12-2005

New efficient MDS array codes for RAID part II: Rabin-like codes for tolerating multiple (>=4) disk failures

Gui-Liang FENG

Robert H. DENG

Singapore Management University, robertdeng@smu.edu.sg

Feng Bao

Singapore Management University, fbao@smu.edu.sg

DOI: https://doi.org/10.1109/TC.2005.200

Published in IEEE Transactions on Computers, 2005 December, Volume 54, Issue 12, Pages 1473-1483
https://doi.org/10.1109/TC.2005.200

New Efficient MDS Array Codes for RAID Part II: Rabin-Like Codes for Tolerating Multiple (≥ 4) Disk Failures

Gui-Liang Feng, *Senior Member*, *IEEE*, Robert H. Deng, Feng Bao, and Jia-Chen Shen

Abstract—A new class of Binary Maximum Distance Separable (MDS) array codes which are based on circular permutation matrices are introduced in this paper. These array codes are used for tolerating multiple (≥ 4) disk failures in Redundant Arrays of Inexpensive Disks (RAID) architecture. The size of the information part is $m \times n$, where n is the number of information disks and $(m+1)$ is a prime integer; the size of the parity-check part is $m \times r$, the minimum distance is $r+1$, and the number of parity-check disks is r. In practical applications, m can be very large and n ranges from 20 to 50. The code rate is $R = \frac{n}{n+r}$. These codes can be used for tolerating up to r disk failures, with very fast encoding and decoding. The complexities of encoding and decoding algorithms are $O(rmn)$ and $O(m^3r^4)$, respectively. When $r = 4$, there need to be $9mn$ XOR operations for encoding and $(9n+95)(m+1)$ XOR operations for decoding.

Index Terms—Rabin codes, MDS array codes, RAID, multiple disk failures.

1 Introduction

A new technique, called RAID, can be used in many applications to store huge amounts of data and it has been used by many companies, universities, and government organizations. Erasure codes are required for protecting data in RAID from multiple disk failures.

In order to retrieve the information lost on r failed (erased) disks, we need at least r redundant disks (in coding theory, this is known as the capacity of the erasure channel [2]). The well-known Reed-Solomon codes [3] can achieve this capacity. However, their encoding and decoding involve operations over finite fields and, hence, are very slow. It would be desirable to have binary linear codes that only involve exclusive-OR (XOR) operations. For $r = 2$, i.e., for tolerating two disk failures, many good codes have already been developed [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. These codes are called MDS array codes. The best results are obtained with EvenOdd codes [5], [14], X-codes [12], and B-codes [13]. But, these codes all have distance 3, meaning they can only be used for tolerating two disk failures. A generalization of EvenOdd codes has been developed [14]. Yet, the encoding and decoding for $r \geq 3$ need to be developed. In practical applications of RAID, the size of each individual symbol (i.e., m) can be as big as a whole sector: During update operations, we prefer to update a minimal number of redundant symbols when a single information symbol is updated. That means the parity-check matrix should be of the following form:

$$H = \begin{bmatrix} 1 & 0 & \dots & 0 & h_{1,1} & h_{1,2} & \dots & h_{1,n} \\ 0 & 1 & \dots & 0 & h_{2,1} & h_{2,2} & \dots & h_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & h_{r,1} & h_{r,2} & \dots & h_{r,n} \end{bmatrix}. \quad (1.1)$$

Recently, a class of Reed-Solomon-like MDS array codes for tolerating three disk failures in RAID with very fast encoding and decoding algorithms has been developed [15]. However, it cannot be used for tolerating more than four disk failures in RAID architectures. In this paper, we address this issue by developing a new class of binary MDS array codes for tolerating multiple (≥ 4) disk failures in RAID in an efficient manner. The binary MDS array codes are a class of binary linear codes, where information bits form an $m \times n$ array and parity bits form an $m \times r$ array. In applications of these new codes in RAID, m indicates the number of "data," which can be bytes or computer words and are stored on a disk, $(m+1)$ is a very large prime, and n denotes the number of information disks on which information "data" are stored. In RAID, n should be $20 \sim 50$. The code rate is $\frac{n}{n+r}$, i.e., it achieves the capacity of erasure channel [1]. Although this class of codes is high-density parity-check codes, the encoding and decoding are still very fast.

This paper is organized as follows: In the next section, we first briefly review the circular permutation matrices (CPM) and their algebra, which are very useful in the subsequent sections. The proof of the lemmas and theorems can be found in [15]. In Section 3, we introduce a class of codes based on the Cauchy matrix and CPMs, called the Rabin-like MDS array codes. Their advantage is that (1.1) is satisfied for any $r \geq 4$. Although these codes are high-density parity-check codes and the encoding cost is three times as much as that of the codes in [15], it is still very fast. In Section 4, we present a decoding algorithm for tolerating up to $r \geq 4$ disk failures. The complexity of such a decoding algorithm is $O(m^3r^4)$ and

- *G.-L. Feng and J.-C. Shen are with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA 70504. E-mail: {glf, jxs2352}@cacs.louisiana.edu.*
- *R.H. Deng and F. Bao are with the Institute for Information Research, 21 Heng Mui Keng Terrace, Singapore. E-mail: {deng, baofeng}@i2r.a-star.edu.sg.*

Manuscript received 11 Jan. 2005; revised 12 May 2005; accepted 21 June 2005; published online 14 Oct. 2005.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0005-0105.

its cost is linear with the decoding cost of the Reed-Solomon-like codes [15]. Finally, conclusions are presented in Section 5.

2 Notations and Main Lemmas

In this section, we present some new mathematical results of CPM matrices. Other results can be found in [15]. Both of them are very important in understanding the new codes and their fast encoding and decoding algorithms.

2.1 Review of CPMs and Their Algebra

We introduce a quasi-inverse matrix of $(I + E^{\mu})$, denoted by $(I + E^{\mu})^{-1}$, as follows:

$$(I + E^{\mu})^{-1}(I + E^{\mu}) = \begin{bmatrix} I_m & \overrightarrow{1}^T \\ \overrightarrow{0} & 0 \end{bmatrix} \triangleq Q \tag{2.1}$$

and

$$(I + E^{\mu})(I + E^{\mu})^{-1} = \begin{bmatrix} 0 & \overrightarrow{1} \\ \overrightarrow{0}^T & I_m \end{bmatrix} \triangleq P. \tag{2.1'}$$

Let

$$S \triangleq I + Q = \begin{bmatrix} O_m & \overrightarrow{1}^T \\ \overrightarrow{0} & 1 \end{bmatrix} \tag{2.2}$$

and

$$S^* \triangleq I + P = \begin{bmatrix} \overrightarrow{1} & 1 \\ O_m & \overrightarrow{0}^T \end{bmatrix}, \tag{2.2'}$$

where O_m denotes $m \times m$ zero matrix.

Definition 2.1. *The modified quasi-left-inverse matrix and modified quasi-right-inverse matrix of* $(I + E^{\mu})$ *for* $\mu \neq 0$, *denoted by* $\overrightarrow{(I + E^{\mu})^{-1}}$ *and* $\overleftarrow{(I + E^{\mu})^{-1}}$, *are defined by*

$$\overrightarrow{(I + E^{\mu})^{-1}} Q (I + E^{\mu}) = Q \tag{2.3}$$

and

$$(I + E^{\mu}) P \overleftarrow{(I + E^{\mu})^{-1}} = P, \tag{2.3'}$$

respectively.

These two modified quasi-inverse matrices are very important in our decoding algorithm.

We can prove the following lemma:

Lemma 2.1. *Let* $\overrightarrow{u} = (u_0, u_1, \ldots, u_m)$ *be the sum of all rows of* $(I + E^{\mu})^{-1}$, *i.e.,* $u_i = 1$ *if the weight of column* i *of* $(I + E^{\mu})^{-1}$ *is odd and, otherwise, 0. We have*

$$\overleftarrow{(I + E^{\mu})^{-1}} = (I + E^{\mu})^{-1} + U, \tag{2.4}$$

where

$$U = \begin{bmatrix} \overrightarrow{0} \\ \overrightarrow{u} \\ \vdots \\ \overrightarrow{u} \end{bmatrix}.$$

Furthermore, $\overleftarrow{(I + E^{\mu})^{-1}}$ *is a nonsingular matrix.*

From the above lemma and results in [15], we know that there is a modified quasi-left-inverse matrix such that (2.3) is true. Let us consider the matrix

$$M_t = (I + E^{\mu_1}) \left(\prod_{j=2}^{t} (I + E^{\mu_j}) \right),$$

where $\mu_j \neq 0$. Since, for each μ_j, there are nonsingular $\overrightarrow{(I + E^{\mu_j})^{-1}}$ and $(I + E^{\mu_j})^{-1}$, from Definition 2.1 in [15] and Definition 2.1, we have

$$\left(\prod_{j=2}^{t} \overrightarrow{(I + E^{\mu_j})^{-1}} \right) (I + E^{\mu_1})^{-1} M_t = Q.$$

2.2 The Rabin Codes

In the theory of error-correcting codes, Rabin codes [16] are very important. They are defined by the Cauchy matrix and are all Maximum Distance Separable (MDS) codes, otherwise known as *optimal* codes (see [17, p. 316]).

