A New Construction and an Efficient Decoding Method for Rabin-Like Codes

Hanxu Hou *Member, IEEE* and Yunghsiang S. Han, *Fellow, IEEE*

***Abstract*— Array codes have been widely used in communication and storage systems. To reduce computational complexity, one important property of the array codes is that only exclusive OR operations are used in the encoding and decoding processes. Cauchy Reed-Solomon codes, Rabin-like codes and circulant Cauchy codes are existing Cauchy maximum-distance separable (MDS) array codes that employ Cauchy matrices over finite fields, circular permutation matrices and circulant Cauchy matrices, respectively. All these codes can correct any number of failures; however, a critical drawback of existing codes is the high decoding complexity. In this work, we propose a new construction of Rabin-like codes based on a quotient ring with a cyclic structure. The newly constructed Rabin-like codes have more supported parameters (prime p is extended to an odd number) such that the world sizes of them are more flexible than the existing Cauchy MDS array codes. An efficient decoding method using LU factorization of the Cauchy matrix can be applied to the newly constructed Rabin-like codes. It is shown that the decoding complexity of the proposed approach is less than that of existing Cauchy MDS array codes. Hence, the Rabin-like codes based on the new construction are attractive to distributed storage systems.**

***Index Terms*—MDS array codes, Rabin-like codes, decoding, storage systems.**

I. Introduction

Array codes are error and burst correcting codes that have been widely employed in communication and storage systems [1], [2], to enhance data reliability. A common property of the array codes is that the encoding and decoding algorithms use only XOR (exclusive OR) operations. A binary array code consists of an array of size $m \times n$, where each element in the array stores a single bit. Among the n columns (or data disks) in the array, the first k columns are *information columns* that store *information bits*, and the last r columns are *parity columns* that store *parity bits*. Note that $n = r + k$. When a data disk fails, the corresponding column of the array code is considered as an *erasure*. If the array code can tolerate arbitrary r erasures, then it is called a maximum-distance separable (MDS) array code. In other words, in an MDS array code, the information bits can be recovered from any k columns.

Besides the MDS property, the performance of an MDS array code also depends on encoding and decoding complexities. *Encoding complexity* is defined as the number of XORs required to construct the parities, and *decoding complexity* is defined as the number of XORs required to recover the erased columns from any surviving k ones.

A. Related Work

Row-diagonal parity (RDP) code [3], EVENODD code [4] and Liber8Tion code [5] can tolerate two arbitrary disk erasures. Due to increasing capacities of hard disks and the requirements of low bit error rates, the protection offered by double parities will soon be inadequate. The issue of reliability is more pronounced in solid-state drives, which have significant wear-out rates when the frequencies of disk writes are high. The use of triple-parity Redundant Arrays of Inexpensive Disks (RAID) has already been advocated in storage technology [6]. Construction of array codes recovering multiple disk erasures is relatively rare, in comparison to array codes recovering double erasures. We name the existing MDS array codes in [3], [4], [7]–[13] as *Vandermonde MDS array codes*, because their constructions are based on Vandermonde matrices.

Among the Vandermonde MDS array codes, the BBV (Blaum, Bruck and Vardy) code [7], [14], which is an extension of EVENODD code with more parity columns, has the best fault-tolerance. Blaum *et al.* proved that an extended BBV code is always an MDS code for three parity columns, but may not be an MDS code for four or more parity columns [7]. A necessary and sufficient condition for the extended BBV code with four parity columns to be an MDS code is given by [7], and some results for no more than eight parity columns are provided.

Another family of MDS array codes is called *Cauchy MDS array codes*, which are constructed based on Cauchy matrices. Cauchy Reed-Solomon (CRS) codes [15], Rabin-like codes [13] and circulant Cauchy codes [16] are examples of Cauchy MDS array codes. Blömer *et al.* constructed CRS codes by employing a Cauchy matrix to perform encoding (and upon failure, decoding) over a finite field instead of a binary field [15]. In this approach, the isomorphism and companion methods, converting a normal finite field operation to a binary field XOR operation, are necessary. The idea is to replace a field symbol with a matrix in another finite field. Schindelhauer and Ortolf [16] considered a special class of CRS

Hanxu Hou is with the School of Electrical Engineering & Intelligentization, Dongguan University of Technology and the Shenzhen Key Lab of Information Theory & Future Internet Architecture, Future Network PKU Lab of National Major Research Infrastructure, Peking University Shenzhen Graduate School (E-mail: houhanxu@163.com), Yunghsiang S. Han is with the School of Electrical Engineering & Intelligentization, Dongguan University of Technology (E-mail: yunghsiangh@gmail.com). This work was partially supported by the National Natural Science Foundation of China (No. 61701115, 61671007, 61671001), National Keystone R&D Program of China (No. 2017YFB0803204, 2016YFB0800101) and Shenzhen Research Program (No. ZDSYS201603311739428).

codes using circulant Cauchy matrices, called circulant Cauchy codes, that have lower encoding and decoding complexities than CRS codes. Based on the concept of a permutation matrix, Feng *et al.* [13] gave a construction method to convert the Cauchy matrix to a sparse matrix. Compared with the Vandermonde MDS array codes, Cauchy MDS array codes have better fault-tolerance, but at a cost of higher computational complexity. In this paper, we construct Rabin-like codes based on a quotient ring with a cyclic structure and propose an efficient decoding method for the newly constructed Rabin-like codes.

B. Cauchy Reed-Solomon Codes

CRS code [15] is one variant of RS codes that has better coding complexity by employing Cauchy matrices. The key to constructing good CRS codes is the selection of Cauchy matrices. Given k data symbols in a finite field $\mathbb{F}_{2^w}$, we can generate r encoded symbols of a CRS code with $k+r \leq 2^w$ in the following way. Let $X = \{x_1, \ldots, x_r\}$, $Y = \{y_1, \ldots, y_k\}$, where $X \cap Y = \emptyset$, such that each x_i and y_i is a distinct element in $\mathbb{F}_{2^w}$. The entry (i, j) of the Cauchy matrix is calculated as $1/(x_i + y_j)$. Since each element of $\mathbb{F}_{2^w}$ can be represented by a $w \times w$ binary matrix, we can transform the $r \times k$ Cauchy matrix into an $rw \times kw$ *binary distribution matrix* using a projection defined by a primitive polynomial of $\mathbb{F}_{2^w}$ [15]. Fig. 1 displays the encoding process of a CRS code with $k = 4$, $r = 2$ and $w = 3$. We divide each data symbol into w strips; here, a strip is the minimum unit of a symbol. In Fig. 1, the first data symbol is divided into three strips a_1, a_2, a_3. The r encoded symbols are created by multiplying the $rw \times kw$ binary distribution matrix by the kw data strips. During the multiplication process, when there exists "1" in every row of the binary distribution matrix, we do XOR operations on the corresponding data strips to obtain the strips of encoded symbols. The first strip of the first encoded symbol and the second strip of the second encoded symbol in Fig. 1 may be calculated as

$$a_2 + b_3 + c_1 + c_3 + d_1 \text{ and } a_1 + a_2 + b_1 + c_1 + c_3 + d_2$$

respectively. The two strips can be calculated by nine XORs.

With a binary distribution matrix, it is possible to create a strip of encoded symbol as the XOR of all data strips whose corresponding columns of the binary distribution matrix have all 1s. In this approach, the expensive matrix multiplication is replaced by binary addition. Hence, this gives a good reduction in the computational complexity. For more information about the encoding and decoding process of CRS codes, please refer to Plank and Lu [17].

There are two approaches for reducing the number of XORs in the coding processes.

1) **Choosing a "good" Cauchy matrix**. Since the Cauchy matrix dictates the number of XORs [17], many researchers [17]–[19] have designed codes with low-density Cauchy matrices. However, the way to find the lowest-density Cauchy matrix is to enumerate all the matrices and select the best one, where the number of matrices is exponential in k and r. Therefore, this method is only feasible for small k and r. For example, when the parameters k, r, w are small, the performance of CRS is optimized [17], [20] by choosing the Cauchy matrix of which the corresponding binary distribution matrix has the fewest "1"s.
2) **Encoding data using a schedule.** The issue of exploiting common sums in the XOR equations was addressed in [21], [22]. However, finding a good schedule with minimum XORs is still an open problem. Some heuristic schedules were proposed in [23]–[26]. In the above example, the two strips containing $c_1 + c_3$ are treated as a subexpression. Therefore, if the bit $c_1 + c_3$ is calculated before the calculation of two strips, then the two strips can be computed recursively by $x_1 = c_1 + c_3, x_2 = a_2 + b_3, x_3 = x_2 + x_1, x_4 = x_3 + d_1$ and $x_5 = a_1 + a_2, x_6 = x_5 + b_1, x_7 = x_6 + x_1, x_8 = x_7 + d_2$ with eight XORs.

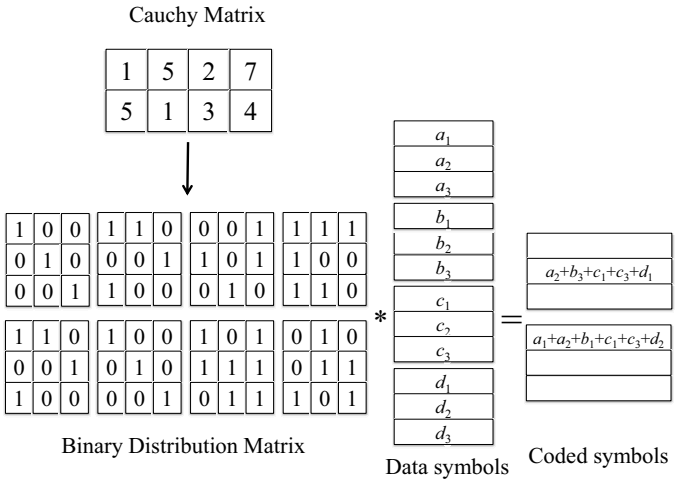


Fig. 1: The encoding process of the CRS code with $k = 4$, $r = 2$ and $w = 3$ over $\mathbb{F}_8$. The finite field is constructed with the primitive polynomial $1 + x + x^3$.

C. Contribution of This Work

In this paper, we propose a new construction of Rabin-like codes based on a specific polynomial ring with a cyclic structure. An efficient decoding algorithm is designed based on LU factorization of the Cauchy matrix, which provides significant simplification of the decoding procedure for the Rabin-like codes. We demonstrate that the proposed decoding algorithm has the lowest decoding complexity among the existing decoding methods of Cauchy array codes. Furthermore, the word size, which constrains data in the disk, is more flexible for the newly constructed Rabin-like codes than the existing Cauchy array codes. Extension of the supported parameters is made possible by using a specific polynomial ring to construct the Rabin-like codes and by exploiting the characterization for the specific polynomial ring. In addition, the new construction has a slightly lower encoding complexity than that proposed in [13].

