Binary MDS Array Codes with Optimal Repair

Hanxu Hou and Patrick P. C. Lee

***Abstract*— Consider a binary maximum distance separable (MDS) array code composed of an $m \times (k+r)$ array of bits with k information columns and r parity columns, such that any k out of $k+r$ columns suffice to reconstruct the k information columns. Our goal is to provide *optimal repair access* for binary MDS array codes, meaning that the bandwidth triggered to repair any single failed information or parity column is minimized. In this paper, we propose a generic transformation framework for binary MDS array codes, using EVENODD codes as a motivating example, to support optimal repair access for $k+1 \leq d \leq k+r-1$, where d denotes the number of non-failed columns that are connected for repair; note that when $d < k+r-1$, some of the chosen d columns in repairing a failed column are specific. In addition, we show how our transformation framework applies to an example of binary MDS array codes with asymptotically optimal repair access of any single information column and enables asymptotically or exactly optimal repair access for any column. Furthermore, we present a new transformation for EVENODD codes with two parity columns such that the existing efficient repair property of any information column is preserved and the repair access of parity column is optimal.**

***Index Terms*—Binary MDS array codes, EVENODD codes, repair bandwidth, repair access.**

I. Introduction

Large-scale storage systems typically introduce redundancy into data storage to provide fault tolerance and maintain storage reliability. Erasure coding is a redundancy technique that significantly achieves higher reliability than replication at the same storage overhead [1], and has been widely adopted in commercial storage systems [2], [3]. One important class of erasure codes is *maximum distance separable (MDS)* codes, which achieve the maximum reliability for a given amount of redundancy. Specifically, an MDS code transforms k information symbols into $k+r$ encoded symbols of the same size for some configurable parameters k and r, such that any k out of $k+r$ symbols are sufficient to retrieve all k information symbols. Reed-Solomon (RS) codes [4] are one well-known example of MDS codes.

In this paper, we examine a special class of MDS codes called *binary MDS array codes*, which have low computational complexity since the encoding and decoding procedures only involve XOR operations. Examples of binary MDS array codes are EVENODD [5]–[7], X-code [8], and RDP [9], [10]. Specifically, we consider a binary MDS array code that is composed of an array of size $m \times (k+r)$, where each element in the array is a bit. In this work, we assume that the code is *systematic*, meaning that k columns are *information columns* that store information bits, and the remaining r columns are *parity columns* that store parity bits encoded from the k information columns. The code is MDS, meaning that any k out of $k+r$ columns can reconstruct all the original k information columns. We distribute the $k+r$ columns across $k+r$ distinct storage nodes, such that the bits in each column are stored in the same node. We use the terms "column" and "node" interchangeably in this paper.

In large-scale storage systems, node failures are common and the majority of all failures are single node failures [11]. Thus, it is critical to design an efficient repair scheme for repairing the lost bits of a single failed node, while providing fault tolerance for multiple node failures. The problem of repairing a single node failure was first formulated by Dimakis *et al.* [12], in which it is shown that the amount of symbols downloaded for repairing a single node failure (called the *repair bandwidth*) of an $m \times (k+r)$ MDS array code is at least (in units of bits):

$$\frac{dm}{d-k+1}, \tag{1}$$

where d ($k \leq d \leq k+r-1$) is the number of nodes connected to repairing the failed node. Many constructions [13]–[17] of MDS array codes have been proposed to achieve the optimal repair bandwidth in (1). If the repair bandwidth of a binary MDS array code achieves the optimal value in (1), we say that the code has optimal repair bandwidth. If the repair does not require any arithmetic operations on the d connected nodes, then the repair is called *uncoded repair*. A binary MDS array code is said to achieve *optimal repair access* if the repair bandwidth is (1) with uncoded repair.

A. Related Work

There are many related studies on binary MDS array codes along different directions, such as new constructions [6], [7], [18]–[20], efficient decoding methods [21]–[26] and the improvement of the repair problem [27]–[33].

In particular, EVENODD is well explored in the literature, and has been extended to STAR codes [21] with three parity columns and generalized EVENODD [6], [7] with more parity columns. The computational complexity of EVENODD is optimized in [20] by a new construction. A sufficient condition for the generalized EVENODD to be MDS with more than eight parity columns is given in [34].

RDP is another important class of binary MDS array codes with two parity columns. It is extended to RTP codes [18] to tolerate three column failures. Blaum [10] generalized RDP to correct more than three column failures and showed that the generalized EVENODD and generalized RDP share the same MDS property condition. The authors in [25] proposed a

Hanxu Hou is with the School of Electrical Engineering & Intelligentization, Dongguan University of Technology and the Department of Computer Science and Engineering, The Chinese University of Hong Kong (E-mail: houhanxu@163.com). Patrick P. C. Lee is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong (E-mail: pclee@cse.cuhk.edu.hk). This work was partially supported by the National Natural Science Foundation of China (No. 61701115) and Research Grants Council of Hong Kong (GRF 14216316 and CRF C7036-15G).

unified form of generalized EVENODD and generalized RDP, and presented an efficient decoding method for some patterns of failures.

The above constructions are based on the Vandermonde matrix. Some constructions of binary MDS array codes based on Cauchy matrix are Cauchy Reed-Solomon codes [35], Rabin-like codes [26], [36] and circulant Cauchy codes [37].

Most of the decoding methods focus on generalized EVENODD [21], [22] and generalized RDP [18], [24] with three parity columns. The study [25] shows an efficient erasure decoding method based on the LU factorization of Vandermonde matrix for EVENODD and RDP with more than two parity columns.

There have been many studies [27], [28], [31]–[33], [38]–[42] on the repair problem of binary MDS array codes. Some optimal repair schemes reduce I/O for RDP [28], X-code [38] and EVENODD [27] by approximately 25%, but the repair bandwidth is sub-optimal. ButterFly codes [41], [42] are binary MDS array codes with optimal repair for information column failures, but only has two parity columns (i.e., $r=2$). MDR codes [39], [40] are constructed with $r = 2$ and have optimal repair bandwidth for k information columns and one parity column. Binary MDS array codes with more than two parity columns are proposed in [31]–[33]; however, the repair bandwidth is asymptotically optimal and the d helper columns are specifically selected.

B. Contributions

The contributions of this paper are summarized as follows.

1) First, we propose a generic transformation for an $m \times (k + r)$ EVENODD code. The transformed EVENODD code is of size $m(d - k + 1) \times (k + r)$ and has three properties: (1) the transformed EVENODD code achieves optimal repair access for the chosen $d - k + 1$ columns; (2) the property of optimal repair access for the chosen $d - k + 1$ columns of the transformed EVENODD code is preserved if we apply the transformation once more for the transformed EVENODD code; and (3) the transformed EVENODD code is MDS.
2) Second, we present a family of $m(d - k + 1)^{\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil} \times (k + r)$ multi-layer transformed EVENODD codes with $r \geq 2$, such that it achieves optimal repair access for all columns based on the EVENODD transformation, where $k+1 \leq d \leq k+r-1$. Some of the d helper columns need to be specifically selected.
3) Third, the efficient decoding method of the original EVENODD code is also applicable to the proposed family of multi-layer transformed EVENODD codes.
4) Lastly, the other binary MDS array codes, such as RDP [10] and codes in [26], [31]–[33], [35]–[37], can also be transformed to achieve optimal repair access and the efficient decoding methods of the original binary MDS array codes are maintained in the transformed codes. By applying the transformation with well-chosen encoding coefficients for an example of binary MDS array codes [33] that have asymptotically optimal repair access for any information column, we show that the obtained transformed codes have asymptotically optimal repair access for any information column and optimal repair access for any parity column. We also show how to design a transformation for EVENODD codes with two parity columns such that the transformed codes have optimal repair access for any single parity column and the repair access of any single information column of the transformed codes is roughly $3/4$ of all the information bits.

A closely related work to ours is [16], which also proposes a transformation for non-binary MDS codes to enable optimal repair access. The main differences between the work in [16] and ours are two-fold. First, our transformation is designed for binary MDS array codes, while the transformation in [16] is designed for non-binary MDS codes. The minimum operation unit of our transformed codes is a bit, so that we can carefully choose the encoding coefficients of the transformation to combine the efficient repair property of existing or newly designed binary MDS array codes for any single information column as well as the optimal repair of the transformed codes for any parity column. In contrast, the minimum operation unit of the transformation [16] for non-binary MDS codes is a field element, so we cannot directly apply the transformation [16] for binary MDS array codes. Even though we can view each column of some binary MDS array codes (such as EVENODD codes with p being a special prime number [6]) as a field element, if we apply the transformation [16] for such binary MDS array codes, the efficient repair property of such binary MDS array codes for any single information column cannot be maintained, as the efficient repair property of binary MDS array codes is achieved by downloading some bits from the chosen columns but not all the bits (field element) from the chosen columns. We illustrate the transformation of an example of binary MDS array codes [33] with asymptotically optimal repair access for any single information column to obtain the transformed array codes that have asymptotically optimal repair for any information column and optimal repair access for any parity column in Section IV. We also design a new transformation for EVENODD codes with $r = 2$ parity columns such that the repair access of any single information column of the transformed codes is roughly $3/4$ of all the information bits and the repair access of each parity column is optimal in Section IV-B. Second, our work allows a more flexible number of nodes connected for repairing the failed node. In particular, our work allows $k + 1 \leq d \leq k + r - 1$, while the work in [16] requires $d = k + r - 1$.

II. Transformation of EVENODD Codes

We first review the definition of EVENODD codes. We then present our transformation approach.

A. Review of EVENODD Codes

An EVENODD code is an array code of size $(p-1)\times(k+r)$, where p is a prime number with $p \geq \max\{k, r\}$. Given the $(p - 1) \times (k + r)$ array $[a_{i,j}]$ for $i = 0, 1, \ldots, p - 2$ and

$j = 0, 1, \ldots, k+r-1$, the $p-1$ bits $a_{0,j}, a_{1,j}, \ldots, a_{p-2,j}$ in column j can be represented as a polynomial

$$a_j(x) = a_{0,j} + a_{1,j}x + \cdots + a_{p-2,j}x^{p-2}.$$

Without loss of generality, we store the information bits in the k leftmost columns and the parity bits in the remaining r columns. The first k polynomials $a_0(x), \ldots, a_{k-1}(x)$ are called *information polynomials*, and the last r polynomials $a_k, \ldots, a_{k+r-1}(x)$ are *parity polynomials*. The r parity polynomials are computed as

$$\begin{bmatrix} a_k(x) & \cdots & a_{k+r-1}(x) \end{bmatrix} = \begin{bmatrix} a_0(x) & \cdots & a_{k-1}(x) \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & x & \cdots & x^{r-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x^{k-1} & \cdots & x^{(r-1)(k-1)} \end{bmatrix} \quad (2)$$

over the ring $\mathbb{F}_2[x]/(1 + x + \cdots + x^{p-1})$. The matrix on the right-hand side of (2) is called the *encoding matrix*.

B. The Transformation

We will present the transformation that can convert a $(p-1) \times (k+r)$ EVENODD code into a $(p-1)(d-k+1) \times (k+r)$ transformed code with optimal repair access for any chosen $d-k+1$ columns, where $k+1 \leq d \leq k+r-1$. For the ease of presentation, we assume that the chosen $d-k+1$ columns are the first $d-k+1$ columns in the following discussion.

1) The First Transformation: Given the codewords of a $(p-1) \times (k+r)$ EVENODD code $a_0(x), \ldots, a_{k+r-1}(x)$, we first generate $d-k+1$ instances $a_{0,\ell}(x), \ldots, a_{k+r-1,\ell}(x)$ for $\ell = 0, 1, \ldots, d-k$. Specifically, the r parity polynomials $a_{k,\ell}(x), \ldots, a_{k+r-1,\ell}(x)$ are computed by the multiplication of $[a_{0,\ell}(x), \ldots, a_{k-1,\ell}(x)]$ and the encoding matrix in (2), where $\ell = 0, 1, \ldots, d-k$. For $i = 0, 1, \ldots, d-k$, the polynomials in column i are

$$\begin{array}{l} a_{i,0}(x) + a_{0,i}(x), a_{i,1}(x) + a_{1,i}(x), \ldots, a_{i,i-1}(x) + a_{i-1,i}(x), \\ a_{i,i}(x), a_{i,i+1}(x) + (1+x^e)a_{i+1,i}(x), \\ a_{i,i+2}(x) + (1+x^e)a_{i+2,i}(x), \ldots, a_{i,d-k}(x) + (1+x^e)a_{d-k,i}(x), \end{array} \quad (3)$$

where e is a positive integer with $1 \leq e \leq p-1$. On the other hand, for $i = d-k+1, \ldots, k+r-1$, the polynomials in column i are

$$a_{i,0}(x), a_{i,1}(x), \ldots, a_{i,d-k}(x).$$

The above transformation is called *the first transformation* and the obtained codes are called transformed EVENODD codes. Each column of the transformed EVENODD codes has $d-k+1$ polynomials. Table I shows an example of the first transformed EVENODD code with $k = 4$, $r = 2$, $d = 5$ and $e = 1$.

