomials $a(x), b(x) \in Z_p[x]$ (p prime) of degree at most n, we can compute $d(x) = \gcd(a(x), b(x))$ in $O(\Gamma(p,n))$ bit operations, where

$$\Gamma(p,n) = \Pi(p,n)\log n + n\mu(\log p)\log\log p \tag{7}$$

The same algorithm also returns $s(x)$ and $t(x)$ such that $a(x)s(x) + b(x)t(x) = d(x)$. The bound (7) follows by a straightforward modification of the polynomial gcd algorithm described in [15] (Section 8.9: the term $n\mu(\log p)\log\log p$ comes from the fact that we must compute the inverse of $O(n)$ elements of Z_p).

2 Inversion in $Z_m[x]/\langle x^n + x + 1\rangle$ Factorization of m Known

In this section we consider the problem of computing the inverse of a RSFMLR circulant matrix over Z_m when the factorization $m = p_1^{k_1} p_2^{k_2} \cdots p_h^{k_h}$ of the modulus m is known. We consider the equivalent problem of inverting a polynomial $f(x)$ over $Z_m[x]/\langle x^n + x + 1\rangle$, and we show that we can compute the inverse by combining known techniques (Chinese remaindering, the extended Euclidean algorithm, and Newton-Hensel lifting). We start by showing that it suffices to find the inverse of $f(x)$ modulo the prime powers p_i^{ki}.

Lemma 2 Let $m = p_1^{k_1} p_2^{k_2} \cdots p_h^{k_h}$, and let $f(x)$ be a polynomial in $Z_m[x]$. Given $g_1(x), \cdots, g_h(x)$ such that $f(x)g_i(x) \equiv 1(\bmod x^n + x + 1)$ in $Z_{p_i}k_i[x]$ for $i = 1, \cdots, h$, we can find $g(x) \in Z_m[x]$ which satisfies (5) at the cost of $O(nh\mu(\log m) + \mu(\log m)\log\log m)$ bit operations.

Proof The proof is constructive. Since $f(x)g_i(x) \equiv 1(\bmod x^n + x + 1)$,in $Z_{p_i^{k_i}}[x]$,we have $f(x)g_i(x) \equiv 1 + \lambda_i(x)\langle x^n + x + 1\rangle(\bmod p_i^{k_i})$ Let $\alpha_i = m/p_i^{k_i}$. Clearly, for $j \neq i$, $\alpha_i \equiv 0(\bmod p_j^{k_j})$. Since $\gcd(\alpha_i, p_i^{k_i}) = 1$, we can find β_i such that $\alpha_i\beta_i \equiv 1(\bmod p_i^{k_i})$. Let $g(x) = \sum_{i-1}^{h} \alpha_i\beta_i g_i(x)$,

$\lambda(x) = \sum_{i-1}^{h} \alpha_i \beta_i \lambda_i(x)$.

By construction, for $i = 1,2\cdots,h$, we have $g(x) \equiv g_i(x) (\mathrm{mod}\ p_i^{k_i})$ and $\lambda(x) \equiv \lambda_i(x) (\mathrm{mod}\ p_i^{k_i})$. Hence, for $i = 1,2\cdots,h$, we have

$$f(x)g(x) \equiv \sum_{\mathrm{j\text{-}1}}^{\mathrm{h}} \alpha_j \beta_j f(x) g_j(x) \equiv f(x) g_i(x) (\mathrm{mod}\ p_i^{k_i})$$
$$\equiv 1 + \lambda_i(x)\langle x^n + x + 1\rangle (\mathrm{mod}\ p_i^{k_i}) \equiv 1 + \lambda(x)\langle x^n + x + 1\rangle$$
$$(\mathrm{mod}\ p_i^{k_i})$$

We conclude that $f(x)g(x) \equiv 1 + \lambda(x)\langle x^n + x + 1\rangle (\mathrm{mod}\, m)$, or, equivalently, $f(x)g(x) \equiv 1 (\mathrm{mod}\ x^n + x + 1)$ in $Z_m[x]$.

