

Error Correction Decoding Algorithms of RS Codes Based on An Earlier Termination Algorithm to Find The Error Locator Polynomial

Zhengyi Jiang, Hao Shi, Zhongyi Huang, Linqi Song, Bo Bai, Gong Zhang, Hanxu Hou

Abstract—Reed-Solomon (RS) codes are widely used to correct errors in storage systems. Finding the error locator polynomial is one of the key steps in the error correction procedure of RS codes. Modular Approach (MA) is an effective algorithm for solving the Welch-Berlekamp (WB) key-equation problem to find the error locator polynomial that needs $2t$ steps, where t is the error correction capability. In this paper, we first present a new MA algorithm that only requires $2e$ steps and then propose two fast decoding algorithms for RS codes based on our MA algorithm, where e is the number of errors and $e \leq t$. We propose Improved-Frequency Domain Modular Approach (I-FDMA) algorithm that needs $2e$ steps to solve the error locator polynomial and present our first decoding algorithm based on the I-FDMA algorithm. We show that, compared with the existing methods based on MA algorithms, our I-FDMA algorithm can effectively reduce the decoding complexity of RS codes when $e < t$. Furthermore, we propose the t_0 -Shortened I-FDMA (t_0 -SI-FDMA) algorithm (t_0 is a predetermined even number less than $2t-1$) based on the new termination mechanism to solve the error number e quickly. We propose our second decoding algorithm based on the SI-FDMA algorithm for RS codes and show that the multiplication complexity of our second decoding algorithm is lower than our first decoding algorithm (the I-FDMA decoding algorithm) when $2e < t_0 + 1$.

I. INTRODUCTION

Reed-Solomon (RS) codes [2] are widely used in data storage systems because of their powerful error correction capability. An (n, k) RS code encodes k data symbols into n coded symbols, and can correct any e errors, where $e \leq t$ and $t = \lfloor \frac{n-k}{2} \rfloor$ is the error correction capability. Many decoding algorithms have been studied in [3]–[12] to correct the errors for RS codes.

Finding the error locator polynomial is the key part of the error correction process of RS codes. The Berlekamp–Massey algorithm [3] and the Modular Approach (MA) [4] are two typical algorithms for solving error locator polynomial when

decoding RS codes. The existing literature [7] has proved that the Berlekamp–Massey algorithm can stop the iteration and find the error locator polynomial in at least $t + e$ steps.

The key equation derived in the process of RS code error correction can be regarded as a special case of Welch-Berlekamp (WB) key equation, and the MA algorithm is effective for solving the WB key equation problem. Based on MA method and LCH-FFT fast algorithm [9], the Frequency Domain Modular Approach (FDMA) algorithm is proposed in [13] to solve the error locator polynomial, which is suitable for decoding short RS codes and hardware implementation. The FDMA algorithm [13] needs to iterate $2t$ steps to get the error locator polynomial. Recently, Chen *et al.* [12] proposed an early termination mechanism for MA algorithm which can find the error locator polynomial by iterating only $t + e$ steps. Similar to the FDMA algorithm, they [12] proposed the enhanced FDMA (eFDMA) algorithm based on their termination mechanism and LCH-FFT algorithm.

The main contributions of this paper are summarized as follows.

- We first theoretically show that the MA algorithm can find the error locator polynomial by iterating only $2e$ steps when $e \leq t$ and propose a new termination mechanism to stop the algorithm at step $2e$.
- We propose Improved-FDMA (I-FDMA) algorithm based on our new termination mechanism and present our first decoding algorithm based on I-FDMA algorithm. We show that our I-FDMA decoding algorithm has 5.64% to 41.79% multiplication reduction, compared with the existing eFDMA algorithm for evaluated parameters $(n, k) = (256, 224)$ and $e \in \{1, 2, \dots, 10\}$.
- We propose the t_0 -Shortened I-FDMA (SI-FDMA) algorithm, which can be used to quickly find the number of errors e . Based on the t_0 -SI-FDMA algorithm, we propose our second decoding algorithm, which has lower decoding complexity than our first decoding algorithm when $2e < t_0 + 1$. We show that our second decoding algorithm has 22.49% to 46.41% multiplication reduction for evaluated parameters $(n, k) = (256, 224)$, $e \in \{1, 2, \dots, 8\}$ and 26.71% to 59.22% multiplication reduction for evaluated parameters $(n, k) = (128, 96)$, $e \in \{1, 2, \dots, 8\}$, than our first decoding algorithm.

The rest of the paper is organized as follows. Section II reviews the WB key equation problem and MA algorithm. Section III derives the new termination mechanism for MA

This paper was presented in part at the IEEE International Symposium on Information Theory (ISIT), 2024 [1]. Z. Jiang, H. Shi and Z. Huang are with the Department of Mathematics Sciences, Tsinghua University, Beijing, China (E-mail: jzy21@mails.tsinghua.edu.cn, shih22@mails.tsinghua.edu.cn, zhongyih@tsinghua.edu.cn). L. Song is with the Department of Computer Science, City University of Hong Kong (E-mail: linqi.song@cityu.edu.hk). B. Bai and G. Zhang are with the Theory Lab, Central Research Institute, 2012 Labs, Huawei Tech. Co. Ltd., Hong Kong SAR (E-mail: baibo8@huawei.com, nicholas.zhang@huawei.com). H. Hou is with the School of Electrical Engineering & Intelligentization, Dongguan University of Technology (E-mail: houhanxu@163.com). (Corresponding author: Hanxu Hou.)

This work was partially supported by the National Key R&D Program of China (No. 2020YFA0712300), the National Natural Science Foundation of China (No. 62071121, 62371411, 12025104), Basic Research Enhancement Program of China under Grant 2021-JCJQ-JJ-0483.

algorithm. Section IV presents I-FDMA algorithm and our first decoding algorithm. Section V presents the t_0 -SI-FDMA algorithm and our second decoding algorithm. Section VI evaluates our two decoding algorithms and the existing decoding algorithms based on MA methods. Section VII concludes the paper.

II. PRELIMINARY

In this section, we first review the WB key equation problem [13], then we introduce the classic MA algorithm and some related results.

Consider the binary finite fields \mathbb{F}_{2^m} with 2^m elements. Let $\{v_i\}_{i=0}^{m-1}$ be the basis of \mathbb{F}_{2^m} over \mathbb{F}_2 , and $\mathbb{F}_{2^m} = \{\omega_i\}_{i=0}^{2^m-1}$, in which

$$\omega_i = i_0 \cdot v_0 + i_1 \cdot v_1 + \cdots + i_{m-1} \cdot v_{m-1}$$

for each $i = i_0 + i_1 \cdot 2 + \cdots + i_{m-1} \cdot 2^{m-1} \in \{0, 1, \dots, 2^m - 1\}$, where $(i_0, i_1, \dots, i_{m-1})$ is the binary representation of i .

A. The WB Key Equation Problem and MA Algorithm

The Welch–Berlekamp key equation problem can be described as follows. Given that positive integer $\rho \geq 1$, find a polynomial pair $(\omega(x), n(x))$ with $\omega(x), n(x) \in \mathbb{F}_{2^m}[x]$ and $\deg(n(x)) < \deg(\omega(x))$ such that the following equations hold,

$$n(x_i) = \omega(x_i)y_i, \forall i = 1, 2, \dots, \rho, \quad (1)$$

where $\{(x_i, y_i)\}_{i=1}^\rho$ are given ρ nonzero points over \mathbb{F}_{2^m} .

Consider an (n, k) RS code over the field \mathbb{F}_{2^m} , we have that the error correction capability $t = \lfloor \frac{n-k}{2} \rfloor$ and $n \leq 2^m$. In the error correction process of an (n, k) RS code, find the error locator polynomial [13] is equivalent to find a polynomial pair $(\lambda(x), z(x))$ with $\deg(z(x)) < \deg(\lambda(x))$ such that the following equation holds,

$$z(x) = s(x)\lambda(x) \mod \prod_{i=0}^{2t-1} (x - \omega_i), \quad (2)$$

where $s(x) \in \mathbb{F}_{2^m}[x]$ is the syndrome polynomial, $\lambda(x) \in \mathbb{F}_{2^m}[x]$ is the error locator polynomial and $z(x) \in \mathbb{F}_{2^m}[x]$ is the error evaluation polynomial. The equation problem in Eq. (2) is called the key equation problem. We can see that the key equation problem in Eq. (2) is a special WB key equation problem in Eq. (1) with $\rho = 2t$ and $x_i = \omega_{i-1}, y_i = s(\omega_{i-1})$ for $i = 1, 2, \dots, 2t$.

Suppose that there are e errors in an (n, k) RS code, where $e \leq t$. The element $\omega_i \in \mathbb{F}_{2^m}$ is the error position if and only if ω_i is the root of the error locator polynomial $\lambda(x)$, i.e., $\lambda(\omega_i) = 0$. Let

$$E := \{i | \lambda(\omega_i) = 0, i \in \{0, 1, \dots, n-1\}\}$$

be the index set of all e error positions. Similar to the assumption in [12], suppose that $E \subset \{i\}_{i=2t}^{n-1}$. In fact, we can always obtain an (n, k) RS code as a shortened code of $(2^m, k+2^m-n)$ RS code by deleting the first 2^m-n codeword symbols. If the number of shortened symbols 2^m-n is not less than $2t$, then the indices of both data symbols and parity

symbols of (n, k) RS code are $\geq 2t$, and we can always have $E \subset \{i\}_{i=2t}^{n-1}$.

We can employ the MA method to solve the key equation problem in Eq. (2). It inputs $2t$ values $\{s(\omega_i)\}_{i=0}^{2t-1}$ and outputs polynomial pair $(\lambda(x), z(x))$ satisfying Eq. (2). We review the MA algorithm [13] in Algorithm 1, which requires $2t$ iterations. Please refer to the literature [13] for the idea behind the MA algorithm.

Algorithm 1 The MA Algorithm [13]

Require: $\{s(\omega_i)\}_{i=0}^{2t-1}$

Ensure: $(\lambda(x), z(x))$ satisfying $z(\omega_i) = s(\omega_i)\lambda(\omega_i)$ for $i = 0, 1, \dots, 2t-1$ and $\deg(\lambda(x)) \leq t, \deg(z(x)) < t$.