Remark. For $i < j \in \{0, 1, \ldots, d-k\}$, columns i and j contain the following two polynomials

$$a_{i,j}(x) + (1+x^e)a_{j,i}(x), a_{j,i}(x) + a_{i,j}(x).$$

We can solve $x^e a_{j,i}(x)$ by summing the above two polynomials. Then, we can obtain $a_{j,i}(x)$ by multiplying $x^e a_{j,i}(x)$ by x^{p-e}, and $a_{i,j}(x)$ by summing $a_{j,i}(x) + a_{i,j}(x)$ and $a_{j,i}(x)$. Therefore, we can solve two information polynomials $a_{j,i}(x), a_{i,j}(x)$ from columns i and j. If ℓ columns are chosen that are in the first $d-k+1$ columns, then we can solve $\ell(\ell-1)$ information polynomials from the chosen ℓ columns, where $\ell = 2, 3, \ldots, d-k+1$.

2) The Second Transformation: Note that the above transformed code is a non-systematic code. To obtain the systematic code, we can first replace $a_{i,\ell}(x) + a_{\ell,i}(x)$ by $a'_{i,\ell}(x)$ and replace $a_{\ell,i}(x) + (1+x^e)a_{i,\ell}(x)$ by $a'_{\ell,i}(x)$ for $\ell < i$, to obtain that

$$\begin{cases} a_{i,\ell}(x) = x^{p-e}a'_{i,\ell}(x) + x^{p-e}a'_{\ell,i}(x), \\ a_{\ell,i}(x) = (1+x^{p-e})a'_{i,\ell}(x) + x^{p-e}a'_{\ell,i}(x). \end{cases} \quad (4)$$

Then, we can show the equivalent systematic transformed code as follows. For $i = 0, 1, \ldots, k-1$, the $d-k+1$ polynomials in column i are $a_{i,\ell}(x)$ for $\ell = 0, 1, \ldots, d-k$. Recall that the polynomial $a_{i,\ell}(x)$ is computed by

$$a_{i,\ell}(x) = \sum_{j=0}^{k-1} x^{j(i-k)} a_{j,\ell}(x)$$

for $i = k, k+1, \ldots, k+r-1$ and $\ell = 0, 1, \ldots, d-k$. We update $r(d-k+1)$ polynomials $a_{i,\ell}(x)$ for $i = k, k+1, \ldots, k+r-1$ and $\ell = 0, 1, \ldots, d-k$, by replacing the component $x^{j(i-k)}a_{j,\ell}(x)$ of $a_{i,\ell}(x)$ by $x^{j(i-k)}(x^{p-e}a_{j,\ell}(x) + (1+x^{p-e})a_{\ell,j}(x))$ for $j < \ell$, and replacing the component $x^{j(i-k)}a_{j,\ell}(x)$ of $a_{i,\ell}(x)$ by $x^{j(i-k)}(x^{p-e}a_{j,\ell}(x) + x^{p-e}a_{\ell,j}(x))$ for $j > \ell$, i.e.,

$$a_{i,\ell}(x) = \left(\sum_{j=0}^{\ell-1} x^{j(i-k)}(x^{p-e}a_{j,\ell}(x) + (1+x^{p-e})a_{\ell,j}(x)) \right) + x^{\ell(i-k)}a_{\ell,\ell}(x) + \left(\sum_{j=\ell+1}^{d-k} x^{j(i-k)}(x^{p-e}a_{j,\ell}(x) + x^{p-e}a_{\ell,j}(x)) \right) + \left(\sum_{j=d-k+1}^{k-1} x^{j(i-k)}a_{j,\ell}(x) \right).$$

The $d-k+1$ polynomials in column i for $i = k, k+1, \ldots, k+r-1$ are $a_{i,\ell}(x)$ for $\ell = 0, 1, \ldots, d-k$. In other words, the $d-k+1$ polynomials in column i for $i = k, k+1, \ldots, k+r-1$ are $a_{i,0}(x), \ldots, a_{i,d-k}(x)$, which are computed by (5) (in the next page) over the ring $\mathbb{F}_2[x]/(1 + x + \cdots + x^{p-1})$. The above transformation is called *the second transformation*. The transformed EVENODD code is denoted by $\mathsf{EVENODD}_1$. Note that each column of EVENODD codes has one polynomial, and each column of $\mathsf{EVENODD}_1$ obtained by applying the transformation for EVENODD codes has $d-k+1$ polynomials.

When $k = 4$, $r = 2$, $d = 5$ and $e = 1$, the $\mathsf{EVENODD}_1$ code with the second transformation is shown in Table II. We claim that we can recover all the information polynomials from any four columns. We can obtain the information polynomials from columns 0, 1, 2 and 3 directly. Consider that we want to recover the information polynomials from one parity column and three information columns, say columns 0, 2, 3 and 4. We can obtain $a_{1,0}(x)$ by

$$x(a_{4,0}(x) + a_{0,0}(x) + x^{p-1}a_{0,1}(x) + a_{2,0}(x) + a_{3,0}(x)),$$

TABLE I: The first transformation for EVENODD codes with $k=4$, $r=2$, $d=5$ and $e=1$.

Column 0	Column 1	Column 2	Column 3	Column 4	Column 5
$a_{0,0}(x)$	$a_{1,0}(x)+$ $a_{0,1}(x)$	$a_{2,0}(x)$	$a_{3,0}(x)$	$a_{4,0}(x)=a_{0,0}(x)+a_{1,0}(x)+$ $a_{2,0}(x)+a_{3,0}(x)$	$a_{5,0}(x)=a_{0,0}(x)+xa_{1,0}(x)+$ $x^2a_{2,0}(x)+x^3a_{3,0}(x)$
$a_{0,1}(x)+$ $(1+x)a_{1,0}(x)$	$a_{1,1}(x)$	$a_{2,1}(x)$	$a_{3,1}(x)$	$a_{4,1}(x)=a_{0,1}(x)+a_{1,1}(x)+$ $a_{2,1}(x)+a_{3,1}(x)$	$a_{5,1}(x)=a_{0,1}(x)+xa_{1,1}(x)+$ $x^2a_{2,1}(x)+x^3a_{3,1}(x)$

$$\begin{bmatrix}1 & x^{i-k} & \cdots & x^{(k-1)(i-k)}\end{bmatrix}\cdot \begin{bmatrix} a_{0,0}(x) & x^{p-e}a_{0,1}(x)+(1+x^{p-e})a_{1,0}(x) & \cdots & x^{p-e}a_{0,d-k}(x)+(1+x^{p-e})a_{d-k,0}(x)\\ x^{p-e}(a_{1,0}(x)+a_{0,1}(x)) & a_{1,1}(x) & \cdots & x^{p-e}a_{1,d-k}(x)+(1+x^{p-e})a_{d-k,1}(x)\\ \vdots & \vdots & \ddots & \vdots\\ x^{p-e}(a_{d-k,0}(x)+a_{0,d-k}(x)) & x^{p-e}(a_{d-k,1}(x)+a_{1,d-k}(x)) & \cdots & a_{d-k,d-k}(x)\\ a_{d-k+1,0}(x) & a_{d-k+1,1}(x) & \cdots & a_{d-k+1,d-k}(x)\\ \vdots & \vdots & \ddots & \vdots\\ a_{k-1,0}(x) & a_{k-1,1}(x) & \cdots & a_{k-1,d-k}(x)\end{bmatrix}. \tag{5}$$

and $a_{1,1}(x)$ by

$$a_{4,1}(x)+x^{p-1}a_{0,1}(x)+(1+x^{p-1})a_{1,0}(x)+a_{2,1}(x)+a_{3,1}(x).$$

Suppose that we want to solve the information polynomials from two information columns and two parity columns, say columns 1, 2, 4 and 5. First, we compute the following two polynomials by subtracting $a_{1,1}(x), a_{2,1}(x)$ from $a_{4,1}(x), a_{5,1}(x)$,

$$\begin{aligned} p_1(x) =& a_{4,1}(x)+(1+x^{p-1})a_{1,0}(x)+a_{1,1}(x)+a_{2,1}(x)\\ =& x^{p-1}a_{0,1}(x)+a_{3,1}(x),\\ p_2(x) =& a_{5,1}(x)+(1+x^{p-1})a_{1,0}(x)+xa_{1,1}(x)+x^2a_{2,1}(x)\\ =& x^{p-1}a_{0,1}(x)+x^3a_{3,1}(x).\end{aligned}$$

Then, we can solve $a_{3,1}(x)$ by $\frac{p_1(x)+p_2(x)}{1+x^3}$,[1] and $a_{0,1}(x)$ by $x(a_{3,1}(x)+p_1(x))$. The other two information polynomials $a_{0,0}(x), a_{3,0}(x)$ can be solved similarly.

The repair access of each of the first two columns is optimal. Suppose that the first column fails. We can first solve $a_{0,0}(x)$ and $x^{p-1}(a_{1,0}(x)+a_{0,1}(x))$ by accessing four polynomials $a_{2,0}(x), a_{3,0}(x), a_{4,0}(x), a_{5,0}(x)$ due to the MDS property of EVENODD codes, and then recover $a_{0,1}(x)$ by computing $x(x^{p-1}(a_{1,0}(x)+a_{0,1}(x))+x^{p-1}a_{1,0}(x))$. Therefore, we can recover two information polynomials by downloading five polynomials from five helper columns, and the repair bandwidth achieves the minimum value in (1). The repair of the second column is similar.

C. Properties of Transformed EVENODD Codes

The next theorem shows that the second transformed EVENODD code is also MDS code.

Theorem 1. *If the $(k+r,k)$ EVENODD code is MDS code, then the second transformed EVENODD code is also MDS.*

Proof. The code is an MDS code if any k out of $k+r$ columns can retrieve all information bits. It is equivalent to show that the k information columns can be reconstructed from any t information columns and any $k-t$ parity columns, where $\max\{0,k-r\}\le t\le k$. When $t=k$, we can obtain the k information columns directly.

In the following, we consider the case of $t<k$. Suppose that columns $i_1,i_2,\ldots,i_t$ and columns $j_1,j_2,\ldots,j_{k-t}$ are connected with $0\le i_1<\ldots<i_t\le k-1$ and $k\le j_1<\cdots<j_{k-t}\le k+r-1$. We need to recover $k-t$ information columns $e_1,e_2,\ldots,e_{k-t}$, where

$$e_1<e_2<\cdots<e_{k-t}\in\{0,1,\ldots,k-1\}\setminus\{i_1,i_2,\ldots,i_t\}.$$

Recall that we can obtain $t(d-k+1)$ information polynomials $a_{i_1,\ell}(x),\ldots,a_{i_t,\ell}(x)$ and $(k-t)(d-k+1)$ parity polynomials $a_{j_1,\ell}(x),\ldots,a_{j_{k-t},\ell}(x)$ from the connected columns, where $\ell=0,1,\ldots,d-k$.

We divide the proof into two cases: $i_1<d-k$ and $i_1\ge d-k$. We first assume that $i_1<d-k$. By subtracting $t(d-k+1)$ information polynomials from $k-t$ parity polynomials $a_{j_1,i_1}(x),\ldots,a_{j_{k-t},i_1}(x)$ each, we obtain $k-t$ syndrome polynomials over $\mathbb{F}_2[x]/(1+x+\cdots+x^{p-1})$ as

$$\begin{bmatrix}x^{p-e}a_{e_1,i_1}(x) & \cdots & x^{p-e}a_{e_\alpha,i_1}(x) & a_{e_{\alpha+1},i_1}(x) & \cdots & a_{e_{k-t},i_1}(x)\end{bmatrix}\begin{bmatrix} x^{e_1(j_1-k)} & x^{e_1(j_2-k)} & \ldots & x^{e_1(j_{k-t}-k)}\\ x^{e_2(j_1-k)} & x^{e_2(j_2-k)} & \ldots & x^{e_2(j_{k-t}-k)}\\ \vdots & \vdots & \ddots & \vdots\\ x^{e_{k-t}(j_1-k)} & x^{e_{k-t}(j_2-k)} & \ldots & x^{e_{k-t}(j_{k-t}-k)}\end{bmatrix},$$

where α is an integer that ranges from 1 to $k-t-1$ with $e_\alpha\le d-k$ and $e_{\alpha+1}\ge d-k+1$. In the decoding process from columns 1, 2, 4 and 5 of the example in Table II, we have $k=4$, $t=2$, $e=1$, $i_1=1$, $e_1=0$, $e_2=3$, $j_1=k$, $j_2=k+1$ and $\alpha=1$. The two syndrome polynomials are

$$\begin{bmatrix}p_1(x) & p_2(x)\end{bmatrix}=\begin{bmatrix}x^{p-1}a_{0,1}(x) & a_{3,1}(x)\end{bmatrix}\cdot\begin{bmatrix}1 & 1\\ 1 & x^3\end{bmatrix}.$$

As the $(k+r,k)$ EVENODD code is MDS, we can recover the polynomials

$$x^{p-e}a_{e_1,i_1}(x),\ldots,x^{p-e}a_{e_\alpha,i_1}(x),a_{e_{\alpha+1},i_1}(x),\ldots,a_{e_{k-t},i_1}(x),$$

and therefore, $a_{e_1,i_1}(x),a_{e_2,i_1}(x),\ldots,a_{e_{k-t},i_1}(x)$ can be recovered. Let c be an integer with $2\le c\le t$ such that

[1] $1+x^3$ is invertible in $\mathbb{F}_2[x]/(1+x+\cdots+x^{p-1})$ due to the MDS property of EVENODD codes given in Proposition 2.2 in [6].