Although polynomial rings have been employed in most array codes (e.g. [7], [9], [10], [12], [27]), the polynomial

ring in this paper is different from those in other array codes. In [7], [9], [10], [12], [27], the construction is based on a ring of polynomials with binary coefficients modulo $1+x+\cdots+x^{p-1}$ for some prime number p. The calculation of the division $1/(1+x^b)$ involves $2p$ XORs at most (see Lemma 1 in [27]), and the computation is reduced to $(3p-5)/2$ (see Lemma 13 in [28]). In this paper, the underlying polynomial ring consists of polynomials in $\mathbb{F}_2[x]/(1+x^p)$ with an even number of non-zero coefficients, where p is an odd number. Computing the division $1/(1+x^b)$ in the polynomial ring in this paper by the simplified Algorithm 2 only takes $p-3$ XORs, which is less than that in [27], [28] (see the remark above Theorem 9).

Even though the newly constructed Rabin-like codes in this paper are similar to Rabin-like codes [13] and circulant Cauchy codes [16], there are several differences between them. First, the word sizes of the newly constructed Rabin-like codes are extended to be more flexible than the existing Cauchy MDS array codes. Second, although LU factorization of the Cauchy matrix is employed in the decoding in [13] and in this paper, the detailed operations and the LU factorization expressions are different. The decoding procedure of Rabin-like codes [13] involves a specific number of XORs only for $r=4$, while the decoding complexity of the newly constructed Rabin-like codes is a specific number of XORs for $r \geq 1$, and it is much less than that of Rabin-like codes [13]. Third, in the coding process, the inverse computation and solving the linear system of circulant Cauchy codes and those of the newly constructed Rabin-like codes are different. The inverse computation in circulant Cauchy codes can be computed with $2p-3$ XORs, while the inverse computation in the newly constructed Rabin-like codes takes only $p-1$ XORs. The linear system of Cauchy matrix form in this paper is solved by LU factorization of the Cauchy matrix, but not in circulant Cauchy codes.

The rest of paper is organized as follows. In Section II, we give the new construction for Rabin-like codes. After proving the MDS property of the newly constructed Rabin-like codes in Section III, we give an efficient decoding method for any number of erasures in Section IV. Section V compares the computational complexity of encoding and decoding of the newly constructed Rabin-like codes with those of existing well-known MDS array codes, in terms of the number of XORs in computation. Conclusions are given in Section VI.

II. New construction of Rabin-like Codes

In this section, we give a new construction of Rabin-like codes, which are based on a specific quotient ring. The reduction on computational complexity is made possible by exploiting the cyclic structure and efficient computation of a division in the quotient ring.

A. Quotient Ring

Let $p>2$ be an odd number and let R_p be the quotient ring

$$R_p \triangleq \mathbb{F}_2[x]/(1+x^p), \tag{1}$$

which is also called a cyclotomic ring in [29]. Every element of R_p will be referred to as a *polynomial* in the sequel. The vector $(a_0, a_1, \ldots, a_{p-1}) \in \mathbb{F}_2^p$ is the codeword corresponding to the polynomial $\sum_{i=0}^{p-1} a_i x^i$. The indeterminate x represents the *cyclic-right-shift* operator in the codewords.

Consider the specific quotient ring, C_p, which consists of polynomials in R_p with an even number of non-zero coefficients,

$$C_p = \{a(x)(1+x) \mod (1+x^p) |\, a(x) \in R_p\}. \tag{2}$$

The *check polynomial* of C_p is $h(x) = 1+x+\cdots+x^{p-1}$. That is, $\forall s(x) \in C_p$ and $c(x) \in R_p$, we have

$$s(x)(c(x)+h(x)) = s(x)c(x) \mod (1+x^p), \tag{3}$$

since

$$\begin{aligned} s(x)h(x) &= (a(x)(1+x) \mod (1+x^p))h(x) \mod (1+x^p) \\ &= a(x)((1+x)h(x)) \mod (1+x^p) \\ &= 0 \mod (1+x^p). \end{aligned}$$

Recall that, in a general ring R with identity, there exists the identity e such that $ue = eu = u$, $\forall u \in R$. The identity element of C_p is

$$e(x) \triangleq 1 + h(x) = x + x^2 + \cdots + x^{p-1}.$$

Note that the ring C_p is discussed in [28], [30] and is used in regenerating codes for computational complexity reduction. From Theorem 2 in [28], we have that C_p is isomorphic to $\mathbb{F}_2[x]/(h(x))$. Note that C_p is isomorphic to a finite field $\mathbb{F}_{2^{p-1}}$ if and only if 2 is a primitive element in $\mathbb{F}_p$ [31]. As far as we know, Silverman was the first to use R_p for performing computations in $\mathbb{F}_{2^{p-1}}$ [29], when p is a prime such that 2 is a primitive element in $\mathbb{F}_p$. In addition, Blaum *et al.* [7], [14] discussed the rings $\mathbb{F}_2[x]/(h(x))$ in detail.

A polynomial $f(x) \in C_p$ is called *invertible* if we can find a polynomial $\bar{f}(x) \in C_p$ such that $f(x)\bar{f}(x)$ is equal to $e(x)$. The polynomial $\bar{f}(x)$ is called the inverse of $f(x)$. It can be shown that the inverse is unique in C_p. The next lemma demonstrates that $x^t + x^{t+b}$ is invertible.

Lemma 1 (Lemma 9 in [28])**.** *Let $p>2$ be an odd number such that all divisors of p except 1 are strictly larger than $k+r-1$, then there exists a polynomial $a(x) \in C_p$ such that*

$$a(x)(x^t + x^{t+b}) = e(x) \mod (1+x^p), \tag{4}$$

where $t, b \geq 0$ and $1 \leq t+b < p$.

The result in Lemma 1 has been independently observed in [32]. Hereafter, we represent the inverse of $x^t + x^{t+b}$ as $1/(x^t + x^{t+b})$. In the following, we present some properties of the inverses, which will be used in the proof of the Cauchy determinant over C_p.

Lemma 2. *Let a, b, c, d be integers between 0 and $p-1$ such that $a \neq b$ and $c \neq d$. For two polynomials $s_1(x), s_2(x) \in R_p$, the following equations hold:*

$$\frac{1}{x^a + x^b} \cdot \frac{1}{x^c + x^d} = \frac{1}{(x^a + x^b)(x^c + x^d)}, \tag{5}$$

$$\frac{s_1(x)}{x^a + x^b} + \frac{s_2(x)}{x^a + x^b} = \frac{s_1(x) + s_2(x)}{x^a + x^b}, \tag{6}$$

$$\frac{1}{x^a + x^b} + \frac{1}{x^c + x^d} = \frac{x^a + x^b + x^c + x^d}{(x^a + x^b)(x^c + x^d)}. \tag{7}$$

Proof. Let $p(x)$ and $q(x)$ be inverses of x^a+x^b and x^c+x^d, respectively. That is, $(x^a+x^b)p(x) = e(x) \bmod (1+x^p)$ and $(x^c+x^d)q(x) = e(x) \bmod (1+x^p)$. Thus we have

$$(x^a+x^b)(x^c+x^d)p(x)q(x) = e(x)e(x) = e(x) \bmod (1+x^p).$$

Note that, $(x^a+x^b)(x^c+x^d) \in C_p$ and $p(x)q(x) \in C_p$. By definition, we have

$$\frac{1}{(x^a+x^b)(x^c+x^d)} = p(x)q(x).$$

Therefore, (5) holds, and (6) follows from

$$p(x)s_1(x) + p(x)s_2(x) = p(x)(s_1(x)+s_2(x)).$$

The right-hand side of equation (7) is

$$\begin{aligned}
&\frac{(x^a+x^b)+(x^c+x^d)}{(x^a+x^b)(x^c+x^d)} \\
=&\frac{(x^a+x^b)+(x^c+x^d)}{(x^a+x^b)} \cdot \frac{1}{(x^c+x^d)} \text{ (by (5))} \\
=&(e(x)+\frac{x^c+x^d}{x^a+x^b}) \cdot \frac{1}{(x^c+x^d)} \text{ (by (6))} \\
=&\frac{1}{x^a+x^b} + \frac{1}{x^c+x^d}.
\end{aligned}$$

□

For a square matrix in C_p, we define the inverse matrix as follows.

Definition 1. *An $\ell \times \ell$ matrix $\mathcal{M}_{\ell\times\ell}$ over C_p is* invertible *if we can find an $\ell \times \ell$ matrix $\mathcal{M}^{-1}_{\ell\times\ell}$ such that $\mathcal{M}_{\ell\times\ell} \cdot \mathcal{M}^{-1}_{\ell\times\ell} = \mathcal{I}_\ell$, $\mathcal{M}^{-1}_{\ell\times\ell}$ is the inverse matrix of $\mathcal{M}_{\ell\times\ell}$, $\mathcal{I}_\ell$ is the $\ell \times \ell$ identity matrix*

$$\mathcal{I}_\ell \triangleq \begin{bmatrix} e(x) & 0 & \cdots & 0 \\ 0 & e(x) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e(x) \end{bmatrix}. \tag{8}$$

B. Construction of Rabin-like Codes

The newly constructed Rabin-like codes can be represented by a $(p-1) \times (k+r)$ array, which is denoted by $\mathcal{C}(k,r,p)$, where $p > 2$ is an odd number such that all divisors of p except 1 are no less than $k+r$. Note that the newly constructed Rabin-like codes have more flexible parameters than Rabin-like codes, as the parameter p in Rabin-like codes should be a prime. We index the columns by $\{0,1,\ldots,k+r-1\}$, and the rows by $\{0,1,\ldots,p-2\}$. Columns 0 to $k-1$ are called the *information columns*, which store the information bits. Columns k to $k+r-1$ are called the *parity columns*, which store the redundant bits.

For $i = 0,1,\ldots,p-2$ and $j = 0,1,\ldots,k-1$, let the i-th information bit in the j-th information column be denoted by $s_{i,j}$. For each $p-1$ information bits $s_{0,j}, s_{1,j}, \ldots, s_{p-2,j}$ stored in j-th information column, one extra *parity-check bit* $s_{p-1,j}$ is computed as

$$s_{p-1,j} \triangleq s_{0,j} + s_{1,j} + \cdots + s_{p-2,j}. \tag{9}$$

We define a *data polynomial* for the j-th information column as

$$s_j(x) \triangleq s_{0,j} + s_{1,j}x + \cdots + s_{p-2,j}x^{p-2} + s_{p-1,j}x^{p-1}. \tag{10}$$

Note that the extra parity-check bit is not stored, and can be computed when necessary. It is easy to see that each data polynomial is an element in C_p.