1: **Initialization:**

$$2: \begin{pmatrix} w^{(0)}(x) & n^{(0)}(x) \\ v^{(0)}(x) & m^{(0)}(x) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}; \begin{pmatrix} d_i^{(0)} \\ g_i^{(0)} \end{pmatrix} = \begin{pmatrix} -s(\omega_i) \\ 1 \end{pmatrix}, i = 0, 1, \dots, 2t-1; \begin{pmatrix} R_0^{(0)} \\ R_1^{(0)} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

3: **for** $r = 0, 1, \dots, 2t-1$ **do**

4: **if** $d_r^{(r)} = 0$ or $((R_0^{(r)} > R_1^{(r)})$ and $g_r^{(r)} \neq 0$) **then**

$$5: \quad \text{Let } \Psi_r = \begin{pmatrix} -g_r^{(r)} & d_r^{(r)} \\ 0 & x - \omega_r \end{pmatrix}$$

$$6: \quad \text{and } \Psi_r(\omega_i) = \begin{pmatrix} -g_r^{(r)} & d_r^{(r)} \\ 0 & \omega_i - \omega_r \end{pmatrix} \text{ for } i = r+1, r+2, \dots, 2t-1$$

$$7: \quad R_0^{(r+1)} = R_0^{(r)}, R_1^{(r+1)} = R_1^{(r)} + 2$$

8: **else**

$$9: \quad \text{Let } \Psi_r = \begin{pmatrix} -g_r^{(r)} & d_r^{(r)} \\ x - \omega_r & 0 \end{pmatrix}$$

$$10: \quad \text{and } \Psi_r(\omega_i) = \begin{pmatrix} -g_r^{(r)} & d_r^{(r)} \\ \omega_i - \omega_r & 0 \end{pmatrix} \text{ for } i = r+1, r+2, \dots, 2t-1$$

$$11: \quad R_0^{(r+1)} = R_1^{(r)}, R_1^{(r+1)} = R_0^{(r)} + 2$$

12: **end if**

13: **for** $i = r+1, r+2, \dots, 2t-1$ **do**

$$14: \quad \begin{pmatrix} d_i^{(r+1)} \\ g_i^{(r+1)} \end{pmatrix} = \Psi_r(\omega_i) \cdot \begin{pmatrix} d_i^{(r)} \\ g_i^{(r)} \end{pmatrix}$$

15: **end for**

16:

$$\begin{pmatrix} w^{(r+1)}(x) & n^{(r+1)}(x) \\ v^{(r+1)}(x) & m^{(r+1)}(x) \end{pmatrix} = \Psi_r \cdot \begin{pmatrix} w^{(r)}(x) & n^{(r)}(x) \\ v^{(r)}(x) & m^{(r)}(x) \end{pmatrix}$$

17: **end for**

18: **if** $R_0^{(2t)} < R_1^{(2t)}$ **then**

$$19: \quad \text{return } (w^{(2t)}(x), n^{(2t)}(x))$$

20: **else**

$$21: \quad \text{return } (v^{(2t)}(x), m^{(2t)}(x))$$

22: **end if**

B. Some Results of MA Algorithm

In the following, we review some existing results about MA algorithm in Algorithm 1, which are useful for deriving our new termination mechanism in the next section. The four variables $g_r^{(r)}, d_r^{(r)}, R_0^{(r)}$ and $R_1^{(r)}$ are used in Algorithm 1 which affect the iterations, where $r = 0, 1, \dots, 2t$. The

two variables $g_r^{(r)}$ and $d_r^{(r)}$ are in \mathbb{F}_{2^m} , while $R_0^{(r)}$ and $R_1^{(r)}$ are nonnegative integers, for $r = 0, 1, \dots, 2t$. Please refer to the literature [13] for the detailed meaning of these four parameters.

The next lemma shows that $g_r^{(r)}$ and $d_r^{(r)}$ cannot be both zero.

Lemma 1. [13, Lemma 4] *Either $g_r^{(r)}$ or $d_r^{(r)}$ is nonzero for $r = 0, 1, \dots, 2t - 1$.*

Lemma 2 shows the relationship between $R_0^{(r)}$ and $R_1^{(r)}$ after each iteration.

Lemma 2. [13, Lemma 5] $R_0^{(r)} + R_1^{(r)} = 2r + 1$ for $r = 0, 1, \dots, 2t$ and if $e \leq t$, $\min\{R_0^{(2t)}, R_1^{(2t)}\} = 2e$.

Lemma 3 reveals the key criterion about the termination mechanism of the MA algorithm.

Lemma 3. [12, Lemma 1] *If $e \leq t$, then there exists some $r \leq t + e$ such that $R_1^{(r)} = 2t + 1$.*

Theorem 4. [12, Theorem 1] *If $e \leq t$, assume that $E \subset \{i\}_{i=2t}^{n-1}$, then $(R_0^{(t+e)}, R_1^{(t+e)}) = (2e, 2t + 1)$, and the roots of $w^{(t+e)}(x)$ are all e error positions.*

Together with Lemma 3 and Theorem 4, we can see that Algorithm 1 can be executed with $t + e$ steps.

III. THE NEW TERMINATION MECHANISM

In this section, we first show that we can execute the MA algorithm by only $2e$ steps to find the error locator polynomial. Note that e is unknown in the error correction procedure, and we further derive a sufficient condition for the MA algorithm to be executed with $2e$ steps.

According to Lemma 2, we can see that the sum of $R_0^{(i)}$ and $R_1^{(i)}$ is an odd number for $0 \leq i \leq 2t$. Therefore, either $R_0^{(i)}$ or $R_1^{(i)}$ is an even number. Let $\text{Even}(R_0^{(i)}, R_1^{(i)})$ be the element in $\{R_0^{(i)}, R_1^{(i)}\}$ which is an even number, where $0 \leq i \leq 2t$. We can obtain the following lemma.

Lemma 5. *For $0 \leq i < j \leq 2t$, we have that*

$$\text{Even}(R_0^{(i)}, R_1^{(i)}) \leq \text{Even}(R_0^{(j)}, R_1^{(j)}).$$

Proof. Without loss of generality, suppose that $R_0^{(i)} = \text{Even}(R_0^{(i)}, R_1^{(i)})$, where $i \in \{0, 1, \dots, 2t - 1\}$. If line 4 in Algorithm 1 is true, then line 7 will be performed, and we have

$$R_0^{(i+1)} = R_0^{(i)} = \text{Even}(R_0^{(i+1)}, R_1^{(i+1)}),$$

i.e., $\text{Even}(R_0^{(i+1)}, R_1^{(i+1)}) = \text{Even}(R_0^{(i)}, R_1^{(i)})$.

On the other hand, if line 4 is false, then line 11 will be performed, and we have

$$R_1^{(i+1)} = R_0^{(i)} + 2 = \text{Even}(R_0^{(i+1)}, R_1^{(i+1)}),$$

i.e., $\text{Even}(R_0^{(i+1)}, R_1^{(i+1)}) > \text{Even}(R_0^{(i)}, R_1^{(i)})$. Therefore, the lemma is proved. \square

The following lemma shows a sufficient condition for finding the e error positions.

Lemma 6. *If $R_0^{(i_0)} = 2e$ and $R_1^{(i_0)} \geq 2e + 1$, where $0 \leq i_0 \leq 2t$, then the roots of $w^{(i_0)}(x)$ are all e error positions.*

Proof. When $R_0^{(i_0)} = 2e$ and $R_1^{(i_0)} \geq 2e + 1$, we have $\text{Even}(R_0^{(i_0)}, R_1^{(i_0)}) = 2e$. According to Lemma 5 and Lemma 2, we have $2e = \text{Even}(R_0^{(i_0)}, R_1^{(i_0)}) \leq \text{Even}(R_0^{(i)}, R_1^{(i)}) \leq \text{Even}(R_0^{(2t)}, R_1^{(2t)}) = 2e$, for any $i \geq i_0$. Therefore, for any $i \geq i_0$, we can obtain that $R_0^{(i)} = R_0^{(i_0)} = 2e$. According to Lemma 2, we have $R_0^{(i)} + R_1^{(i)} = 2i + 1$ for $i \geq i_0$, $R_1^{(i_0)} = 2i_0 + 1 - 2e$, and further obtain that

$$R_1^{(i)} = 2i + 1 - 2e = 2(i - i_0) + R_1^{(i_0)} \geq 2e + 1 > R_0^{(i)}. \quad (3)$$

Since $R_0^{(i)} = R_0^{(i_0)} = 2e$ for any $i \geq i_0$, line 7 of Algorithm 1 must be performed. Therefore, the condition in line 4 of Algorithm 1 must be true. Note that for $i \geq i_0$, we have $R_0^{(i)} < R_1^{(i)}$ by Eq. (3). Then we obtain that $d_i^{(i)} = 0$; otherwise, the condition in line 4 of Algorithm 1 is false. We can further obtain that $g_i^{(i)} \neq 0$ according to Lemma 1. By line 16 of Algorithm 1, for $i \geq i_0$, we have

$$\begin{pmatrix} w^{(i+1)}(x) & n^{(i+1)}(x) \\ v^{(i+1)}(x) & m^{(i+1)}(x) \end{pmatrix} = \begin{pmatrix} -g_i^{(i)} & 0 \\ 0 & x - \omega_i \end{pmatrix} \cdot \begin{pmatrix} w^{(i)}(x) & n^{(i)}(x) \\ v^{(i)}(x) & m^{(i)}(x) \end{pmatrix}.$$

Thus $w^{(i+1)}(x) = -g_i^{(i)}w^{(i)}(x)$, and two polynomials $w^{(i+1)}(x)$ and $w^{(i)}(x)$ have the same roots.

To sum up, $w^{(i)}(x)$ and $w^{(t+e)}(x)$ have the same roots for any $i \geq i_0$. According to Theorem 4, we have that the roots of $w^{(t+e)}(x)$ are all e error positions. Therefore, the roots of $w^{(i_0)}(x)$ are all e error positions. \square

According to Lemma 6, we need to find a value of i_0 such that the condition in Lemma 6 holds. The next lemma shows that the condition in Lemma 6 holds when $i_0 = 2e$.

Lemma 7. *When $i_0 = 2e$, we have $(R_0^{(2e)}, R_1^{(2e)}) = (2e, 2e + 1)$, i.e., the condition in Lemma 6 holds.*

Proof. Let r_0 be the smallest integer such that $R_0^{(r_0)} = 2e$. Obviously, $r_0 \leq t + e$ since $R_0^{(t+e)} = 2e$ according to Theorem 4. Next, we show that $R_1^{(r_0)} = 2e + 1$ by contradiction.

On the one hand, suppose that $R_1^{(r_0)} < 2e + 1$, then

$$R_0^{(r_0)} + R_1^{(r_0)} = 2r_0 + 1 < 2e + (2e + 1),$$

where the above equality comes from Lemma 2. We obtain that $r_0 < 2e$ from the above inequality. According to Lemma 5, for any $2t - 1 \geq i \geq r_0$, we have

$$\begin{aligned} 2e = \text{Even}(R_0^{(r_0)}, R_1^{(r_0)}) &\leq \text{Even}(R_0^{(i)}, R_1^{(i)}) \\ &\leq \text{Even}(R_0^{(2t)}, R_1^{(2t)}) = 2e. \end{aligned}$$

Therefore, $R_0^{(i)} = R_0^{(r_0)} = 2e$ and $R_1^{(i)} = 2i + 1 - 2e$ for $2t \geq i \geq r_0$. Since $r_0 < 2e$, we have $(R_0^{(2e)}, R_1^{(2e)}) = (2e, 2e + 1)$.