TABLE II: The second transformation for EVENODD codes with $k=4$, $r=2$, $d=5$ and $e=1$.

Column 0	Column 1	Column 2	Column 3	Column 4	Column 5
$a_{0,0}(x)$	$a_{1,0}(x)$	$a_{2,0}(x)$	$a_{3,0}(x)$	$a_{0,0}(x)+x^{p-1}a_{1,0}(x)+$ $x^{p-1}a_{0,1}(x)+a_{2,0}(x)+a_{3,0}(x)$	$a_{0,0}(x)+a_{1,0}(x)+a_{0,1}(x)$ $+x^2a_{2,0}(x)+x^3a_{3,0}(x)$
$a_{0,1}(x)$	$a_{1,1}(x)$	$a_{2,1}(x)$	$a_{3,1}(x)$	$x^{p-1}a_{0,1}(x)+(1+x^{p-1})a_{1,0}(x)+$ $a_{1,1}(x)+a_{2,1}(x)+a_{3,1}(x)$	$x^{p-1}a_{0,1}(x)+(1+x^{p-1})a_{1,0}(x)+$ $xa_{1,1}(x)+x^2a_{2,1}(x)+x^3a_{3,1}(x)$

$i_{c-1} < d-k$ and $i_c \geq d-k$. By the same argument, we can recover polynomials $a_{e_1,i_h}(x), a_{e_2,i_h}(x), \ldots, a_{e_{k-t},i_h}(x)$ for $h=2,3,\ldots,c-1$. Once the polynomials

$$a_{e_1,i_h}(x), a_{e_2,i_h}(x), \ldots, a_{e_{k-t},i_h}(x)$$

for $h=1,2,\ldots,c-1$ are known, we can recover all the other failed polynomials by first subtracting all $t(d-k+1)$ information polynomials and the known polynomials $a_{e_1,i_h}(x), a_{e_2,i_h}(x), \ldots, a_{e_{k-t},i_h}(x)$ with $h=1,2,\ldots,c-1$ from parity polynomials $a_{j_1,i_\ell}(x), \ldots, a_{j_{k-t},i_\ell}(x)$, followed by solving the failed polynomials according to the MDS property of the $(k+r,k)$ EVENODD code, where $\ell \in \{0,1,\ldots,d-k\} \setminus \{i_1, i_2, \ldots, i_{c-1}\}$.

If $i_1 \geq d-k$, then $i_t > \cdots > i_1 \geq d-k$ and we can obtain the following syndrome polynomials by subtracting all $t(d-k+1)$ information polynomials from all $(k-t)(d-k+1)$ parity polynomials

$$\begin{bmatrix} a^*_{e_1,\ell}(x) & a^*_{e_2,\ell}(x) & \cdots & a^*_{e_{k-t},\ell}(x) \end{bmatrix} \cdot \begin{bmatrix} x^{e_1(j_1-k)} & x^{e_1(j_2-k)} & \ldots & x^{e_1(j_{k-t}-k)} \\ x^{e_2(j_1-k)} & x^{e_2(j_2-k)} & \ldots & x^{e_2(j_{k-t}-k)} \\ \vdots & \vdots & \ddots & \vdots \\ x^{e_{k-t}(j_1-k)} & x^{e_{k-t}(j_2-k)} & \ldots & x^{e_{k-t}(j_{k-t}-k)} \end{bmatrix},$$

where

$$a^*_{e_i,\ell}(x) = \begin{cases} a_{e_i,\ell}(x) & \text{if } e_i = \ell, \\ x^{p-e}a_{e_i,\ell}(x) + (1+x^{p-e})a_{\ell,e_i}(x) & \text{if } e_i < \ell, \\ x^{p-e}a_{e_i,\ell}(x) + x^{p-e}a_{\ell,e_i}(x) & \text{if } e_i > \ell, \end{cases} \tag{6}$$

$\ell = 0,1,\ldots,d-k$. The polynomials $a^*_{e_1,\ell}(x)$, $a^*_{e_2,\ell}(x), \ldots, a^*_{e_{k-t},\ell}(x)$ can be recovered, because $(k+r,k)$ EVENODD code is MDS. Then, we can obtain $a_{\ell,\ell}(x)$ directly, $a_{e_i,\ell}(x)$ for $e_i > \ell$ by $a^*_{e_i,\ell}(x) + a^*_{\ell,e_i}(x)$, and $a_{e_i,\ell}(x)$ for $e_i < \ell$ by $x^e(a_{\ell,e_i}(x) + a^*_{\ell,e_i}(x))$. □

We show in the next theorem that the second transformed EVENODD code has optimal access for the first $d-k+1$ columns.

Theorem 2. *The repair bandwidth and repair access of column i of the second transformed EVENODD code for $i=0,1,\ldots,d-k$ is optimal.*

Proof. For $i=0,1,\ldots,d-k$, column i can be repaired by downloading one polynomial from each of d helper columns. Among d columns, $d-k$ columns are columns $0,1,\ldots,i-1,i+1,\ldots,d-k$ and other k columns are chosen from columns $d-k+1,\ldots,k+r-1$. Specifically, we can recover the polynomials $a_{i,i}(x)$, $x^{p-e}a_{j,i}(x) + (1+x^{p-e})a_{i,j}(x)$ for $j < i$ and $x^{p-e}(a_{\ell,i}(x) + a_{i,\ell}(x))$ for $\ell > i$, by downloading k polynomials $a_{h_1,i}(x), \ldots, a_{h_k,i}(x)$ from columns $h_1, \ldots, h_k$, where $h_1 \neq \ldots \neq h_k \in \{d-k+1, \ldots, k+r-1\}$, due to the MDS property of EVENODD codes. Then we download $d-k$ polynomials $a_{0,i}(x), \ldots, a_{i-1,i}(x), a_{i+1,i}(x), \ldots, a_{d-k,i}(x)$ from columns $0,1,\ldots,i-1,i+1,\ldots,d-k$. Finally, we subtract the downloaded polynomials $a_{0,i}(x), \ldots, a_{i-1,i}(x), a_{i+1,i}(x), \ldots, a_{d-k,i}(x)$ from the recovered polynomials $x^{p-e}a_{j,i}(x) + (1+x^{p-e})a_{i,j}(x)$ for $j < i$ and $x^{p-e}(a_{\ell,i}(x) + a_{i,\ell}(x))$ for $\ell > i$, to obtain polynomials $a_{i,0}(x), a_{i,1}(x), \ldots, a_{i,i-1}(x), a_{i,i+1}(x), \ldots, a_{i,d-k}(x)$. This completes the proof. □

Note that $\mathsf{EVENODD}_1$ with the first transformation also satisfies Theorem 1 and Theorem 2, as the two transformations are equivalent. In the following, let $\mathsf{EVENODD}_1$ be the transformed code with the first transformation and $\mathsf{EVENODD}_2$ be the transformed code by applying the first transformation for the columns from $d-k+1$ to $2d-2k+1$ of $\mathsf{EVENODD}_1$. Specifically, we can obtain $\mathsf{EVENODD}_2$ as follows. Let

$$t = d-k+1.$$

We first generate t instances of the code $\mathsf{EVENODD}_1$ and view the t polynomials stored in each column of $\mathsf{EVENODD}_1$ as a vector. For $\ell = 0,1,\ldots,d-k$ and $h=0,1,\ldots,n-1$, the vector stored in column h of instance ℓ of $\mathsf{EVENODD}_1$ is denoted as $\mathbf{v}_h^\ell$. For $i=0,1,\ldots,d-k$, column $t+i$ of $\mathsf{EVENODD}_2$ stores the following t vectors (t^2 polynomials)

$$\begin{array}{l} \mathbf{v}^0_{t+i} + \mathbf{v}^i_t, \\ \mathbf{v}^1_{t+i} + \mathbf{v}^i_{t+1}, \ldots, \\ \mathbf{v}^{i-1}_{t+i} + \mathbf{v}^i_{t+i-1}, \\ \mathbf{v}^i_{t+i}, \\ \mathbf{v}^{i+1}_{t+i} + (1+x^e)\mathbf{v}^i_{t+i+1}, \\ \mathbf{v}^{i+2}_{t+i} + (1+x^e)\mathbf{v}^i_{t+i+2}, \ldots, \\ \mathbf{v}^{d-k}_{t+i} + (1+x^e)\mathbf{v}^i_{t+d-k}, \end{array} \tag{7}$$

where $1 \leq e \leq p-1$. Note that the multiplication of a polynomial x^e and a vector

$$\mathbf{v} = \begin{bmatrix} v_0 & v_1 & \ldots & v_{d-k} \end{bmatrix}$$

is defined as

$$x^e\mathbf{v} = \begin{bmatrix} x^e v_0 & x^e v_1 & \ldots & x^e v_{d-k} \end{bmatrix}$$

and the addition of two vectors

$$\mathbf{v}^1 = \begin{bmatrix} v^1_0 & v^1_1 & \ldots & v^1_{d-k} \end{bmatrix}$$

and

$$\mathbf{v}^2 = \begin{bmatrix} v^2_0 & v^2_1 & \ldots & v^2_{d-k} \end{bmatrix}$$

is

$$\mathbf{v}^1 + \mathbf{v}^2 = \begin{bmatrix} v^1_0 + v^2_0 & v^1_1 + v^2_1 & \ldots & v^1_{d-k} + v^2_{d-k} \end{bmatrix}.$$

For $h \in \{0,1,\ldots,n-1\} \setminus \{t,t+1,\ldots,2t-1\}$, column h stores t vectors (t^2 polynomials)

$$\mathbf{v}_h^0, \mathbf{v}_h^1, \ldots, \mathbf{v}_h^{d-k}.$$

For $\ell = 0,1,\ldots,d-k$ and $h = t,t+1,\ldots,n-1$, we have that

$$\mathbf{v}_h^\ell = \begin{bmatrix} a_{h,\ell t}(x) & a_{h,\ell t+1}(x) & \cdots & a_{h,(\ell+1)t-1}(x) \end{bmatrix}$$

according to the first transformation. Table III shows the storage of the first $2t$ columns of $\mathsf{EVENODD}_2$ by (7). We show in the next theorem that the optimal repair access property of the first $d-k+1$ columns of $\mathsf{EVENODD}_1$ code is maintained in $\mathsf{EVENODD}_2$.

Theorem 3. *The repair access of column i of $\mathsf{EVENODD}_2$ code for $i = 0,1,\ldots,2d-2k+1$ is optimal.*

Proof. By Theorem 2, we can repair t vectors (t^2 polynomials) in column i for $i = t,t+1\ldots,2t-1$ by downloading k vectors (kt polynomials)

$$\mathbf{v}_{h_1}^{i-t}, \mathbf{v}_{h_2}^{i-t}, \ldots, \mathbf{v}_{h_k}^{i-t} \tag{8}$$

from columns h_j with $j = 0,1,\ldots,k-1$, where $h_j = j$ for $j = 0,1,\ldots,t-1$ and $h_j \in \{2t,\ldots,k+r-1\}$ for $j = t,t+1,\ldots,k-1$, and the following $d-k$ vectors ($(d-k)t$ polynomials)

$$\begin{matrix} \mathbf{v}_t^{i-t} + (1+x^e)\mathbf{v}_i^0, \ldots, \mathbf{v}_{i-1}^{i-t} + (1+x^e)\mathbf{v}_i^{i-t-1}, \\ \mathbf{v}_{i+1}^{i-t} + \mathbf{v}_i^{i-t+1}, \ldots, \mathbf{v}_{2t-1}^{i-t} + \mathbf{v}_i^{t-1}. \end{matrix} \tag{9}$$

Specifically, we can first compute vectors

$$\mathbf{v}_t^{i-t}, \mathbf{v}_{t+1}^{i-t}, \ldots, \mathbf{v}_{2t-1}^{i-t}$$

from k vectors in (8), and then recover the t vectors in column i with the above t vectors and the downloaded $d-k$ vectors in (9).