The computation of $g(x)$ consists in n (one for each coefficient) applications of Chinese remaindering. Obviously, the computation of α_i, β_i, $i = 1,\cdots,h$, should be done only once. Since integer division has the same asymptotic cost as multiplication, we can compute $\alpha_1,\cdots,\alpha_h$ in $O(h\mu(\log m))$ bit operations. Since each β_i is obtained through an inversion in $Z_{p_i^{k_j}}$, computing the $\beta_1,\cdots\beta_h$ takes $O(\sum_{j=1}^{h} \mu(\log p_j^{k_j}) \log\log p_j^{k_j})$ bit operations. Finally, given $\alpha_1,\cdots,\alpha_h$, $\beta_1,\cdots\beta_h$, $g(x)_1,\cdots,g_h(x)$ we can compute $g(x)$ in $O(nh\mu(\log m))$ bit operations. The thesis follows using the inequality

$$\mu(\log a)\log\log a + \mu(\log b)\log\log b$$
$$\leq \mu(\log(ab))\log\log(ab)$$

In view of Lemma 2, we can restrict ourselves to the problem of inverting a polynomial over $\mathrm{Z}_m[x]/\langle x^n + x + 1\rangle$ when $m = p^k$ is a prime power. Next lemma shows how to solve this particular problem.

Lemma 3 Let $f(x)$ be a polynomial in $Z_{p^k}[x]$. If $\gcd(f(x), x^n + x + 1) = 1$ in $Z_p[x]$, then $f(x)$ is invertible in $\mathrm{Z}_{p^k}[x]/\langle x^n + x + 1\rangle$. In this case, the inverse of $f(x)$ can be computed in $O(\Gamma(p,n) + \Pi(p^k,n))$ bit operations, where $\Gamma(p,n)$ and $\Pi(p^k,n)$ are defined by (7) and (6) respectively.

Proof If $\gcd(f(x), x^n + x + 1) = 1$ in $Z_p[x]$, by Bezout's lemma there exist $s(x)$, $t(x)$ such that $f(x)s(x) + \langle x^n + x + 1\rangle t(x) \equiv 1 (\mathrm{mod}\ p)$

Next we consider the sequence

$$g_0(x) = s(x),$$
$$g_i(x) = 2g_{i-1}(x) - [g_{i-1}(x)]^2 f(x) \bmod x^n + x + 1,$$

known as Newton-Hensel lifting. It is straightforward to verify by induction that $g_i(x) f(x) \equiv 1 + p^{2^i} \lambda_i(x) (\mathrm{mod}\ x^n + x + 1)$. Hence, the inverse of $f(x)$ in $\mathrm{Z}_{p^k}[x]/\langle x^n + x + 1\rangle$ is $g\lceil \log k \rceil(x)$.

The computation of $s(x)$ takes $O(\Gamma(p,n))$ bit operations. For computing the sequence $g_1(x), g_2(x), \cdots, g_{\lceil \log k \rceil}(x)$ we observe that it suffices to compute each g_i modulo p^{2^i}. Hence, the cost of obtaining the whole sequence is

$$O(\Pi(p^2,n))+\Pi(p^4,n)+\cdots+\Pi(p^{2^{\lceil \log k\rceil}},n))$$
$$=O(\Pi(p^k,n))$$

bit operations.

Note that from Lemmas 2 and 3, we find that the condition given in Theorem 1 is indeed a sufficient condition for invertibility of a RSFMLR circulant matrix. Combining the above lemmas we obtain Algorithm 1 for the inversion of a polynomial $f(x)$ over $\mathrm{Z}_m[x]/\langle x^n+x+1\rangle$. The cost of the algorithm is

$$T(m,n)=O(nh\mu(\log m)+\mu(\log m)\log\log m$$
$$+\sum_{j=1}^{h}\Gamma(p_j,n)+\Pi(p_j^{k_j},n))$$

bit operations. In order to get a more manageable expression, we bound h with $\log m$ and p_j with $p_j^{k_j}$. In addition, we use the inequalities $\Pi(a,n)+\Pi(b,n)\le\Pi(ab,n)$ and $\Gamma(a,n)+\Gamma(b,n)\le\Gamma(ab,n)$. We get

$$T(m,n)=O(n\log m\mu(\log m)+\mu(\log m)\log\log m$$
$$+\Gamma(m,n)+\Pi(m,n))$$
$$=O(n\log m\mu(\log m))+\Pi(m,n)\log n).$$

Note that if $m=O(n)$ the dominant term is $\Pi(m,n)\log n$. That is, the cost of inverting $f(x)$ is asymptotically bounded by the cost of executing $\log n$ multiplications in $\mathrm{Z}_m[x]$.