On the other hand, suppose that $R_1^{(r_0)} > 2e + 1$. Note that $R_0^{(r_0)} = 2e$ and $R_0^{(r_0-1)} < 2e$. According to line 7 and line 11, we can obtain that $R_0^{(r_0-1)} = R_1^{(r_0)} - 2$; otherwise, we have

$R_0^{(r_0-1)} = R_0^{(r_0)} = 2e$, which contradicts with $R_0^{(r_0-1)} < 2e$. Therefore, when $r = r_0 - 1$ in line 3, we will perform line 11 and the condition in line 4 is false. Since $R_1^{(r_0)}$ is an odd number, we have $R_1^{(r_0)} \geq 2e + 3$ and

$$R_0^{(r_0-1)} = R_1^{(r_0)} - 2 \geq 2e + 1 > 2e = R_1^{(r_0-1)},$$

then we have $d_{r_0-1}^{(r_0-1)} \neq 0$ and $g_{r_0-1}^{(r_0-1)} = 0$. By line 16 with $r = r_0 - 1$, we have

$$\begin{pmatrix} w^{(r_0)}(x) & n^{(r_0)}(x) \\ v^{(r_0)}(x) & m^{(r_0)}(x) \end{pmatrix} = \begin{pmatrix} 0 & d_{r_0-1}^{(r_0-1)} \\ x - \omega_{r_0-1} & 0 \end{pmatrix} \cdot \begin{pmatrix} w^{(r_0-1)}(x) & n^{(r_0-1)}(x) \\ v^{(r_0-1)}(x) & m^{(r_0-1)}(x) \end{pmatrix},$$

and $w^{(r_0)}(x) = d_{r_0-1}^{(r_0-1)} v^{(r_0-1)}(x)$. Therefore, two polynomials $w^{(r_0)}(x)$ and $v^{(r_0-1)}(x)$ have the same roots. According to Lemma 6, the roots of $w^{(r_0)}(x)$ are all e error positions, thus the roots of $v^{(r_0-1)}(x)$ are also all error positions.

When $r = r_0 - 2$, if the condition in line 4 is true, then we have $v^{(r_0-1)}(x) = (x - \omega_{r_0-2}) v^{(r_0-2)}(x)$ by line 16, which contradicts with the assumption that $v^{(r_0-1)}(x)$ does not contain the root whose index less than $2t$. Otherwise, if the condition in line 4 is false, then we have $v^{(r_0-1)}(x) = (x - \omega_{r_0-2}) w^{(r_0-2)}(x)$ by line 16, which also contradicts with the assumption that the root index of $v^{(r_0-1)}(x)$ is no less than $2t$.

Therefore, we have $R_1^{(r_0)} = 2e + 1$. According to Lemma 2, $R_0^{(r_0)} + R_1^{(r_0)} = 2r_0 + 1 = 4e + 1$, then we have $r_0 = 2e$. The lemma is proved. \square

Combining Lemma 5 to Lemma 7, we can know that $(R_0^{(2e)}, R_1^{(2e)}) = (2e, 2e + 1)$ and the roots of $w^{(2e)}(x)$ are all e error positions when $e \leq t$ and $E \subset \{i\}_{i=2t}^{n-1}$. Therefore, we can obtain the error locator polynomial after $2e$ iterations in Algorithm 1. However, the value of e is unknown. Next, we present a sufficient condition to find the error locator polynomial by only $2e$ iterations.

Lemma 8. *Given an integer i_1 with $2t - 1 \geq i_1 \geq 0$, if $d_j^{(i_1)} = 0$ for all $2t - 1 \geq j \geq i_1$, then $w^{(i_1)}(x)$ is the error locator polynomial.*

Proof. Since $d_j^{(i_1)} = 0$ for any $2t - 1 \geq j \geq i_1$, according to line 14 in Algorithm 1 with $i = j$, we have

$$d_j^{(i_1+1)} = -g_{i_1}^{(i_1)} d_j^{(i_1)} + d_{i_1}^{(i_1)} g_j^{(i_1)} = 0,$$

for any $2t - 1 \geq j \geq i_1 + 1$. By analogy, we can know that $d_j^{(i)} = 0$ for any $i_1 \leq i \leq 2t - 1$ and $i \leq j \leq 2t - 1$ by repeatedly using line 14 in Algorithm 1.

Therefore, for any $r \geq i_1$, we have $d_r^{(r)} = 0$ and the condition in line 4 of Algorithm 1 is true. Then we have

$$\begin{pmatrix} w^{(r+1)}(x) & n^{(r+1)}(x) \\ v^{(r+1)}(x) & m^{(r+1)}(x) \end{pmatrix} = \begin{pmatrix} -g_r^{(r)} & 0 \\ 0 & x - \omega_r \end{pmatrix} \cdot \begin{pmatrix} w^{(r)}(x) & n^{(r)}(x) \\ v^{(r)}(x) & m^{(r)}(x) \end{pmatrix},$$

i.e., $w^{(r+1)}(x) = -g_r^{(r)} w^{(r)}(x)$. By Lemma 1, we can see that $g_r^{(r)} \neq 0$ for $r \geq i_1$. Therefore, polynomials $w^{(r+1)}(x)$

and $w^{(r)}(x)$ have the same roots for any $r \geq i_1$. We can obtain that $w^{(i_1)}(x)$ and $w^{(2t)}(x)$ have the same roots. Since $w^{(2t)}(x)$ is the error locator polynomial, then $w^{(i_1)}(x)$ is also an error locator polynomial. \square

Next, we show that the condition in Lemma 8 holds for $i_1 = 2e$.

Lemma 9. *For any $2t - 1 \geq j \geq 2e$, we have $d_j^{(2e)} = 0$.*

Proof. According to Lemma 6 and Lemma 7, we have that $R_0^{(2e)} = 2e$, the roots of $w^{(2e)}(x)$ are all e error positions, and $d_i^{(i)} = 0$ for all $2e \leq i \leq 2t - 1$. Next, we show that $d_j^{(2e)} = 0$ for any $j \geq 2e$ by contradiction.

Suppose there exists j_0 with $2e \leq j_0 \leq 2t - 1$ such that $d_{j_0}^{(2e)} \neq 0$. According to line 14 in Algorithm 1, we have

$$d_{j_0}^{(2e+1)} = -g_{2e}^{(2e)} d_{j_0}^{(2e)} + d_{2e}^{(2e)} g_{j_0}^{(2e)} = -g_{2e}^{(2e)} d_{j_0}^{(2e)}.$$

Similarly, we can obtain

$$\begin{aligned} d_{j_0}^{(2e+2)} &= -g_{2e+1}^{(2e+1)} d_{j_0}^{(2e+1)} + d_{2e+1}^{(2e+1)} g_{j_0}^{(2e+1)} \\ &= -g_{2e+1}^{(2e+1)} d_{j_0}^{(2e+1)} \\ &= (-g_{2e+1}^{(2e+1)}) (-g_{2e}^{(2e)}) d_{j_0}^{(2e)}, \end{aligned}$$

and further, obtain

$$d_{j_0}^{(j_0)} = \prod_{j=2e}^{j_0-1} (-g_j^{(j)}) \cdot d_{j_0}^{(2e)}. \quad (4)$$

By Lemma 1, we have $g_j^{(j)} \neq 0$ for all $2e \leq j \leq j_0 - 1$. Since $d_{j_0}^{(2e)} \neq 0$, i.e., the right side of Eq. (4) is non-zero, which contradicts with $d_{j_0}^{(j_0)} = 0$. Therefore, the lemma is finished. \square

According to Lemma 8 and Lemma 9, we only need to check whether $d_j^{(r)} = 0$ for all $r \leq j \leq 2t - 1$ in iteration r . If $d_j^{(r)} = 0$ for all $r \leq j \leq 2t - 1$ in iteration r , then we have $r = 2e$ and we can stop Algorithm 1 to output the error locator polynomial.

IV. OUR FIRST DECODING ALGORITHM BASED ON I-FDMA ALGORITHM

In this section, we present our first decoding algorithm for LCH-FFT-based RS codes [9]. We first review the LCH-FFT related algorithms [9], [13], then propose the I-FDMA algorithm based on the new termination mechanism derived in Section III, and finally give our first decoding algorithm based on the LCH-FFT related algorithms and the I-FDMA algorithm.

A. LCH-FFT Related Algorithms

The encoding of LCH-FFT-based RS codes [9] is based on LCH-FFT algorithm under LCH-basis $\bar{X} = \{\bar{X}_0(x), \bar{X}_1(x), \dots, \bar{X}_{2m-1}(x)\}$ in linear space $\mathbb{F}_{2^m}[x]/(x^{2^m} - x)$, where

$$\bar{X}_i(x) := \frac{\prod_{j=0}^{m-1} (s_j(x))^{i_j}}{p_i}, \quad p_i = \prod_{j=0}^{m-1} (s_j(v_j))^{i_j}$$

for any $i = \sum_{j=0}^{m-1} i_j \cdot 2^j \in \{0, 1, \dots, 2^m - 1\}$, and for any $j \in \{0, 1, \dots, m\}$, the subspace polynomial $s_j(x) := \prod_{i=0}^{2^j-1} (x - \omega_j)$.

The LCH-FFT algorithm is used to calculate the following estimation problem: given a polynomial $f(x) = \sum_{i=0}^{2^k-1} f_i \bar{X}_i(x)$ in $\mathbb{F}_{2^m}[x]/(x^{2^m} - x)$ with $\deg(f(x)) < 2^k$, $k \leq m$, and any $\beta \in \mathbb{F}_{2^m}$, calculate the values of $f(x)$ at 2^k points $\{\omega_i + \beta\}_{i=0}^{2^k-1}$, i.e., $\{f(\omega_i + \beta)\}_{i=0}^{2^k-1}$. Let $\mathbf{f}_{\bar{X}} := (f_0, f_1, \dots, f_{2^k-1})$ represent the coefficient vector of polynomial $f(x)$ under basis \bar{X} . We give the LCH-FFT algorithm in Algorithm 2.

The inverse process of LCH-FFT algorithm, namely LCH-IFFT algorithm, corresponds to the interpolation problem: given 2^k values $\{d_i = f(\omega_i + \beta)\}_{i=0}^{2^k-1}$ in \mathbb{F}_{2^m} , where $\beta \in \mathbb{F}_{2^m}$ and $k \leq m$, calculate the coefficient vector $\mathbf{f}_{\bar{X}}$ of polynomial $f(x) = \sum_{i=0}^{2^k-1} f_i \bar{X}_i(x)$ based on basis \bar{X} . We give the LCH-IFFT algorithm in Algorithm 3.