Next, we present the method to compute the encoded symbols in parity columns. For $i = 0,1,\ldots,p-2$ and $j = 0,1,\ldots,r-1$, let the i-th redundant bit stored in the j-th parity column be denoted by $c_{i,j}$. We define a *coded polynomial* for the j-th parity column as

$$c_j(x) \triangleq c_{0,j} + c_{1,j}x + \cdots + c_{p-2,j}x^{p-2} + c_{p-1,j}x^{p-1}. \tag{11}$$

It will be clear later that

$$c_{p-1,j} = c_{0,j} + c_{1,j} + \cdots + c_{p-2,j}.$$

The coded polynomial can be generated by

$$\begin{aligned}
&\begin{bmatrix} c_0(x) & c_1(x) & \cdots & c_{r-1}(x) \end{bmatrix}^T \\
\triangleq\ &\mathbf{C}_{r\times k} \cdot \begin{bmatrix} s_0(x) & s_1(x) & \cdots & s_{k-1}(x) \end{bmatrix}^T,
\end{aligned} \tag{12}$$

where

$$\mathbf{C}_{r\times k} \triangleq \begin{bmatrix} \frac{1}{1+x^r} & \frac{1}{1+x^{r+1}} & \cdots & \frac{1}{1+x^{k+r-1}} \\ \frac{1}{x+x^r} & \frac{1}{x+x^{r+1}} & \cdots & \frac{1}{x+x^{k+r-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{x^{r-1}+x^r} & \frac{1}{x^{r-1}+x^{r+1}} & \cdots & \frac{1}{x^{r-1}+x^{k+r-1}} \end{bmatrix} \tag{13}$$

is a $r \times k$ rectangular Cauchy matrix. Each entry of the matrix in (13) is the inverse of x^t+x^{t+b} and all arithmetic operations in (12) are performed in C_p. The coded polynomials $c_j(x)$ in (12), for $0 \le j \le r-1$, are in C_p. Hence, the $(k+r) \times k$ generator matrix $\mathbf{G}_{(k+r)\times k}$ of the codewords

$$s_0(x), s_1(x), \ldots, s_{k-1}(x), c_0(x), c_1(x), \cdots, c_{r-1}(x)$$

is given by

$$\mathbf{G}_{(k+r)\times k} = \begin{bmatrix} \mathcal{I}_k \\ \mathbf{C}_{r\times k} \end{bmatrix},$$

where $\mathcal{I}_k$ is the $k \times k$ matrix given in (8). Note that all entries in $\mathbf{G}_{(k+r)\times k}$ are in C_p.

The above encoding procedure can be summarized as three steps: (i) given $k(p-1)$ information bits, append k extra parity-check bits as given in (9) and obtain the k polynomials

$$s_0(x), s_1(x), \cdots, s_{k-1}(x);$$

(ii) generate r coded polynomials as given in (12); and (iii) ignore the terms with degree $p-1$ of the coded polynomials and store the coefficients of the terms in the coded polynomials of degrees from 0 to $p-2$. Next, we give an example to demonstrate the procedure of encoding.

Example 1. *Consider a code $\mathcal{C}(3,2,5)$ with two data polynomials $s_i(x) = s_{0,i} + s_{1,i}x + s_{2,i}x^2 + s_{3,i}x^3 + (s_{0,i} + s_{1,i} + s_{2,i} + s_{3,i})x^4$, for $i = 0,1$. Two coded polynomials*

$c_0(x) = c_{0,0} + c_{1,0}x + c_{2,0}x^2 + c_{3,0}x^3 + c_{4,0}x^4$, $c_1(x) = c_{0,1} + c_{1,1}x + c_{2,1}x^2 + c_{3,1}x^3 + c_{4,1}x^4$ *are computed by*

$$c_0(x) \triangleq \frac{1}{1+x^2}s_0(x) + \frac{1}{1+x^3}s_1(x),$$
$$c_1(x) \triangleq \frac{1}{x+x^2}s_0(x) + \frac{1}{x+x^3}s_1(x).$$

$\frac{s(x)}{1+x^b}$ *can be solved with two XORs by the simplified Algorithm 2 which is presented below in Section IV-C2. For* $c(x) = \frac{s_0(x)}{1+x^2}$, *we have* $s_{0,0} = c_0 + c_3$, $s_{1,0} = c_1 + c_4$, $s_{2,0} = c_2 + c_0$, $s_{3,0} = c_3 + c_1$ *and* $s_{4,0} = c_4 + c_2$. *First, we set* $c_4 = 0$, *and have* $c_1 = s_{1,0}$ *and* $c_2 = s_{4,0}$. *Then, we can compute* $c_0 = s_{2,0} + s_{4,0}$ *from* $c_0 = s_{2,0} + c_2$, *and* $c_3 = c_0 + s_{0,0} = s_{2,0} + s_{4,0} + s_{0,0}$. *The total number of XORs involved in computing* $\frac{s_0(x)}{1+x^2}$ *is two.*

TABLE I: The array code $\mathcal{C}(3, 2, 5)$.

Disk 0	Disk 1	Disk 2	Disk 3
$s_{0,0}$	$s_{0,1}$	$(s_{2,0} + s_{4,0}) + (s_{1,1} + s_{3,1} + s_{4,1})$	$(s_{0,0} + s_{1,0} + s_{3,0}) + (s_{0,1} + s_{3,1})$
$s_{1,0}$	$s_{1,1}$	$s_{1,0} + s_{4,1}$	$s_{1,0} + s_{2,1}$
$s_{2,0}$	$s_{2,1}$	$s_{4,0} + s_{2,1}$	$s_{4,0} + s_{0,1}$
$s_{3,0}$	$s_{3,1}$	$(s_{0,0} + s_{2,0} + s_{4,0}) + (s_{1,1} + s_{4,1})$	$(s_{1,0} + s_{3,0}) + (s_{0,1} + s_{1,1} + s_{3,1})$
$s_{4,0}$	$s_{4,1}$	0	0

The code given in the above example is shown in Table I. The last row of the array code in Table I does not need to be stored, as the last bit of each information column is the parity-check bit of the first $p-1$ *bits and the last bit of each parity column is 0.*

Assume that two data polynomials $s_0(x), s_1(x)$ *are* $1 + x$ *and* $x + x^3$ *respectively, then the two coded polynomials are computed as* $c_0(x) = x$ *and* $c_1(x) = x + x^2 + x^3$.

Before we present a fast decoding algorithm, we first present the MDS property of the newly constructed Rabin-like codes.

III. The MDS Property

A $(p-1) \times n$ array code that encodes $k(p-1)$ information bits is said to be an *MDS array code* if the $k(p-1)$ information bits can be recovered by downloading any k columns.[1] We are going to prove that the newly constructed Rabin-like code satisfies the MDS property for $k + r \leq p$. The next lemma shows a sufficient MDS property condition of the array code $\mathcal{C}(k, r, p)$.

Lemma 3 (Theorem 2 in [28])**.** *If any* $k \times k$ *sub-matrix of* $\mathbf{G}_{(k+r)\times k}$, *after reduction modulo* $h(x)$, *is a nonsingular matrix over* $\mathbb{F}_2[x]/(h(x))$, *then* $\mathcal{C}(k, r, p)$ *satisfies the MDS property.*

We need the following result about the Cauchy determinant in ring C_p before giving a characterization of the MDS property in terms of determinants.

Lemma 4. *Let* $x^{a_1}, x^{a_2}, \ldots, x^{a_\ell}, x^{b_1}, x^{b_2}, \ldots, x^{b_\ell}$ *be* 2ℓ *distinct monomials, where* $0 \leq a_i, b_i < p$ *for* $i = 1, 2, \ldots, \ell$ *and* p *is an odd number. The determinant of the Cauchy matrix in* (15) *is*

$$D_\ell(x) = \frac{\prod_{\ell \geq j > i \geq 1}(x^{a_j} + x^{a_i})(x^{b_i} + x^{b_j})}{\prod_{\ell \geq j, i \geq 1}(x^{a_i} + x^{b_j})}. \tag{14}$$

$$\mathbf{C}(x^{a_{1:\ell}}, x^{b_{1:\ell}}) \triangleq \begin{bmatrix} \frac{1}{x^{a_1}+x^{b_1}} & \frac{1}{x^{a_1}+x^{b_2}} & \cdots & \frac{1}{x^{a_1}+x^{b_\ell}} \\ \frac{1}{x^{a_2}+x^{b_1}} & \frac{1}{x^{a_2}+x^{b_2}} & \cdots & \frac{1}{x^{a_2}+x^{b_\ell}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{x^{a_\ell}+x^{b_1}} & \frac{1}{x^{a_\ell}+x^{b_2}} & \cdots & \frac{1}{x^{a_\ell}+x^{b_\ell}} \end{bmatrix}. \tag{15}$$

Proof. Recall that the polynomial $x^i + x^j$ is invertible in C_p from Lemma 1 for $0 \leq i < j < k + r$, and $\frac{1}{x^i+x^j}$ is the inverse of $x^i + x^j$. For the determinant of the Cauchy matrix in (15), adding column 1 to each of columns 2 to ℓ, we have the entry in the i-th row and the j-th column as

$$\begin{aligned} \frac{1}{x^{a_i}+x^{b_j}} + \frac{1}{x^{a_i}+x^{b_1}} &= \frac{(x^{a_i}+x^{b_1}) + (x^{a_i}+x^{b_j})}{(x^{a_i}+x^{b_j})(x^{a_i}+x^{b_1})} \text{ (by (7))} \\ &= \frac{(x^{b_j}+x^{b_1})}{(x^{a_i}+x^{b_1})} \cdot \frac{1}{(x^{a_i}+x^{b_j})} \text{ (by (5))}, \end{aligned}$$

where $1 \leq i \leq \ell$ and $2 \leq j \leq \ell$. There is no effect on the value of the determinant from the multiple of a row added to a row of the determinant. Thus, the determinant $D_\ell(x)$ is

$$\begin{vmatrix} \frac{1}{x^{a_1}+x^{b_1}} & \frac{(x^{b_2}+x^{b_1})}{(x^{a_1}+x^{b_1})} \cdot \frac{1}{(x^{a_1}+x^{b_2})} & \cdots & \frac{(x^{b_\ell}+x^{b_1})}{(x^{a_1}+x^{b_1})} \cdot \frac{1}{(x^{a_1}+x^{b_\ell})} \\ \frac{1}{x^{a_2}+x^{b_1}} & \frac{(x^{b_2}+x^{b_1})}{(x^{a_2}+x^{b_1})} \cdot \frac{1}{(x^{a_2}+x^{b_2})} & \cdots & \frac{(x^{b_\ell}+x^{b_1})}{(x^{a_2}+x^{b_1})} \cdot \frac{1}{(x^{a_2}+x^{b_\ell})} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{x^{a_\ell}+x^{b_1}} & \frac{(x^{b_2}+x^{b_1})}{(x^{a_\ell}+x^{b_1})} \cdot \frac{1}{(x^{a_\ell}+x^{b_2})} & \cdots & \frac{(x^{b_\ell}+x^{b_1})}{(x^{a_\ell}+x^{b_1})} \cdot \frac{1}{(x^{a_\ell}+x^{b_\ell})} \end{vmatrix}.$$

Extracting the factor $\frac{1}{x^{a_i}+x^{b_1}}$ from the i-th row for $i = 1, 2, \ldots, \ell$, and the factor $x^{b_j} + x^{b_1}$ from the j-th column for $j = 2, 3, \ldots, \ell$, the determinant is

$$\left(\prod_{i=1}^{\ell} \frac{1}{x^{a_i}+x^{b_1}}\right) \cdot \left(\prod_{j=2}^{\ell} (x^{b_j}+x^{b_1})\right) \cdot \begin{vmatrix} 1 & \frac{1}{(x^{a_1}+x^{b_2})} & \cdots & \frac{1}{(x^{a_1}+x^{b_\ell})} \\ 1 & \frac{1}{(x^{a_2}+x^{b_2})} & \cdots & \frac{1}{(x^{a_2}+x^{b_\ell})} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \frac{1}{(x^{a_\ell}+x^{b_2})} & \cdots & \frac{1}{(x^{a_\ell}+x^{b_\ell})} \end{vmatrix}.$$