Inverse 1 $(f(x),m,n)\to g(x)$

{Computes the inverse $g(x)$ of the polynomial $f(x)$ in $\mathrm{Z}_m[x]/\langle x^n+x+1\rangle$ }

1. **let** $m=p_1^{k_1}p_2^{k_2}\cdots p_h^{k_h}$;
2. **for** $j=1,2,\cdots,h$ do
3. **if** $\gcd(f(x),x^n+x+1)=1$ **in** $\mathrm{Z}_{\mathrm{p_j}}[x]$ then
4. compute $g_j(x)$ such that

   $f(x)g_j(x)\equiv 1(\mathrm{mod}\ x^n+x+1)$ in $Z_{p_j^{k_j}}[x]$
5. using Newton-Hensel lifting (Lemma 3);
6. **else**
7. **return** " $f(x)$ is not invertible";
8. **endif**
9. **endfor**
10. compute $g(x)$ using Chinese remaindering (Lemma 2).

Algorithm 1 *Inversion in* $\mathrm{Z}_m[x]/\langle x^n+x+1\rangle$. *Factorization of* m *known.*

3 A General Inversion Algorithm in $\mathrm{Z}_m[x]/\langle x^n+x+1\rangle$

The algorithm described in Section 2 relies on the fact that the factorization of the modulus m is known. If this is not the case and the factorization must be computed beforehand, the increase in the running time may be significant since the fastest known factorization algorithms require time

exponential in $\log m$ (see for example [17]). In this section we show how to compute the inverse of $f(x)$ without knowing the factorization of the modulus. The number of bit operations of the new algorithm is only a factor $O(\log m)$ greater than in the previous case.

Our idea consists in trying to compute $\gcd(f(x), x^n + x + 1)$ in $Z_m[x]$ using the gcd algorithm for $Z_p[x]$. Such algorithm requires the inversion of some scalars, which is not a problem in $Z_p[x]$, but it is not always possible if m is not prime. Therefore, the computation of $\gcd(f(x), x^n + x + 1)$ may fail. However, if the gcd algorithm terminates we have solved the problem. In fact, together with the alleged gcd $a(x)$ the algorithm also returns $s(x)$, $t(x)$ such that $f(x)s(x) + (x^n + x + 1)t(x) = a(x)$ in $Z_m[x]$. If $a(x) = 1$, then $s(x)$ is the inverse of $f(x)$. If $\deg(a(x)) \neq 0$, one can easily prove that $f(x)$ is not invertible in $\mathrm{Z}_m[x]/\langle x^n + x + 1\rangle$. Note that we must force the gcd algorithm to return a monic polynomial.

If the computation of $\gcd(f(x), x^n + x + 1)$ fails, we use recursion. In fact, the gcd algorithm fails if it cannot invert an element $y \in Z_m$. Inversion is done by using the integer gcd algorithm. If y is not invertible, the integer gcd algorithm returns $d = \gcd(m, y)$, with $d > 1$. Hence, d is a nontrivial factor of m. We use d to compute either a pair m_1, m_2 such that $\gcd(m_1, m_2) = 1$ and $m_1 m_2 = m$, or a single factor m_1 such that $m_1 | m$ and $m | (m_1)^2$. In the first case we invert $f(x)$ in $\mathrm{Z}_{m_1}[x]/\langle x^n + x + 1\rangle$ and $\mathrm{Z}_{m_2}[x]/\langle x^n + x + 1\rangle$, and we use Chinese remaindering to get the desired result. In the second case, we invert $f(x)$ in $\mathrm{Z}_{m_1}[x]/\langle x^n + x + 1\rangle$ and we use one step of Newton-Hensel lifting to get the inverse in $\mathrm{Z}_m[x]/\langle x^n + x + 1\rangle$.

The computation of the factors m_1, m_2 is done by procedure GetFactors whose correctness is proven by Lemmas 4 and 5. Combining these procedures together we get Algorithm 2.

Inverse 2 $(f(x), m) \rightarrow g(x)$

{Computes the inverse $g(x)$ of the polynomial $f(x)$ in $\mathrm{Z}_m[x]/\langle x^n + x + 1\rangle$ }

1. **if** $\gcd(f(x), x^n + x + 1) = 1$ **then**
2. **let** $s(x)$, $t(x)$ such that $f(x)s(x) + (x^n + x + 1)t(x) = 1$ in $Z_m[x]$;
3. **return** $s(x)$;
4. **else if** $\gcd(f(x), x^n + x + 1)t(x) = a(x)$, $\deg(a(x)) > 0$ **then**
5. **return** " $f(x)$ is not invertible";
6. **else if** $\gcd(f(x), x^n + x + 1)$ fails let d be such that $d | m$;
7. **let** $(m_1, m_2) \leftarrow$ **GetFactors** (m, d);
8. **if** $m_2 \neq 1$, then
9. $g_1(x) \leftarrow$ **Inverse 2** $(f(x), m_1)$;
10. $g_2(x) \leftarrow$ **Inverse 2** $(f(x), m_2)$;
11. compute $g(x)$ using Chinese remaindering (Lemma 2);