Tang *et al.* proposed the LCH-Extended IFFT algorithm [13] to solve the $2^k + 1$ -point interpolation problem, which is based on LCH-IFFT Algorithm 3 and can be applied to the decoding process of RS codes. We give the LCH-Extended IFFT algorithm in Algorithm 4. Let $h = 2^k$, the complexity of each of Algorithms 2, 3 and 4 is $O(h \lg(h))$ [9], [13].

Algorithm 2 $FFT_{\bar{X}}(\mathbf{f}_{\bar{X}}, k, \beta)$ [9]

Require: $\mathbf{f}_{\bar{X}} = (f_0, f_1, \dots, f_{2^k-1})$, $0 \leq k \leq m$, $\beta \in \mathbb{F}_{2^m}$

Ensure: 2^k values: $(f(\omega_0 + \beta), f(\omega_1 + \beta), \dots, f(\omega_{2^k-1} + \beta))$

```

1: if  $k = 0$  then
2:   return  $f_0$ 
3: end if
4: for  $i = 0, 1, \dots, 2^{k-1} - 1$  do
5:    $g_i^{(0)} = f_i + f_{i+2^{k-1}} \cdot \frac{s_{k-1}(\beta)}{s_{k-1}(v_{k-1})}$ 
6:    $g_i^{(1)} = g_i^{(0)} + f_{i+2^{k-1}}$ 
7: end for
8:  $V_0 = FFT_{\bar{X}}(g^{(0)}, k - 1, \beta)$ , where  $g^{(0)} = (g_0^{(0)}, g_1^{(0)}, \dots, g_{2^{k-1}-1}^{(0)})$ 
9:  $V_1 = FFT_{\bar{X}}(g^{(1)}, k - 1, \beta)$ , where  $g^{(1)} = (g_0^{(1)}, g_1^{(1)}, \dots, g_{2^{k-1}-1}^{(1)})$ 
10: return  $(V_0, V_1)$ 

```

In the following, we consider the encoding of LCH-FFT-based $(n = 2^m, k = 2^m - 2^\mu)$ RS codes over \mathbb{F}_{2^m} whose error correction capability is $t = 2^{\mu-1}$, where $\mu < m$. Let $T := n - k = 2^\mu$. Any $1 \times n$ codeword vector \mathbf{F} can be represented as $\mathbf{F} = (\mathbf{F}_1, \mathbf{F}_2, \dots, \mathbf{F}_{2^m-\mu})$, where $\mathbf{F}_i = (f(\omega_{(i-1)2^\mu}), f(\omega_{(i-1)2^\mu+1}), \dots, f(\omega_{i2^\mu-1}))$ for any $i \in \{1, 2, \dots, 2^{m-\mu}\}$. The following lemma [9, Lemma 10] is the key to the encoding of systematic code.

Lemma 10. [9, Lemma 10]

$$IFFT_{\bar{X}}(\mathbf{F}_1, \mu, \omega_0) + IFFT_{\bar{X}}(\mathbf{F}_2, \mu, \omega_{2^\mu}) + \dots + IFFT_{\bar{X}}(\mathbf{F}_{2^m-\mu}, \mu, \omega_{2^m-2^\mu}) = \mathbf{0}_{1 \times 2^\mu}, \quad (5)$$

in which the addition between two vectors means adding the elements corresponding to each position, and $\mathbf{0}_{a \times b}$ denotes a $a \times b$ matrix with each element being 0.

Algorithm 3 $IFFT_{\bar{X}}(\mathbf{D}_{2^k}, k, \beta)$ [9]

Require: $\mathbf{D}_{2^k} = (d_0, d_1, \dots, d_{2^k-1})$, where $d_i = f(\omega_i + \beta)$ and $0 \leq k \leq m$, $\beta \in \mathbb{F}_{2^m}$

Ensure: $\mathbf{f}_{\bar{X}} = (f_0, f_1, \dots, f_{2^k-1})$

```

1: if  $k = 0$  then
2:   return  $d_0$ 
3: end if
4:  $(g_0^{(0)}, g_1^{(0)}, \dots, g_{2^{k-1}-1}^{(0)}) = IFFT_{\bar{X}}(V_0, k - 1, \beta)$ , where  $V_0 = (d_0, d_1, \dots, d_{2^{k-1}-1})$ 
5:  $(g_0^{(1)}, g_1^{(1)}, \dots, g_{2^{k-1}-1}^{(1)}) = IFFT_{\bar{X}}(V_1, k - 1, \beta)$ , where  $V_1 = (d_{2^{k-1}}, d_{2^{k-1}+1}, \dots, d_{2^k-1})$ 
6: for  $i = 0, 1, \dots, 2^{k-1} - 1$  do
7:    $f_{i+2^{k-1}} = g_i^{(0)} + g_i^{(1)}$ 
8:    $f_i = g_i^{(0)} + \frac{s_{k-1}(\beta)}{s_{k-1}(v_{k-1})} \cdot f_{i+2^{k-1}}$ 
9: end for
10: return  $(f_0, f_1, \dots, f_{2^k-1})$ 

```

Algorithm 4 *Extended IFFT* $_{\bar{X}}(k, \beta)$ [13]

Require: $\{f(\omega_i + \beta)\}_{i=0}^{2^k}$, $0 \leq k \leq m$, $\beta \in \mathbb{F}_{2^m}$

Ensure: $\mathbf{f}_{\bar{X}} = (f_0, f_1, \dots, f_{2^k})$

```

1: Call Algorithm3 with input  $(f(\omega_0 + \beta), f(\omega_1 + \beta), \dots, f(\omega_{2^k-1} + \beta))$ ,  $k, \beta$  to obtain  $\hat{f}(x)$ 
2: Calculate  $\hat{f}(\omega_{2^k} + \beta)$ 
3:  $f(x) = (f(\omega_{2^k} + \beta) - \hat{f}(\omega_{2^k} + \beta)) \cdot \bar{X}_{2^k}(x) + \hat{f}(x) - \frac{s_k(\beta)}{s_k(v_k)} \cdot (f(\omega_{2^k} + \beta) - \hat{f}(\omega_{2^k} + \beta)) \cdot \bar{X}_0(x)$ 
4: return  $f(x)$ 

```

According to Lemma 10, we can perform systematic encoding for $(n = 2^m, k = 2^m - 2^\mu)$ RS codes by setting $\mathbf{F}_2, \mathbf{F}_3, \dots, \mathbf{F}_{2^m-\mu}$ as data symbols, and the parity symbol vector \mathbf{F}_1 can be calculated as follows

$$\mathbf{F}_1 = FFT_{\bar{X}}(IFFT_{\bar{X}}(\mathbf{F}_2, \mu, \omega_{2^\mu}) + \dots + IFFT_{\bar{X}}(\mathbf{F}_{2^m-\mu}, \mu, \omega_{2^m-2^\mu}), \mu, \omega_0).$$

Since the encoding process involves $(n/T - 1)$ T -point LCH-FFT algorithms and one T -point LCH-IFFT algorithm, its complexity is $O(T \lg(T)) + (n/T - 1)O(T \lg(T)) = O(n \lg(n - k))$.

B. The I-FDMA Algorithm

In the following, we present the I-FDMA algorithm for solving the error locator polynomial, which is based on our new termination mechanism of MA method in Section III.

We show the I-FDMA algorithm in Algorithm 5. Recall that in Eq. (2), when $e \leq t$, we have $\deg(z(x)) < \deg(\lambda(x)) = e \leq t = 2^{\mu-1}$. Our I-FDMA algorithm only needs to update $t + 1$ evaluations $\{w^{(r)}(\omega_i), v^{(r)}(\omega_i)\}_{i=0}^t$ of $w^{(r)}(x), v^{(r)}(x)$ for $r \in \{1, 2, \dots, 2t\}$, and output the $t + 1$ evaluations $\{\lambda(\omega_i)\}_{i=0}^t$ of the error locator polynomial. Note that our I-FDMA algorithm does not need to update the evaluations of two polynomials $n^{(r)}(x)$ and $v^{(r)}(x)$. Given an integer r with $r \in \{0, 1, \dots, 2t - 1\}$, when $d_i^{(r)} = 0$ for all $i = r, r + 1, \dots, 2t - 1$, then our I-FDMA algorithm can output $\{w^{(r)}(\omega_i)\}_{i=0}^t = \{\lambda(\omega_i)\}_{i=0}^t$. This is because the roots

of $w^{(r)}(x)$ are the e error positions when $r = 2e$ according to Lemma 8 and Lemma 9. Once $\{\lambda(\omega_i)\}_{i=0}^t$ are obtained, we can solve the error locator polynomial $\lambda(x)$ by employing Algorithm 4.

Note that in Algorithm 5, line 14 requires three multiplications and two additions, and line 17 requires three multiplications and two additions. Therefore, the number of multiplications involved in I-FDMA algorithm is

$$\sum_{r=0}^{2e-1} (3(2t-r-1) + 3(t+1)) = 18et - 6e^2 + 3e, \quad (6)$$

and the number of additions involved in I-FDMA algorithm is

$$\sum_{r=0}^{2e-1} (2(2t-r-1) + 2(t+1)) = 12et - 4e^2 + 2e. \quad (7)$$

Algorithm 5 The I-FDMA Algorithm

Require: $\{s(\omega_i)\}_{i=0}^{2t-1}$
Ensure: $(\lambda^{(r+1)}(\omega_0), \lambda^{(r+1)}(\omega_1), \dots, \lambda^{(r+1)}(\omega_t))$

- 1: Initialization:
- 2: $\begin{pmatrix} d_i^{(0)} \\ g_i^{(0)} \end{pmatrix} = \begin{pmatrix} -s(\omega_i) \\ 1 \end{pmatrix}, i = 0, 1, \dots, 2t-1;$
- 3: $\begin{pmatrix} W_i^{(0)} \\ V_i^{(0)} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, i = 0, 1, \dots, t;$
- 4: $\begin{pmatrix} R_0^{(0)} \\ R_1^{(0)} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$
- 5: **for** $r = 0, 1, \dots, 2t-1$ **do**
- 6: **if** $(d_r^{(r)} = 0)$ or $((R_0^{(r)} > R_1^{(r)})$ and $g_r^{(r)} \neq 0)$ **then**
- 7: Let $\Psi_r(\omega_i) = \begin{pmatrix} -g_r^{(r)} & d_r^{(r)} \\ 0 & \omega_i - \omega_r \end{pmatrix}$ for $i = r+1, r+2, \dots, 2t-1$
- 8: $R_0^{(r+1)} = R_0^{(r)}, R_1^{(r+1)} = R_1^{(r)} + 2$
- 9: **else**
- 10: Let $\Psi_r(\omega_i) = \begin{pmatrix} -g_r^{(r)} & d_r^{(r)} \\ \omega_i - \omega_r & 0 \end{pmatrix}$ for $i = r+1, r+2, \dots, 2t-1$
- 11: $R_0^{(r+1)} = R_1^{(r)}, R_1^{(r+1)} = R_0^{(r)} + 2$
- 12: **end if**
- 13: **for** $i = r+1, r+2, \dots, 2t-1$ **do**
- 14: $\begin{pmatrix} d_i^{(r+1)} \\ g_i^{(r+1)} \end{pmatrix} = \Psi_r(\omega_i) \cdot \begin{pmatrix} d_i^{(r)} \\ g_i^{(r)} \end{pmatrix}$
- 15: **end for**
- 16: **for** $i = 0, 1, \dots, t$ **do**
- 17: $\begin{pmatrix} W_i^{(r+1)} \\ V_i^{(r+1)} \end{pmatrix} = \Psi_r(\omega_i) \cdot \begin{pmatrix} W_i^{(r)} \\ V_i^{(r)} \end{pmatrix}$
- 18: **end for**
- 19: **if** $(d_i^{(r+1)} = 0$ for $r+1 \leq i \leq 2t-1)$ or $(r = 2t-1)$ **then**
- 20: **return** $(W_0^{(r+1)}, W_1^{(r+1)}, \dots, W_t^{(r+1)})$
- 21: **end if**
- 22: **end for**

C. Our First Decoding Algorithm

In the following, we introduce our first decoding algorithm for LCH-FFT-based RS codes based on the I-FDMA algorithm.