Consider the repair of column i with $i = 0,1,\ldots,d-k$. We can repair t^2 polynomials in column i by downloading $(2t-1)t$ polynomials from $2t-1$ columns $0,1,\ldots,i-1,i+1,\ldots,t-1,t,t+1,\ldots,2t-1$ in rows $i+1,i+1+t,\ldots,i+1+(t-1)t$, and $(d-2t+1)t$ polynomials from $d-2t+1$ columns $h_1,\ldots,h_{d-2t+1}$ in rows $i+1,i+1+t,\ldots,i+1+(t-1)t$ with indices $2t \le h_1 < \ldots < h_{d-2t+1} \le n-1$. Note that the t^2 polynomials downloaded from columns t to $2t-1$ are in (10) (in the next page). We can compute $a_{t+1,i}(x)$ and $a_{t,t+i}(x)$ from $a_{t+1,i}(x) + a_{t,t+i}(x)$ and $a_{t,t+i}(x) + (1+x^e)a_{t+1,i}(x)$ by

$$\frac{(a_{t,t+i}(x) + (1+x^e)a_{t+1,i}(x)) - (a_{t+1,i}(x) + a_{t,t+i}(x))}{x^e}$$

and

$$\frac{(1+x^e)(a_{t+1,i}(x) + a_{t,t+i}(x)) - (a_{t,t+i}(x) + (1+x^e)a_{t+1,i}(x))}{x^e},$$

respectively. Similarly, we can compute t^2 polynomials

$$a_{t,\ell t+i}(x), a_{t+1,\ell t+i}(x), \ldots, a_{2t-1,\ell t+i}(x),$$

from t^2 polynomials in (10) (in the next page), where $\ell = 0,1,\ldots,d-k$. Together with $(d-2t+1)t$ polynomials

$$a_{h_1,\ell t+i}(x), a_{h_2,\ell t+i}(x), \ldots, a_{h_{d-2t+1},\ell t+i}(x),$$

with $\ell = 0,1,\ldots,d-k$ downloaded from $d-2t+1$ columns $h_1,\ldots,h_{d-2t+1}$, we can compute the following t^2 polynomials

$$a_{0,\ell t+i}(x), a_{1,\ell t+i}(x), \ldots, a_{d-k,\ell t+i}(x),$$

with $\ell = 0,1,\ldots,d-k$. Finally, we can recover all t^2 polynomials with the above t^2 polynomials and the downloaded $(t-1)t$ polynomials from $t-1$ columns $0,1,\ldots,i-1,i+1,\ldots,t-1$. □

Each column of $\mathsf{EVENODD}_1$ stores $d-k+1$ polynomials, we can repair each of the first $d-k+1$ columns by accessing one polynomial from each of the d columns according to Theorem 2 and the repair access is optimal according to (1). In $\mathsf{EVENODD}_2$, each column has $(d-k+1)^2$ polynomials. According to Theorem 3, the $(d-k+1)^2$ polynomial in each of the first $2(d-k+1)$ columns can be recovered by accessing $d-k+1$ polynomials from each of the d columns and the repair access is optimal according to (1).

Table IV shows the $\mathsf{EVENODD}_2$ by applying the first transformation twice for EVENODD codes with $k=4$, $r=2$, $d=5$ and $e=1$. We can repair column 0 by downloading the following 10 polynomials

$$\begin{aligned} &a_{1,0}(x) + a_{0,1}(x), a_{2,0}(x), a_{3,0}(x) + a_{2,2}(x), a_{4,0}(x), \\ &a_{5,0}(x), a_{1,2}(x) + a_{0,3}(x), a_{2,2}(x) + (1+x)a_{3,0}(x), \\ &a_{3,2}(x), a_{4,2}(x), a_{5,2}(x). \end{aligned}$$

Specifically, we first compute $a_{2,2}(x)$ and $a_{3,0}(x)$ from $a_{3,0}(x) + a_{2,2}(x)$ and $a_{2,2}(x) + (1+x)a_{3,0}(x)$. Then, we can compute $a_{0,0}(x), a_{1,0}(x)$ and $a_{0,2}(x), a_{1,2}(x)$ from

$$a_{2,0}(x), a_{3,0}(x), a_{4,0}(x), a_{5,0}(x),$$

and

$$a_{2,2}(x), a_{3,2}(x), a_{4,2}(x), a_{5,2}(x),$$

respectively, according to the MDS property of EVENODD codes. Finally, we can recover $a_{0,1}(x) + (1+x)a_{1,0}(x)$ and $a_{0,3}(x) + (1+x)a_{1,2}(x)$ by $(a_{1,0}(x) + a_{0,1}(x)) + xa_{1,0}(x)$ and $(a_{1,2}(x) + a_{0,3}(x)) + xa_{1,2}(x)$, respectively. According to Theorem 3, we can repair each of the first four columns by downloading 10 polynomials and the repair access is optimal.

Consider the systematic $\mathsf{EVENODD}_2$ code with $k=4$, $r=2$, $d=5$ and $e=1$ in Table V. The repair access of column i for $i = 0,1,2,3$ is optimal. We can recover the four polynomials in column 0 by downloading 10 polynomials

$$\begin{aligned} &a_{1,0}(x), a_{1,2}(x), a_{2,0}(x), a_{2,2}(x), a_{3,0}(x), a_{3,2}(x), \\ &a_{4,0}(x), a_{4,2}(x), a_{5,0}(x), a_{5,2}(x). \end{aligned}$$

By subtracting polynomials $a_{1,0}(x), a_{2,0}(x), a_{3,0}(x)$ and $a_{2,2}(x)$ from $a_{4,0}(x)$ and $a_{5,0}(x)$ each, we can obtain two polynomials

$$\begin{aligned} p_1(x) =& a_{0,0}(x) + x^{p-1}a_{0,1}(x), \\ p_2(x) =& a_{0,0}(x) + a_{0,1}(x). \end{aligned}$$

Thus, we can recover $a_{0,0}(x)$ and $a_{0,1}(x)$ by $\frac{p_1(x)+x^{p-1}p_2(x)}{1+x^{p-1}}$ and $\frac{p_1(x)+p_2(x)}{1+x^{p-1}}$, respectively. Similarly, we can first obtain

$$\begin{aligned} p_3(x) =& a_{0,2}(x) + x^{p-1}a_{0,3}(x), \\ p_4(x) =& a_{0,2}(x) + a_{0,3}(x), \end{aligned}$$

TABLE III: The storage of the first $2t$ columns of $\mathsf{EVENODD}_2$, where $t = d - k + 1$.

Column 0	Column 1	$\cdots$	Column $d-k$
$a_{0,0}(x)$	$a_{1,0}(x) + a_{0,1}(x)$	$\cdots$	$a_{d-k,0}(x) + a_{0,d-k}(x)$
$a_{0,1}(x) + (1+x^e)a_{1,0}(x)$	$a_{1,1}(x)$	$\cdots$	$a_{d-k,1}(x) + a_{1,d-k}(x)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$a_{0,d-k}(x) + (1+x^e)a_{d-k,0}(x)$	$a_{1,d-k}(x) + (1+x^e)a_{d-k,1}(x)$	$\cdots$	$a_{d-k,d-k}(x)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$a_{0,(d-k)t}(x)$	$a_{1,(d-k)t}(x) + a_{0,(d-k)t+1}(x)$	$\cdots$	$a_{d-k,(d-k)t}(x) + a_{0,t^2-1}(x)$
$a_{0,(d-k)t+1}(x) + (1+x^e)a_{1,(d-k)t}(x)$	$a_{1,(d-k)t+1}(x)$	$\cdots$	$a_{d-k,(d-k)t+1}(x) + a_{1,t^2-1}(x)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$a_{0,t^2-1}(x) + (1+x^e)a_{d-k,(d-k)t}(x)$	$a_{1,t^2-1}(x) + (1+x^e)a_{d-k,(d-k)t+1}(x)$	$\cdots$	$a_{d-k,t^2-1}(x)$

Column t	Column $t+1$	$\cdots$	Column $2t-1$
$a_{t,0}(x)$	$a_{t+1,0}(x) + a_{t,t}(x)$	$\cdots$	$a_{2t-1,0}(x) + a_{t,(d-k)t}(x)$
$a_{t,1}(x)$	$a_{t+1,1}(x) + a_{t,t+1}(x)$	$\cdots$	$a_{2t-1,1}(x) + a_{t,(d-k)t+1}(x)$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$a_{t,d-k}(x)$	$a_{t+1,d-k}(x) + a_{t,2t-1}(x)$	$\cdots$	$a_{2t-1,d-k}(x) + a_{t,t^2-1}(x)$
$a_{t,t}(x) + (1+x^e)a_{t+1,0}(x)$	$a_{t+1,t}(x)$	$\cdots$	$a_{2t-1,t}(x) + a_{t+1,(d-k)t}(x)$
$a_{t,t+1}(x) + (1+x^e)a_{t+1,1}(x)$	$a_{t+1,t+1}(x)$	$\cdots$	$a_{2t-1,t+1}(x) + a_{t+1,(d-k)t+1}(x)$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$a_{t,2t-1}(x) + (1+x^e)a_{t+1,d-k}(x)$	$a_{t+1,2t-1}(x)$	$\cdots$	$a_{2t-1,2t-1}(x) + a_{t+1,t^2-1}(x)$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$a_{t,(d-k)t}(x) + (1+x^e)a_{2t-1,0}(x)$	$a_{t+1,(d-k)t}(x) + (1+x^e)a_{2t-1,t}(x)$	$\cdots$	$a_{2t-1,(d-k)t}(x)$
$a_{t,(d-k)t+1}(x) + (1+x^e)a_{2t-1,1}(x)$	$a_{t+1,(d-k)t+1}(x) + (1+x^e)a_{2t-1,t+1}(x)$	$\cdots$	$a_{2t-1,(d-k)t+1}(x)$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$a_{t,t^2-1}(x) + (1+x^e)a_{2t-1,d-k}(x)$	$a_{t+1,t^2-1}(x) + (1+x^e)a_{2t-1,2t-1}(x)$	$\cdots$	$a_{2t-1,t^2-1}(x)$

$$\begin{bmatrix} a_{t,i}(x) & a_{t+1,i}(x) + a_{t,t+i}(x) & \cdots & a_{2t-1,i}(x) + a_{t,(d-k)t+i}(x) \\ a_{t,t+i}(x) + (1+x^e)a_{t+1,i}(x) & a_{t+1,t+i}(x) & \cdots & a_{2t-1,t+i}(x) + a_{t+1,(d-k)t+i}(x) \\ \vdots & \vdots & \ddots & \vdots \\ a_{t,(d-k)t+i}(x) + (1+x^e)a_{2t-1,i}(x) & a_{t+1,(d-k)t+i}(x) + (1+x^e)a_{2t-1,t+i}(x) & \cdots & a_{2t-1,(d-k)t+i}(x) \end{bmatrix}. \quad (10)$$

TABLE IV: $\mathsf{EVENODD}_2$ by applying the first transformation twice for EVENODD codes with $k = 4$, $r = 2$, $d = 5$ and $e = 1$.

Column 0	Column 1	Column 2	Column 3	Column 4	Column 5
$a_{0,0}(x)$	$a_{1,0}(x) + a_{0,1}(x)$	$a_{2,0}(x)$	$a_{3,0}(x) + a_{2,2}(x)$	$a_{4,0}(x)$	$a_{5,0}(x)$
$a_{0,1}(x) + (1+x)a_{1,0}(x)$	$a_{1,1}(x)$	$a_{2,1}(x)$	$a_{3,1}(x) + a_{2,3}(x)$	$a_{4,1}(x)$	$a_{5,1}(x)$
$a_{0,2}(x)$	$a_{1,2}(x) + a_{0,3}(x)$	$a_{2,2}(x) + (1+x)a_{3,0}(x)$	$a_{3,2}(x)$	$a_{4,2}(x)$	$a_{5,2}(x)$
$a_{0,3}(x) + (1+x)a_{1,2}(x)$	$a_{1,3}(x)$	$a_{2,3}(x) + (1+x)a_{3,1}(x)$	$a_{3,3}(x)$	$a_{4,3}(x)$	$a_{5,3}(x)$

by subtracting polynomials $a_{1,2}(x), a_{2,2}(x), a_{3,2}(x)$ and $a_{3,0}(x)$ from $a_{4,2}(x)$ and $a_{5,2}(x)$ each, and then recover the other two polynomials in column 0 by $\frac{p_3(x)+x^{p-1}p_4(x)}{1+x^{p-1}}$ and $\frac{p_3(x)+p_4(x)}{1+x^{p-1}}$. Column 1 can be recovered by downloading

$$a_{0,1}(x), a_{0,3}(x), a_{2,1}(x), a_{2,3}(x), a_{3,1}(x), a_{3,3}(x),$$
$$a_{4,1}(x), a_{4,3}(x), a_{5,1}(x), a_{5,3}(x).$$

The first transformed EVENODD codes also satisfy the above three theorems, as the two transformations are equivalent. The $\mathsf{EVENODD}_1$ codes have optimal repair for each of the first $d-k+1$ columns according to Theorem 2. By applying the second transformation for the columns from $d-k+1$ to $2d-2k+1$ of $\mathsf{EVENODD}_1$ codes, we obtain $\mathsf{EVENODD}_2$ codes that have optimal repair for each of the columns from 0 to $2d-2k+1$ according to Theorem 3. Similarly, we can transform the original EVENODD codes for the columns between $i(d-k+1)$ and $((i+1)(d-k+1)-1) \bmod n$ to obtain the transformed EVENODD codes with optimal repair with the columns between $i(d-k+1)$ and $((i+1)(d-k+1)-1) \bmod n$, where $i = 1, 2, \ldots, \lceil \frac{n}{d-k+1} \rceil - 1$. The polynomial $1+x^e$ used in the transformation in (3) is called the *encoding coefficient* associated with the transformation. We may replace the encoding coefficient $1+x^e$ by other polynomials in $\mathbb{F}_2[x]/(1+x+\cdots+x^{p-1})$, such as x^e, as long as the three theorems in Section II-C still hold under the specific binary MDS array codes. It is easy to check that the three theorems in Section II-C hold for EVENODD codes, if we replace the encoding coefficient $1+x^e$ by x^e. More generally, we can view the $d-k+1$ polynomials in (3) as $d-k+1$ vectors with length $p-1$ and compute the vectors by the summation of some permutated vectors, as long as the three theorems in Section II-C hold. With more general transformation, we may combine the transformation and the existing binary MDS array codes with efficient repair for any single information column to obtain the transformed codes that have efficient repair for both information and parity columns. Recall that there are some efficient repair schemes for any

TABLE V: Systematic $\mathsf{EVENODD}_2$ code by applying the second transformation twice for EVENODD code with $k=4$, $r=2$, $d=5$ and $e=1$.