12. **else**
13. $g_1(x) \leftarrow$ **Inverse 2** $(f(x), m_1)$;
14. compute $g(x)$ using Newton-Hensel lifting (Lemma 3);
15. **endif**
16. **return** $g(x)$;
17. **endif**

GetFactors (m;d)!(m1;m2)

18. **let** $m_1 \leftarrow \gcd(m, d^{\lfloor \log m \rfloor})$;
19. **if** $(m/m_1) \neq 1$ **then**
20. **return** $(m_1, m/m_1)$;
21. **endif**
22. **let** $e \leftarrow m|d$;
23. **let** $m_1 \leftarrow \gcd(m, e^{\lfloor \log m \rfloor})$;
24. **if** $(m/m_1) \neq 1$ **then**
25. **return** $(m_1, m/m_1)$;
26. **endif**
27. **let** $m_1 \leftarrow \mathrm{lcm}(d, e)$;
28. **return** $(m_1, 1)$;

Algorithm 2 *Inversion in* $\mathrm{Z}_m[x]/\langle x^n + x + 1\rangle$.

Factorization of m unknown.

The following Lemma 4 and Lemma 5 proved in [3].

Lemma 4 Let α, $\alpha > 1$, be a divisor of m and let $\alpha' = \gcd(m, \alpha^{\lfloor \log m \rfloor})$. Then, α' is a divisor of m and $\gcd(\alpha', m/\alpha') = 1$.

Lemma 5 Let α, β be such that $\alpha\beta = m$ and $\gcd(m, \alpha^{\lfloor \log m \rfloor}) = \gcd(m, \beta^{\lfloor \log m \rfloor}) = m$ .Then $\gamma = \mathrm{lcm}(\alpha, \beta) = m/\gcd(\alpha, \beta)$ is such that γ / m and γ / m^2.

Theorem 2 If $f(x)$ is invertible in $\mathrm{Z}_m[x]/\langle x^n + x + 1\rangle$, Algorithm 2 returns the inverse $g(x)$ in $O(\Gamma(m,n)\log m)$ bit operations.

Proof One can easily prove the correctness of the algorithm by induction on m, the base on the induction being the case in which m is prime where the inverse is computed by the gcd algorithm.

To prove the bound on the number of bit operations we first consider the cost of the single steps. By (7) we know that computing $\gcd(f(x), x^n + x + 1)$ takes $O(\Gamma(m,n)) = \mathrm{O}(\Pi(m,n)\log n + n\mu(\log m)\log\log m)$ bit operations. By Lemma 2, we know that Chinese remaindering at Step 11 takes $O(n\mu(\log m)) + \mu(\log m)\log\log m)$ bit operations. By Lemma 3 we know that Newton-Hensel lifting at Step 14 takes $O(\Pi(m,n))$ bit operations. Finally, it is straightforward to verify that GetFactors computes (m_1, m_2) in $O(\mu(\log m)\log\log m)$ bit operations. We conclude that, apart from the recursive calls, the cost of the algorithm is dominated by the cost of the gcd computation no matter which is the output of the gcd algorithm. Hence, there exists a constant c such that the total number of bit operations satisfies the recurrence $T(m,n) \le c\Gamma(m,n) + T(m_1,n) + T(m_2,n)$, where we assume $T(m_2,n) = 0$ if $m_2 = 1$. Let

$m = p_1^{k_1} p_2^{k_2} \cdots p_h^{k_h}$ denote the prime factorization of m. Define $l(m) = k_1 + k_2 + \cdots + k_h$. We now show that $T(m,n) \le cl(m)\Gamma(m,n)$. Since $l(m) \le \log(m)$ this will prove the theorem. We prove the result by induction on $l(m)$. If $l(m) = 1$, then m is prime and the inequality holds since the computation is done without any recursive call. Let $l(m) > 1$. By induction we have $T(m_1,n) \le cl(m_1)\Gamma(m_1,n), T(m_2,n) \le cl(m_2)\Gamma(m_2,n).$ We have $T(m,n) \le c\Gamma(m,n) + c[l(m)-1][\Gamma(m_1,n) + \Gamma(m_2,n)]$, which implies the thesis since $\Gamma(m_1,n) + \Gamma(m_2,n) \le \Gamma(m,n)$.