Assume that the received vector is

$$\begin{aligned} \mathbf{r} &= (r_0, r_1, \dots, r_{2^m-1}) \\ &= \mathbf{F} + \mathbf{e} \\ &= (f(\omega_0), f(\omega_1), \dots, f(\omega_{2^m-1})) + (e_0, e_1, \dots, e_{2^m-1}), \end{aligned}$$

where \mathbf{e} is the error pattern vector. The index set of error positions is $E = \{i | e_i \neq 0, i \in \{0, 1, \dots, 2^m-1\}\} \subset [2t, 2^m-1]$, and $|E| = e \leq t$. Let $\mathbf{r}_i := \{r_{i \cdot 2^\mu}, r_{i \cdot 2^\mu+1}, \dots, r_{i \cdot 2^\mu+2^\mu-1}\}$, for $i \in \{0, 1, \dots, 2^{m-\mu}-1\}$. Now, we introduce the decoding process.

First, we compute the syndrome polynomial $s(x)$ by Algorithm 2 as follows [13],

$$\sum_{i=0}^{2^\mu-1} IFFT_{\bar{X}}(\mathbf{r}_i, \mu, \omega_{i \cdot 2^\mu}) / p_{2^m-2^\mu}.$$

Then we can calculate $2t$ syndromes $\{s(\omega_i)\}_{i=0}^{2t-1}$ (i.e., the input of Algorithm 5) by $FFT_{\bar{X}}(\mathbf{s}_{\bar{X}}, \mu, \omega_0)$. Then we can get the $t+1$ values $\{\lambda(\omega_i)\}_{i=0}^t$ of the error locator polynomial $\lambda(x)$ by I-FDMA Algorithm 5. Next, according to Eq. (2), we can compute

$$z(\omega_i) = s(\omega_i)\lambda(\omega_i) = s(\omega_i)w(\omega_i), i = 0, 1, \dots, t. \quad (8)$$

Then we can compute $z(x)$ and $\lambda(x)$ by Algorithm 4. After obtaining $\lambda(x)$, we can use the Chien search method to find the roots of $\lambda(x)$ by Algorithm 2 as follows,

$$FFT_{\bar{X}}(\lambda_{\bar{X}}, \mu, \omega_{i \cdot 2^\mu}), i = 0, 1, \dots, 2^{m-\mu}-1. \quad (9)$$

Finally, we use Forney's formula to compute the error values as follows,

$$f(\omega_\ell) - r(\omega_\ell) = \frac{z(\omega_\ell)}{s_\mu(\omega_\ell)\lambda'(\omega_\ell)}, \forall \ell \in E, \quad (10)$$

where $\lambda'(x)$ represents the formal derivative of $\lambda(x)$. Please refer to [13] for the derivation of the above processes.

We summarize our first decoding algorithm in Algorithm 6. The computational complexity of computing syndrome polynomial and syndromes, Chien search, formal derivative, and Forney's formula is $O(n \lg(n-k))$ [9], [13]. Note that Algorithm 6 can be generalized to any parameters n and k with asymptotic complexity $O(n \lg(n-k) + (n-k) \lg^2(n-k))$ [14]. Please refer to [14] for relevant details.

V. OUR SECOND DECODING ALGORITHM

In this section, we give our second decoding algorithm for RS codes. Our second decoding algorithm has lower decoding complexity than our first decoding algorithm when $e \ll t$. In the following, we still consider the LCH-FFT-based ($n = 2^m, k = 2^m - 2^\mu$) RS codes as in Section IV.

The general idea of our second decoding algorithm is as follows: **Step (1)**: we first determine the number of errors e by the t_0 -Shortened I-FDMA (t_0 -SI-FDMA) algorithm (which is presented in the next subsection); **Step (2)**: then we iterate the

Algorithm 6 Our First Decoding Algorithm

Require: Received vector $\mathbf{r} = \mathbf{F} + \mathbf{e}$ **Ensure:** The codeword \mathbf{F}

- 1: Compute the syndrome polynomial $s(x)$ and syndromes $\{s(\omega_i)\}_{i=0}^{2^\mu-1}$
 - 2: Compute $\{\lambda(\omega_i)\}_{i=0}^{2^\mu-1}$ by I-FDMA Algorithm 5, and get the error locator polynomial $\lambda(x)$ by Algorithm 4
 - 3: Compute $\{z(\omega_i)\}_{i=0}^{2^\mu-1}$, and get $z(x)$ by Algorithm 4
 - 4: Find all error positions by Chien search
 - 5: Compute the formal derivative of $\lambda(x)$, and then get the error pattern \mathbf{e} by Forney's formula
 - 6: **return** The codeword $\mathbf{F} = \mathbf{r} + \mathbf{e}$
-

Early-Stopped Berlekamp–Massey (ESBM) algorithm [7] $2e$ steps to find the error locator polynomial; **Step (3)**: finally, together with LCH-FFT related algorithms in Section IV-A, we find the e error locations and error values. The main difference between our second decoding algorithm and our first decoding algorithm in Algorithm 6 is as follows. We employ I-FDMA algorithm to solve the error locator polynomial in our first decoding algorithm. While in our second decoding algorithm, we first propose the t_0 -SI-FDMA algorithm to determine the number of errors e and then employ ESBM algorithm to solve the error locator polynomial. We will show that our second decoding algorithm has lower multiplication complexity than our first decoding algorithm in Algorithm 6 when $e \ll t$.

A. The t_0 -SI-FDMA Algorithm

In the following, we propose the t_0 -SI-FDMA algorithm that can quickly determine the number of errors e .

The idea behind the t_0 -SI-FDMA algorithm is based on the following two observations. First, we can see that lines 16 to 18 in I-FDMA Algorithm 5 do not affect the values of $\{d_r^{(r)}, g_r^{(r)}, R_0^{(r)}, R_1^{(r)}\}_{r=0}^{2t}$, which are the core parameters that affect the iterations of Algorithm 5. Note that lines 16 to 18 in Algorithm 5 are just updating the $t+1$ values of the error locator polynomial. Therefore, if we delete lines 16 to 18 from Algorithm 5, and when the algorithm terminates, we output $R_0^{(r)}/2$ (note that $R_0^{(2e)} = 2e$ according to Lemma 8), then $R_0^{(r)}/2$ must be the number of errors e . Second, for a given even number t_0 and $t_0 < 2t$, we will show (refer to Theorem 11) that Algorithm 5 does not need to input all $2t$ syndromes $\{s(\omega_i)\}_{i=0}^{2t-1}$, but only $t_0 + 1$ syndromes $\{s(\omega_i)\}_{i=0}^{t_0}$, and we can still output the number of errors e after $2e$ iterations when $2e < t_0 + 1$.

We give the t_0 -SI-FDMA algorithm in Algorithm 7. The following theorem shows the key properties of Algorithm 7.

Theorem 11. *Let t_0 be an even number. When $2e < t_0 + 1$, Algorithm 7 can output the number of errors e ; if the algorithm has no output, then there must be $2e > t_0 + 1$.*

Proof. If $2e < t_0 + 1$, we can see that $d_j^{(2e)} = 0$ for $j = 2e, 2e+1, \dots, t_0$ by Lemma 9, which means that line 15 of Algorithm 7 is true when $r = 2e - 1$, and Algorithm 7 must be terminated in no more than $2e$ iterations. Note that the

Algorithm 7 The t_0 -SI-FDMA Algorithm

Require: $\{s(\omega_i)\}_{i=0}^{t_0}$ **Ensure:** The number of errors e

- 1: Initialization:
 - 2: $\begin{pmatrix} d_i^{(0)} \\ g_i^{(0)} \end{pmatrix} = \begin{pmatrix} -s(\omega_i) \\ 1 \end{pmatrix}, i = 0, 1, \dots, t_0;$
 - 3: $\begin{pmatrix} R_0^{(0)} \\ R_1^{(0)} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$
 - 4: **for** $r = 0, 1, \dots, t_0$ **do**
 - 5: **if** $(d_r^{(r)} = 0)$ or $((R_0^{(r)} > R_1^{(r)})$ and $g_r^{(r)} \neq 0)$ **then**
 - 6: Let $\Psi_r(\omega_i) = \begin{pmatrix} -g_r^{(r)} & d_r^{(r)} \\ 0 & \omega_i - \omega_r \end{pmatrix}$ for $i = r+1, r+2, \dots, t_0$
 - 7: $R_0^{(r+1)} = R_0^{(r)}, R_1^{(r+1)} = R_1^{(r)} + 2$
 - 8: **else**
 - 9: Let $\Psi_r(\omega_i) = \begin{pmatrix} -g_r^{(r)} & d_r^{(r)} \\ \omega_i - \omega_r & 0 \end{pmatrix}$ for $i = r+1, r+2, \dots, t_0$
 - 10: $R_0^{(r+1)} = R_0^{(r)}, R_1^{(r+1)} = R_0^{(r)} + 2$
 - 11: **end if**
 - 12: **for** $i = r+1, r+2, \dots, t_0$ **do**
 - 13: $\begin{pmatrix} d_i^{(r+1)} \\ g_i^{(r+1)} \end{pmatrix} = \Psi_r(\omega_i) \cdot \begin{pmatrix} d_i^{(r)} \\ g_i^{(r)} \end{pmatrix}$
 - 14: **end for**
 - 15: **if** $d_i^{(r+1)} = 0$ for $r+1 \leq i \leq t_0$ **then**
 - 16: **return** $R_0^{(r)}/2$
 - 17: **end if**
 - 18: **end for**
-

above analysis has shown that if Algorithm 7 does not output any value, it means that there must be $2e > t_0 + 1$.