First, we briefly review the Cauchy matrix:

$$H_C = \begin{bmatrix} \frac{1}{x_1 - y_1} & \frac{1}{x_2 - y_1} & \cdots & \frac{1}{x_n - y_1} \\ \frac{1}{x_1 - y_2} & \frac{1}{x_2 - y_2} & \cdots & \frac{1}{x_n - y_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{x_1 - y_r} & \frac{1}{x_2 - y_r} & \cdots & \frac{1}{x_n - y_r} \end{bmatrix}, \tag{2.5}$$

where x_is and y_js are distinct from each other for $1 \leq i \leq n$ and $1 \leq j \leq r$. It is well-known that a submatrix consisting of any r columns of H_C is a full rank matrix. A linear code C_{Rabin} defined by H_C as a parity-check matrix is called a Rabin code [16], which is also an MDS code.

For any r, adding r columns, we have the following matrix:

$$H_{EC} = \begin{bmatrix} 1 & 0 & \ldots & 0 & \frac{1}{x_1 - y_1} & \frac{1}{x_2 - y_1} & \cdots & \frac{1}{x_n - y_1} \\ 0 & 1 & \ldots & 0 & \frac{1}{x_1 - y_2} & \frac{1}{x_2 - y_2} & \cdots & \frac{1}{x_n - y_2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 & \frac{1}{x_1 - y_r} & \frac{1}{x_2 - y_r} & \cdots & \frac{1}{x_n - y_r} \end{bmatrix}, \tag{2.6}$$

where x_i and y_j for $1 \leq i \leq n$ and $1 \leq j \leq r$ are distinct from each other. It can be easily checked that a submatrix consisting of any r columns of H_{EC} is a full rank matrix. Thus, a linear code C_{ERabin} defined by H_{EC} as a parity-check matrix is called an extended Rabin code, which is also an MDS code.

3 The Extended Rabin-Like Codes Based on CPM

Before introducing the extended Rabin-like codes based on CPM, we present some important properties of CPM.

Proposition 3.1.

$$(E^i + E^j)(E^x + E^y) = (E^x + E^y)(E^i + E^j). \tag{3.1}$$

Proposition 3.2.

$$(E^i + E^j)Q = (E^i + E^j), \quad P(E^x + E^y) = (E^x + E^y). \quad (3.2)$$

Proof. From (2.1) and (2.2), we have

$$(E^i + E^j)Q = (E^i + E^j)(I + S) = (E^i + E^j) + (E^i + E^j)S.$$

Since the first m columns of S are all columns of 0s, the first m columns of $(E^i + E^j)S$ are also columns of 0s. On the other hand, the last column is a column of 1s and each row of $(E^i + E^j)$ has exactly two 1s, so the last column of $(E^i + E^j)S$ is also a column of 0s. Therefore, $(E^i + E^j)S = O$, i.e.,

$$(E^i + E^j)Q = (E^i + E^j).$$

By the same token, we have the second result. □

Proposition 3.3.

$$P \times P = P. \quad (3.2')$$

Proof. Since $P = (I + E)(I + E)^{-1}$, we know that

$$\begin{aligned} P \times P &= (I + E)(I + E)^{-1}(I + E)(I + E)^{-1} \\ &= (I + E)Q(I + E)^{-1} = (I + E)(I + E)^{-1} = P. \end{aligned}$$

Thus, the proof is completed. □

From these properties, we have the following theorem:

Theorem 3.1. *Let an $r(m+1) \times r(m+1)$ matrix:*

$$H = \begin{bmatrix} X_1(x_1+y_1)^{-1}Y_1 & X_2(x_2+y_1)^{-1}Y_1 & X_3(x_3+y_1)^{-1}Y_1 & \dots & X_r(x_r+y_1)^{-1}Y_1 \\ X_1(x_1+y_2)^{-1}Y_2 & X_2(x_2+y_2)^{-1}Y_2 & X_3(x_3+y_2)^{-1}Y_2 & \dots & X_r(x_r+y_2)^{-1}Y_2 \\ X_1(x_1+y_3)^{-1}Y_3 & X_2(x_2+y_3)^{-1}Y_3 & X_3(x_3+y_3)^{-1}Y_3 & \dots & X_r(x_r+y_3)^{-1}Y_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X_1(x_1+y_r)^{-1}Y_r & X_2(x_2+y_r)^{-1}Y_r & X_3(x_3+y_r)^{-1}Y_r & \dots & X_r(x_r+y_r)^{-1}Y_r \end{bmatrix}, \quad (3.3)$$

where $X_1, X_2, \dots, X_r, Y_1, Y_2, \dots, Y_r$ are products of sequences of $(E^i + E^j)$, and $x_1, x_2, \dots, x_r$ and $y_1, y_2, \dots, y_r$ are terms of E^i.

Then, the matrix (3.3) has rank rm.

Proof. We use mathematical induction to prove this theorem. From Section 2, we know that the rank of A is m. Thus, the rank of $X \times (x + a)^{-1} \times A$ is m, i.e., the theorem is true for $r = 1$.

Assume that the theorem is true for $(r-1)$. Let us consider the following $r(m+1) \times r(m+1)$ matrix:

$$H_1 = \begin{bmatrix} X_1(x_1+y_1)^{-1}Y_1 & X_2(x_2+y_1)^{-1}Y_1 & X_3(x_3+y_1)^{-1}Y_1 & \dots & X_r(x_r+y_1)^{-1}Y_1 \\ X_1(x_1+y_2)^{-1}Y_2 & X_2(x_2+y_2)^{-1}Y_2 & X_3(x_3+y_2)^{-1}Y_2 & \dots & X_r(x_r+y_2)^{-1}Y_2 \\ X_1(x_1+y_3)^{-1}Y_3 & X_2(x_2+y_3)^{-1}Y_3 & X_3(x_3+y_3)^{-1}Y_3 & \dots & X_r(x_r+y_3)^{-1}Y_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X_1(x_1+y_r)^{-1}Y_r & X_2(x_2+y_r)^{-1}Y_r & X_3(x_3+y_r)^{-1}Y_r & \dots & X_r(x_r+y_r)^{-1}Y_r \end{bmatrix}. \quad (3.4)$$

Let

$$H_2 = \begin{bmatrix} (x_1+y_1) & O & O & \dots & O \\ Y_2(x_1+y_1) & Y_1(x_1+y_2) & O & \dots & O \\ Y_3(x_1+y_1) & O & Y_1(x_1+y_3) & \dots & O \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ Y_r(x_1+y_1) & O & O & \dots & Y_1(x_1+y_r) \end{bmatrix} \times H_1 \times \begin{bmatrix} I & O & O & \dots & O \\ O & (x_2+y_1) & O & \dots & O \\ O & O & (x_3+y_1) & \dots & O \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ O & O & O & \dots & (x_r+y_1) \end{bmatrix},$$

we have

$$H_2 = \begin{bmatrix} X_1Y_1 & X_2(x_1+y_1)Y_1 & X_3(x_1+y_1)Y_1 & \dots & X_r(x_1+y_1)Y_1 \\ O & Y_1X_2(x_1+x_2)(x_2+y_2)^{-1}(y_1+y_2)Y_2 & Y_1X_3(x_1+x_3)(x_3+y_2)^{-1}(y_1+y_2)Y_2 & \dots & Y_1X_r(x_1+x_r)(x_r+y_2)^{-1}(y_1+y_2)Y_2 \\ O & Y_1X_2(x_1+x_2)(x_2+y_3)^{-1}(y_1+y_3)Y_3 & Y_1X_3(x_1+x_3)(x_3+y_3)^{-1}(y_1+y_3)Y_3 & \dots & Y_1X_r(x_1+x_r)(x_r+y_3)^{-1}(y_1+y_3)Y_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ O & Y_1X_2(x_1+x_2)(x_2+y_r)^{-1}(y_1+y_r)Y_r & Y_1X_3(x_1+x_3)(x_3+y_r)^{-1}(y_1+y_r)Y_r & \dots & Y_1X_r(x_1+x_r)(x_r+y_r)^{-1}(y_1+y_r)Y_r \end{bmatrix}, \quad (3.5)$$

where we use (3.2), (3.3), and the following formula:

$$\begin{aligned} (x_1 + y_1)(x_2 + y_1)^{-1} &= (x_1 + x_2 + x_2 + y_1)(x_2 + y_1)^{-1} \\ &= (x_1 + x_2)(x_2 + y_1)^{-1} + P. \end{aligned}$$

Now, let us consider the submatrix consisting of the last $(r-1)$ rows and the last $(r-1)$ columns:

$$H^* = \begin{bmatrix} Y_1X_2(x_1+x_2)(x_2+y_2)^{-1}(y_1+y_2)Y_2 & \dots & Y_1X_r(x_1+x_r)(x_r+y_2)^{-1}(y_1+y_2)Y_2 \\ Y_1X_2(x_1+x_2)(x_2+y_3)^{-1}(y_1+y_3)Y_3 & \dots & Y_1X_r(x_1+x_r)(x_r+y_3)^{-1}(y_1+y_3)Y_3 \\ \vdots & \ddots & \vdots \\ Y_1X_2(x_1+x_2)(x_2+y_r)^{-1}(y_1+y_r)Y_r & \dots & Y_1X_r(x_1+x_r)(x_r+y_r)^{-1}(y_1+y_r)Y_r \end{bmatrix}.$$