For $i = 2, 3, \ldots, \ell$, adding the first row to rows 2 to ℓ, the determinant is

$$\begin{vmatrix} 1 & \frac{1}{(x^{a_1}+x^{b_2})} & \cdots & \frac{1}{(x^{a_1}+x^{b_\ell})} \\ 0 & \frac{(x^{a_1}+x^{a_2})}{(x^{a_1}+x^{b_2})} \cdot \frac{1}{(x^{a_2}+x^{b_2})} & \cdots & \frac{(x^{a_1}+x^{a_2})}{(x^{a_1}+x^{b_\ell})} \cdot \frac{1}{(x^{a_2}+x^{b_\ell})} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \frac{(x^{a_1}+x^{a_\ell})}{(x^{a_1}+x^{b_2})} \cdot \frac{1}{(x^{a_\ell}+x^{b_2})} & \cdots & \frac{(x^{a_1}+x^{a_\ell})}{(x^{a_1}+x^{b_\ell})} \cdot \frac{1}{(x^{a_\ell}+x^{b_\ell})} \end{vmatrix}.$$

Again, extracting the factor $x^{a_1} + x^{a_i}$ from the i-th row for $i = 2, 3, \ldots, \ell$, and the factor $\frac{1}{x^{a_1}+x^{b_j}}$ from the j-th column

[1] In total, one needs to download $k \times (p-1)$ bits.

for $j = 2, 3, \ldots, \ell$, the determinant is

$$\left(\prod_{i=1}^{\ell} \frac{1}{x^{a_i}+x^{b_1}}\right)\left(\prod_{j=2}^{\ell} \frac{1}{x^{a_1}+x^{b_j}}\right)\left(\prod_{j=2}^{\ell}(x^{b_j}+x^{b_1})\right)\cdot \left(\prod_{i=2}^{\ell}(x^{a_1}+x^{a_i})\right)\cdot \begin{vmatrix} 1 & 1 & \cdots & 1 \\ 0 & \frac{1}{(x^{a_2}+x^{b_2})} & \cdots & \frac{1}{(x^{a_2}+x^{b_\ell})} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \frac{1}{(x^{a_\ell}+x^{b_2})} & \cdots & \frac{1}{(x^{a_\ell}+x^{b_\ell})} \end{vmatrix}$$

$$= \frac{\prod_{i=2}^{\ell}(x^{a_i}+x^{a_1})(x^{b_i}+x^{b_1})}{\prod_{1\le i,j\le \ell}(x^{a_i}+x^{b_1})(x^{a_1}+x^{b_j})} \cdot \begin{vmatrix} \frac{1}{(x^{a_2}+x^{b_2})} & \frac{1}{(x^{a_2}+x^{b_3})} & \cdots & \frac{1}{(x^{a_2}+x^{b_\ell})} \\ \frac{1}{(x^{a_3}+x^{b_2})} & \frac{1}{(x^{a_3}+x^{b_3})} & \cdots & \frac{1}{(x^{a_3}+x^{b_\ell})} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{(x^{a_\ell}+x^{b_2})} & \frac{1}{(x^{a_\ell}+x^{b_3})} & \cdots & \frac{1}{(x^{a_\ell}+x^{b_\ell})} \end{vmatrix}.$$

Repeating the above process for the remaining $(\ell-1)\times(\ell-1)$ Cauchy determinant, we can obtain the determinant given in (14). □

The next lemma gives a characterization of the MDS property in terms of determinants.

Lemma 5. *Let $p > 2$ be an odd number such that all divisors of p except 1 are no less than $k+r$. Then the determinant of any $k \times k$ sub-matrix of the generator matrix $\mathbf{G}_{(k+r)\times k}$, after reduction modulo $h(x)$, is invertible over $\mathbb{F}_2[x]/(h(x))$.*

Proof. Note that the determinant of any square matrix of $\mathbf{G}_{(k+r)\times k}$ after reduction modulo $h(x)$ can be computed by first reducing each entry of the square matrix by $h(x)$, and then computing the determinant by reducing $h(x)$. It is sufficient to show that the determinant of any $\ell \times \ell$ sub-matrix of the matrix $\mathbf{C}_{r\times k}$, after reduction modulo $h(x)$, is invertible over $\mathbb{F}_2[x]/(h(x))$, for $1 \le \ell \le \min\{k, r\}$. Considering the matrix $\mathbf{C}_{r\times k}$ given in (13), for any ℓ distinct rows indexed by $a_1, a_2, \cdots, a_\ell$ between 0 and $r-1$ and any ℓ distinct columns indexed by $b_1, b_2, \cdots, b_\ell$ between r and $k+r-1$, the corresponding $\ell \times \ell$ sub-matrix is of the form of the Cauchy matrix given in (15). Hence, the determinant is the polynomial given in (14). As the polynomial $x^i + x^j$ is invertible in C_p from Lemma 1, where $0 \le i < j < k+r$, (14) is invertible in C_p. By the definition of an invertible, there exists a polynomial $a(x) \in C_p$ such that

$$D_\ell(x)a(x) = e(x) \mod (1+x^p)$$

holds. Therefore, we have

$$D_\ell(x)a(x) + (1+x^p)b(x) = 1 + h(x)$$

for some polynomial $b(x) \in C_p$, and

$$D_\ell(x)a(x) + h(x)((1+x)b(x)+1) = 1.$$

Hence, $D_\ell(x)a(x) = 1 \bmod h(x)$, and this proves that the polynomial $(D_\ell(x) \bmod h(x))$ is invertible in $\mathbb{F}_2[x]/(h(x))$. □

By applying Lemma 3 and Lemma 5, we have the following theorem.

Theorem 6. *Let $p > 2$ be an odd number such that all divisors of p except 1 are no less than $k+r$. For any positive integer $r \ge 1$ and $k \ge 2$, the code $\mathcal{C}(k, r, p)$ satisfies the MDS property.*

IV. An Efficient Decoding Method

In this section, we give a decoding method based on the LU factorization of the Cauchy matrix in (15), which is very efficient in decoding the newly constructed Rabin-like codes. Expressing a matrix as a product of a lower triangular matrix $\mathbf{L}$ and an upper triangular matrix $\mathbf{U}$ is called an *LU factorization*. Some results of Cauchy matrix LU factorization over a field can be found in [33]–[35]. We first give an LU factorization of the Cauchy matrix over C_p, and then present the efficient decoding algorithm based on the LU factorization.

A. LU Factorization of the Cauchy Matrix over C_p

Given 2ℓ distinct variables $a_1, a_2, \ldots, a_\ell, b_1, b_2, \ldots, b_\ell$ between 0 and $p-1$, the $\ell \times \ell$ square Cauchy matrix $\mathbf{C}(x^{a_{1:\ell}}, x^{b_{1:\ell}})$ over ring C_p is of the form given in (15). By Theorem 6, the matrix $\mathbf{C}(x^{a_{1:\ell}}, x^{b_{1:\ell}})$ is invertible, and the inverse matrix is denoted as $\mathbf{C}(x^{a_{1:\ell}}, x^{b_{1:\ell}})^{-1}$. A factorization of $\mathbf{C}(x^{a_{1:\ell}}, x^{b_{1:\ell}})^{-1}$ is derived, which is stated in the following theorem. No proof is given as it is similar to Theorem 3.1 in [34].

Theorem 7. *Let $x^{a_1}, x^{a_2}, \ldots, x^{a_\ell}, x^{b_1}, x^{b_2}, \ldots, x^{b_\ell}$ be 2ℓ distinct monomials in R_p, where $0 \le a_i, b_i < p$ for $i = 1, 2, \ldots, \ell$. The inverse matrix $\mathbf{C}(x^{a_{1:\ell}}, x^{b_{1:\ell}})^{-1}$ can be decomposed as*

$$\mathbf{C}(x^{a_{1:\ell}}, x^{b_{1:\ell}})^{-1} = \mathbf{U}_\ell^1 \mathbf{U}_\ell^2 \ldots \mathbf{U}_\ell^{\ell-1} \mathbf{D}_\ell \mathbf{L}_\ell^{\ell-1} \ldots \mathbf{L}_\ell^2 \mathbf{L}_\ell^1, \quad (16)$$

where

$$\mathbf{L}_\ell^i = \begin{bmatrix} \mathcal{I}_i & & & \\ & \frac{1}{x^{a_{i+1}}+x^{a_1}} & & \\ & & \ddots & \\ & & & \frac{1}{x^{a_\ell}+x^{a_{\ell-i}}} \end{bmatrix} \begin{bmatrix} \mathcal{I}_{i-1} & & & \\ & e(x) & 0 & \\ & x^{a_1}+x^{b_i} & x^{a_{i+1}}+x^{b_i} & \\ & & \ddots & \ddots \\ & & & x^{a_{\ell-i}}+x^{b_i} \quad x^{a_\ell}+x^{b_i} \end{bmatrix},$$

$$\mathbf{U}_\ell^i = \begin{bmatrix} \mathcal{I}_{i-1} & & & \\ & e(x) & x^{a_i}+x^{b_1} & \\ & 0 & x^{a_i}+x^{b_{i+1}} & \ddots \\ & & & \ddots \quad x^{a_i}+x^{b_{\ell-i}} \\ & & & \quad\quad x^{a_i}+x^{b_\ell} \end{bmatrix} \begin{bmatrix} \mathcal{I}_i & & & \\ & \frac{1}{x^{b_1}+x^{b_{i+1}}} & & \\ & & \ddots & \\ & & & \frac{1}{x^{b_{\ell-i}}+x^{b_\ell}} \end{bmatrix},$$

for $i = 1, 2, \ldots, \ell - 1$, *and*

$$\mathbf{D}_\ell = diag\{x^{a_1} + x^{b_1}, x^{a_2} + x^{b_2}, \ldots, x^{a_\ell} + x^{b_\ell}\}. \quad (17)$$

Considering $\ell = 2$, the matrix $\mathbf{C}(x^{a_{1:2}}, x^{b_{1:2}})^{-1}$ can be factorized into

$$\begin{bmatrix} e(x) & x^{a_1} + x^{b_1} \\ 0 & x^{a_1} + x^{b_2} \end{bmatrix} \begin{bmatrix} e(x) & 0 \\ 0 & \frac{1}{x^{b_1} + x^{b_2}} \end{bmatrix} \begin{bmatrix} x^{a_1} + x^{b_1} & 0 \\ 0 & x^{a_2} + x^{b_2} \end{bmatrix} \begin{bmatrix} e(x) & 0 \\ 0 & \frac{1}{x^{a_2} + x^{a_1}} \end{bmatrix} \begin{bmatrix} e(x) & 0 \\ x^{a_1} + x^{b_1} & x^{a_2} + x^{b_1} \end{bmatrix}.$$

Based on the factorization in Theorem 7, we have a fast algorithm for solving a Cauchy system of linear equations over C_p. Given an $\ell \times \ell$ linear system in Cauchy matrix form

$$\mathbf{C}(x^{a_{1:\ell}}, x^{b_{1:\ell}})\mathbf{s} = \mathbf{c}, \quad (18)$$

where $\mathbf{s} = (s_1(x), s_2(x), \ldots, s_\ell(x))^t$ and $\mathbf{c} = (c_1(x), c_2(x), \ldots, c_\ell(x))^t$ are two columns of length ℓ over C_p. We can solve the equation for $\mathbf{s}$, given $\mathbf{C}(x^{a_{1:\ell}}, x^{b_{1:\ell}})$ and $\mathbf{c}$, by computing

$$\mathbf{U}_\ell^1 \mathbf{U}_\ell^2 \ldots \mathbf{U}_\ell^{\ell-1} \mathbf{D}_\ell \mathbf{L}_\ell^{\ell-1} \ldots \mathbf{L}_\ell^2 \mathbf{L}_\ell^1 \mathbf{c}. \quad (19)$$

The pseudocode is stated in Algorithm 1. In Algorithm 1, Steps 4 and 5 are the computation of the right matrix of $\mathbf{L}_j^i$ and the column vector $\mathbf{c}$. Steps 6 and 7 calculate the left matrix of $\mathbf{L}_\ell^i$ and the above resulting column vector $\mathbf{c}$. Steps 8 and 9 compute the multiplication of $\mathbf{D}_\ell$ and the vector. Steps 11 and 12 compute multiplication of the right matrix of $\mathbf{U}_\ell^i$ and the vector, while Steps 13 to 18 calculate multiplication of the left matrix $\mathbf{U}_\ell^i$ and the vector.