Next, we only need to show that Algorithm 7 will be terminated strictly after $2e$ iterations. We prove it by contradiction. Assume that Algorithm 7 terminates after u iterations, where $u < 2e$. Then we have $d_j^{(u)} = 0$ for $j = u, u+1, \dots, t_0$. According to line 13, we have

$$d_j^{(u+1)} = -g_u^{(u)} d_j^{(u)} + d_u^{(u)} g_j^{(u)} = 0,$$

for $j = u+1, u+2, \dots, t_0$. By analogy, for any $a = u, u+1, \dots, t_0$ and $b = a, a+1, \dots, t_0$, we have $d_b^{(a)} = 0$. By taking $a = 2e-1 < t_0$ and $b = a$, we have $d_b^{(a)} = d_{2e-1}^{(2e-1)} = 0$. Then, the judgment condition of line 5 in the $2e$ -th iteration (i.e., $r = 2e-1$) of Algorithm 7 is true, so that line 7 is executed, and thus there is $R_0^{(2e)} = R_0^{(2e-1)}$. However, according to Lemma 7, the smallest integer r_0 such that $R_0^{(r_0)} = 2e$ is $r_0 = 2e$, which contradicts with $R_0^{(2e)} = R_0^{(2e-1)} = 2e$. To sum up, Algorithm 7 will terminate strictly after $2e$ iterations and the theorem is proved. \square

According to Theorem 11, given an even number t_0 , the t_0 -SI-FDMA algorithm is more suitable for the case where e is relatively small (i.e., $2e < t_0 + 1$). If the t_0 -SI-FDMA algorithm does not return any value, it means that e is relatively large (i.e., $2e > t_0 + 1$), then we can employ the first decoding algorithm in Algorithm 6. Note that the values

of $\{d_j^{(i)}, g_j^{(i)}\}_{i=0,1,\dots,t_0}^{j=i,i+1,\dots,t_0}$ have been calculated by the t_0 -SI-FDMA algorithm and they do not need to be calculated again when performing I-FDMA Algorithm in our first decoding algorithm. Therefore, the computational complexity of Algorithm 6 will not be increased, when $2e > t_0 + 1$.

B. Parity-Check Matrix and Syndromes

In the following, unless otherwise specified, we assume that $2e < t_0 + 1$. After **Step (1)**, we can get the number of errors e by t_0 -SI-FDMA algorithm. We first analyze the parity-check matrix and the corresponding syndromes for LCH-FFT-based $(n = 2^m, k = 2^m - 2^\mu)$ RS codes to show that we can apply the ESBM algorithm [7] to find the error locator polynomial in **Step (2)**.

First, we introduce the key Theorem [15, p.167] for deriving the parity-check matrix of LCH-FFT-based RS codes.

Theorem 12. [15, p.167] Consider any (n, k) RS code over $\mathbb{F}_{2^m} = \{\omega_i\}_{i=0}^{2^m-1}$ ($0 < k < n \leq 2^m$) which is given by

$$\{f(\omega_0), f(\omega_1), \dots, f(\omega_{n-1}) | f(x) \in \mathbb{F}_{2^m}[x], \deg(f(x)) < k\},$$

then the dual code of the (n, k) RS code can be given by

$$\{\mu_0 g(\omega_0), \mu_1 g(\omega_1), \dots, \mu_{n-1} g(\omega_{n-1}) | g(x) \in \mathbb{F}_{2^m}[x], \deg(g(x)) < n - k\},$$

$$\text{where } \mu_i = \frac{1}{\prod_{i \neq j} (\omega_i - \omega_j)}.$$

According to Theorem 12, when $n = 2^m$, $\{\omega_i\}_{i=0}^{n-1}$ form a linear space over \mathbb{F}_2 , then $\mu_0 = \mu_1 = \dots = \mu_{n-1}$, and the parity-check matrix of LCH-FFT-based $(n = 2^m, k = 2^m - 2^\mu)$ RS codes \mathbf{H} is an $(n - k) \times n$ Vandermonde matrix, specially,

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ \omega_0 & \omega_1 & \omega_2 & \dots & \omega_{2^m-1} \\ \omega_0^2 & \omega_1^2 & \omega_2^2 & \dots & \omega_{2^m-1}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega_0^{n-k-1} & \omega_1^{n-k-1} & \omega_2^{n-k-1} & \dots & \omega_{2^m-1}^{n-k-1} \end{bmatrix}. \quad (11)$$

Based on Eq. (11), we have

$$\mathbf{H} \cdot \mathbf{r}^T = \mathbf{H} \cdot \mathbf{F}^T + \mathbf{H} \cdot \mathbf{e}^T = \mathbf{H} \cdot \mathbf{e}^T. \quad (12)$$

Let $\mathbf{H} \cdot \mathbf{r}^T := (S_0, S_1, \dots, S_{2t-1})^T$, and $\{S_i\}_{i=0}^{2t-1}$ be the $2t$ syndromes. Suppose there are $e < \frac{t_0}{2}$ errors, and $e_{i_j} \neq 0$ for any $j \in [e]$, where $E = \{i_j\}_{j \in [e]} \in [2t, n-1]$. Let $\beta_j := \omega_{i_j}$ and $\delta_j = e_{i_j}$, for $j \in [e]$. According to Eq. (12), we have

$$S_i = \beta_1^i \delta_1 + \beta_2^i \delta_2 + \dots + \beta_e^i \delta_e, \quad \forall i \in \{0, 1, \dots, 2t-1\}. \quad (13)$$

Let

$$\sigma(x) := \prod_{j \in [e]} (1 - \beta_j x) = 1 + \sigma_1 x + \sigma_2 x^2 + \dots + \sigma_e x^e$$

be the error locator polynomial of our second decoding method (which differs from that of Algorithm 6). Once the unknown coefficients $\{\sigma_i\}_{i \in [e]}$ of $\sigma(x)$ are found, all error positions can be obtained by searching the roots of $\sigma(x)$.

According to the definition of $\sigma(x)$ and Eq. (13), the relationship between the syndromes and the coefficients of the error locator polynomial $\sigma(x)$ can be obtained as follows [16, p.261],

$$\sum_{\ell=0}^e \sigma_\ell S_{j+e-\ell} = 0, \quad \forall j \in \{0, 2t-1-e\}, \quad (14)$$

where, we define $\sigma_0 = 1$.

Let \mathbf{S} be the $t \times (t+1)$ Hankel matrix [7] formed by all $2t$ syndromes $\{S_i\}_{i=0}^{2t-1}$, i.e.,

$$\mathbf{S} = \begin{pmatrix} S_0 & S_1 & \dots & S_{t-1} & S_t \\ S_1 & S_2 & \dots & S_t & S_{t+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ S_{t-1} & S_t & \dots & S_{2t-2} & S_{2t-1} \end{pmatrix}.$$

Next, we show that the $e \times e$ sub-matrix in the upper left corner of \mathbf{S} is invertible.

Lemma 13. The matrix $\mathbf{S}_{e \times e}$ is invertible, where $\mathbf{S}_{e \times e}$ represents the $e \times e$ sub-matrix in the upper left corner of \mathbf{S} , i.e.,

$$\mathbf{S}_{e \times e} = \begin{pmatrix} S_0 & S_1 & \dots & S_{e-1} \\ S_1 & S_2 & \dots & S_e \\ \vdots & \vdots & \ddots & \vdots \\ S_{e-1} & S_e & \dots & S_{2e-2} \end{pmatrix}.$$

Proof. Let \mathbf{W} be the $e \times e$ Vandermonde matrix as follows,

$$\mathbf{W} := \begin{pmatrix} 1 & 1 & \dots & 1 \\ \beta_1 & \beta_2 & \dots & \beta_e \\ \vdots & \vdots & \ddots & \vdots \\ \beta_1^{e-1} & \beta_2^{e-1} & \dots & \beta_e^{e-1} \end{pmatrix}.$$

Then we can verify that

$$\mathbf{W} \cdot \text{diag}(\delta_1, \delta_2, \dots, \delta_e) \cdot \mathbf{W}^T = \mathbf{S}_{e \times e},$$

where $\text{diag}(\delta_1, \delta_2, \dots, \delta_e)$ represents the diagonal matrix whose main diagonal elements are $\delta_1, \delta_2, \dots, \delta_e$ respectively.

Therefore, we have

$$\det(\mathbf{S}_{e \times e}) = \det(\mathbf{W})^2 \cdot \prod_{j=1}^e \delta_j.$$

Note that $\delta_j = e_{i_j} \neq 0$, and the Vandermonde matrix \mathbf{W} is invertible since $\beta_i \neq \beta_j$ for $i \neq j \in [e]$. We have $\det(\mathbf{S}_{e \times e}) \neq 0$, i.e., $\mathbf{S}_{e \times e}$ is invertible. \square

According to Eq. (14), we have

$$\mathbf{S}_{e \times e} \cdot \begin{bmatrix} -\sigma_e \\ -\sigma_{e-1} \\ \vdots \\ -\sigma_1 \end{bmatrix} = \begin{bmatrix} S_e \\ S_{e+1} \\ \vdots \\ S_{2e-1} \end{bmatrix}. \quad (15)$$

Recall that $\mathbf{S}_{e \times e}$ is invertible by Lemma 13, thus Eq. (15) has a unique solution, and we can see that we only need to calculate $2e$ syndromes $\{S_i\}_{i=0}^{2e-1}$ to solve the e coefficients $\{\sigma_i\}_{i=1}^e$ of the error locator polynomial $\sigma(x)$.

C. The S-ESBM Algorithm

In the following, we review the Early-Stopped Berlekamp–Massey (ESBM) algorithm [7], which can be used to efficiently solve Eq. (15) to get the error locator polynomial $\sigma(x)$.

We summarize the problem that can be solved by ESBM algorithm in Theorem 14. Please refer to [7] for more details.

Theorem 14. [7] *Given a positive integer t and a $t \times (t+1)$ Hankel matrix \mathbf{S} , where*

$$\mathbf{S} = \begin{pmatrix} S_0 & S_1 & \cdots & S_{t-1} & S_t \\ S_1 & S_2 & \cdots & S_t & S_{t+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ S_{t-1} & S_t & \cdots & S_{2t-2} & S_{2t-1} \end{pmatrix}.$$

For $i \in \{1, 2, \dots, t+1\}$, let ξ_i represent the i -th column of \mathbf{S} . If there exists a positive integer $e \leq t$, and an $e \times 1$ unknown vector $(\Lambda_e, \Lambda_{e-1}, \dots, \Lambda_1)^T$, such that:

- (1) The $e \times e$ sub-matrix in the upper left corner of \mathbf{S} is invertible;
- (2) For any $1 \leq j \leq t+1-e$,

$$\Lambda_e \xi_j + \Lambda_{e-1} \xi_{j+1} + \cdots + \Lambda_1 \xi_{e+j-1} + \xi_{e+j} = \mathbf{0}_{t \times 1}.$$

Then, if e is unknown, the ESBM algorithm needs to iterate $t+e$ steps to obtain $(\Lambda_e, \Lambda_{e-1}, \dots, \Lambda_1)^T$; if e is known, the ESBM algorithm needs to iterate $2e$ steps to obtain $(\Lambda_e, \Lambda_{e-1}, \dots, \Lambda_1)^T$.