Let

$$\begin{cases} X_2^* &= Y_1X_2(x_1 + x_2) \\ X_3^* &= Y_1X_3(x_1 + x_3) \\ X_r^* &= Y_1X_r(x_1 + x_r) \\ Y_2^* &= (y_1 + y_2)Y_2 \\ Y_3^* &= (y_1 + y_3)Y_3 \\ Y_r^* &= (y_1 + y_r)Y_r. \end{cases} \quad (3.6)$$

We have

$$H^* = \begin{bmatrix} X_2^*(x_2+y_2)^{-1}Y_2^* & X_3^*(x_3+y_2)^{-1}Y_2^* & \dots & X_r^*(x_r+y_2)^{-1}Y_2^* \\ X_2^*(x_2+y_3)^{-1}Y_3^* & X_3^*(x_3+y_3)^{-1}Y_3^* & \dots & X_r^*(x_r+y_3)^{-1}Y_3^* \\ \vdots & \vdots & \ddots & \vdots \\ X_2^*(x_2+y_r)^{-1}Y_r^* & X_3^*(x_3+y_r)^{-1}Y_r^* & \dots & X_r^*(x_r+y_r)^{-1}Y_r^* \end{bmatrix},$$

where $X_2^*, X_3^*, \dots, X_r^*$ and $Y_2^*, Y_3^*, \dots, Y_r^*$ are also products of sequence of $(E^i + E^j)$.

By mathematical induction, the rank of H^* is $(r-1)m$. Thus, the rank of H_1 is at least $(r-1)m + m = rm$.

On the other hand, the rank of each block row of H_1 is at most m because the ranks of $X_1, X_2, \dots, X_r$ and $Y_1, Y_2, \dots, Y_r$ are all at most m. Therefore, the rank of H_1 is at most rm.

Hence, the proof is completed. □

We are now going to introduce the extended Rabin-like codes based on CPM. Let us consider the following matrix: $H =$

$$\begin{bmatrix} I & 0 & \dots & 0 & (I+E^r)^{-1} & (I+E^{r+1})^{-1} & \dots & (I+E^{r+n-1})^{-1} \\ 0 & I & \dots & 0 & (E+E^r)^{-1} & (E+E^{r+1})^{-1} & \dots & (E+E^{r+n-1})^{-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & I & (E^{r-1}+E^r)^{-1} & (E^{r-1}+E^{r+1})^{-1} & \dots & (E^{r-1}+E^{r+n-1})^{-1} \\ \vec{0} & \vec{0} & \dots & \vec{0} & \vec{1} & \vec{0} & \dots & \vec{0} \\ \vec{0} & \vec{0} & \dots & \vec{0} & \vec{0} & \vec{1} & \dots & \vec{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \vec{0} & \vec{0} & \dots & \vec{0} & \vec{0} & \vec{0} & \dots & \vec{1} \end{bmatrix}. \tag{3.7}$$

From Theorem 3.1, it can be easily seen that the rank of H is $r(m+1)$. Thus, we can use H as a parity-check matrix to define a binary linear code:

$$C = \{\vec{c} = (\vec{c_0}, \vec{c_1}, ..., \vec{c_{n+r-1}})\ H^* \vec{c^*}^T = \vec{0}^T\}, \tag{3.8}$$

where

$$\vec{c_i} = (c_{i,0}, c_{i,1}, c_{i,2}, ..., c_{i,m})\ for\ 0 \le i \le n+r-1$$

and

$$c_{i,m} = 0\ for\ 0 \le i \le r-1, \tag{3.9}$$

$$c_{j,0} = \sum_{\nu=1}^{m} c_{j,\nu}\ for\ r \le j \le n+r-1. \tag{3.9'}$$

In the codeword, $c_{i,\mu}$, for $0 \le i \le r-1$ and $\mu < m$, and $c_{j,0}$, for $r \le j \le n+r-1$, are parity-check bits, while $c_{j,\mu}$, for $r \le j \le n+r-1$ and $\mu \ne 0$, are information bits.

Remark. It can be easily seen that the code length is $(r+n)(m+1)$ and code dimension is nm. However, $c_{i,m} = 0$ and $c_{j,0} = \sum_{\mu=1}^{m} c_{i,\mu}$, for $0 \le i \le r-1$ and $r \le j \le n+r-1$. Hence, the data on these bits do not need to be stored on disks and only data on the other $(n+r)m$ bits need to be stored on the disks. The nm bits of these data are information and the rm bits are redundant. Therefore, the code rate is $\frac{n}{n+r}$. Since the submatrix consisting of any $t \le r$ block columns has rank $t(m+1)$, any erasure error in t block columns can be corrected.

For the same reason as in (3.7), this code C can be defined on any Abelian group $(G, \oplus)$, i.e., $c_{i,\nu} \in G$.

Example 3.1. Let us consider the example: $r = 4$ and $n = 4$, the parity-check matrix H is

$$\begin{bmatrix} I & O & O & O & (I+E^i)^{-1} & (I+E^j)^{-1} & (I+E^k)^{-1} & (I+E^s)^{-1} & (I+E^t)^{-1} \\ O & I & O & O & (E+E^i)^{-1} & (E+E^j)^{-1} & (E+E^k)^{-1} & (E+E^s)^{-1} & (E+E^t)^{-1} \\ O & O & I & O & (E^2+E^i)^{-1} & (E^2+E^j)^{-1} & (E^2+E^k)^{-1} & (E^2+E^s)^{-1} & (E^2+E^t)^{-1} \\ O & O & O & I & (E^3+E^i)^{-1} & (E^3+E^j)^{-1} & (E^3+E^k)^{-1} & (E^3+E^s)^{-1} & (E^3+E^t)^{-1} \\ \vec{0} & \vec{0} & \vec{0} & \vec{0} & \vec{1} & \vec{0} & \vec{0} & \vec{0} & \vec{0} \\ \vec{0} & \vec{0} & \vec{0} & \vec{0} & \vec{0} & \vec{1} & \vec{0} & \vec{0} & \vec{0} \\ \vec{0} & \vec{0} & \vec{0} & \vec{0} & \vec{0} & \vec{0} & \vec{1} & \vec{0} & \vec{0} \\ \vec{0} & \vec{0} & \vec{0} & \vec{0} & \vec{0} & \vec{0} & \vec{0} & \vec{1} & \vec{0} \\ \vec{0} & \vec{0} & \vec{0} & \vec{0} & \vec{0} & \vec{0} & \vec{0} & \vec{0} & \vec{1} \end{bmatrix}.$$

From Theorem 3.1, we know that the rank of H is $4(m+1)$. Thus, we can use H as a parity-check matrix to define a binary linear code

$$C = \left\{ \overrightarrow{c} = (\overrightarrow{c}_0, \overrightarrow{c}_1, \dots, \overrightarrow{c}_{n+3})\ H^* \overrightarrow{c}^T = \overrightarrow{0}^T \right\}, \tag{3.10}$$

where

$$\overrightarrow{c}_i = (c_{i,0}, c_{i,1}, c_{i,2}, \dots, c_{i,m})\ for\ 0 \le i \le n+3$$

and

$$c_{i,m} = 0\ for\ 0 \le i \le 3, \tag{3.11}$$

$$c_{j,0} = \sum_{\nu=1}^{m} c_{j,\nu}\ for\ 4 \le j \le n+3, \tag{3.12}$$

where $c_{i,\mu}$, for $0 \le i \le 3$ and $0 \le \mu \le m$, and $c_{j,0}$, for $4 \le j \le n+3$, are parity-check bits and $c_{j,\mu}$, for $4 \le j \le n+3$ and $\mu \ne 0$, are information bits. It should be noted that both $c_{i,\mu}$, for $0 \le i \le 3$, and $c_{j,0}$, for $4 \le j \le n+3$, are present for formal convenience. Specifically, $c_{i,m} = 0$, for $0 \le i \le 3$, are constants and are therefore not used and $c_{j,0}$, for $4 \le j \le n+3$, are not used either because they are virtual parity-check bits, for the information on these bits is calculated with (3.12) from other information bits in the decoding process. Thus, with only the transmitted information bits and parity-check bits considered, i.e., $c_{i,\mu}$, for $0 \le i \le 3$ and $0 \le \mu \le m-1$, and $c_{j,\nu}$, for $4 \le j \le n+3$ and $1 \le \nu \le m$, the code rate is $\frac{nm}{(n+4)m} = \frac{n}{n+4}$.

From (2.1') and (3.12), we have

$$P \times \overrightarrow{c}_j = \overrightarrow{c}_j \quad for \quad 4 \le j \le n+3. \tag{3.13}$$

From Theorem 3.1, it is clear that any four or less block-columns are linearly independent with coefficients of E^μ-type. That means up to four erasure errors can be corrected.