Algorithm 1 Solving a Cauchy linear system over C_p.

Inputs:

Integer ℓ, odd number $p > 2$, the values of $\mathbf{c} = (c_1(x), \ldots, c_\ell(x))^t$, $a_1, \ldots, a_\ell$ and $b_1, \ldots, b_\ell$.

Outputs:

The values of $\mathbf{s} = (s_1(x), s_2(x), \ldots, s_\ell(x))^t$.

Require: All 2ℓ distinct non-negative integers $a_1, \ldots, a_\ell, b_1, \ldots, b_\ell$ are less than p.

1: **for** $i = 1, 2, \ldots, \ell$ **do**
2: $\quad s_i(x) = c_i(x)$.
3: **for** $i = 1, 2, \ldots, \ell - 1$ **do**
4: $\quad$ **for** $j = i + 1, i + 2, \ldots, \ell$ **do**
5: $\quad\quad s_j(x) = (x^{a_{j-i}} + x^{b_i}) s_{j-1}(x) + (x^{a_j} + x^{b_i}) s_j(x)$.
6: $\quad$ **for** $j = i + 1, i + 2, \ldots, \ell$ **do**
7: $\quad\quad s_j(x) = \frac{1}{x^{a_j} + x^{a_{j-i}}} s_j(x)$.
8: **for** $i = 1, 2, \ldots, \ell$ **do**
9: $\quad s_i(x) = (x^{a_i} + x^{b_i}) s_i(x)$.
10: **for** $i = \ell - 1, \ell - 2, \ldots, 1$ **do**
11: $\quad$ **for** $j = i + 1, i + 2, \ldots, \ell$ **do**
12: $\quad\quad s_j(x) = \frac{1}{x^{b_{j-i}} + x^{b_j}} s_j(x)$.
13: $\quad$ **for** $j = 1, 2, \ldots, \ell$ **do**
14: $\quad\quad$ **if** $j - i = 0$ **then**
15: $\quad\quad\quad s_j(x) = s_j(x) + (x^{a_i} + x^{b_{j-i+1}}) s_{j+1}(x)$.
16: $\quad\quad$ **if** $\ell - 1 \geq j - i \geq 1$ **then**
17: $\quad\quad\quad s_j(x) = (x^{a_i} + x^{b_j}) s_j(x) + (x^{a_i} + x^{b_{j-i+1}}) s_{j+1}(x)$.
18: $\quad s_\ell(x) = (x^{a_i} + x^{b_\ell}) s_\ell(x)$.

B. Decoding Algorithm of Erasures

We now describe the decoding procedure of any $\rho \leq r$ erasures for $\mathcal{C}(k, r, p)$. Suppose that γ information columns $a_1, \ldots, a_\gamma$ and δ parity columns $b_1, \ldots, b_\delta$ are erased with $0 \leq a_1 < \ldots < a_\gamma \leq k - 1$ and $0 \leq b_1 < \ldots < b_\delta \leq r - 1$, where $k \geq \gamma \geq 0$, $r \geq \delta \geq 0$ and $\gamma + \delta = \rho \leq r$. Let

$$\mathcal{A} := \{0, 1, \ldots, k - 1\} \setminus \{a_1, a_2, \ldots, a_\gamma\}$$

be a set of indices of the available information columns, and let

$$\mathcal{B} := \{0, 1, \ldots, r - 1\} \setminus \{b_1, b_2, \ldots, b_\delta\}$$

be a set of indices of the available parity columns. We want to first recover the lost information columns by reading $k - \gamma$ information columns with indices $i_1, i_2, \ldots, i_{k-\gamma} \in \mathcal{A}$, and γ parity columns with indices $\ell_1, \ell_2, \ldots, \ell_\gamma \in \mathcal{B}$, and then recover the failure parity column by multiplying the corresponding encoding vector and the k data polynomials.

We add the extra parity-check bit for information column i_τ to obtain the data polynomial

$$s_{i_\tau}(x) = s_{0,i_\tau} + s_{1,i_\tau} x + \cdots + s_{p-2,i_\tau} x^{p-2} + (\sum_{j=0}^{p-2} s_{j,i_\tau}) x^{p-1},$$

for $\tau = 1, 2, \ldots, k - \gamma$. For $h = 1, 2, \ldots, \gamma$, since the coded polynomial $c_{\ell_h}(x) \in C_p$, we have

$$c_{\ell_h}(x) = c_{0,\ell_h} + c_{1,\ell_h} x + \cdots + c_{p-2,\ell_h} x^{p-2} + (\sum_{j=0}^{p-2} c_{j,\ell_h}) x^{p-1}.$$

Let $p_{\ell_1}(x), p_{\ell_2}(x), \ldots, p_{\ell_\gamma}(x)$ be the polynomials obtained by subtracting the chosen $k - \gamma$ data polynomials $s_{i_1}(x), s_{i_2}(x), \ldots, s_{i_{k-\gamma}}(x)$ from γ coded polynomials $c_{\ell_1}(x), c_{\ell_2}(x), \ldots, c_{\ell_\gamma}(x)$, i.e.,

$$p_{\ell_h}(x) \triangleq c_{\ell_h}(x) + \sum_{j=1}^{k-r} \frac{1}{x^{\ell_h} + x^{\ell_h + i_j + r - 1}} s_{i_j}(x), \quad (20)$$

for $h = 1, 2, \ldots, \gamma$. The γ information erasures can be recovered by solving the linear equations

$$\begin{bmatrix} \frac{1}{x^{\ell_1} + x^{a_1 + r}} & \frac{1}{x^{\ell_1} + x^{a_2 + r}} & \cdots & \frac{1}{x^{\ell_1} + x^{a_\gamma + r}} \\ \frac{1}{x^{\ell_2} + x^{a_1 + r}} & \frac{1}{x^{\ell_2} + x^{a_2 + r}} & \cdots & \frac{1}{x^{\ell_2} + x^{a_\gamma + r}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{x^{\ell_\gamma} + x^{a_1 + r}} & \frac{1}{x^{\ell_\gamma} + x^{a_2 + r}} & \cdots & \frac{1}{x^{\ell_\gamma} + x^{a_\gamma + r}} \end{bmatrix} \begin{bmatrix} s_{a_1}(x) \\ s_{a_2}(x) \\ \vdots \\ s_{a_\gamma}(x) \end{bmatrix} = \begin{bmatrix} p_{\ell_1}(x) & p_{\ell_2}(x) & \cdots & p_{\ell_\gamma}(x) \end{bmatrix}^T. \quad (21)$$

The above system of linear equations is of the form of (18) such that Algorithm 1 can be applied to obtain the γ failure data polynomials. Then, we can recover the δ coded polynomials by multiplying the corresponding encoding vectors and k data polynomials.

C. Computation Complexity of the Linear System in a Cauchy Matrix

1) Algorithm for division: In computing the coded polynomial in (12) and in Algorithm 1, we should compute many divisions of the form $\frac{s(x)}{x^t+x^{t+b}}$, where $s(x) \in C_p$, b and t are non-negative integers such that $b+t < k+r$. Let's first consider the calculation of

$$\frac{s(x)}{x^t + x^{t+b}} = c(x) \mod (1+x^p), \tag{22}$$

where $s(x) \in C_p$ and $c(x) \in R_p$. If $c(x) \notin C_p$, we can take $c(x) + h(x)$, which is in C_p, instead. Later we will show that the above step is not necessary in encoding and decoding processes if we allow some coded polynomials to be of the form $c(x) + h(x)$. The following lemma demonstrates an efficient method to compute (22).

Algorithm 2 Solving the division given in (22).

Inputs:

Non-negative integers b, t and $s_0, s_1, \ldots, s_{p-1}$, where $s_i \in \{0,1\}$ for $i = 0, 1, \ldots, p-1$.

Outputs:

$c_0, c_1, \ldots, c_{p-1}$, where $c_i \in \{0,1\}$ for $i = 0, 1, \ldots, p-1$.

Require: Both $b + t < k + r$ and $s_0 + s_1 + \ldots + s_{p-1} = 0$ hold.

1: $c_{p-1} = 0$.
2: $c_{p-b-1} = s_{(t-1) \bmod p}$.
3: $c_{b-1} = s_{(t+b-1) \bmod p}$.
4: **for** $i = 2, 3, \ldots, p-2$ **do**
5: $\quad c_{(p-ib-1) \bmod p} = s_{(t-(i-1)b-1) \bmod p} + c_{p-(i-1)b-1}$.
6: **if** $\sum_{i=0}^{p-1} c_i \neq 0$ **then**
7: $\quad$ **for** $i = 0, 1, \ldots, p-1$ **do**
8: $\quad\quad c_i = c_i + 1$.

*Steps 6, 7 and 8 are deleted in the simplified version.