Information columns				Parity column 0
$a_{0,0}(x)$	$a_{1,0}(x)$	$a_{2,0}(x)$	$a_{3,0}(x)$	$a_{0,0}(x)+(x^{p-1}a_{1,0}(x)+x^{p-1}a_{0,1}(x))+a_{2,0}(x)+(x^{p-1}a_{3,0}(x)+x^{p-1}a_{2,2}(x))$
$a_{0,1}(x)$	$a_{1,1}(x)$	$a_{2,1}(x)$	$a_{3,1}(x)$	$x^{p-1}a_{0,1}(x)+(1+x^{p-1})a_{1,0}(x)+a_{1,1}(x)+a_{2,1}(x)+x^{p-1}a_{3,1}(x)+x^{p-1}a_{2,3}(x)$
$a_{0,2}(x)$	$a_{1,2}(x)$	$a_{2,2}(x)$	$a_{3,2}(x)$	$a_{0,2}(x)+(x^{p-1}a_{1,2}(x)+x^{p-1}a_{0,3}(x))+x^{p-1}a_{2,2}(x)+(1+x^{p-1})a_{3,0}(x)+a_{3,2}(x)$
$a_{0,3}(x)$	$a_{1,3}(x)$	$a_{2,3}(x)$	$a_{3,3}(x)$	$x^{p-1}a_{0,3}(x)+(1+x^{p-1})a_{1,2}(x)+a_{1,3}(x)+x^{p-1}a_{2,3}(x)+(1+x^{p-1})a_{3,1}(x)+a_{3,3}(x)$

Parity column 1
$a_{0,0}(x)+a_{1,0}(x)+a_{0,1}(x)+x^2a_{2,0}(x)+(x^2a_{3,0}(x)+x^2a_{2,2}(x))$
$x^{p-1}a_{0,1}(x)+(1+x^{p-1})a_{1,0}(x)+xa_{1,1}(x)+x^2a_{2,1}(x)+x^2a_{3,1}(x)+x^2a_{2,3}(x)$
$a_{0,2}(x)+(a_{1,2}(x)+a_{0,3}(x))+xa_{2,2}(x)+(x+x^2)a_{3,0}(x)+x^3a_{3,2}(x)$
$x^{p-1}a_{0,3}(x)+(1+x^{p-1})a_{1,2}(x)+xa_{1,3}(x)+xa_{2,3}(x)+(x+x^2)a_{3,1}(x)+x^3a_{3,3}(x)$

information column of RDP [28], X-code [38], EVENODD [27] and the binary MDS array codes [33]. In Section IV, we will take an example of EVENODD to show how to design the specific transformation for EVENODD to enable optimal repair for each of the parity columns and preserve the efficient repair property for any single information column. We also give the transformation for the binary MDS array codes [33] with efficient repair access for any single information column such that the transformed codes have optimal repair access for any single parity column and asymptotically optimal repair access for any single information column in Section IV.

III. Construction of Multi-layer Transformed EVENODD Codes

In the section, we first present the construction of multi-layer transformed EVENODD codes by applying the transformation given in Section II. We then give a repair algorithm for any single column with optimal repair access.

A. Construction

The codes considered herein have k information columns and r parity columns. For notational convenience, we divide k information columns into $\lceil \frac{k}{d-k+1} \rceil$ *information partitions*, each of the information partitions has $d-k+1$ columns. For $i=1,2,\ldots,\lceil \frac{k}{d-k+1} \rceil - 1$, information partition i contains columns between $(i-1)(d-k+1)$ and $(i(d-k+1)-1)$. Information partition $\lceil \frac{k}{d-k+1} \rceil$ contains the last $d-k+1$ information columns. Similarly, the r parity columns are divided into $\lceil \frac{r}{d-k+1} \rceil$ *parity partitions* and parity column i contains parity columns between $(i-1)(d-k+1)$ and $(i(d-k+1)-1)$ for $i=1,2,\ldots,\lceil \frac{r}{d-k+1} \rceil - 1$. Parity partition $\lceil \frac{r}{d-k+1} \rceil$ contains the last $d-k+1$ parity columns. Therefore, we obtain $\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil$ partitions, contains $\lceil \frac{k}{d-k+1} \rceil$ information partitions and $\lceil \frac{r}{d-k+1} \rceil$ parity partitions. We label the index of the partitions from 1 to $\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil$. The construction is given in the following.

By applying the transformation for the first information partition (the first $d-k+1$ columns) of EVENODD code, we can obtain $\mathsf{EVENODD}_1$ with each column having $d-k+1$ polynomials, such that $\mathsf{EVENODD}_1$ is MDS according to Theorem 1 and has optimal repair bandwidth for the first $d-k+1$ information columns according to Theorem 2. By applying the transformation for the second partition (columns between $d-k+1$ and $2(d-k+1)-1$) of $\mathsf{EVENODD}_1$, we obtain $\mathsf{EVENODD}_2$ with each column having $(d-k+1)^2$ polynomials that is MDS code according to Theorem 1 and has optimal repair bandwidth for the first $2(d-k+1)$ information columns according to Theorem 2 and Theorem 3.

For $j=1,2,\ldots,\lceil \frac{k}{d-k+1} \rceil - 1$, by recursively applying the transformation for information partition $i+1$ of $\mathsf{EVENODD}_j$ code, we can obtain $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil}$. Specifically, we can obtain $\mathsf{EVENODD}_{j+1}$ by applying the transformation for $\mathsf{EVENODD}_j$ as follows, where $j=1,2,\ldots,\lceil \frac{k}{d-k+1} \rceil - 1$. We generate $d-k+1$ instances of the code $\mathsf{EVENODD}_j$ and view the $(d-k+1)^j$ polynomials stored in each column of $\mathsf{EVENODD}_j$ as a vector. For $\ell=0,1,\ldots,d-k$ and $h=0,1,\ldots,n-1$, denote $(d-k+1)^j$ polynomials stored in column h of instance ℓ of $\mathsf{EVENODD}_j$ as the vector $\mathbf{v}_h^\ell$. For $i=0,1,\ldots,d-k$, column $j(d-k+1)+i$ of $\mathsf{EVENODD}_{j+1}$ stores the following $d-k+1$ vectors ($(d-k+1)^{j+1}$ polynomials)

$$\begin{aligned}
&\mathbf{v}^0_{(d-k+1)+i}+\mathbf{v}^i_{(d-k+1)},\\
&\mathbf{v}^1_{(d-k+1)+i}+\mathbf{v}^i_{(d-k+1)+1},\ldots,\\
&\mathbf{v}^{i-1}_{(d-k+1)+i}+\mathbf{v}^i_{(d-k+1)+i-1},\\
&\mathbf{v}^i_{(d-k+1)+i},\\
&\mathbf{v}^{i+1}_{(d-k+1)+i}+(1+x^e)\mathbf{v}^i_{(d-k+1)+i+1},\\
&\mathbf{v}^{i+2}_{(d-k+1)+i}+(1+x^e)\mathbf{v}^i_{(d-k+1)+i+2},\ldots,\\
&\mathbf{v}^{d-k}_{(d-k+1)+i}+(1+x^e)\mathbf{v}^i_{(d-k+1)+d-k},
\end{aligned}$$

where $1\le e\le p-1$. For $h\in\{0,1,\ldots,n-1\}\setminus\{j(d-k+1),j(d-k+1)+1,\ldots,(j+1)(d-k+1)-1\}$, column h stores $d-k+1$ vectors ($(d-k+1)^{j+1}$ polynomials)

$$\mathbf{v}^0_h,\mathbf{v}^1_h,\ldots,\mathbf{v}^{d-k}_h.$$

Note that the transformation from $\mathsf{EVENODD}_j$ to $\mathsf{EVENODD}_{j+1}$ is the same as the transformation from $\mathsf{EVENODD}_1$ to $\mathsf{EVENODD}_2$, and we can show that the optimal repair property of the first $j(d-k+1)$ columns of $\mathsf{EVENODD}_j$ is maintained in $\mathsf{EVENODD}_{j+1}$ by the similar proof of Theorem 3. Therefore, we can obtain the code $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil}$ and each column has $(d-k+1)^{\lceil \frac{k}{d-k+1} \rceil}$ polynomials. $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil}$ is MDS code according to Theorem 1 and we can repair each of the first k columns by downloading $(d-k+1)^{\lceil \frac{k}{d-k+1} \rceil - 1}$ polynomials from each of the chosen d columns, the repair bandwidth of each

of the first k columns is optimal according to Theorem 2 and Theorem 3. To obtain optimal repair bandwidth for parity columns, we can transform $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil}$ code $\lceil \frac{r}{d-k+1} \rceil$ times into $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil}$ code by the first transformation. In $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil}$ code, each column has $(d-k+1)^{\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil}$ polynomials and we can repair each column by downloading $(d-k+1)^{\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil - 1}$ polynomials from each of the chosen d columns. Therefore, $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil}$ code has optimal repair bandwidth for any column.

B. Repair Algorithm

Algorithm 1 Algorithm of repairing a single failed column f, where $0 \leq f \leq k+r-1$.

1: The column f is failed.
2: **if** $f \in \{0, 1, \ldots, k-1\}$, denote $f = (d-k+1)m_f + r_f$, where m_f, r_f are two integers with $0 \leq m_f$ and $0 \leq r_f \leq d-k$. **then**
3: Repair the polynomials in column f by downloading $a_{h_1,\ell}(x), a_{h_2,\ell}(x), \ldots, a_{h_d,\ell}(x)$, for $\ell \bmod (d-k+1)^{m_f+1} \in \{r_f \cdot (d-k+1)^{m_f}, r_f \cdot (d-k+1)^{m_f}+1, \ldots, (r_f+1)\cdot(d-k+1)^{m_f}-1\}$, where $\{h_1, h_2, \ldots, h_{d-k}\} = \{f-m_f, f-m_f+1, \ldots, f-1, f+1, \ldots, f-m_f+d-k\}$ and columns $h_{d-k+1}, \ldots, h_d$ are chosen as follows. For $i = d-k+1, \ldots, d$, if h_i belongs to a partition ℓ with $\ell > m_f+1$, then all $d-k+1$ columns of the partition ℓ are in $\{h_{d-k+1}, \ldots, h_d\}$.
4: **return**
5: **if** $f \in \{k, k+1 \ldots, k+r-1\}$, denote $f-k = (d-k+1)m_f + r_f$, where m_f, r_f are two integers with $0 \leq m_f$ and $0 \leq r_f \leq d-k$. **then**
6: Repair the polynomials in column f by downloading $a_{h_1,\ell}(x), a_{h_2,\ell}(x), \ldots, a_{h_d,\ell}(x)$, for $\ell \bmod (d-k+1)^{\lceil \frac{k}{d-k+1} \rceil + m_f + 1} \in \{r_f \cdot (d-k+1)^{\lceil \frac{k}{d-k+1} \rceil + m_f}, r_f \cdot (d-k+1)^{\lceil \frac{k}{d-k+1} \rceil + m_f}+1, \ldots, (r_f+1)\cdot(d-k+1)^{\lceil \frac{k}{d-k+1} \rceil + m_f}-1\}$, where $\{h_1, h_2, \ldots, h_{d-k}\} = \{f-m_f, f-m_f+1, \ldots, f-1, f+1, \ldots, f-m_f+d-k\}$ and columns $h_{d-k+1}, \ldots, h_d$ are chosen as follows. For $i = d-k+1, \ldots, d$, if h_i belongs to a partition ℓ with $\ell > m_f+1$, then all $d-k+1$ columns of the partition ℓ are in $\{h_{d-k+1}, \ldots, h_d\}$.
7: **return**

In the following, we present the repair algorithm for a single column failure that is stated in Algorithm 1. Note that there is a requirement when choosing the d helper columns in Algorithm 1. We show in the next lemma that we can always choose the d helper columns that satisfy the requirement for all f.

Lemma 4. *For $f = 0, 1, \ldots, k+r-1$, column f belongs to partition m_f+1. The first $d-k$ helper columns are chosen to be the other surviving columns of partition m_f+1, and the other k helper columns h_i for $i = d-k+1, \ldots, d$ satisfy that if h_i belongs to a partition ℓ with $\ell > m_f+1$, then all $d-k+1$ columns of the partition ℓ are in $\{h_{d-k+1}, \ldots, h_d\}$.*

Proof. We first consider the information failure, i.e., $0 \leq f \leq k-1$. The case of $k \leq f \leq k+r-1$ can be proven similarly.