This code C can also be defined on any Abelian group $(G, \oplus)$, i.e., $c_{i,\nu} \in G$. In practical applications, G can be a group of computer words, i.e., binary vectors of 32 bits. Therefore, the 32 codewords of C can be simultaneously encoded/decoded, which will lead to a 32-fold improvement in efficiency.

Now, we discuss the encoding process. In this code C, information bits are $c_{j,\nu}$ for $r \le j \le n+r-1$ and $1 \le \nu \le m$, from which $c_{j,0} = \sum_{\nu=1}^{m} c_{j,\nu}$ for $r \le j \le n+r-1$ are calculated. Then, $c_{i,\mu}$, for $0 \le i \le r-1$ and $0 \le \mu \le m$, are calculated with (3.8). Finally, we need to store only $c_{i,\mu}$, for $0 \le i \le r-1$ and $0 \le \mu \le m-1$, and $c_{j,\nu}$, for $r \le j \le n+r-1$ and $1 \le \nu \le m$, on the n information disks and the four parity-check disks, respectively.

1. To calculate $c_{i,0} = \sum_{\mu=1}^{m} c_{i,\mu}$ for $r \le i \le n+r-1$: For each $c_{i,0}$, there need to be $(m-1)$ XOR operations. Thus, a total of $n(m-1)$ XOR operations are needed.
2. To calculate $c_{i,\mu}$ for $0 \le i \le r-1$ and $0 \le \mu \le m-1$: For each j, $r \le j \le n+r-1$, we first calculate the sums $s_{j,i,\mu}$ of rows (i,μ) and block-columns j. For each block-section (i,j), from Section 2, there need to be m XOR operations to calculate $s_{j,i,\mu}$. Thus, a total of $4mn$ XOR operations are needed for all $s_{j,i,\mu}$s. To calculate a single $c_{i,\mu} = \sum_{j=r}^{n+r-1} s_{j,i,\mu}$ from $s_{j,i,\mu}$s, there need to be $(n-1)$ XOR operations. Thus, a total of $rm(n-1)$ XOR operations are needed for all $c_{i,\mu}$s from $s_{j,i,\mu}$s. Therefore, to calculate all $c_{i,\mu}$s from $c_{j,\nu}$s, there need to be $2rmn$ XOR operations.

Consequently, in the encoding process, at most $(2r+1)mn$ XOR operations are needed. Thus, the encoding cost of code C is roughly three times as much as that of the codes in [15]. However, it is still very fast.

To calculate $c_{i,0}$, we need m XOR operations for each $r \le i \le n+r-1$. Thus, we need nm XOR operations. Then, from all $c_{i,j}$, for $r \le i \le n+r-1$ and $0 \le j \le m$, we need to calculate $c_{h,j}$, for $0 \le h \le r-1$. For this step, we need to calculate $(n \times r)$ multiplications of $(E^i + E^j)^{-1}$ and $\vec{c_\mu}$. From Section 2, each such multiplication needs m XOR operations. Thus, we need a total of nrm XOR operations.

4 Decoding of the Extended Rabin-Like Codes Based on CPM

Assume that a codeword $\mathbf{c} = (\vec{c}_0, \vec{c}_1, \ldots, \vec{c}_{n+r-1})$ is transmitted and that t packets, say $\vec{c}_{\mu_i}$ for $i = 1, 2, \ldots, t \le r$, are lost. Then, the received codeword is given by

$$\mathbf{y} = (\vec{y}_0, \vec{y}_1, \ldots, \vec{y}_{n+r-1}),$$

where

$$\vec{y}_i = \begin{cases} \vec{c}_i & i \notin \{\mu_1, \mu_2, ..., \mu_t\} \\ \vec{0} & i \in \{\mu_1, \mu_2, ..., \mu_t\}. \end{cases}$$

We define the syndromes of the received codeword $\mathbf{y}$ as

$$\widetilde{H}\mathbf{y}^T = \mathbf{s}^T, \tag{4.1}$$

where $\mathbf{s} = (\vec{s}_0, \vec{s}_1, \ldots, \vec{s}_{r-1})$, $\vec{s}_i = (s_{i,0}, s_{i,1}, \ldots, s_{i,m})$, and $s_{i,j} \in G$.

Let

$$\mathbf{z} = (\vec{z}_0, \vec{z}_1, \ldots, \vec{z}_{n+r-1})$$

and

$$\vec{z}_i = \begin{cases} \vec{0} & i \notin \{\mu_1, \mu_2, \ldots, \mu_t\} \\ \vec{c}_i & i \in \{\mu_1, \mu_2, \ldots, \mu_t\}. \end{cases}$$

It follows that

$$\mathbf{c} = \mathbf{y} + \mathbf{z}.$$

Since $\widetilde{H}\mathbf{c}^T = \vec{0}^T$ and (4.1), we have

$$\widetilde{H}\mathbf{z}^T = \mathbf{s}^T. \tag{4.2}$$

Then, correcting these erasure errors is equivalent to solving the following set of linear equations with a Cauchy matrix:

$$\begin{bmatrix} (E^{\lambda_0}+E^{\mu_s})^{-1} & (E^{\lambda_0}+E^{\mu_{s+1}})^{-1} & \ldots & (E^{\lambda_0}+E^{\mu_t})^{-1} \\ (E^{\lambda_1}+E^{\mu_s})^{-1} & (E^{\lambda_1}+E^{\mu_{s+1}})^{-1} & \ldots & (E^{\lambda_1}+E^{\mu_t})^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ (E^{\lambda_{t-s}}+E^{\mu_s})^{-1} & (E^{\lambda_{t-s}}+E^{\mu_{s+1}})^{-1} & \ldots & (E^{\lambda_{t-s}}+E^{\mu_t})^{-1} \end{bmatrix} \times \begin{bmatrix} \vec{c_{\mu_1}} \\ \vec{c_{\mu_2}} \\ \vdots \\ \vec{c_{\mu_t}} \end{bmatrix} = \begin{bmatrix} \vec{s_0^{(0)}} \\ \vec{s_1^{(0)}} \\ \vdots \\ \vec{s_{t-1}^{(0)}} \end{bmatrix}, \tag{4.3}$$

where we assume that

$$0 \le \mu_1 < \ldots < \mu_{s-1} \le r-1 < \mu_s < \ldots < \mu_t,$$

i.e., the first $(s-1)$ errors are in the parity-check bits and the last $(t+s+1)$ errors are in the information bits,

$$\{\lambda_0, \lambda_1, \ldots, \lambda_{t-s}\} \subseteq \{0, 1, \ldots, r-1\} \setminus \{\mu_1, \mu_2, \ldots, \mu_{s-1}\}. \tag{4.4}$$

Remark. When $t < r$, there are many such sets of λs and we can choose any one.

The decoding process can be briefly summarized as follows: Given a received codeword y and the locations of the lost packets $\mu_1, \mu_2, \cdots, \mu_r$, we first compute the syndromes from (4.1) and then determine the values of the lost packets $\vec{c}_{\mu_i}$ for $1 \le i \le r$ by solving (4.4).

We are now going to derive a recursive algorithm for solving (4.4). The algorithm consists of two parts: the forward steps and the backward steps.

First, we explain the forward steps.

Let us consider the equation

$$L_i = \left[\begin{array}{cccc|ccccc} I & O & \cdots & O & & & & & \\ O & I & \cdots & O & & & O & & \\ \vdots & \vdots & \ddots & \vdots & & & & & \\ O & O & \cdots & I & & & & & \\ \hline & & & & (x_i+a_i) & O & \cdots & O \\ & O & & & A_{i+1}^{(i-1)}(x_i+a_i) & A_i^{(i-1)}(x_i+a_{i+1}) & \cdots & O \\ & & & & \vdots & \vdots & \ddots & \vdots \\ & & & & A_t^{(i-1)}(x_i+a_i) & O & \cdots & A_i^{(i-1)}(x_i+a_t) \end{array}\right],$$

Fig. 1.