Lemma 8. *The coefficients of $c(x)$ in (22) can be computed by Algorithm 2, where $t, b \geq 0$, $0 < b + t < p$, $s(x) = \sum_{i=0}^{p-1} s_i x^i \in C_p$, and $c(x) = \sum_{i=0}^{p-1} c_i x^i \in C_p$.*

Proof. By (22) and Lemma 1, we have

$$s(x)\left(x^{p-t}(1 + x^{2b} + \cdots + x^{(p-1)b})\right) = c(x) \bmod (1+x^p). \tag{23}$$

Multiplied by $x^t + x^{t+b}$, (23) becomes

$$s(x)(1+h(x)) = c(x)(x^t + x^{t+b}) \mod (1+x^p). \tag{24}$$

By (3), (24) is equivalent to

$$s(x) = c(x)(x^t + x^{t+b}) \mod (1+x^p). \tag{25}$$

Then, the coefficients of $s(x)$ and $c(x)$ satisfy

$$\begin{aligned} s_t &= c_0 + c_{p-b}, \\ s_{t+1} &= c_1 + c_{p-b+1}, \\ s_{(t+2) \bmod p} &= c_2 + c_{p-b+2}, \\ &\vdots \\ s_{(t-1) \bmod p} &= c_{p-1} + c_{p-b-1}. \end{aligned} \tag{26}$$

Recall that, given $s(x)$, t and b, there are two polynomials $c(x) = \sum_{i=0}^{p-1} c_i x^i$ and $c(x) + h(x) = \sum_{i=0}^{p-1}(c_i+1)x^i$ such that

$$\Big(\sum_{i=0}^{p-1} c_i x^i\Big)(x^t + x^{t+b}) = \Big(\sum_{i=0}^{p-1}(c_i+1)x^i\Big)(x^t + x^{t+b}) = s(x).$$

We can choose one coefficient c_i of $c(x)$ to be 0, and all the other coefficients can be computed iteratively. Specifically, in Algorithm 2, we let $c_{p-1} = 0$. Then, we obtain $c_{p-b-1} = s_{(t-1) \bmod p}$ and $c_{b-1} = s_{(t+b-1) \bmod p}$. Substituting c_{p-b-1} into the corresponding equation in (26), we have

$$c_{(p-2b-1) \bmod p} = s_{(t-b-1) \bmod p} + c_{p-b-1}.$$

In general,

$$c_{(p-ib-1) \bmod p} = s_{(t-(i-1)b-1) \bmod p} + c_{(p-(i-1)b-1) \bmod p}$$

for $2 \leq i \leq p-2$. Note that each coefficient can be calculated iteratively with at most one XOR operation involved. Next, we need to prove that

$$\{(p-ib-1) \bmod p | 1 \leq i \leq p-2\} = \{0, 1, 2, \ldots, p-2\}.$$

First we prove that if $i \neq j$, then $(p-ib-1) \bmod p \neq (p-jb-1) \bmod p$. If $(p-ib-1) \bmod p = (p-jb-1) \bmod p$ for $1 \leq j < i \leq p-2$, then there exists an integer ℓ such that

$$p - jb - 1 = \ell p + p - ib - 1.$$

The above equation can be further reduced to

$$(i-j)b = \ell p.$$

Since either $b = 1$ or $b \not| p$, we have $(i-j)|p$. However, this is impossible due to the fact that $1 \leq j < i \leq p-2$. Similarly, we can prove that, for $1 \leq i \leq p-2$,

$$p - ib - 1 \bmod p \neq p - 1.$$

Hence, $\{(p-ib-1) \bmod p | 1 \leq i \leq p-2\} = \{0, 1, 2, \ldots, p-2\}$. Finally, if $\sum_{i=0}^{p-1} c_i \neq 0$, then $c(x) \notin C_p$; however, $c(x) + h(x) \in C_p$. □

2) Simplified algorithm for division: Next, we prove that Steps 6, 7 and 8 in Algorithm 2 are not necessary. Hence, the computation complexity of Algorithm 2 can be reduced drastically. We call Algorithm 2 without Steps 6, 7 and 8 the simplified Algorithm 2.

Remark. We remark that the calculation of division in (22) is also given in Lemma 19 in [28] with $(3p-5)/2$ XORs. The similar division of circulant Cauchy codes in [16] is $2p-3$ XORs, and the division over $\mathbb{F}_2[x]/(h(x))$ takes $2p$ XORs at most in [7]. Computing the division in (22) by the simplified Algorithm 2 takes $p-3$ XORs, which is less than that in [7], [16], [28].

Recall that, after dropping Steps 6, 7 and 8 in Algorithm 2, the output of the algorithm might be $c(x) + h(x)$ instead of $c(x)$; however, we will show that the data polynomials can be recovered after performing the proposed decoding algorithm no matter which algorithm is performed.

Theorem 9. *The proposed decoding algorithm outputs the same data polynomials no matter if Algorithm 2 or the simplified Algorithm 2 is performed.*

Proof. According to Algorithm 1, there are two steps (7 and 12) in it that involve (simplified) Algorithm 2. In addition, after encoding, $c(x)$ might become $c(x)+h(x)$ when we apply the simplified Algorithm 2 for encoding. When applying the simplified Algorithm 2 in the encoding process, the coded polynomials might be $c(x)+h(x)$ instead of $c(x)$. Hence, the input of Algorithm 1 becomes $c_1(x)+a_1h(x), c_2(x)+a_2h(x), \ldots, c_\ell(x)+a_\ell h(x)$, where $a_i \in \{0,1\}$ for $1 \le i \le \ell$. Note that since $h(x)$ is the check polynomial of C_p, from (3), we have

$$s(x)(c(x)+h(x)) = s(x)c(x) \mod (1+x^p) \tag{27}$$

$\forall s(x) \in C_p$ and $c(x) \in R_p$. Hence, after performing Step 5 in Algorithm 1, $s_j \in C_p$ for $2 \le j \le \ell$. However, after performing Step 7, $s_j(x)$ might become $s_j(x)+h(x)$ for $2 \le j \le \ell$ due to performing the simplified Algorithm 2. Again, after performing Step 9, the effect of $h(x)$ has been eliminated according to (27). A similar argument can be applied for performing Step 12, Step 15 (or Step 17) and Step 18. Hence, we can conclude that the output of Algorithm 1 becomes the same no matter whether Algorithm 2 or the simplified Algorithm 2 is applied. □

3) Computation complexity: Recall that the coded polynomial $c_j(x)$ is computed by

$$\frac{1}{x^j+x^r}s_0(x)+\cdots+\frac{1}{x^j+x^{k+r-1}}s_{k-1}(x), \tag{28}$$

for $j=0,1,\ldots,r-1$. With Lemma 8, we have that the last coefficient of $\frac{1}{x^j+x^{r+\ell}}s_\ell(x)$ is equal to 0, $\ell=0,1,\ldots,k-1$. Therefore, the last coefficient $c_{p-1,j}$ of $c_j(x)$ is equal to 0. Hence, there are $p-3$ XORs involved in computing $\frac{1}{x^j+x^{r+\ell}}s_\ell(x)$ by the simplified Algorithm 2.

Note that, in Algorithm 1, we only need to compute three different operations: (i) multiplication of $c_i(x)$ and x^{a_i}, (ii) division of the form $\frac{c_i(x)}{x^{a_j}+x^{b_{j-i}}}$ and (iii) addition between $c_i(x)$ and $c_j(x)$. Hence, in Algorithm 1, there are a total of $4\ell(\ell-1)+2\ell$ multiplications of the first type, $\ell(\ell-1)$ divisions of the second type and $\ell+3\ell(\ell-1)$ additions.

The multiplication of x^i and a polynomial $s(x)$ over R_p can be obtained by cyclically shifting the polynomial $s(x)$ by i bits, which takes no XORs. The second operation over R_p requires $p-3$ XORs when performing the simplified Algorithm 2. One addition needs p XORs. In Algorithm 1, Steps 4 and 5 are the computation of the right matrix of $\mathbf{L}_\ell^i$ and the column vector $\mathbf{c}$ of length ℓ, with each component being a polynomial in R_p, of which the complexity is at most $3(\ell-i)p$ XORs. In the resultant column vector $\mathbf{c}$, the first i components are in C_p and the last $\ell-i$ components are in C_p. Steps 6 to 7 calculate the left matrix of $\mathbf{L}_\ell^i$ and the above resultant column vector $\mathbf{c}$. As the last $\ell-i$ components are in C_p, all the divisions of the form $\frac{c_i(x)}{x^j+x^{j-i}}$ can be computed by the simplified Algorithm 2, which takes $(\ell-i)(p-3)$ XORs. Recall that the last bit of polynomial $\frac{c_i(x)}{x^j+x^{j-i}}$ is 0, and the multiplication of x^i+x^j and $\frac{c_i(x)}{x^j+x^{j-i}}$ thus requires $p-2$ XORs. Therefore, the total number of XORs involved in Steps 4 to 7 is

$$\underbrace{3p\ell(\ell-1)/2}_{\text{Steps 4 to 5}}+\underbrace{(p-3)\ell(\ell-1)/2}_{\text{Steps 6 to 7}}.$$

Steps 8 and 9 compute the multiplication of diagonal matrix $\mathbf{D}_\ell$ and the above resultant column vector, where the number of XORs involved are $p+(p-2)(\ell-1)$. Steps 11 and 12 compute multiplication of the right matrix of $\mathbf{U}_\ell^i$ and the above column vector, where $(p-3)\ell(\ell-1)/2$ XORs are required. Steps 13 to 18 calculate multiplication of the left matrix $\mathbf{U}_\ell^i$ and the above column vector, where $(2(p-2)+p)\ell(\ell-1)/2$ XORs are needed. Therefore, the total number of XORs involved in Algorithm 1 over R_p is at most

$$\underbrace{3p\ell(\ell-1)/2}_{\text{Steps 4 to 5}}+\underbrace{(p-3)\ell(\ell-1)/2}_{\text{Steps 6 to 7}}+\underbrace{p+(p-2)(\ell-1)}_{\text{Steps 8 to 9}}+$$
$$\underbrace{(p-3)\ell(\ell-1)/2}_{\text{Steps 11 to 12}}+\underbrace{(2(p-2)+p)\ell(\ell-1)/2}_{\text{Steps 13 to 18}}$$
$$=4\ell^2p-3\ell p-5\ell^2+3\ell+2.$$

Adding overall parity-checks to $k-\gamma$ data polynomials takes $(k-\gamma)(p-2)$ XORs. Computing γ polynomials in (20) requires $\gamma((k-\gamma)(p-3)+(k-\gamma)(p-1))=\gamma(k-\gamma)(2p-4)$ XORs. The number of XORs involved in solving the $\gamma\times\gamma$ Cauchy system is $4\gamma^2p-3\gamma p-5\gamma^2+3\gamma+2$. In recovering the δ parity columns, there are $\delta(k(p-3)+(k-1)(p-1))$ XORs involved. Therefore, the decoding complexity of recovering γ information erasures and δ parity erasures is

$$(k-\gamma)(p-2)+\gamma(k-\gamma)(2p-4)+4\gamma^2p-3\gamma p-5\gamma^2+$$
$$3\gamma+2+\delta(k(p-3)+(k-1)(p-1)) \text{ XORs.}$$

When $\delta=0$, i.e., only the information column fails, the decoding complexity is

$$(k-\gamma)(p-2)+\gamma(k-\gamma)(2p-4)+4\gamma^2p-3\gamma p-5\gamma^2+3\gamma+2 \text{ XORs.}$$

Although an LU factorization of the Cauchy matrix is also employed in the decoding of Rabin-like codes, the explicit expression is different from the LU factorization given in this paper and the calculations in solving the Cauchy matrix of the two codes are different. Using the LU factorization in this paper, we only need to take three different types of operations when solving the Cauchy linear system, which is more efficient than the decoding method of Rabin-like codes.