According to Lemma 13, the matrix $\mathbf{S}_{e \times e}$ is invertible. According to Eq. (14), we can verify that for any $1 \leq j \leq t+1-e$,

$$\sigma_e \xi_j + \sigma_{e-1} \xi_{j+1} + \cdots + \sigma_1 \xi_{e+j-1} + \xi_{e+j} = \mathbf{0}_{t \times 1}.$$

Therefore, the two conditions of Theorem 14 are satisfied, and we can use the ESBM algorithm to solve Eq. (15) and obtain $(\sigma_e, \sigma_{e-1}, \dots, \sigma_1)$. Note that the number of errors e is known according to the t_0 -SI-FDMA algorithm, so the ESBM algorithm only needs $2e$ syndromes $\{S_i\}_{i=0}^{2e-1}$ and $2e$ iterations to output the e unknown coefficients of $\sigma(x)$. Recall that ESBM algorithm [7] needs $t+e$ steps to solve the error locator polynomial, where e is unknown. We propose the Shortened-ESBM (S-ESBM) algorithm which is given in Algorithm 8, that can solve the error locator polynomial with $2e$ steps, where e is known. In Algorithm 8, for any $1 \times n$ vector, the notation $\mathbf{v}|_\ell$ ($\ell \leq n$) represents the right ℓ -truncate of the vector \mathbf{v} , i.e., $\mathbf{v}|_\ell = (v_{n-\ell+1}, \dots, v_{n-1}, v_n)$. The multiplication complexity of S-ESBM is $2e^2 - 1$ [7].

Remark: Note that the operation in line 17 of I-FDMA Algorithm 5 requires three field multiplications. Therefore, compared with the t_0 -SI-FDMA algorithm, solving the error locator polynomial by I-FDMA algorithm requires at least

$$2e \cdot 3(t+1) = 6e(t+1)$$

more multiplications. When $e \ll t$, we have $2e^2 - 1 \ll 6e(t+1)$, which means that the new algorithm for solving the error locator polynomials (i.e., using the t_0 -SI-FDMA algorithm and S-ESBM algorithm) will further effectively reduce the multiplication complexity for the case of $e \ll t$, which is also

Algorithm 8 The S-ESBM Algorithm

Require: $e, \{S_i\}_{i=0}^{2e-1}$

Ensure: The coefficient vector of error locator polynomial Λ

```

1: for  $r = 1$  to  $2e$  do
2:   Calculate  $d_r$  by  $d_r = \Lambda \cdot (S_{r-L}, \dots, S_r)$ 
3:   if  $d_r = 0$  then
4:      $\tilde{\Lambda} \leftarrow (\tilde{\Lambda}, 0)$ 
5:   else if  $d_r \neq 0$  and  $r \leq 2L$  then
6:      $\Lambda \leftarrow \Lambda - d_r \cdot D^{-1} \cdot (\tilde{\Lambda}, 0)|_{L+1}$ 
7:      $\tilde{\Lambda} \leftarrow (\tilde{\Lambda}, 0)$ 
8:   else if  $d_r \neq 0$  and  $r > 2L$  then
9:      $\Lambda_{temp} \leftarrow \Lambda$ 
10:     $\Lambda \leftarrow (\mathbf{0}_{1 \times (r-2L)}, \Lambda) - d_r \cdot D^{-1} \cdot (\tilde{\Lambda}, 0)|_{r-L+1}$ 
11:     $\tilde{\Lambda} \leftarrow (\mathbf{0}_{1 \times (r-2L)}, \Lambda_{temp})$ 
12:     $D \leftarrow d_r$ 
13:     $L \leftarrow r - L$ 
14:   end if
15: end for
16: return  $\Lambda$ 

```

a major reason for the multiplication complexity reduction in our second decoding algorithm.

D. Transformation of Coefficient Vector

Note that the coefficient vector $(1, \sigma_1, \sigma_2, \dots, \sigma_e)$ of $\sigma(x)$ is based on the standard basis $\{1, x, x^2, \dots, x^{2^m-1}\}$ in $\mathbb{F}_{2^m}/(x^{2^m} - x)$. In order to employ the LCH-FFT algorithm in Section IV-A in our second decoding algorithm, we need to convert the coefficient vector into the form based on the LCH-basis \bar{X} . According to the definition of \bar{X} , there exists $(e+1) \times (e+1)$ invertible matrix \mathbf{T}_{e+1} such that

$$\begin{aligned} \sigma(x) &= (1, \sigma_1, \sigma_2, \dots, \sigma_e) \cdot \begin{bmatrix} 1 \\ x \\ \vdots \\ x^e \end{bmatrix} \\ &= (1, \sigma_1, \sigma_2, \dots, \sigma_e) \cdot \mathbf{T}_{e+1} \cdot \begin{bmatrix} \bar{X}_0(x) \\ \bar{X}_1(x) \\ \vdots \\ \bar{X}_e(x) \end{bmatrix}. \end{aligned} \quad (16)$$

Note that \mathbf{T}_{e+1} is a lower triangular matrix and can be pre-calculated, and the element in the first row and the first column of \mathbf{T}_{e+1} is 1 since $\bar{X}_0(x) = 1$, so the number of multiplications of the coefficient vector transformation of $\sigma(x)$ is less than or equal to $\frac{e^2+3e}{2}$. To distinguish, in the following, for any polynomial $f(x)$, let $\mathbf{f}_{\bar{X}}$ be the coefficient vector of $f(x)$ under the basis \bar{X} and \mathbf{f} be the coefficient vector under the standard basis.

E. Our Second Decoding Algorithm

Let $s := \lfloor \log(e) \rfloor + 1$ and $R := 2^s$, then $e < R \leq 2e < t_0 + 1$.

We present decoding process for **Step (3)** of our second decoding algorithm as follows. Once $\sigma_{\bar{X}}$ is calculated by

S-ESBM Algorithm 8 and Eq. (16), we can find the roots $\{\beta_j^{-1} = \omega_{i_j}^{-1}\}_{j=1}^e$ of $\sigma(x)$ through the Chien search and FFT Algorithm 2 as follows,

$$FFT_{\bar{X}}(\sigma_{\bar{X}}, s, \omega_{\ell \cdot 2^s}), \forall \ell = 0, 1, \dots, 2^{m-s} - 1, \quad (17)$$

and the complexity is $O(n \lg(R))$. Then we can calculate the error locator polynomial $\lambda(x) = \prod_{j=1}^e (x - \omega_{i_j})$ (here, we still call $\lambda(x)$ as the error locator polynomial as in Section IV), in which the number of multiplications is $\frac{e^2 - e}{2}$. Then we can get $\lambda_{\bar{X}}$ according to Eq. (16). Next, we can obtain $z_{\bar{X}}$ by following three steps: first, we compute $\{\lambda(\omega_i)\}_{i=0}^{R-1}$ by $FFT_{\bar{X}}(\lambda_{\bar{X}}, s, \omega_0)$; second, we compute $\{z(\omega_i)\}_{i=0}^{R-1}$ by Eq. (8); finally, we get $z_{\bar{X}}$ by $IFFT_{\bar{X}}(\mathbf{D}_R, s, \omega_0)$, where $\mathbf{D}_R := (z(\omega_0), z(\omega_1), \dots, z(\omega_{R-1}))$, and the complexity of these three steps is $O(n \lg(R))$. Finally, we can obtain the error values $\{e_{i_j}\}_{j=1}^e$ by Forney's formula by Eq. (10), and the complexity is $O(n \lg(R))$.

We summarize our second decoding algorithm in Algorithm 9. Note that we can use the Reed-Muller (RM) transform [17]–[19] to calculate the $2e$ syndromes in line 3 in Algorithm 9 to further reduce the number of multiplications. When $e = 1$, the number of multiplications is $3 \lfloor \log(n) \rfloor - 1$ [17], and when $e \geq 2$, there is no exact expression for the number of multiplications of RM transform. The asymptotic multiplications of RM transform for $e \geq 2$ is given in [19], please refer to [19] for more details.

Algorithm 9 Our Second Decoding Algorithm

Require: Received vector $\mathbf{r} = \mathbf{F} + \mathbf{e}$

Ensure: The codeword \mathbf{F}

- 1: Compute the syndrome polynomial $s(x)$ and syndromes $\{s(\omega_i)\}_{i=0}^{2^\mu-1}$
 - 2: Compute the number of errors e by t_0 -SI-FDMA Algorithm 7
 - 3: Compute $2e$ syndromes $\{S_i\}_{i=0}^{2e-1}$
 - 4: Compute the coefficient vector based on the standard basis of $\sigma(x)$ by S-ESBM Algorithm 8, and obtain $\sigma_{\bar{X}}$
 - 5: Find all error locations by Chien search
 - 6: Compute $\{\lambda(\omega_i)\}_{i=0}^{R-1}$ by $FFT_{\bar{X}}(\lambda_{\bar{X}}, s, \omega_0)$, then we compute $\{z(\omega_i)\}_{i=0}^{R-1}$ by Eq. (8), and get $z_{\bar{X}}$ by IFFT Algorithm 3
 - 7: Compute the formal derivative of $\lambda(x)$, and then get the error pattern \mathbf{e} by Forney's formula
 - 8: **return** The codeword $\mathbf{F} = \mathbf{r} + \mathbf{e}$
-

VI. COMPARATIVE ANALYSIS

A. Comparison of MA-based Algorithms

In the following, we focus on evaluating the decoding complexity for our I-FDMA algorithm and eFDMA algorithm [12]. The difference between our I-FDMA algorithm and eFDMA algorithm is that our I-FDMA algorithm only needs $2e$ steps to find the e error positions, while eFDMA algorithm requires $t + e$ steps. Therefore, the decoding complexity of our I-FDMA algorithm is strictly less than that of eFDMA algorithm when $e < t$.

First, we show the number of multiplications and the number of additions involved in finding the e error positions by FDMA algorithm [13], eFDMA algorithm [12], and I-FDMA algorithm in Table I for (n, k) RS code with error correction capability t , where $e \leq t$.

TABLE I: The number of multiplications and additions involved in finding the e error positions by FDMA algorithm [13], eFDMA algorithm [12] and I-FDMA algorithm for (n, k) RS code with error correction capability t , where $e \leq t$.

| MA-based algorithms | Multiplication complexity | Addition complexity |
|---------------------|--|--|
| FDMA [13] | $12t^2 + 3t$ | $8t^2 + 2t$ |
| eFDMA [12] | $\frac{1}{2}(7t^2 + 26et + 3(e+t) - 9r^2)$ | $\frac{1}{2}(5t^2 + 16et - 5e^2 + 3e + t)$ |
| Our I-FDMA | $18et - 6e^2 + 3e$ | $12et - 4e^2 + 2e$ |

According to Table I, when $e \ll t$, the complexity of I-FDMA algorithm is only $O(t)$ level, while the complexity of FDMA and eFDMA algorithms is both $O(t^2)$ level.