$$\begin{bmatrix} X_1^{(0)}(x_1+a_1)^{-1}A_1^{(0)} & X_2^{(0)}(x_2+a_1)^{-1}A_1^{(0)} & X_3^{(0)}(x_3+a_1)^{-1}A_1^{(0)} \\ X_1^{(0)}(x_1+a_2)^{-1}A_2^{(0)} & X_2^{(0)}(x_2+a_2)^{-1}A_2^{(0)} & X_3^{(0)}(x_3+a_2)^{-1}A_2^{(0)} \\ X_1^{(0)}(x_1+a_3)^{-1}A_3^{(0)} & X_2^{(0)}(x_2+a_3)^{-1}A_3^{(0)} & X_3^{(0)}(x_3+a_3)^{-1}A_3^{(0)} \\ \vdots & \vdots & \vdots \\ X_1^{(0)}(x_1+a_t)^{-1}A_t^{(0)} & X_2^{(0)}(x_2+a_t)^{-1}A_t^{(0)} & X_3^{(0)}(x_3+a_t)^{-1}A_t^{(0)} \end{bmatrix}$$
$$\begin{bmatrix} \cdots & X_t^{(0)}(x_t+a_1)^{-1}A_1^{(0)} \\ \cdots & X_t^{(0)}(x_t+a_2)^{-1}A_2^{(0)} \\ \cdots & X_t^{(0)}(x_t+a_3)^{-1}A_3^{(0)} \\ \ddots & \vdots \\ \cdots & X_t^{(0)}(x_t+a_t)^{-1}A_t^{(0)} \end{bmatrix} \times \begin{bmatrix} \alpha_1^{(0)} \\ \alpha_2^{(0)} \\ \alpha_3^{(0)} \\ \vdots \\ \alpha_t^{(0)} \end{bmatrix} = \begin{bmatrix} \vec{s_1^{(0)}} \\ \vec{s_2^{(0)}} \\ \vec{s_3^{(0)}} \\ \vdots \\ \vec{s_t^{(0)}} \end{bmatrix}, \tag{4.5}$$

$$H^{(0)} \times \vec{\Psi^{(0)}} = \vec{S^{(0)}}. \tag{4.5'}$$

Let L_1, R_1, and R_1^* be the following matrices, respectively:

$$L_1 \triangleq \begin{bmatrix} (x_1+a_1) & O & O & \cdots & O \\ A_2^{(0)}(x_1+a_1) & A_1^{(0)}(x_1+a_2) & O & \cdots & O \\ A_3^{(0)}(x_1+a_1) & O & A_1^{(0)}(x_1+a_3) & \cdots & O \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_t^{(0)}(x_1+a_1) & O & O & \cdots & A_1^{(0)}(x_1+a_t) \end{bmatrix},$$

$$R_1 \triangleq \begin{bmatrix} I & O & O & \cdots & O \\ O & (x_2+a_1) & O & \cdots & O \\ O & O & (x_3+a_1) & \cdots & O \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ O & O & O & \cdots & (x_t+a_1) \end{bmatrix},$$

$$R_1^* \triangleq \begin{bmatrix} I & O & O & \cdots & O \\ O & (x_2+a_1)^{-1} & O & \cdots & O \\ O & O & \overleftarrow{P(x_3+a_1)^{-1}} & \cdots & O \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ O & O & O & \cdots & \overleftarrow{P(x_t+a_1)^{-1}} \end{bmatrix}. \tag{4.6}$$

From (4.5'), we have

$$L_1 \times H^{(0)} \times R_1 \times R_1^* \times \vec{\Psi} = L_1 \times \vec{S^{(0)}}, \tag{4.7}$$

and it is denoted by

$$H^{(1)} \times \vec{\Psi^{(1)}} = \vec{S^{(1)}}, \tag{4.7'}$$

where

$$H^{(1)} \triangleq \begin{bmatrix} X_1^{(0)}A_1^{(0)} & X_2^{(0)}(x_1+a_1)A_1^{(0)} & X_3^{(0)}(x_1+a_1)A_1^{(0)} & \cdots & X_t^{(0)}(x_1+a_1)A_1^{(0)} \\ O & X_2^{(1)}(x_2+a_2)^{-1}A_2^{(1)} & X_3^{(1)}(x_3+a_2)^{-1}A_2^{(1)} & \cdots & X_t^{(1)}(x_t+a_2)^{-1}A_2^{(1)} \\ O & X_2^{(1)}(x_2+a_3)^{-1}A_3^{(1)} & X_3^{(1)}(x_3+a_3)^{-1}A_3^{(1)} & \cdots & X_t^{(1)}(x_t+a_3)^{-1}A_3^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ O & X_2^{(1)}(x_2+a_t)^{-1}A_t^{(1)} & X_3^{(1)}(x_3+a_t)^{-1}A_t^{(1)} & \cdots & X_t^{(1)}(x_t+a_t)^{-1}A_t^{(1)} \end{bmatrix}, \tag{4.8}$$

$$\begin{cases} A_i^{(1)} \triangleq (a_1+a_i)A_i^{(0)} & for\ 2 \le i \le t \\ X_j^{(1)} \triangleq A_1^{(0)}X_j^{(0)}(x_1+x_j) & for\ 2 \le j \le t, \end{cases}$$

$$\vec{\Psi^{(1)}} \triangleq R_1^* \times \vec{\Psi^{(0)}},$$

and

$$\vec{S^{(1)}} \triangleq L_1 \times \vec{S^{(0)}}.$$

Likewise, we give the recursive algorithm: for any $1 \le i \le t$, as shown in Fig. 1, where the first $(i-1)$ block-rows and first $(i-1)$ block-columns of L_i form an $(i-1)(m+1) \times (i-1)(m+1)$ identity matrix and, as shown in Fig. 2, where the first i block-rows and the first i block-columns of both R_i and R_i^* form an $i(m+1) \times i(m+1)$ identity matrix.

And, let

$$\begin{cases} H^{(i)} = L_i \times H^{(i-1)} \times R_i \\ \vec{\Psi^{(i)}} = R_i^* \times \vec{\Psi^{(i-1)}} \\ \vec{S^{(i)}} = L_i \times \vec{S^{(i-1)}} \end{cases} \tag{4.8'}$$

and

$$R_i = \left[\begin{array}{cccc|cccc} I & O & \cdots & O & & & & \\ O & I & \cdots & O & & & & \\ \vdots & \vdots & \ddots & \vdots & & O & & \\ O & O & \cdots & I & & & & \\ \hline & & & & (x_{i+1}+a_i) & O & \cdots & O \\ & O & & & O & (x_{i+2}+a_i) & \cdots & O \\ & & & & \vdots & \vdots & \ddots & \vdots \\ & & & & O & O & \cdots & (x_t+a_i) \end{array}\right],$$

$$R_i^* = \left[\begin{array}{cccc|cccc} I & O & \cdots & O & & & & \\ O & I & \cdots & O & & & & \\ \vdots & \vdots & \ddots & \vdots & & O & & \\ O & O & \cdots & I & & & & \\ \hline & & & & (x_{i+1}+a_i)^{-1} & O & \cdots & O \\ & O & & & O & \overleftarrow{P(x_{i+2}+a_i)^{-1}} & \cdots & O \\ & & & & \vdots & \vdots & \ddots & \vdots \\ & & & & O & O & \cdots & \overleftarrow{P(x_t+a_i)^{-1}} \end{array}\right], \quad (4.6')$$

Fig. 2.

$$X_j^{(i)} = \begin{cases} Y_i^{(i-1)} X_j^{(i-1)}(x_i + x_j) & if \quad j > i \\ X_j^{(i-1)} & if \quad j < i, \end{cases}$$

$$Y_j^{(i)} = \begin{cases} (y_i + y_j) Y_j^{(i-1)} & if \quad j > i \\ Y_j^{(i-1)} & if \quad j < i. \end{cases}$$

We get

$$H^{(i)} = \begin{bmatrix} H_{1,1}^{(i)} & H_{1,2}^{(i)} \\ O & H_{2,2}^{(i)} \end{bmatrix} \quad (4.9)$$

and

$$H^{(i)} \cdot \vec{\Psi^{(i)}} = \vec{S^{(i)}},$$

where

$$H_{1,1}^{(i)} = \begin{bmatrix} X_1^{(0)} A_1^{(0)} & X_2^{(0)}(x_1+a_1) A_1^{(0)} & \cdots & X_i^{(0)}(x_1+a_1) A_1^{(0)} \cdot \prod_{m=2}^{i-1}(x_i+a_m) \\ O & X_2^{(1)} A_2^{(1)} & \cdots & X_i^{(1)}(x_2+a_2) A_2^{(1)} \cdot \prod_{m=3}^{i-1}(x_i+a_m) \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \cdots & X_i^{(i-1)} A_i^{(i-1)} \end{bmatrix},$$

$$H_{1,2}^{(i)} = \begin{bmatrix} X_{i+1}^{(0)}(x_1+a_1) A_1^{(0)} \cdot \prod_{m=2}^{i-1}(x_{i+1}+a_m) \\ X_{i+1}^{(1)}(x_2+a_2) A_2^{(1)} \cdot \prod_{m=3}^{i-1}(x_{i+1}+a_m) \\ \vdots \\ X_{i+1}^{(i-1)}(x_i+a_i) A_i^{(i-1)} \end{bmatrix}$$

$$\begin{bmatrix} X_{i+2}^{(0)}(x_1+a_1) A_1^{(0)} \cdot \prod_{m=2}^{i-1}(x_{i+1}+a_m) & \cdots & X_t^{(0)}(x_1+a_1) A_1^{(0)} \cdot \prod_{m=2}^{i-1}(x_{i+1}+a_m) \\ X_{i+2}^{(1)}(x_2+a_2) A_2^{(1)} \cdot \prod_{m=3}^{i-1}(x_{i+1}+a_m) & \cdots & X_t^{(1)}(x_2+a_2) A_2^{(1)} \cdot \prod_{m=3}^{i-1}(x_{i+1}+a_m) \\ \vdots & \ddots & \vdots \\ X_{i+2}^{(i-1)}(x_i+a_i) A_i^{(i-1)} & \cdots & X_t^{(i-1)}(x_i+a_i) A_i^{(i-1)} \end{bmatrix},$$

$$H_{2,2}^{(i)} = \begin{bmatrix} X_{i+1}^{(i)}(x_{i+1}+a_{i+1})^{-1} A_{i+1}^{(i)} & X_{i+2}^{(i)}(x_{i+2}+a_{i+1})^{-1} A_{i+1}^{(i)} & \cdots & X_t^{(i)}(x_t+a_{i+1})^{-1} A_{i+1}^{(i)} \\ X_{i+1}^{(i)}(x_{i+1}+a_{i+2})^{-1} A_{i+2}^{(i)} & X_{i+2}^{(i)}(x_{i+2}+a_{i+2})^{-1} A_{i+2}^{(i)} & \cdots & X_t^{(i)}(x_t+a_{i+2})^{-1} A_{i+2}^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ X_{i+1}^{(i)}(x_{i+1}+a_t)^{-1} A_t^{(i)} & X_{i+2}^{(i)}(x_{i+2}+a_t)^{-1} A_t^{(i)} & \cdots & X_t^{(i)}(x_t+a_t)^{-1} A_t^{(i)} \end{bmatrix}.$$