Example 2. *Continue from Example 1, by Theorem 7, the inverse matrix of the 2×2 Cauchy matrix can be factorized into*

$$\mathbf{U}_2^1\cdot\mathbf{D}_2\cdot\mathbf{L}_2^1=\begin{bmatrix}e(x) & 1+x^2\\ 0 & 1+x^3\end{bmatrix}\begin{bmatrix}e(x) & 0\\ 0 & \frac{1}{x^2+x^3}\end{bmatrix}\cdot$$
$$\begin{bmatrix}1+x^2 & 0\\ 0 & x+x^3\end{bmatrix}\cdot\begin{bmatrix}e(x) & 0\\ 0 & \frac{1}{1+x}\end{bmatrix}\begin{bmatrix}e(x) & 0\\ 1+x^2 & x+x^2\end{bmatrix}.$$

We can check that the two data polynomials can be recovered by

$$\mathbf{U}_2^1\cdot\mathbf{D}_2\cdot\mathbf{L}_2^1\cdot\begin{bmatrix}x\\ x+x^2+x^3\end{bmatrix}=\begin{bmatrix}1+x\\ x+x^3\end{bmatrix},$$

with 32 XORs involved.

V. Performance Comparisons

In this section, we evaluate the encoding/decoding complexities for $\mathcal{C}(k,r,p)$ as well as other existing Cauchy family array codes, such as Rabin-like codes [13], circulant Cauchy codes [16] and CRS codes [15], which are widely employed in many practical distributed storage systems such as Facebook data centers [36].

CRS codes are constructed by Cauchy matrices [37]. It uses projections that convert the operations of finite field multiplication into XORs. This leads to a reduction in the coding complexity because the standard RS algorithm [37] consumes most of the time over finite field multiplications. As the state-of-the-art works in correcting four or more erasures, Rabin-like codes, circulant Cauchy codes and CRS codes are used as the main comparison to the proposed codes. Note that the coding algorithm of the CRS codes involves Cauchy matrices, and it is hard to calculate the exact number of 1s in the Cauchy matrices. We run simulations for CRS codes and record the average numbers from simulations to estimate the encoding/decoding complexity.

We determine the *normalized encoding complexity* as the ratio of the encoding complexity to the number of information bits, and *normalized decoding complexity* as the ratio of the decoding complexity to the number of information bits.

A. Encoding Complexity

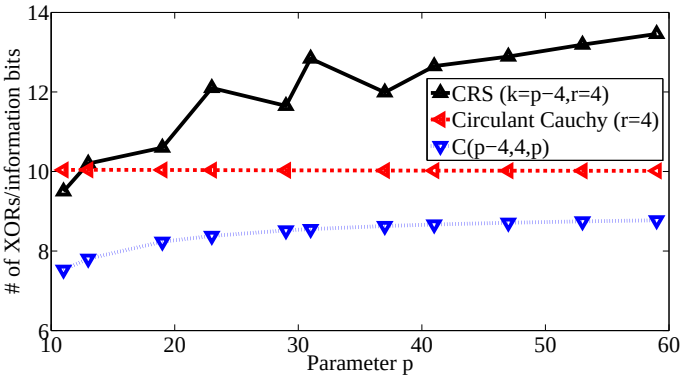
In the $p-1\times(k+r)$ array of code $\mathcal{C}(k,r,p)$, there are k information columns and r parity columns. First, we should compute a parity-check bit for each information column to obtain k data polynomials, with $k(p-2)$ XORs being involved. Second, we need to compute r coded polynomials by (12). There are $p-3$ XORs required to compute a division of the form x^t+x^{t+b} by the simplified Algorithm 2. Each coded polynomial is generated by computing k divisions of the form $1+x^b$ and $k-1$ additions. As the last coefficient is 0 (by Lemma 8), the $k-1$ additions take $(k-1)(p-1)$ XORs. Therefore, $k(p-3)+(k-1)(p-1)$ XORs are required to obtain a coded polynomial. The total number of XORs required for construction of r parity columns is $k(p-2)+r(2kp-4k-p+1)$, and the normalized encoding complexity is

$$\frac{k(p-2)+r(2kp-4k-p+1)}{k(p-1)}.$$

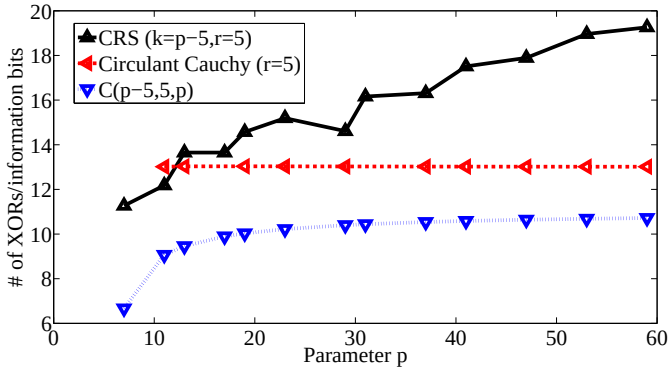
In Rabin-like code, a parity-check bit is computed for each information column and the one division involved in the encoding process takes $p-1$ XORs. The normalized encoding complexity can be computed as $\frac{k(p-2)+r(k(p-1)+(k-1)p)}{k(p-1)}$. When we compare the encoding complexity for Rabin-like code and $\mathcal{C}(k,r,p)$, we can see that $\mathcal{C}(k,r,p)$ has a slightly lower encoding complexity, as the division takes less XORs. The normalized encoding complexity circulant Cauchy codes is $3r-2+\frac{k-r}{k(p-1)}$ [16].

We compare the encoding complexity of circulant Cauchy codes, CRS codes and $\mathcal{C}(k,r,p)$ in the following. For fair comparison, we set $k=p-r$ for the three codes; we can thus have the normalized encoding complexity of the proposed $\mathcal{C}(p-r,r,p)$ as

$$\frac{(p-r)(p-2)+r(2p(p-r)-4(p-r)-p+1)}{(p-r)(p-1)}.$$



(a) $r=4$.



(b) $r=5$.

Fig. 2: The normalized encoding complexity.

The normalized encoding complexities of circulant Cauchy codes, CRS codes and $\mathcal{C}(p-r,r,p)$ for $r=4$ and $r=5$ are shown in Fig. 2. For all the values of parameter p, the encoding complexity of $\mathcal{C}(p-r,r,p)$ is less than that of circulant Cauchy codes and CRS codes. Note that the differences between the newly constructed Rabin-like codes and others becomes larger when r increases. When $r=4$, the reductions in the encoding complexity of $\mathcal{C}(p-4,4,p)$ over circulant Cauchy codes and CRS codes are 12.4%-22.3% and 23.6%-34.9%, respectively. When $r=5$, they increase to 17.7%-47.8% and 25.6%-44.4%, respectively. In general, the encoding complexity of $\mathcal{C}(p-r,r,p)$ is less than that of circulant Cauchy codes; the main reason is that the division in (22) of $\mathcal{C}(p-r,r,p)$ involves less XORs than that of circulant Cauchy codes.

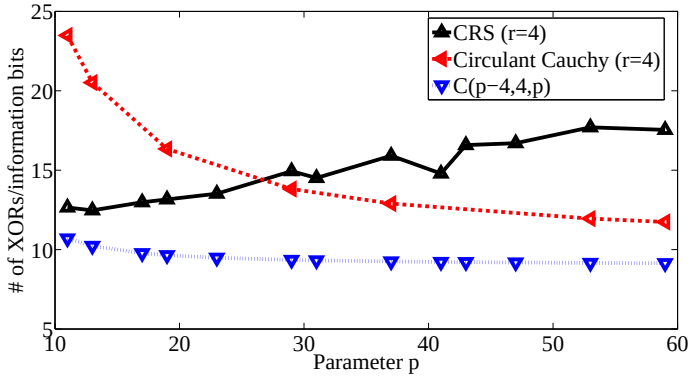
B. Decoding Complexity

As pointed out in [13], circulant Cauchy codes have a lower decoding complexity than Rabin-like codes. Therefore, we evaluate the decoding complexity of the proposed array codes $\mathcal{C}(k,r,p)$, CRS codes and circulant Cauchy codes in the following. If no information column fails, then the decoding procedure of parity column failure can be viewed as a special case of the encoding procedure. Hence, we only consider the case with at least one information column fail.

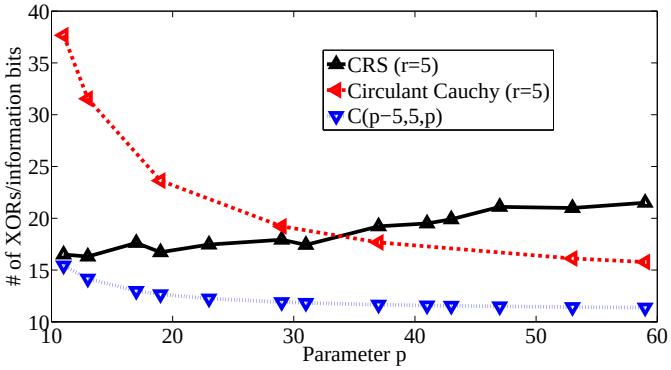
We let $k=p-r$ for the three codes, and we have the normalized decoding complexity of the newly constructed $\mathcal{C}(p-r,r,p)$ as

$$\frac{(p-2r)(p-2)+r(p-2r)(2p-4)+4r^2p-3rp-5r^2+3r+2}{(p-r)(p-1)}.$$

Schindelhauer and Ortolf [16] gave the normalized decoding complexity of circulant Cauchy codes as $\frac{3r(p-1)(p-r)+6r^2(p-1)-2(p-r)(p-1)-4r}{(p-r)(p-1)} = 3r - 2 + \frac{6r^2}{p-r} - \frac{4r}{(p-r)(p-1)}$.



(a) $r = 4$.



(b) $r = 5$.

Fig. 3: The normalized decoding complexity.

The normalized decoding complexities of $r = 4$ and $r = 5$ are shown in Fig. 3. We observe that the decoding complexity of CRS codes increases as p increases, and the decoding complexity of circulant Cauchy codes decreases while p increases, where r is fixed. However, the normalized decoding complexity of $\mathcal{C}(p-r, r, p)$ is almost the same for different values of p when r is constant. In general, the decoding complexity of $\mathcal{C}(p-r, r, p)$ is much less than that of CRS codes and circulant Cauchy codes, and the complexity difference between $\mathcal{C}(p-r, r, p)$ and CRS codes becomes larger when p increases. When $r = 4$, the percentage improvement over CRS codes and circulant Cauchy codes varies between 15.4% and 47.9%, and 22.1%-54.4%, respectively. When $r = 5$, the percentage improvement over CRS codes and circulant Cauchy codes varies between 6.5% and 47.1%, and 27.9%-59.0%, respectively.

$\mathcal{C}(p-r, r, p)$ has a much lower decoding complexity than that of circulant Cauchy codes, for two reasons. First, the division (22) of $\mathcal{C}(p-r, r, p)$ has less computational complexity. Second, an LU factorization of the Cauchy matrix is employed in the decoding of $\mathcal{C}(p-r, r, p)$, which is much more efficient than the decoding method of circulant Cauchy codes.