If k is a multiple of $d-k+1$, then we can choose k helper columns h_i for $i = d-k+1, \ldots, d$ as all the columns of any $k/(d-k+1)$ partitions, except partition m_f+1. If k is not a multiple of $d-k+1$, we can divide the proof into two cases: $m_f = 0$ and $m_f > 0$. When $m_f = 0$, We can choose k helper columns h_i for $i = d-k+1, \ldots, d$ as all the columns of information partitions $\lceil \frac{k}{d-k+1} \rceil - 1$ and $\lceil \frac{k}{d-k+1} \rceil$, and any other $\lceil \frac{k}{d-k+1} \rceil - 2$ partitions except partition m_f+1. When $m_f > 0$, we can choose k helper columns h_i for $i = d-k+1, \ldots, d$ as all the columns of α partitions except partition m_f+1 and β columns that belong to information partition ℓ with $1 \leq \ell \leq m_f$, where $(d-k+1)\alpha + \beta = k$. This completes the proof. □

We first consider the repair algorithm of information column f, i.e., $0 \leq f \leq k-1$. There exist two integers m_f and r_f such that $f = (d-k+1)m_f + r_f$, where $0 \leq m_f$ and $0 \leq r_f \leq d-k$.

Note that $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil}$ code is transformed from EVENODD code for $\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil$ times. The optimal repair of columns in partition i is enabled by the i-th transformation, where $i = 1, 2, \ldots, \lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil$. According to Theorem 3, the optimal repair property of columns in partition i of $\mathsf{EVENODD}_i$ is preserved in $\mathsf{EVENODD}_{i+1}$ (also in $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil}$) for $i = 1, 2, \ldots, \lceil \frac{k}{d-k+1} \rceil - 1$ if either all $d-k+1$ columns of partition $i+1$ are chosen as helper columns or all $d-k+1$ columns of partition $i+1$ are not chosen as helper columns. In addition, the other $d-k$ surviving columns of partition i are required to recover the failed column in partition i. Therefore, the d helper columns of the failed column f are comprised of $d-k$ columns in partition m_f+1, and other k columns. If a column of partition ℓ with $\ell > m_f+1$ is chosen as helper column, then all $d-k+1$ columns of partition ℓ are chosen as the helper columns. By Lemma 4, we can always find the d helper columns that satisfy the requirement in Algorithm 1.

We can recover column f of $\mathsf{EVENODD}_{m_f+1}$ by downloading polynomials $a_{h_1,\ell}(x), a_{h_2,\ell}(x), \ldots, a_{h_d,\ell}(x)$ from the chosen d helper columns, for $\ell \in \{r_f \cdot (d-k+1)^{m_f}, r_f \cdot (d-k+1)^{m_f}+1, \ldots, 2r_f \cdot (d-k+1)^{m_f}-1\}$. Because the optimal repair algorithm of column f of $\mathsf{EVENODD}_{m_f+1}$ is preserved in $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil}$ and the number of polynomials of $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil}$ is extended to be $(d-k+1)^{\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil}$ that is a multiple of $(d-k+1)^{m_f+1}$ (the number of polynomials of $\mathsf{EVENODD}_{m_f+1}$). Therefore, we can recover column f by step 3 in Algorithm 1. It can be counted that the number of polynomials that are downloaded to recover column f is $d(d-k+1)^{\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil}$, which is optimal by (1).

When $f = k, k+1, \ldots, k+r-1$, the repair algorithm of column f is similar to the repair algorithm of an information column. The only difference is that the optimal repair property of parity column is enabled by the first transformation, while

the optimal repair property of information column is enabled by the second transformation.

C. Decoding Method

We present the decoding method of any $\rho \leq r$ erasures for $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil}$ code. 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 $k > \gamma > 0$, $r \geq \delta \geq 0$ 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 $s_1, s_2, \ldots, s_{k-\gamma} \in \mathcal{A}$, and γ parity columns with indices $c_1, c_2, \ldots, c_\gamma \in \mathcal{B}$, and then recover the failure parity column by multiplying the corresponding encoding vector and the information polynomials.

In the construction, $k+r$ columns are divided into $\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil$ partitions with each partition contains $d-k+1$ columns. Columns in each partition are enabled optimal repair access by recursively applying a transformation for each partition. In the decoding procedure, we need to first return to the original EVENODD codes and then decode the failed information polynomials for some rows recursively. The decoding procedure is briefly described as follows.

For $i = 1, 2, \ldots, \gamma$, there exist two integers m_{c_i} and r_{c_i} such that $c_i = (d-k+1)m_{c_i} + r_{c_i}$, where $0 \leq m_{c_i}$ and $0 \leq r_{c_i} \leq d-k$. Similarly, we have $s_i = (d-k+1)m_{s_i} + r_{s_i}$ for $i = 1, 2, \ldots, k-\gamma$, where $0 \leq m_{s_i}$ and $0 \leq r_{s_i} \leq d-k$.

We consider the case that $m_{c_1} \neq m_{c_2} \neq \cdots \neq m_{c_\gamma}$. The parity polynomials are either linear combinations of the corresponding information polynomials or the summations of some linear combinations of the information polynomials. According to Theorem 1, there exists at least one row of the array codes, of which we can first obtain γ syndrome polynomials by subtracting $(k-\gamma)(d-k+1)$ information polynomials from γ parity polynomials and then solve the γ failed information polynomials by computing the $\gamma \times \gamma$ linear equations of the γ syndrome polynomials. Finally, we can recover the failed information polynomials in other rows recursively by subtracting the known information polynomials from the chosen parity polynomials.

For $1 \leq i < j \leq \gamma$, if $m_{c_i} = m_{c_j}$, then we can obtain two parity polynomials of $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil}$ according to the remark at the end of Section II-B1. After solving the parity polynomials of $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil + 1}$ for all i, j with $m_{c_i} = m_{c_j}$, then we can find at least $(d-k+1)^{\lceil \frac{k}{d-k+1} \rceil + m_{c_1}}$ rows such that all the γ parity polynomials in the each of the chosen rows are parity polynomials of $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil}$. By the similar decoding procedure of $m_{c_1} \neq m_{c_2} \neq \cdots \neq m_{c_\gamma}$, all the failed information polynomials can be solved.

D. Example

We present an example of $k=4$, $r=2$, $d=5$ and $e=1$ to illustrate the main ideas. Table II shows the systematic transformed $\mathsf{EVENODD}_1$ code and Table V shows the systematic transformed $\mathsf{EVENODD}_2$ code. While the systematic transformed $\mathsf{EVENODD}_{2+1}$ code is shown in Table VI, and we focus on the transformed $\mathsf{EVENODD}_{2+1}$ code in the following.

1) Decoding Procedure: We claim that we can recover all the information polynomials from any four columns. From the first four columns, we can obtain all the information polynomials directly. Suppose that the data collector connects to three information columns and one parity column, say columns 0, 1, 2 and 5. From columns 0, 1 and 2, one can download information polynomials $a_{0,\ell}(x), a_{1,\ell}(x), a_{2,\ell}(x)$ for $\ell = 0, 1, \ldots, 7$ directly. By subtracting the downloaded information polynomials from the parity polynomials $a_{5,\ell}(x)$, we can obtain the following 8 polynomials

$$\begin{aligned}
&x^2 a_{3,0}(x) + x^{p-1} a_{3,4}(x), x^2 a_{3,1}(x) + x^{p-1} a_{3,5}(x),\\
&(x+x^2) a_{3,0}(x) + x^3 a_{3,2}(x) + (1+x^{p-1}) a_{3,4}(x) + a_{3,6}(x),\\
&(x+x^2) a_{3,1}(x) + x^3 a_{3,3}(x) + (1+x^{p-1}) a_{3,5}(x) + a_{3,7}(x),\\
&x^2 a_{3,4}(x), x^2 a_{3,5}(x), (x+x^2) a_{3,4}(x) + x^3 a_{3,6}(x),\\
&(x+x^2) a_{3,5}(x) + x^3 a_{3,7}(x).
\end{aligned}$$

It is easy to recover the information polynomials $a_{3,\ell}(x)$ for $\ell = 0, 1, \ldots, 7$ from the above polynomials. The decoding of any three information columns and the first parity column is similar.

Suppose that we want to decode the information polynomials from two information columns and two parity columns, say columns 0, 2, 4 and 5. Denote

$$\begin{aligned}
b_0(x) =& a_{0,4}(x) + (x^{p-1} a_{1,4}(x) + x^{p-1} a_{0,5}(x)) + a_{2,4}(x) +\\
&(x^{p-1} a_{3,4}(x) + x^{p-1} a_{2,6}(x)),\\
b_1(x) =& x^{p-1} a_{0,5}(x) + (1+x^{p-1}) a_{1,4}(x) + a_{1,5}(x) +\\
&a_{2,5}(x) + x^{p-1} a_{3,5}(x) + x^{p-1} a_{2,7}(x),\\
b_2(x) =& a_{0,6}(x) + (x^{p-1} a_{1,6}(x) + x^{p-1} a_{0,7}(x)) +\\
&x^{p-1} a_{2,6}(x) + (1+x^{p-1}) a_{3,4}(x) + a_{3,6}(x),\\
b_3(x) =& x^{p-1} a_{0,7}(x) + (1+x^{p-1}) a_{1,6}(x) + a_{1,7}(x) +\\
&x^{p-1} a_{2,7}(x) + (1+x^{p-1}) a_{3,5}(x) + a_{3,7}(x),
\end{aligned}$$

and

$$\begin{aligned}
c_0(x) =& a_{0,0}(x) + a_{1,0}(x) + a_{0,1}(x) + x^2 a_{2,0}(x) +\\
&(x^2 a_{3,0}(x) + x^2 a_{2,2}(x)),\\
c_1(x) =& x^{p-1} a_{0,1}(x) + (1+x^{p-1}) a_{1,0}(x) + x a_{1,1}(x) +\\
&x^2 a_{2,1}(x) + x^2 a_{3,1}(x) + x^2 a_{2,3}(x),\\
c_2(x) =& a_{0,2}(x) + (a_{1,2}(x) + a_{0,3}(x)) + x a_{2,2}(x) +\\
&(x+x^2) a_{3,0}(x) + x^3 a_{3,2}(x)\\
c_3(x) =& x^{p-1} a_{0,3}(x) + (1+x^{p-1}) a_{1,2}(x) + x a_{1,3}(x) +\\
&x a_{2,3}(x) + (x+x^2) a_{3,1}(x) + x^3 a_{3,3}(x).
\end{aligned}$$

TABLE VI: Systematic transformed $\mathsf{EVENODD}_{2+1}$ code with $k=4$, $r=2$, $d=5$ and $e=1$.

Information columns				Parity column 0
$a_{0,0}(x)$	$a_{1,0}(x)$	$a_{2,0}(x)$	$a_{3,0}(x)$	$a_{0,0}(x)+(x^{p-1}a_{1,0}(x)+x^{p-1}a_{0,1}(x))+a_{2,0}(x)+(x^{p-1}a_{3,0}(x)+x^{p-1}a_{2,2}(x))$
$a_{0,1}(x)$	$a_{1,1}(x)$	$a_{2,1}(x)$	$a_{3,1}(x)$	$x^{p-1}a_{0,1}(x)+(1+x^{p-1})a_{1,0}(x)+a_{1,1}(x)+a_{2,1}(x)+x^{p-1}a_{3,1}(x)+x^{p-1}a_{2,3}(x)$
$a_{0,2}(x)$	$a_{1,2}(x)$	$a_{2,2}(x)$	$a_{3,2}(x)$	$a_{0,2}(x)+(x^{p-1}a_{1,2}(x)+x^{p-1}a_{0,3}(x))+x^{p-1}a_{2,2}(x)+(1+x^{p-1})a_{3,0}(x)+a_{3,2}(x)$
$a_{0,3}(x)$	$a_{1,3}(x)$	$a_{2,3}(x)$	$a_{3,3}(x)$	$x^{p-1}a_{0,3}(x)+(1+x^{p-1})a_{1,2}(x)+a_{1,3}(x)+x^{p-1}a_{2,3}(x)+(1+x^{p-1})a_{3,1}(x)+a_{3,3}(x)$
$a_{0,4}(x)$	$a_{1,4}(x)$	$a_{2,4}(x)$	$a_{3,4}(x)$	$a_{0,4}(x)+(x^{p-1}a_{1,4}(x)+x^{p-1}a_{0,5}(x))+a_{2,4}(x)+(x^{p-1}a_{3,4}(x)+x^{p-1}a_{2,6}(x))+$ $(1+x)(a_{0,0}(x)+a_{1,0}(x)+a_{0,1}(x)+x^2a_{2,0}(x)+(x^2a_{3,0}(x)+x^2a_{2,2}(x)))$
$a_{0,5}(x)$	$a_{1,5}(x)$	$a_{2,5}(x)$	$a_{3,5}(x)$	$x^{p-1}a_{0,5}(x)+(1+x^{p-1})a_{1,4}(x)+a_{1,5}(x)+a_{2,5}(x)+x^{p-1}a_{3,5}(x)+x^{p-1}a_{2,7}(x)+$ $(1+x)(x^{p-1}a_{0,1}(x)+(1+x^{p-1})a_{1,0}(x)+xa_{1,1}(x)+x^2a_{2,1}(x)+x^2a_{3,1}(x)+x^2a_{2,3}(x))$
$a_{0,6}(x)$	$a_{1,6}(x)$	$a_{2,6}(x)$	$a_{3,6}(x)$	$a_{0,6}(x)+(x^{p-1}a_{1,6}(x)+x^{p-1}a_{0,7}(x))+x^{p-1}a_{2,6}(x)+(1+x^{p-1})a_{3,4}(x)+a_{3,6}(x)+$ $(1+x)(a_{0,2}(x)+(a_{1,2}(x)+a_{0,3}(x))+xa_{2,2}(x)+(x+x^2)a_{3,0}(x)+x^3a_{3,2}(x))$
$a_{0,7}(x)$	$a_{1,7}(x)$	$a_{2,7}(x)$	$a_{3,7}(x)$	$x^{p-1}a_{0,7}(x)+(1+x^{p-1})a_{1,6}(x)+a_{1,7}(x)+x^{p-1}a_{2,7}(x)+(1+x^{p-1})a_{3,5}(x)+a_{3,7}(x)+$ $(1+x)(x^{p-1}a_{0,3}(x)+(1+x^{p-1})a_{1,2}(x)+xa_{1,3}(x)+xa_{2,3}(x)+(x+x^2)a_{3,1}(x)+x^3a_{3,3}(x))$