Thus, we get

$$H^{(t)} \cdot \vec{\Psi^{(t)}} = \vec{S^{(t)}},$$

where

$$\begin{cases} H^{(t)} = L_t \times L_{t-1} \times \cdots \times L_1 \times H^{(0)} \times R_1 \times \cdots \times R_{t-1} \times R_t \\ \vec{\Psi^{(t)}} = R_t^* \times R_{t-1}^* \times \cdots \times R_1^* = R_{t-1}^* \times R_{t-2}^* \times \cdots \times R_1^* \times \vec{\Psi^{(0)}} \\ \vec{S^{(t)}} = L_t \times L_{t-1} \times \cdots \times L_1 \times \vec{S^{(0)}}. \end{cases}$$

Before calculating the complexity of the decoding, we first introduce some new notations.

Let $h_i^{(\mu)}$ be the number of factors $(E^\alpha + E^\beta)$s in $A_i^{(\mu)}$ as well as $k_j^{(\nu)}$ for $X_j^{(\nu)}$. Thus, from (4.8′), we have

$$\begin{cases} k_j^{(\nu+1)} &= h_1^{(\nu)} + k_j^{(\nu)} + 1 \\ h_j^{(\nu+1)} &= h_j^{(\nu)} + 1. \end{cases} \tag{4.10}$$

We need to calculate $\vec{S^{(1)}} = L_1 \times \vec{S^{(0)}}$, i.e.,

$$\begin{cases} s_1^{(1)} \triangleq & (x_1+a_1)s_1^{(0)}; & p\ XORs \\ s_2^{(1)} \triangleq & A_2^{(0)}s_1^{(1)} + A_1^{(0)}(x_1+a_1)s_2^{(0)}; & (h_2^{(0)}+h_1^{(0)}+1+1)p\ XORs \\ s_3^{(1)} \triangleq & A_3^{(0)}s_1^{(1)} + A_1^{(0)}(x_1+a_1)s_3^{(0)}; & (h_3^{(0)}+h_1^{(0)}+1+1)p\ XORs \\ \vdots & \vdots & \vdots \\ s_t^{(1)} \triangleq & A_t^{(0)}s_1^{(1)} + A_1^{(0)}(x_1+a_1)s_r^{(0)}; & (h_t^{(0)}+h_1^{(0)}+1+1)p\ XORs. \end{cases} \tag{4.11}$$

For example, in calculating $A_2^{(0)}s_1^{(1)} + A_1^{(0)}(x_1+a_1)s_2^{(0)}$, we need to multiply $s_1^{(0)}$ and $s_2^{(0)}$ by A_2 and $A_1(x_1+a_1)$, respectively, which needs $(h_2^{(0)} + h_1^{(0)} + 1)p$ XORs. Next, adding them needs p XORs. Thus, a total of $(h_2^{(0)} + h_1^{(0)} + 1 + 1)p$ XORs are needed for calculating $A_2^{(0)}s_1^{(0)} + A_1^{(0)}(x_1+a_1)s_2^{(0)}$.

From (4.11), in this recursive step, there need to be

$$\left(\sum_{i=2}^{t} h_i^{(0)} + (t-1)h_1^{(0)} + 2t - 1\right)p\ XORs. \tag{4.12}$$

By the same token, we know that there need to be $(\sum_{i=j+1}^{t} h_i^{(j-1)} + (t-j)h_j^{(j-1)} + 2t - 2j + 1)p$ XORs to calculate $\vec{S^{(j)}} = L_{j-1} \times \vec{S^{(j-1)}}$. Therefore, a total of $\sum_{j=1}^{t}(\sum_{i=j+1}^{t} h_i^{(j-1)} + (t-j)h_j^{(j-1)} + 2t - 2j + 1)p$ XOR operations are needed to get $\vec{S^{(t)}}$.

From (4.3), we have

$$\begin{cases} h_i^{(0)} = 0 & for \quad 1 \le i \le t \\ k_i^{(0)} = 0 & for \quad 1 \le i \le t. \end{cases}$$

From (4.9), we have

$$\begin{cases} h_i^{(\lambda)} = \lambda & for \quad 0 \le \lambda \le t, \lambda \le i \le t \\ k_i^{(\lambda)} = \frac{\lambda(\lambda+1)}{2} & for \quad 0 \le \lambda \le t, \lambda \le i \le t. \end{cases} \tag{4.13}$$

Thus,

$$\sum_{j=1}^{t}\left(\sum_{i=j+1}^{t} h_i^{(j-1)} + (t-j)h_j^{(j-1)} + 2t - 2j + 1\right)p = \frac{t^3+2t}{3} \cdot p.$$

Now, we come to the backward steps. First, we give a definition.

Definition 4.1. *For any given A, where*

$$A = \prod_{i=1}^{n}(x_i + y_i),$$

let

$$A^{-1} = \prod_{i=2}^{n} \overrightarrow{(x_i+y_i)^{-1}} \cdot (x_1+y_1)^{-1}$$

and

$$A^{-1^*} = \prod_{i=1}^{n} \overrightarrow{(x_i+y_i)^{-1}}.$$

Then, obviously,

$$\begin{cases} A^{-1} \times A = Q \\ A^{-1^*} \times Q \times A = Q. \end{cases}$$

Since we know that

$$H^{(t)} = \begin{bmatrix} X_1^{(0)}A_1^{(0)} & X_2^{(0)}(x_1+a_1)A_1^{(0)} & \cdots & X_t^{(0)}(x_1+a_1)A_1^{(0)} \cdot \prod_{m=2}^{t-1}(x_t+a_m) \\ O & X_2^{(1)}A_2^{(1)} & \cdots & X_t^{(1)}(x_2+a_2)A_2^{(1)} \cdot \prod_{m=3}^{t-1}(x_t+a_m) \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \cdots & X_t^{(t-1)}A_t^{(t-1)} \end{bmatrix},$$

let

$$L_i^* = \left[l_{j,k}\right],$$

where

$$l_{j,k} = \begin{cases} I & j=k \\ O & k \ne i, j \ne k \text{ or } j > k \\ X_i^{(j-1)}(x_j+a_j)A_j^{(j-1)} \prod_{m=j+1}^{i-1}(x_i+a_m)(X_i^{(i-1)})^{-1^*}(A_i^{(i-1)})^{-1} & k=i, j<k \end{cases} \tag{4.14}$$

for $2 \le i \le t$.

Let

$$H_i = \begin{cases} L_i^* \times H_{i+1} & 2 \le i \le t-1 \\ L_t^* \times H^{(t)} & i = t, \end{cases}$$

$$\vec{S_i} = \begin{cases} L_i^* \times \vec{S_{i+1}} & 2 \le i \le t-1 \\ L_t^* \times \vec{S^{(t)}} & i = t. \end{cases}$$

Thus, $H_i \times \vec{\Psi(t)} = \vec{S_i}$ for $2 \le i \le t$ and

$$H_2 = \begin{bmatrix} X_1^{(0)}A_1^{(0)} & O & \cdots & O \\ O & X_2^{(1)}A_2^{(1)} & \cdots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \cdots & X_i^{(i-1)}A_i^{(i-1)} \end{bmatrix}.$$

Finally, let

$$L_1^* = \begin{bmatrix} (X_1^{(0)})^{-1^*}(A_1^{(0)})^{-1} & O & \cdots & O \\ O & (X_2^{(1)})^{-1^*}(A_2^{(1)})^{-1} & \cdots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \cdots & (X_i^{(i-1)})^{-1^*}(A_i^{(i-1)})^{-1} \end{bmatrix},$$

(4.14′)

$$H_1 = L_1^* \times H_2 = \begin{bmatrix} P & O & O & \cdots & O \\ O & Q & O & \cdots & O \\ O & O & Q & \cdots & O \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ O & O & O & \cdots & Q \end{bmatrix},$$

and

$$\vec{S^*} = L_1^* \times \vec{S_2} = \begin{bmatrix} s_1^* \\ s_2^* \\ \vdots \\ s_t^* \end{bmatrix}.$$