VI. Conclusions

We present a new construction for Rabin-like codes over a specific quotient ring that employ XOR and bit-wise cyclic shifts. These codes have been proved with the MDS property. This paper has two main contributions to the field of Cauchy MDS array codes. First, the word size $p-1$ is extended to be more flexible, i.e., p should be an odd number and all divisors of p except 1 are strictly larger than $k+r-1$. The extension is important since it allows coding strips to fit evenly within file system blocks. Second, the newly constructed Rabin-like code improve the decoding complexity over existing codes. An efficient decoding algorithm based on the LU factorization of Cauchy matrix is proposed to reduce the decoding complexity.

We conclude with proposed future work. In the constructed array codes, the parameter p is restricted to being an odd number. It would be interesting to investigate whether there exist MDS Cauchy array codes without this restriction. When a single column fails, the total number of bits downloaded from the surviving columns is termed a repair bandwidth. It would also be interesting to study how to recover the failed column with a repair bandwidth that is as low as possible.

References

[1] D. A. Patterson, P. Chen, G. Gibson, and R. H. Katz, "Introduction to Redundant Arrays of Inexpensive Disks (RAID)," in *Proc. IEEE COMPCON*, vol. 89, 1989, pp. 112–117.

[2] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: high-performance, reliable secondary storage," University of California at Berkeley, Berkeley, Tech. Rep. CSD 03-778, 1993.

[3] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-diagonal parity for double disk failure correction," in *Proc. of the 3rd USENIX Conf. on File and Storage Technologies (FAST)*, 2004, pp. 1–14.

[4] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures," *IEEE Trans. Computers*, vol. 44, no. 2, pp. 192–202, 1995.

[5] J. S. Plank, "The RAID-6 liber8tion code." *International Journal of High Performance Computing Applications*, vol. 23, no. 3, pp. 242–251, 2009.

[6] A. H. Leventhal, "Triple-parity RAID and beyond," *Comm. of the ACM*, vol. 53, no. 1, pp. 58–63, January 2010.

[7] M. Blaum, J. Bruck, and A. Vardy, "MDS array codes with independent parity symbols," *IEEE Trans. Information Theory*, vol. 42, no. 2, pp. 529–542, 1996.

[8] L. Xiang, Y. Xu, J. Lui, and Q. Chang, "Optimal recovery of single disk failure in RDP code storage systems," in *ACM SIGMETRICS Performance Evaluation Rev.*, vol. 38, no. 1. ACM, 2010, pp. 119–130.

[9] M. Blaum, J. Brady, J. Bruck, J. Menon, and A. Vardy, "The EVENODD code and its generalization," *High Performance Mass Storage and Parallel I/O*, pp. 187–208, 2001.

[10] C. Huang and L. Xu, "STAR: An efficient coding scheme for correcting triple storage node failures," *IEEE Trans. Computers*, vol. 57, no. 7, pp. 889–901, 2008.

[11] Y. Wang, G. Li, and X. Zhong, "Triple-Star: A coding scheme with optimal encoding complexity for tolerating triple disk failures in RAID," *International Journal of innovative Computing, Information and Control*, vol. 3, pp. 1731–1472, 2012.

[12] M. Blaum, "A family of MDS array codes with minimal number of encoding operations," in *IEEE Int. Symp. on Inf. Theory*, 2006, pp. 2784–2788.

[13] G.-L. Feng, R. H. Deng, F. Bao, and J.-C. Shen, "New efficient MDS array codes for RAID. Part II. Rabin-like codes for tolerating multiple (≥ 4) disk failures," *IEEE Trans. Computers*, vol. 54, no. 12, pp. 1473–1483, 2005.

[14] M. Blaum, J. Brady, J. Bruck, J. Menon, and A. Vardy, "The EVENODD code and its generalization: An effcient scheme for tolerating multiple disk failures in RAID architectures," in *High Performance Mass Storage and Parallel I/O*. Wiley-IEEE Press, 2002, ch. 8, pp. 187–208.

[15] J. Blomer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman, "An XOR-based erasure-resilient coding scheme," *Proc ACM Sigcomm*, 1999.
[16] C. Schindelhauer and C. Ortolf, "Maximum distance separable codes based on circulant Cauchy matrices," in *Structural Information and Communication Complexity*. Springer, 2013, pp. 334–345.
[17] J. S. Plank and L. Xu, "Optimizing Cauchy Reed-Solomon codes for fault-tolerant network storage applications." in *IEEE International Symposium on Network Computing and Applications*, 2006, pp. 173–180.
[18] M. Blaum and R. M. Roth, "On lowest density MDS codes," *IEEE Transactions on Information Theory*, vol. 45, no. 1, pp. 46–59, 1999.
[19] J. S. Plank, J. Luo, C. D. Schuman, L. Xu, and Z. Wilcox-O'Hearn, "A performance evaluation and examination of open-source erasure coding libraries for storage," in *Proccedings of Conference on File & Storage Technologies*, 2009, pp. 253–265.
[20] J. S. Plank, S. Simmerman, and C. D. Schuman, "Jerasure: A library in C/C++ facilitating erasure coding for storage applications-version 1.2," *University of Tennessee, Tech. Rep. CS-08-627*, vol. 23, 2008.
[21] C. Huang, J. Li, and M. Chen, "On optimizing XOR-based codes for fault-tolerant storage applications," *ITW07 Information Theory Workshop IEEE*, pp. 218–223, 2005.
[22] J. S. Plank, "XOR's, lower bounds and MDS codes for storage," *ITW07 Information Theory Workshop IEEE*, pp. 503 – 507, 2011.
[23] J. L. Hafner, V. Deenadhayalan, K. K. Rao, and J. A. Tomlin, "Matrix methods for lost data reconstruction in erasure codes," in *Conference on Usenix Conference on File & Storage Technologies-volume*, 2005, pp. 183–196.
[24] C. Yin, J. Wang, H. Lv, Z. Cui, L. Cheng, Q. Zhan, and T. Li, "Acoustic emission testing research of composites bearing based on neural network," in *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2011 International Conference on*, 2011, pp. 165–168.
[25] J. S. Plank, C. D. Schuman, and B. D. Robison, "Heuristics for optimizing matrix-based erasure codes for fault-tolerant storage systems," in *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2012, pp. 1–12.
[26] G. Zhang, G. Wu, S. Wang, and J. Shu, "CaCo: An efficient cauchy coding approach for cloud storage systems," *IEEE Transactions on Computers*, pp. 1–13, 2015.
[27] M. Blaum and R. M. Roth, "New array codes for multiple phased burst correction," *IEEE Trans. Information Theory*, vol. 39, no. 1, pp. 66–77, January 1993.
[28] H. Hou, K. W. Shum, M. Chen, and H. Li, "BASIC codes: Low-complexity regenerating codes for distributed storage systems," *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 3053–3069, 2016.
[29] J. H. Silverman, "Fast multiplication in finite fields GF(2^n)," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 1999, pp. 122–134.
[30] K. W. Shum, H. Hou, M. Chen, H. Xu, and H. Li, "Regenerating codes over a binary cyclic code," in *Proc. IEEE Int. Symp. Inf. Theory*, Honolulu, July 2014, pp. 1046–1050.
[31] S. T. J. Fenn, M. G. Parker, M. Benaissa, and D. Taylor, "Bit-serial multiplication in GF(2^m) using irreducible all-one polynomials," *IEEE Proceedings on Computers and Digital Techniques*, vol. 144, no. 6, pp. 391–393, 1997.
[32] C. Schindelhauer, A. Jakoby, and S. Koehler, "Cyclone codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Aachen, June 2017, pp. 156–160.
[33] I. Gohberg, V. Olshevsky, and T. Kailath, "Fast Gaussian elimination with partial pivoting for matrices with displacement structure," *Mathematics of Computation*, vol. 64, no. 212, pp. 1557–1576, 1995.
[34] T. Boros, T. Kailath, and V. Olshevsky, "A fast parallel Björck–Pereyra-type algorithm for solving Cauchy linear equations," *Linear Algebra and Its Applications*, vol. 302, pp. 265–293, 1999.
[35] D. Calvetti and L. Reichel, "Factorizations of Cauchy matrices," *Journal of Computational and Applied Mathematics*, vol. 86, no. 1, pp. 103–123, 1997.
[36] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "XORing elephants: Novel erasure codes for big data," in *Proc. of the 39th Int. Conf. on Very Large Data Bases*, Trento, August 2013.
[37] J. S. Plank *et al.*, "A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems," *Softw., Pract. Exper.*, vol. 27, no. 9, pp. 995–1012, 1997.

Hanxu Hou (S'11-M'16) was born in Anhui, China, 1987. He received the B.Eng. degree in Information Security from Xidian University, Xian, China, in 2010, and Ph.D. degrees in the Dept. of Information Engineering from The Chinese University of Hong Kong in 2015 and in the School of Electronic and Computer Engineering, Peking University. He is now an Assistant Professor with the School of Electrical Engineering & Intelligentization, Dongguan University of Technology and part-time Research Fellow with the Shenzhen Key Lab of Information Theory & Future Internet Architecture, Future Network PKU Lab of National Major Research Infrastructure, Peking University Shenzhen Graduate School. His research interests include erasure coding and coding for distributed storage systems.

Yunghsiang S. Han (S'90-M'93-SM'08-F'11) was born in Taipei, Taiwan, 1962. He received B.Sc. and M.Sc. degrees in electrical engineering from the National Tsing Hua University, Hsinchu, Taiwan, in 1984 and 1986, respectively, and a Ph.D. degree from the School of Computer and Information Science, Syracuse University, Syracuse, NY, in 1993. He was from 1986 to 1988 a lecturer at Ming-Hsin Engineering College, Hsinchu, Taiwan. He was a teaching assistant from 1989 to 1992, and a research associate in the School of Computer and Information Science, Syracuse University from 1992 to 1993. He was, from 1993 to 1997, an Associate Professor in the Department of Electronic Engineering at Hua Fan College of Humanities and Technology, Taipei Hsien, Taiwan. He was with the Department of Computer Science and Information Engineering at National Chi Nan University, Nantou, Taiwan from 1997 to 2004. He was promoted to Professor in 1998. He was a visiting scholar in the Department of Electrical Engineering at University of Hawaii at Manoa, HI from June to October 2001, the SUPRIA visiting research scholar in the Department of Electrical Engineering and Computer Science and CASE center at Syracuse University, NY from September 2002 to January 2004 and July 2012 to June 2013, and the visiting scholar in the Department of Electrical and Computer Engineering at University of Texas at Austin, TX from August 2008 to June 2009. He was with the Graduate Institute of Communication Engineering at National Taipei University, Taipei, Taiwan from August 2004 to July 2010. From August 2010 to January 2017, he was with the Department of Electrical Engineering at National Taiwan University of Science and Technology as Chair Professor. Now he is with School of Electrical Engineering & Intelligentization at Dongguan University of Technology, China. He is also a Chair Professor at National Taipei University from February 2015. His research interests are in error-control coding, wireless networks, and security.

Dr. Han was a winner of the 1994 Syracuse University Doctoral Prize and a Fellow of IEEE. One of his papers won the prestigious 2013 ACM CCS Test-of-Time Award in cybersecurity.