Parity column 1
$a_{5,0}(x)=a_{0,0}(x)+a_{1,0}(x)+a_{0,1}(x)+x^2a_{2,0}(x)+(x^2a_{3,0}(x)+x^2a_{2,2}(x))+$ $a_{0,4}(x)+(x^{p-1}a_{1,4}(x)+x^{p-1}a_{0,5}(x))+a_{2,4}(x)+(x^{p-1}a_{3,4}(x)+x^{p-1}a_{2,6}(x))$
$a_{5,1}(x)=x^{p-1}a_{0,1}(x)+(1+x^{p-1})a_{1,0}(x)+xa_{1,1}(x)+x^2a_{2,1}(x)+x^2a_{3,1}(x)+x^2a_{2,3}(x)+$ $x^{p-1}a_{0,5}(x)+(1+x^{p-1})a_{1,4}(x)+a_{1,5}(x)+a_{2,5}(x)+x^{p-1}a_{3,5}(x)+x^{p-1}a_{2,7}(x)$
$a_{5,2}(x)=a_{0,2}(x)+(a_{1,2}(x)+a_{0,3}(x))+xa_{2,2}(x)+(x+x^2)a_{3,0}(x)+x^3a_{3,2}(x)+$ $a_{0,6}(x)+(x^{p-1}a_{1,6}(x)+x^{p-1}a_{0,7}(x))+x^{p-1}a_{2,6}(x)+(1+x^{p-1})a_{3,4}(x)+a_{3,6}(x)$
$a_{5,3}(x)=x^{p-1}a_{0,3}(x)+(1+x^{p-1})a_{1,2}(x)+xa_{1,3}(x)+xa_{2,3}(x)+(x+x^2)a_{3,1}(x)+x^3a_{3,3}(x)+$ $x^{p-1}a_{0,7}(x)+(1+x^{p-1})a_{1,6}(x)+a_{1,7}(x)+x^{p-1}a_{2,7}(x)+(1+x^{p-1})a_{3,5}(x)+a_{3,7}(x)$
$a_{5,4}(x)=a_{0,4}(x)+a_{1,4}(x)+a_{0,5}(x)+x^2a_{2,4}(x)+(x^2a_{3,4}(x)+x^2a_{2,6}(x))$
$a_{5,5}(x)=x^{p-1}a_{0,5}(x)+(1+x^{p-1})a_{1,4}(x)+xa_{1,5}(x)+x^2a_{2,5}(x)+x^2a_{3,5}(x)+x^2a_{2,7}(x)$
$a_{5,6}(x)=a_{0,6}(x)+(a_{1,6}(x)+a_{0,7}(x))+xa_{2,6}(x)+(x+x^2)a_{3,4}(x)+x^3a_{3,6}(x)$
$a_{5,7}(x)=x^{p-1}a_{0,7}(x)+(1+x^{p-1})a_{1,6}(x)+xa_{1,7}(x)+xa_{2,7}(x)+(x+x^2)a_{3,5}(x)+x^3a_{3,7}(x)$

We have that

$$a_{4,4}(x)=b_0(x)+(1+x)c_0(x), a_{5,0}(x)=b_0(x)+c_0(x),$$
$$a_{4,5}(x)=b_1(x)+(1+x)c_1(x), a_{5,1}(x)=b_1(x)+c_1(x),$$
$$a_{4,6}(x)=b_2(x)+(1+x)c_2(x), a_{5,2}(x)=b_2(x)+c_2(x),$$
$$a_{4,7}(x)=b_3(x)+(1+x)c_3(x), a_{5,3}(x)=b_3(x)+c_3(x).$$

Therefore, we can solve $c_\ell(x)$ by

$$c_\ell(x)=x^{p-1}(a_{4,4+\ell}(x)+a_{5,\ell}(x))$$

and $b_\ell(x)$ by $c_\ell(x)+a_{5,\ell}(x)$ for $\ell=0,1,2,3$. Then, we can subtract the information polynomials $a_{0,\ell}(x), a_{2,\ell}(x)$ for $\ell=0,1,\ldots,7$ from the polynomials $a_{4,\ell}(x), a_{5,4+\ell}(x)$, $b_\ell(x), c_\ell(x)$ for $\ell=0,1,2,3$, and obtain the polynomials

$$p_0(x)=x^{p-1}a_{1,0}(x)+x^{p-1}a_{3,0}(x),$$
$$p_1(x)=a_{1,0}(x)+x^2a_{3,0}(x),$$
$$p_2(x)=(1+x^{p-1})a_{1,0}(x)+a_{1,1}(x)+x^{p-1}a_{3,1}(x),$$
$$p_3(x)=(1+x^{p-1})a_{1,0}(x)+xa_{1,1}(x)+x^2a_{3,1}(x),$$
$$p_4(x)=x^{p-1}a_{1,2}(x)+(1+x^{p-1})a_{3,0}(x)+a_{3,2}(x),$$
$$p_5(x)=a_{1,2}(x)+(x+x^2)a_{3,0}(x)+x^3a_{3,2}(x),$$
$$p_6(x)=(1+x^{p-1})a_{1,2}(x)+a_{1,3}(x)+(1+x^{p-1})a_{3,1}(x)+a_{3,3}(x),$$
$$p_7(x)=(1+x^{p-1})a_{1,2}(x)+xa_{1,3}(x)+(x+x^2)a_{3,1}(x)+x^3a_{3,3}(x),$$
$$p_8(x)=a_{1,4}(x)+x^2a_{3,4}(x),$$
$$p_9(x)=x^{p-1}a_{1,4}(x)+x^{p-1}a_{3,4}(x),$$
$$p_{10}(x)=(1+x^{p-1})a_{1,4}(x)+xa_{1,5}(x)+x^2a_{3,5}(x),$$
$$p_{11}(x)=(1+x^{p-1})a_{1,4}(x)+a_{1,5}(x)+x^{p-1}a_{3,5}(x),$$
$$p_{12}(x)=x^{p-1}a_{1,6}(x)+(1+x^{p-1})a_{3,4}(x)+a_{3,6}(x),$$
$$p_{13}(x)=a_{1,6}(x)+(x+x^2)a_{3,4}(x)+x^3a_{3,6}(x),$$
$$p_{14}(x)=(1+x^{p-1})a_{1,6}(x)+a_{1,7}+(1+x^{p-1})a_{3,5}(x)+a_{3,7}(x),$$
$$p_{15}(x)=(1+x^{p-1})a_{1,6}(x)+xa_{1,7}+(x+x^2)a_{3,5}(x)+x^3a_{3,7}(x).$$

We can first compute

$$a_{3,0}(x)=\frac{p_0(x)+x^{p-1}p_1(x)}{x+x^{p-1}}$$

and then compute $a_{1,0}(x)=p_1(x)+x^2a_{3,0}(x)$. The other polynomials

$$a_{1,1}(x), a_{3,1}(x); a_{1,2}(x), a_{3,2}(x); a_{1,3}(x), a_{3,3}(x)$$

can be computed by recursively subtracting the known polynomials from

$$p_2(x), p_3(x); p_4(x), p_5(x); p_6(x), p_7(x).$$

Similarly, polynomials

$$a_{1,4}(x), a_{3,4}(x); a_{1,5}(x), a_{3,5}(x); a_{1,6}(x), a_{3,6}(x); a_{1,7}(x), a_{3,7}(x)$$

can be computed from

$$p_8(x), p_9(x); p_{10}(x), p_{11}(x); p_{12}(x), p_{13}(x); p_{14}(x), p_{15}(x).$$

2) Repair Procedure: Next we show that any one column can be recovered by Algorithm 1 with optimal repair bandwidth. Suppose that column 4 (parity column 0) is failed, i.e., $f=4$. As $k=4$, $r=2$ and $d=5$, we have $m_f=0$ and $r_f=0$ and all the surviving five columns are selected as helper columns. By step 6 of Algorithm 1, we need to download polynomials

$$a_{0,\ell}(x), a_{1,\ell}(x), a_{2,\ell}(x), a_{3,\ell}(x), a_{5,\ell}(x)$$

from columns $0,1,2,3,5$ for $\ell=0,1,2,3$ to recover column 4. First, we can directly compute the following four parity polynomials

$$\begin{aligned}a_{4,0}(x)=&a_{0,0}(x)+(x^{p-1}a_{1,0}(x)+x^{p-1}a_{0,1}(x))+\\&a_{2,0}(x)+(x^{p-1}a_{3,0}(x)+x^{p-1}a_{2,2}(x)),\\a_{4,1}(x)=&x^{p-1}a_{0,1}(x)+(1+x^{p-1})a_{1,0}(x)+a_{1,1}(x)+\\&a_{2,1}(x)+x^{p-1}a_{3,1}(x)+x^{p-1}a_{2,3}(x),\\a_{4,2}(x)=&a_{0,2}(x)+(x^{p-1}a_{1,2}(x)+x^{p-1}a_{0,3}(x))+\\&x^{p-1}a_{2,2}(x)+(1+x^{p-1})a_{3,0}(x)+a_{3,2}(x),\\a_{4,3}(x)=&x^{p-1}a_{0,3}(x)+(1+x^{p-1})a_{1,2}(x)+a_{1,3}(x)+\\&x^{p-1}a_{2,3}(x)+(1+x^{p-1})a_{3,1}(x)+a_{3,3}(x),\end{aligned}$$

from the downloaded information polynomials. Then, we can compute the following four polynomials

$$\begin{aligned}
c_0(x) =& a_{0,0}(x) + a_{1,0}(x) + a_{0,1}(x) + x^2 a_{2,0}(x) + \\
& (x^2 a_{3,0}(x) + x^2 a_{2,2}(x)), \\
c_1(x) =& x^{p-1} a_{0,1}(x) + (1 + x^{p-1}) a_{1,0}(x) + x a_{1,1}(x) + \\
& x^2 a_{2,1}(x) + x^2 a_{3,1}(x) + x^2 a_{2,3}(x), \\
c_2(x) =& a_{0,2}(x) + (a_{1,2}(x) + a_{0,3}(x)) + x a_{2,2}(x) + \\
& (x + x^2) a_{3,0}(x) + x^3 a_{3,2}(x) \\
c_3(x) =& x^{p-1} a_{0,3}(x) + (1 + x^{p-1}) a_{1,2}(x) + x a_{1,3}(x) + \\
& x a_{2,3}(x) + (x + x^2) a_{3,1}(x) + x^3 a_{3,3}(x),
\end{aligned}$$

from the downloaded information polynomials. Lastly, we can recover the polynomials $a_{4,4}(x)$, $a_{4,5}(x)$, $a_{4,6}(x)$ and $a_{4,7}(x)$ by computing

$$\begin{aligned}
a_{4,4}(x) =& a_{5,0}(x) + x c_0(x), \\
a_{4,5}(x) =& a_{5,1}(x) + x c_1(x), \\
a_{4,6}(x) =& a_{5,2}(x) + x c_2(x), \\
a_{4,7}(x) =& a_{5,3}(x) + x c_3(x).
\end{aligned}$$

It can be checked that column 5 can be recovered by downloading $a_{0,\ell}(x)$, $a_{1,\ell}(x)$, $a_{2,\ell}(x)$, $a_{3,\ell}(x)$ and $a_{4,\ell}(x)$ from columns $0, 1, 2, 3, 4$ for $\ell = 4, 5, 6, 7$ according to Algorithm 1. Similarly, we can recover column 2 and column 3 by downloading $a_{0,\ell}(x), a_{1,\ell}(x), a_{3,\ell}(x), a_{4,\ell}(x), a_{5,\ell}(x)$ for $\ell = 0, 2, 4, 6$, and $a_{0,\ell}(x), a_{1,\ell}(x), a_{2,\ell}(x), a_{4,\ell}(x), a_{5,\ell}(x)$ for $\ell = 2, 3, 6, 7$, respectively. According to Algorithm 1, column 0 and column 1 can be recovered by downloading $a_{1,\ell}(x), a_{2,\ell}(x), a_{3,\ell}(x)$, $a_{4,\ell}(x), a_{5,\ell}(x)$ for $\ell = 0, 2, 4, 6$, and $a_{0,\ell}(x), a_{2,\ell}(x), a_{3,\ell}(x), a_{4,\ell}(x), a_{5,\ell}(x)$ for $\ell = 1, 3, 5, 7$, respectively.