Therefore,

$$\begin{aligned} &H_1 \times \vec{\Psi^{(t)}} = H_1 \times R_t^* \times R_{t-1}^* \times \cdots \times R_1^* \times \vec{\Psi^{(0)}} \\ &= \begin{bmatrix} P\alpha_1^{(0)} \\ Q(x_2+a_1)^{-1}\alpha_2^{(0)} \\ \vdots \\ Q(x_t+a_{t-1})^{-1}\prod_{i=1}^{t-2}(P\overrightarrow{(x_t+a_i)^{-1}})\alpha_t^{(0)} \end{bmatrix} = \vec{S^*} = \begin{bmatrix} s_1^* \\ s_2^* \\ \vdots \\ s_t^* \end{bmatrix}. \end{aligned} \tag{4.15}$$

Left multiplying both sides of (4.15) by

$$\begin{bmatrix} I & O & O & \cdots & O \\ O & (x_2+a_1) & O & \cdots & O \\ O & O & (x_3+a_1)(x_3+a_2) & \cdots & O \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ O & O & O & \cdots & \prod_{i=1}^{t-1}(x_t+a_i) \end{bmatrix},$$

we will get

$$\begin{bmatrix} \alpha_1^{(0)} \\ \alpha_2^{(0)} \\ \alpha_3^{(0)} \\ \vdots \\ \alpha_t^{(0)} \end{bmatrix} = \begin{bmatrix} \vec{S_1^*} \\ (x_2+a_1)\vec{S_2^*} \\ (x_3+a_1)(x_3+a_2)\vec{S_3^*} \\ \vdots \\ \prod_{i=1}^{t-1}(x_t+a_i)\vec{S_t^*} \end{bmatrix}.$$

Thus, the decoding procedure is completed.

It can be easily checked that there need to be $O(p^3 \cdot i^2)$ XORs to calculate $l_{j,k}$ in L_i^*. Thus, $O(p^3 \cdot t^3)$ XORs are needed to get $\vec{S_i}$ from $\vec{S_{i+1}}$. So, the complexity of calculating $\vec{S^*}$ from $\vec{S^{(t)}}$, i.e., the complexity of the backward steps, is $O(p^3 \cdot t^4)$.

On the other hand, we already know that the complexity of the forward steps is $O(p \cdot t^3)$. Therefore, with the forward steps and the backward steps combined, calculation of the syndromes in the decoding process requires a total of $O(p^3 \cdot t^4)$ XOR operations.

Theorem 4.1. *For the extended Rabin-like codes based on CPM with r packages erased, the decoding cost is $O(m^3 \cdot r^4)$ XOR operations at the most.*

In the following, we show how the decoding algorithm corrects four errors.

For convenience, let us consider the worst case where all four errors are in $\overrightarrow{c}_j$ for $4 \le j \le n+3$, i.e., on the information disks. Correcting these erasure errors is equivalent to solving the following set of linear equations with a Cauchy matrix.

$$\begin{bmatrix} (I+E^i)^{-1} & (I+E^j)^{-1} & (I+E^k)^{-1} & (I+E^s)^{-1} \\ (E+E^i)^{-1} & (E+E^j)^{-1} & (E+E^k)^{-1} & (E+E^s)^{-1} \\ (E^2+E^i)^{-1} & (E^2+E^j)^{-1} & (E^2+E^k)^{-1} & (E^2+E^s)^{-1} \\ (E^3+E^i)^{-1} & (E^3+E^j)^{-1} & (E^3+E^k)^{-1} & (E^3+E^s)^{-1} \end{bmatrix} \times \begin{bmatrix} \overrightarrow{c}_i \\ \overrightarrow{c}_j \\ \overrightarrow{c}_k \\ \overrightarrow{c}_s \end{bmatrix} = \begin{bmatrix} \overrightarrow{s}_0^{(0)} \\ \overrightarrow{s}_0^{(1)} \\ \overrightarrow{s}_0^{(2)} \\ \overrightarrow{s}_0^{(3)} \end{bmatrix}. \tag{4.16}$$

The decoding process can be briefly summarized as follows: Given a received codeword y and the locations of the lost packets $\mu_1, \mu_2, \mu_3, \mu_4$, we first compute the syndromes from (4.1) and then determine the values of the lost packets $\overrightarrow{c}_{\mu_i}$ for $1 \le i \le 4$ by solving (4.16).

Let

$$H^{(0)} = \begin{bmatrix} (I+E^i)^{-1} & (I+E^j)^{-1} & (I+E^k)^{-1} & (I+E^s)^{-1} \\ (E+E^i)^{-1} & (E+E^j)^{-1} & (E+E^k)^{-1} & (E+E^s)^{-1} \\ (E^2+E^i)^{-1} & (E^2+E^j)^{-1} & (E^2+E^k)^{-1} & (E^2+E^s)^{-1} \\ (E^3+E^i)^{-1} & (E^3+E^j)^{-1} & (E^3+E^k)^{-1} & (E^3+E^s)^{-1} \end{bmatrix},$$

$$\overrightarrow{\Psi}^{(0)} = \begin{bmatrix} \overrightarrow{c}_i \\ \overrightarrow{c}_j \\ \overrightarrow{c}_k \\ \overrightarrow{c}_s \end{bmatrix}, \text{ and } \overrightarrow{S}^{(0)} = \begin{bmatrix} \overrightarrow{s}_0^{(0)} \\ \overrightarrow{s}_1^{(0)} \\ \overrightarrow{s}_2^{(0)} \\ \overrightarrow{s}_3^{(0)} \end{bmatrix}. \tag{4.16'}$$

Thus, (4.16) can be written as

$$H^{(0)} \times \overrightarrow{\Psi}^{(0)} = \overrightarrow{S}^{(0)}. \tag{4.16''}$$

After the forward steps, we have

$$H^{(4)} = L_4 \times L_3 \times L_2 \times L_1 \times H^{(0)} \times R_1 \times R_2 \times R_3, \tag{4.17.1}$$

$$\overrightarrow{\Psi}^{(3)} = R_3^* \times R_2^* \times R_1^* \times \overrightarrow{\Psi}^{(0)}, \tag{4.17.2}$$

and

$$\overrightarrow{S}^{(4)} = L_4 \times L_3 \times L_2 \times L_1 \times \overrightarrow{S}^{(0)}, \tag{4.17.3}$$

where L_i for $i = 1, 2, 3, 4$, R_j and R_j^* for $j = 1, 2, 3$ are defined as in (4.6) and (4.6').

During the forward steps, we need to calculate $\overrightarrow{S}^{(4)}$. To calculate $L_2 \times \overrightarrow{S}^{(1)}$, there need to be nine $((E^\mu + E^\nu)\overrightarrow{s})$-type operations and two $(\overrightarrow{s} + \overrightarrow{t})$-type operations. From Lemmas 2.8 and 2.9 in [15], there need to be $(9+2) \times p = 11p$ XOR operations. For the same reason, calculating $L_1 \times \overrightarrow{S}^{(0)}$, $L_3 \times \overrightarrow{S}^{(2)}$, and $L_4 \times \overrightarrow{S}^{(3)}$ involves $10p$, $8p$, and p XOR

operations, respectively. Thus, in the forward steps, a total of $30p$ XOR operations are needed.

After the backward steps, we have

$$H_1 \times R_3^* \times R_2^* \times R_1^* \times \overrightarrow{\Psi}^{(0)} = \overrightarrow{S}^{(8)}, \tag{4.18}$$

where

$$H_1 = L_1^* \times L_2^* \times L_3^* \times L_4^* \times H^{(4)} = \begin{bmatrix} P & O & O & O \\ O & Q & O & O \\ O & O & Q & O \\ O & O & O & Q \end{bmatrix}, \tag{4.19}$$

$$\overrightarrow{S}^{(8)} = L_1^* \times L_2^* \times L_3^* \times L_4^* \times \overrightarrow{S}^{(4)},$$

and L_i^* for $i = 1, 2, 3, 4$ are defined in (4.14) and (4.14'), namely

$$\begin{bmatrix} P\overrightarrow{c}_i \\ Q(I+E^j)^{-1}\overrightarrow{c}_j \\ Q(E+E^k)^{-1}\overleftarrow{P(I+E^k)^{-1}}\overrightarrow{c}_k \\ Q(E^2+E^s)^{-1}\overleftarrow{P(E+E^s)^{-1}}\overleftarrow{P(I+E^s)^{-1}}\overrightarrow{c}_s \end{bmatrix} \begin{bmatrix} \overrightarrow{s}_0^{(8)} \\ \overrightarrow{s}_1^{(8)} \\ \overrightarrow{s}_2^{(8)} \\ \overrightarrow{s}_3^{(8)} \end{bmatrix}. \tag{4.20}$$

Both sides are multiplied by

$$\begin{bmatrix} I & O & O & O \\ O & (I+E^j) & O & O \\ O & O & (I+E^k)(E+E^k) & O \\ O & O & O & (I+E^s)(E+E^s)(E^2+E^s) \end{bmatrix}.$$