In addition, by Lemma 1 (i) (ii), Algorithm 1 and Algorithm 2, it is easily to get two algorithms for inverting RSLMFL circulant matrices over Z_m, respectively.

4 Conclusion

In this paper, the problem of inverting an $n \times n$ RSFMLR circulant matrix with entries over Z_m is studied. Two different algorithms are presented. Furthermore, for each algorithm the cost in terms of bit operation are given. Finally, the extended algorithms is used to solve the problem of inverting RSFMLR circulant matrices over Z_m.

Acknowledgements

This work was supported by the GRRC program of Gyeonggi Province [(GRRC SUWON 2016-B5) and the Natural Science Foundation of China under Grant No. 11271175. Their support is gratefully acknowledged.

References

[1] P. Davis. “Circulant Matrices”, Wiley, New York, (1979).

[2] Z. L. Jiang, Z. X. Zhou. “Circulant Matrices”, Chengdu Technology University Publishing Company, Chengdu, (1999).

[3] D. Bini, Corso, G.M.D, G. Manzini, L. Margara. “Inversion of circulant matrices over Z_m”, *Math.Comp.*, **70**, pp. 1169–1182, (2000).

[4] Z. L Jiang, Z. B. Xu, S. P. Gao. “Algorithms for finding the inverses of factor block circulant matrices”, *Numer. Math*, **15(1)**, pp.1–11, (2006).

[5] Z. L. Jiang, Z. B. Xu. “A new algorithm for computing the inverse and generalized inverse of the scaled factor circulant matrix”, *J. Comput. Math.*, **26**, pp.112–122, (2008).

[6] Z. L. Jiang, S.Y. Liu. “Efficient algorithms for computing the minimal polynomial and the inverse of level- kΠ -circulant matrices”, *Bull. Korean Math. Soc*., **40(3**), pp.425–435, (2003).

[7] Z. L. Jiang, S. Y. Liu. “Level-*m* scaled circulant factor matrix over the complex number field and the quaternion division algebra”, *J. Appl. Math. Comput.*, **14(1-2)**, pp.81–96, (2004).

[8] S. G. Zhang, Z. L. Jiang, S. Y. Liu. “An application of the Gröbner basis in computation for the minimal polynomials and inverses of block circulant matrices”, *Linear Algebra Appl.*, **347**, pp. 101–114, (2002).

[9] C. David. “Regular representations of semisimple algebras, separable field extensions, group characters, generalized circulants, and generalized cyclic codes”, *Linear Algebra Appl.* **218**, pp. 147–183, (1995).

[10] Z. L. Jiang, Z. B. Xu. “Efficient algorithm for finding the inverse and group inverse of FLS *r*-circulant matrix”, *J. Appl. Math. Comput.*, **18(1-2)**, pp. 45–57, (2005).

[11] Z. L. Jiang, D. H. Sun. “Fast algorithms for solving the inverse problem of Ax = b”, *Proceedings of the Eighth International Conference on Matrix Theory and Its Applications in China*, pp. **121–124**, (2008).

[12] Z. L. Jiang. “Fast algorithms for solving FLS *r*-circulant linear systems”, *SCET 2012 (Xi’an)*, pp. **141–144**, (2012).
[13] J. Li, Z. L. Jiang, N. Shen. “Explicit determinants of the Fibonacci RFPLR circulant and Lucas RFPLL circulant matrix”, *JP J. Algebra Number Theory Appl.*, **28(2)**, pp. 167–179, (2013).
[14] Z. L. Jiang, J. Li, N. Shen. “On the explicit determinants of the RFPLR and RFPLL circulant matrices involving Pell numbers in information theory”, *ICICA, 2012, Part II, Commun. Comput. Inf. Sci.*, **308**, pp. 364–370, (2012).
[15] A. V. Aho, J. E. Hopcroft, J. D. Ullman. “The Design and Analysis of Computer Algorithms”, Addison-Wesley, Reading, Massachussets, (1974).
[16] D. Bini, V. Y. Pan. “Polynomial and Matrix Computations”, Fundam. Algorithm. 1, Birkh¨auser (1994).
[17] A. K. Lenstra, H. W. Lenstra. “Algorithms in Number Theory”, In J. van Leeuwen, editor, Handbook of Theoretical Computer Science. Algorithms and Complexity. A, The MIT Press/Elsevier, (1990).
[18] A. Schönhage, V. Strassen. “Schnelle Multiplikation grosse Zahlen”, *Comput.* **7**, pp. 281–292, (1971).