Because the multiplication in finite fields is more time-consuming than the addition, we show the number of multiplications involved in finding the e error positions by our I-FDMA algorithm and eFDMA algorithm for $(n = 256, k = 224)$ RS code in Table II. The results in Table II demonstrate that our I-FDMA algorithm can reduce the number of multiplications by 9.94%-74.67%, compared with eFDMA algorithm.

TABLE II: The number of multiplications involved in finding the error positions by I-FDMA algorithm and eFDMA algorithm for $(256, 224)$ RS code. In the table, the improvement is defined as the ratio of the multiplication reduction to the number of multiplications involved in finding the error positions by eFDMA algorithm.

| e ($t=16$) | eFDMA | I-FDMA | Improvement |
|----------------|-------|--------|-------------|
| 1 | 1125 | 285 | 74.67% |
| 2 | 1321 | 558 | 57.76% |
| 3 | 1508 | 819 | 45.69% |
| 4 | 1686 | 1068 | 36.65% |
| 5 | 1855 | 1305 | 29.65% |
| 6 | 2015 | 1530 | 24.07% |
| 7 | 2166 | 1743 | 19.53% |
| 8 | 2308 | 1944 | 15.77% |
| 9 | 2441 | 2133 | 12.62% |
| 10 | 2565 | 2310 | 9.94% |

Moreover, we show the total number of multiplications involved in our first decoding algorithm in Algorithm 6 of $(256, 224)$ RS code and eFDMA algorithm in Table III. The results show that our first decoding algorithm can reduce the number of multiplications by 5.64%-41.79%, compared with that based on eFDMA algorithm.

TABLE III: The number of multiplications involved in our first decoding algorithm in Algorithm 6 and eFDMA algorithm for (256, 224) RS code. In the table, the improvement is defined as the ratio of the multiplication reduction to the number of multiplications involved in the decoding algorithm based on eFDMA algorithm.

| e ($t=16$) | eFDMA | Our Algorithm 6 | Improvement |
|----------------|-------|-----------------|-------------|
| 1 | 2010 | 1170 | 41.79% |
| 2 | 2352 | 1589 | 32.44% |
| 3 | 2549 | 1860 | 27.03% |
| 4 | 2935 | 2317 | 21.05% |
| 5 | 3130 | 2580 | 17.57% |
| 6 | 3316 | 2831 | 18.12% |
| 7 | 3493 | 3070 | 14.62% |
| 8 | 4125 | 3761 | 12.10% |
| 9 | 4324 | 4016 | 8.82% |
| 10 | 4514 | 4259 | 5.64% |

B. Algorithm 9 VS Algorithm 6

Next, we compare the multiplication complexity for the proposed two decoding algorithms, namely our first decoding algorithm in Algorithm 6 (based on I-FDMA algorithm) and our second decoding algorithm in Algorithm 9 (based on t_0 -SI-FDMA algorithm and S-ESBM algorithm). We take $t_0 = t$ in the following comparisons.

In Table IV, we give the number of multiplications of decoding Algorithm 6 and decoding Algorithm 9 for ($n = 256, k = 224$) RS code error correction procedure, and we take $e \in \{1, 2, \dots, 8\}$. In Table V, we give the number of multiplications of decoding Algorithm 6 and decoding Algorithm 9 for ($n = 128, k = 96$) RS code error correction procedure, and we take $e \in \{1, 2, \dots, 8\}$.

TABLE IV: The number of multiplications of decoding Algorithm 6 and decoding Algorithm 9 for ($n = 256, k = 224$) RS code error correction decoding process. In the table, the improvement is defined as the ratio of the multiplication reduction to the number of multiplications by Algorithm 6.

| e ($t=16$) | Decoding Algorithm 6 | Decoding Algorithm 9 | Improvement |
|----------------|----------------------|----------------------|-------------|
| 1 | 1170 | 769 | 34.27% |
| 2 | 1589 | 1057 | 33.48% |
| 3 | 1860 | 1193 | 35.86% |
| 4 | 2317 | 1591 | 31.33% |
| 5 | 2580 | 1707 | 33.83% |
| 6 | 2831 | 1928 | 31.89% |
| 7 | 3070 | 2117 | 31.04% |
| 8 | 3761 | 2931 | 22.06% |

According to Table IV and Table V, when $e \leq t/2$, the results show that Algorithm 9 can reduce the number of multiplications by 22.06%-35.86% for parameters $n = 256, k = 224$

TABLE V: The number of multiplications of decoding Algorithm 6 and decoding Algorithm 9 for ($n = 128, k = 96$) RS code error correction decoding process. In the table, the improvement is defined as the ratio of the multiplication reduction to the number of multiplications by Algorithm 6.

| e ($t=16$) | Decoding Algorithm 6 | Decoding Algorithm 9 | Improvement |
|----------------|----------------------|----------------------|-------------|
| 1 | 786 | 449 | 42.87% |
| 2 | 1141 | 673 | 41.01% |
| 3 | 1412 | 809 | 42.70% |
| 4 | 1805 | 1143 | 36.67% |
| 5 | 2068 | 1259 | 39.11% |
| 6 | 2319 | 1480 | 36.17% |
| 7 | 2558 | 1669 | 34.75% |
| 8 | 3185 | 2419 | 24.05% |

and by 24.05%-42.87% for parameters $n = 128, k = 96$, compared with Algorithm 6.

To sum up, our second decoding algorithm in Algorithm 9 is more suitable for the case of $2e < t_0 + 1$. We summarize the essential reasons as follows: (1) I-FDMA algorithm needs to multiply matrix and vector to iterate the value of the error locator polynomial, which will cause much more multiplication operations when $e \ll t$, while the t_0 -SI-FDMA Algorithm 11 and the S-ESBM Algorithm 8 have lower multiplication complexity when $e \ll t$; (2) when $e \ll t$, the RM transform can effectively reduce the multiplication complexity for syndrome calculation (i.e., line 3 of Algorithm 9), while with the increase of e , the number of multiplications required by RM transform will also increase rapidly [19].

VII. CONCLUSION

In this paper, we propose an efficient termination mechanism of the MA method to find the error locator polynomials by only $2e$ steps in the error correction decoding process of RS codes, where e is the number of errors. Based on the new termination mechanism, we propose two types of error correction decoding algorithms for LCH-FFT-based RS codes, which have lower multiplication complexity than the existing correlation decoding algorithms.

REFERENCES

- [1] Z. Jiang, H. Shi, Z. Huang, B. Bai, G. Zhang, and H. Hou, "An Earlier Termination Algorithm to Find Error Locator Polynomial in Error Correction of RS Codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2024.
- [2] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial & Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.
- [3] E. R. Berlekamp, "Algebraic coding theory," in *McGraw-Hill series in systems science*, 1984. [Online]. Available: <https://api.semanticscholar.org/CorpusID:33309098>
- [4] D. Dabiri and I. Blake, "Fast parallel algorithms for decoding reed-solomon codes based on remainder polynomials," *IEEE Transactions on Information Theory*, vol. 41, no. 4, pp. 873–885, 1995.
- [5] C.-L. Chen, "High-speed decoding of bch codes (corresp.)," *IEEE Transactions on Information Theory*, vol. 27, no. 2, pp. 254–256, 1981.
- [6] T.-K. Truong, J.-H. Jeng, and T. Cheng, "A new decoding algorithm for correcting both erasures and errors of reed-solomon codes," *IEEE Transactions on Communications*, vol. 51, no. 3, pp. 381–388, 2003.

- [7] C.-W. Liu and C.-C. Lu, "A view of gaussian elimination applied to early-stopped berlekamp-massey algorithm," *IEEE Transactions on Communications*, vol. 55, no. 6, pp. 1131–1143, 2007.
- [8] Y. Wu, "Fast chase decoding algorithms and architectures for reed-solomon codes," *IEEE Transactions on Information Theory*, vol. 58, no. 1, pp. 109–129, 2012.
- [9] S.-J. Lin, T. Y. Al-Naffouri, and Y. S. Han, "FFT Algorithm for Binary Extension Finite Fields and Its Application to Reed-Solomon Codes," *IEEE Transactions on Information Theory*, vol. 62, no. 10, pp. 5343–5358, 2016.
- [10] N. Tang and Y. Lin, "Fast encoding and decoding algorithms for arbitrary (n, k) reed-solomon codes over \mathbb{F}_{2^m} ," *IEEE Communications Letters*, vol. 24, no. 4, pp. 716–719, 2020.
- [11] S. V. Fedorenko, "Efficient algorithm for finding roots of error-locator polynomials," *IEEE Access*, vol. 9, pp. 38 673–38 686, 2021.
- [12] C. Chen, Y. S. Han, N. Tang, S.-J. Lin, B. Bai, and X. Ma, "An early-termination method for the welch-berlekamp algorithm," in *2023 IEEE International Symposium on Information Theory (ISIT)*, 2023.
- [13] N. Tang and Y. S. Han, "A new decoding method for reed-solomon codes based on fft and modular approach," *IEEE Transactions on Communications*, vol. 70, no. 12, pp. 7790–7801, 2022.
- [14] —, "A new decoding method for reed-solomon codes based on fft and modular approach," *IEEE Transactions on Communications*, vol. 70, no. 12, p. 7790–7801, December 2022. [Online]. Available: <http://dx.doi.org/10.1109/TCOMM.2022.3215998>
- [15] R. Roth, "Introduction to coding theory," in *Cambridge University Press*, 2006. [Online]. Available: <https://doi.org/10.1017/CBO9780511808968>
- [16] T. K. Moon, "Error correction coding: Mathematical methods and algorithms," in *Wiley-Interscience*, 2005. [Online]. Available: <https://api.semanticscholar.org/CorpusID:115824719>
- [17] S.-J. Lin, "An encoding algorithm of triply extended reed-solomon codes with asymptotically optimal complexities," *IEEE Transactions on Communications*, vol. 66, no. 8, pp. 3235–3244, 2018.
- [18] L. Yu, Z. Lin, S.-J. Lin, Y. S. Han, and N. Yu, "Fast encoding algorithms for reed-solomon codes with between four and seven parity symbols," *IEEE Transactions on Computers*, vol. 69, no. 5, pp. 699–705, 2020.
- [19] L. Yu, S.-J. Lin, H. Hou, and Z. Li, "Reed-solomon coding algorithms based on reed-muller transform for any number of parities," *IEEE Transactions on Computers*, vol. 72, no. 9, pp. 2677–2688, 2023.