Using (2.3'), (3.2), (3.2'), and (3.13), we have

$$\begin{bmatrix} \overrightarrow{c}_i \\ \overrightarrow{c}_j \\ \overrightarrow{c}_k \\ \overrightarrow{c}_s \end{bmatrix} = \begin{bmatrix} \overrightarrow{s}_0^{(8)} \\ (I+E^j)\overrightarrow{s}_1^{(8)} \\ (I+E^k)(E+E^k)\overrightarrow{s}_2^{(8)} \\ (I+E^s)(E+E^s)(E^2+E^s)^{-1}\overrightarrow{s}_3^{(8)} \end{bmatrix}. \tag{4.21}$$

Remark. As an example, let us consider

$$(I+E^k)(E+E^k) \times Q(E+E^k)^{-1}\overleftarrow{P(I+E^k)^{-1}}\overrightarrow{c}_k.$$

From (3.2), it is equal to

$$(I+E^k)(E+E^k)(E+E^k)^{-1}\overleftarrow{P(I+E^k)^{-1}}\overrightarrow{c}_k.$$

Using (2.1'), (2.3'), (3.2'), and (3.13), it is equal to

$$(I+E^k)\overleftarrow{P(I+E^k)^{-1}}\overrightarrow{c}_k P\overrightarrow{c}_k = \overrightarrow{c}_k.$$

During the backward steps, from Lemmas 2.8 and 2.9 in [15], in order to calculate $\overrightarrow{S}^{(8)} = L_1^* \times L_2^* \times L_3^* \times L_4^* \times \overrightarrow{S}^{(4)}$, a total of $65p$ XOR operations are needed.

With the forward steps and the backward steps combined, calculation of the syndromes in the decoding process requires a total of $(95+9n)(m+1)$ XOR operations.

Theorem 4.2. *For the extended Rabin-like codes with $r = 4$ based on CPM, the decoding cost is $(95+9n)(m+1)$ XOR operations at the most.*

Remark. If G is a group of binary 32-dimension vectors, i.e., each vector is regarded as a computer word (32 bits), then each codeword of this extended Rabin code in fact contains 32 binary codewords. These 32 binary codewords can be encoded/decoded simultaneously, reducing the cost to $\frac{(95+9n)(m+1)}{32}$.

5 Conclusions

In this paper, we have presented a class of MDS array codes based on CPM in Cauchy matrix. Although the parity-check matrix is a high-density parity-check matrix, these codes are still highly efficient for tolerating multiple package losses in network-based storage systems, with very fast encoding and decoding. There need to be at most $2nr(m+1)$ XOR operations for encoding and at most $O(m^3 \cdot r^4)$ XOR operations for decoding. When 32/64 codewords are encoded/decoded simultaneously, a 32/64-fold improvement can be achieved in terms of efficiency.

This scheme can increase the performance of network-based storage systems as well as tolerating multipackage loss. This goal is achieved with a recovery algorithm invoked when $(n-r)$ disk data arrive as well as through avoiding sending/receiving some of the packages when a data update request arises.

On the other hand, data consistency, which is also an important issue in network-based storage systems, is not our main concern and, therefore, is not discussed in this paper. It still remains an open problem in our scheme.

Acknowledgments

The authors would like to thank Mr. Hua Qian and Ms. Anna Robin for helpful discussion.

References

[1] D. Patterson, G. Gibson, and R. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," *Proc. ACM SIGMOD '88,* pp. 109-116, June 1988.

[2] P. Elias, "Coding for Two Noisy Channels," *Information Theory, Proc. Third London Symp.,* pp. 61-76, Sept. 1955.

[3] M. Blahut, *Algebraic Codes for Data Transmission.* Cambridge Univ. Press, 2003.

[4] M. Blaum, "A Class of Byte-Correcting Array Code," IBM Research Report, RJ 5652(57151), May 1987.

[5] M. Blaum, J. Bradt, J. Bruck, and J. Menon, "EVEN-ODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures," *IEEE Trans. Computers,* vol. 44, no. 2, pp. 192-202, Feb. 1995.

[6] M. Blaum, J. Bruck, and A. Vardy, "MDS Array Codes with Independent Parity Symbols," *IEEE Trans. Information Theory,* pp. 529-542, Mar. 1996.

[7] M. Blaum, H. Hao, R. Mattson, and J. Menon, "A Coding Technique for Double Disk Failures in Disk Arrays," US Patent 5,271,012, Dec. 1993.

[8] M. Blaum and R. Roth, "New Array Codes for Multiple Phased Burst Correction," *IEEE Trans. Information Theory,* pp. 66-77, Jan. 1993.

[9] M. Blaum and R. Roth, "On Lowest-Density MDS Codes," *IEEE Trans. Information Theory,* pp. 46-59, Jan. 1999.

[10] T. Fuja, C. Heegard, and M. Blaum, "Cross Parity Check Convolutional Code," *IEEE Trans. Information Theory,* pp. 1264-1276, July 1989.
[11] R. Goodman, R.J. McEliece, and M. Sayano, "Phased Burst Correcting Array Codes," *IEEE Trans. Information Theory,* pp. 684-693, Mar. 1993.
[12] L. Xu and J. Bruck, "X-Code: MDS Array Codes with Optimal Encoding," *IEEE Trans. Information Theory,* pp. 272-276, Jan. 1999.
[13] L. Xu, V. Bohossian, J. Bruck, and D.G. Wagner, "Low-Density MDS Codes and Factors of Complete Graphs," *IEEE Trans. Information Theory,* pp. 1817-1826, Sept. 1999.
[14] M. Blaum, J. Bradt, J. Bruck, J. Menon, and A. Vardy, "The EVENODD Code and Its Generalization: An Efficient Scheme for Tolerating Multiple Disk Failures in RAID Architectures," *High Performance Mass Storage and Parallel I/O,* chapter 14, 2002.
[15] G.-L. Feng, R. Deng, F. Bao, and J.-C. Shen, "New Efficient MDS Array Codes for RAID, Part I: Reed-Solomon-Like Codes for Tolerating Three Disk Failures," *IEEE Trans. Computers,* vol. 54, no. 9, pp. 1071-1080, Sept. 2005.
[16] M.O. Rabin, "Efficient Dispersal of Information for Security, Load Balance, and Fault Tolerance," *J. ACM,* vol. 36, no. 2, pp. 335-348, Apr. 1989.
[17] F.J. MacWilliams and N.J.A. Slone, *The Theory of Error-Correcting Codes.* Elsevier Science, 1977.

Gui-Liang Feng (S'89-M'92-SM'95) received the BS degree from Fudan University, Shanghai, in 1968 and the MS degree from Jiaotong University, Shanghai, in 1982, both in applied mathematics, and the PhD degree in computer science from Lehigh University, Bethlehem, Pennsylvania, in 1990. From 1970 to 1979, he was an electrical engineer in the Hudong Shipyard, Shanghai. From 1982 to 1986, he was an assistant researcher at the Shanghai Institute of Computer Technology. Since 1991, he has been with the Center for Advanced Computer Studies at the University of Louisiana at Lafayette, where he is currently an associate professor. His research interests include error-correcting codes, data compression, fault-tolerant computing, cryptography, and computational algebra. He has publisheded more then 50 papers in journals and proceedings of international symposia and conferences. He has received some research grants from the US National Science Foundation, US Office of Naval Research, NASA, US Army Research Office, and LEQSF. Dr. Feng was a member of the Governing Board of the Information Theory Society of the Chinese Institute of Electronics. He received the 1994 IEEE Information Theory Society Paper Award and the Outstanding Paper Award of the Chinese Institute of Electronics for 1984-1987. He served as an associate editor for coding and fault-tolerant computing for the *IEEE Transactions on Computers* from 1996-2000. He is a senior member of the IEEE and the IEEE Computer Society.

Robert Deng received the BE degree from the National University of Defense Technology, China, and the MS and PhD degrees from the Illinois Institute of Technology, Chicago. He is currently a professor and director of the SIS Research Center, School of Information Systems, Singapore Management University. Prior to this, he was a principal scientist and manager of the Infocomm Security Department, Institute for Infocomm Research. He has 21 patents and more than 140 technical publications in international conferences and journals in the areas of digital communications, network and distributed system security, and information security.

Feng Bao received the BS degree in mathematics and the MS degree in computer science from Peking University and the PhD degree in computer science from Gunma University. Since 1996, he has been with the Institute for Infocomm Research, Singapore. Currently, he is a lead scientist and the head of the Infocomm Security Department and Cryptography Lab of the institute. His research areas include algorithm, automata theory, complexity, cryptography, distributed computing, fault tolerance, and information security. He has more than 120 technical publications in international conferences and journals and 16 patents.

Jia-Chen Shen received the BS degree from Shanghai Jiaotong University, Shanghai, in 2001 and the MS degree from the University of Louisiana at Lafayette, in 2003, both in computer science. He is currently working toward the PhD degree in computer science at University of Louisiana at Lafayette. His research interests include error-correcting codes, cryptography, and computing algebra.