IV. Transformation for Other Binary MDS Array Codes

The transformation given in Section II-B can also be employed in other binary MDS array codes, such as RDP and codes in [26], [31]–[33], [35]–[37].

Specifically, for RDP and codes in [26], [35]–[37], the transformation is similar to that of EVENODD in Section III-A. We only need to replace the original EVENODD by the new codes and transform them for $\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil$ times. We can show that the multi-layer transformed codes also have optimal repair access for all columns, as in $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil}$.

For codes with optimal repair access or asymptotically optimal repair access only for information column, such as codes in [31]–[33], [41], [42], we can transform them for $\lceil \frac{r}{d-k+1} \rceil$ times to enable optimal repair access for all r parity columns and the optimal repair property of any information column is preserved. In the following, we take an example with $k = 2$, $r = 2$ and $d = 3$ of the construction in [33] to illustrate how to apply the transformation to obtain the transformed codes with optimal repair access for each of the two parity columns and asymptotically optimal repair access for each of the two information columns.

TABLE VII: An example of the array code in [33] with $k = 2$, $r = 2$, $d = 3$, $p = 3$ and $\tau = 4$.

Column 0	Column 1	Column 2	Column 3
$a_{0,0}$	$a_{0,1}$	$a_{0,2} = a_{0,0} + a_{0,1}$	$a_{0,3} = a_{11,0} + a_{10,1}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2} = a_{1,0} + a_{1,1}$	$a_{1,3} = a_{0,0} + a_{11,1}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2} = a_{2,0} + a_{2,1}$	$a_{2,3} = a_{1,0} + a_{0,1}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2} = a_{3,0} + a_{3,1}$	$a_{3,3} = a_{2,0} + a_{1,1}$
$a_{4,0}$	$a_{4,1}$	$a_{4,2} = a_{4,0} + a_{4,1}$	$a_{4,3} = a_{3,0} + a_{2,1}$
$a_{5,0}$	$a_{5,1}$	$a_{5,2} = a_{5,0} + a_{5,1}$	$a_{5,3} = a_{4,0} + a_{3,1}$
$a_{6,0}$	$a_{6,1}$	$a_{6,2} = a_{6,0} + a_{6,1}$	$a_{6,3} = a_{5,0} + a_{4,1}$
$a_{7,0}$	$a_{7,1}$	$a_{7,2} = a_{7,0} + a_{7,1}$	$a_{7,3} = a_{6,0} + a_{5,1}$

A. Transformation for Array Codes in [33]

The array code in [33] is specified by parameters k, r, d, p and τ. Let $k = 2$, $r = 2$, $d = 3$, $p = 3$ and $\tau = 4$. The array of the example is of size 8×4. The first two columns are information columns that store information bits and the last two columns are parity columns that store parity bits. Let $a_{i,j}$ be the i-th bit in column j, where $i = 0, 1, \ldots, 7$ and $j = 0, 1, 2, 3$. For $j = 0, 1$, we define four *extra bits* $a_{8,j}, a_{9,j}, a_{10,j}, a_{11,j}$ associated with column j as

$$\begin{aligned}
a_{8,j} =& \quad a_{0,j} + a_{4,j}, \\
a_{9,j} =& \quad a_{1,j} + a_{5,j}, \\
a_{10,j} =& \quad a_{2,j} + a_{6,j}, \\
a_{11,j} =& \quad a_{3,j} + a_{7,j}.
\end{aligned} \tag{11}$$

Given the information bits, the parity bits $a_{0,2}, a_{1,2}, \ldots, a_{7,2}$ in column 2 are computed by

$$a_{i,2} = a_{i,0} + a_{i,1} \text{ for } i = 0, 1, \ldots, 7,$$

and the parity bits $a_{0,3}, a_{1,3}, \ldots, a_{7,3}$ in column 3 are computed by

$$a_{i,3} = a_{i-1,0} + a_{i-2,1} \text{ for } i = 0, 1, \ldots, 7.$$

Note that all the subscripts in the example are computed by modulo 12. Table VII shows the example. Similar to the information column, we also define four *extra bits* $a_{8,j}, a_{9,j}, a_{10,j}, a_{11,j}$ associated with column j as in (11). Note that we only store eight bits $a_{0,j}, a_{1,j}, \ldots, a_{7,j}$ in column j, where $j = 0, 1, 2, 3$, and we can compute the extra bits when necessary by (11).

We can repair four bits $a_{0,0}, a_{2,0}, a_{4,0}, a_{6,0}$ in column 0 by

$$\begin{aligned}
a_{0,0} =& a_{0,1} + a_{0,2}, \text{ where } a_{0,2} = a_{0,0} + a_{0,1}, \\
a_{2,0} =& a_{2,1} + a_{2,2}, \text{ where } a_{2,2} = a_{2,0} + a_{2,1}, \\
a_{4,0} =& a_{4,1} + a_{4,2}, \text{ where } a_{4,2} = a_{4,0} + a_{4,1}, \\
a_{6,0} =& a_{6,1} + a_{6,2}, \text{ where } a_{6,2} = a_{6,0} + a_{6,1},
\end{aligned}$$

and the other bits $a_{1,0}, a_{3,0}, a_{5,0}, a_{7,0}$ in column 0 by

$$\begin{aligned}
a_{1,0} =& a_{0,1} + a_{2,3}, \text{ where } a_{2,3} = a_{1,0} + a_{0,1}, \\
a_{3,0} =& a_{2,1} + a_{4,3}, \text{ where } a_{4,3} = a_{3,0} + a_{2,1}, \\
a_{5,0} =& a_{4,1} + a_{6,3}, \text{ where } a_{6,3} = a_{5,0} + a_{4,1}, \\
a_{7,0} =& a_{6,1} + a_{4,3} + a_{0,3}, \text{ where } a_{4,3} = a_{3,0} + a_{2,1} \\
& \text{and } a_{0,3} = a_{11,0} + a_{10,1}.
\end{aligned}$$

We need to download 12 bits

$$a_{0,1}, a_{2,1}, a_{4,1}, a_{6,1}, a_{0,2}, a_{2,2}, a_{4,2}, a_{6,2}, a_{0,3}, a_{2,3}, a_{4,3}, a_{6,3},$$

to recover the eight bits in column 0. Therefore, the repair access of column 0 is optimal according to (1). Column 1 can be recovered by downloading 14 bits

$$a_{0,0}, a_{1,0}, a_{3,0}, a_{4,0}, a_{5,0}, a_{7,0}, a_{0,2}, a_{1,2},$$
$$a_{4,2}, a_{5,2}, a_{0,3}, a_{1,3}, a_{4,3}, a_{5,3},$$

which are two bits more than the optimal repair access.

We argue that we can recover the information bits from any two columns. We can directly obtain the information bits from column 0 and column 1. We can also obtain the information bits from any one information column and any one parity column. For example, if we want to decode the information bits from column 0 and column 2, we can subtract $a_{i,0}$ from $a_{i,2}$ to obtain $a_{i,1}$, for $i = 0, 1, \ldots, 7$. Finally, we can decode the information bits from column 2 and column 3 as follows. We can compute $a_{7,1}$ by

$$a_{7,1} = a_{0,2} + a_{1,2} + a_{2,2} + a_{3,2} + a_{1,3} + a_{2,3} + a_{3,3} + a_{4,3}.$$

Similarly, we can compute $a_{8,1}, a_{9,1}, a_{10,1}$ by

$$\begin{aligned} a_{8,1} =& a_{1,2} + a_{2,2} + a_{3,2} + a_{4,2} + a_{2,3} + a_{3,3} + a_{4,3} + a_{5,3}, \\ a_{9,1} =& a_{2,2} + a_{3,2} + a_{4,2} + a_{5,2} + a_{3,3} + a_{4,3} + a_{5,3} + a_{6,3}, \\ a_{10,1} =& a_{3,2} + a_{4,2} + a_{5,2} + a_{6,2} + a_{4,3} + a_{5,3} + a_{6,3} + a_{7,3}. \end{aligned}$$

Then, we can compute the other information bits iteratively.

Next, we show how to apply the first transformation for the above example to obtain the transformed codes that have optimal repair access for each of columns 2 and 3, while the efficient repair property of each of columns 0 and 1 is also preserved. For $j = 0, 1, 2, 3$, we can represent the eight bits in column j and the four extra bits associated with column j by polynomial

$$a_j(x) = a_{0,j} + a_{1,j}x + \ldots + a_{11,j}x^{11}.$$

First, we can generate two instances $a_0(x), a_1(x), a_2(x), a_3(x)$ and $b_0(x), b_1(x), b_2(x), b_3(x)$. Then, we compute polynomials $b_2(x) + x^4 a_3(x), a_3(x) + b_2(x)$ over $\mathbb{F}_2[x]/(1 + x^{12})$. For $j = 0, 1$, column j stores the eight coefficients of degrees from zero to seven of the polynomials $a_j(x)$ and $b_j(x)$; while column 2 and column 3 stores the eight coefficients of degrees from zero to seven of the polynomials $a_2(x), b_2(x) + x^4 a_3(x)$ and $a_3(x) + b_2(x), b_3(x)$, respectively. Table VIII shows the transformed array codes.

First, we show that the efficient repair property of any one information column is preserved in the transformed array codes. Consider the repair method of column 0. We can repair column 0 by downloading 24 bits

$$a_{0,1}, a_{2,1}, a_{4,1}, a_{6,1}, a_{0,2}, a_{2,2}, a_{4,2}, a_{6,2}, a_{0,3} + b_{0,2}, a_{2,3} + b_{2,2},$$
$$a_{4,3} + b_{4,2}, a_{6,3} + b_{6,2}, b_{0,1}, b_{2,1}, b_{4,1}, b_{6,1}, b_{0,2} + a_{8,3},$$
$$b_{2,2} + a_{10,3}, b_{4,2} + a_{0,3}, b_{6,2} + a_{2,3}, b_{0,3}, b_{2,3}, b_{4,3}, b_{6,3}.$$

Specifically, we can compute the four bits $a_{0,0}$, $a_{2,0}$, $a_{4,0}$, $a_{6,0}$ in column 0 by

$$\begin{aligned} a_{0,0} =& a_{0,1} + a_{0,2}, \text{ where } a_{0,2} = a_{0,0} + a_{0,1}, \\ a_{2,0} =& a_{2,1} + a_{2,2}, \text{ where } a_{2,2} = a_{2,0} + a_{2,1}, \\ a_{4,0} =& a_{4,1} + a_{4,2}, \text{ where } a_{4,2} = a_{4,0} + a_{4,1}, \\ a_{6,0} =& a_{6,1} + a_{6,2}, \text{ where } a_{6,2} = a_{6,0} + a_{6,1}, \end{aligned}$$

and $a_{1,0}$, $a_{3,0}$, $a_{5,0}$, $a_{7,0}$ in column 0 by

$$\begin{aligned} a_{3,0} =& (b_{0,2} + a_{8,3}) + (a_{0,3} + b_{0,2}) + b_{2,1}, \\ a_{5,0} =& (b_{2,2} + a_{10,3}) + (a_{2,3} + b_{2,2}) + b_{4,1}, \\ a_{7,0} =& (b_{4,2} + a_{0,3}) + (a_{4,3} + b_{4,2}) + b_{6,1}, \\ a_{1,0} =& (b_{6,2} + a_{2,3}) + (a_{6,3} + b_{6,2}) + a_{0,1} + a_{4,1} + a_{5,0}. \end{aligned}$$

Similarly, we can compute $b_{1,0}, b_{3,0}, b_{5,0}, b_{7,0}$ in column 0 by

$$\begin{aligned} b_{1,0} =& b_{0,1} + b_{2,3}, \text{ where } b_{2,3} = b_{1,0} + b_{0,1}, \\ b_{3,0} =& b_{2,1} + b_{4,3}, \text{ where } b_{4,3} = b_{3,0} + b_{2,1}, \\ b_{5,0} =& b_{4,1} + b_{6,3}, \text{ where } b_{6,3} = b_{5,0} + b_{4,1}, \\ b_{7,0} =& b_{6,1} + b_{0,3} + b_{4,3}, \text{ where } b_{4,3} = b_{3,0} + b_{2,1} \\ & \text{and } b_{0,3} = b_{11,0} + b_{10,1}, \end{aligned}$$

and $b_{0,0}, b_{2,0}, b_{4,0}, b_{6,0}$ in column 0 by

$$\begin{aligned} b_{0,0} =& (b_{0,2} + a_{8,3}) + b_{0,1} + a_{7,0} + a_{6,1}, \\ b_{2,0} =& (b_{2,2} + a_{10,3}) + b_{2,1} + a_{1,0} + a_{5,0} + a_{0,1} + a_{4,1}, \\ b_{4,0} =& (b_{4,2} + a_{0,3}) + b_{4,1} + a_{3,0} + a_{7,0} + a_{2,1} + a_{6,1}, \\ b_{6,0} =& (b_{6,2} + a_{2,3}) + b_{6,1} + a_{1,0} + a_{0,1}. \end{aligned}$$

The repair access of column 0 is also optimal. Column 1 can be recovered by downloading 28 bits

$$a_{0,0}, a_{1,0}, a_{3,0}, a_{4,0}, a_{5,0}, a_{7,0}, a_{0,2}, a_{1,2}, a_{4,2}, a_{5,2}, a_{0,3} + b_{0,2},$$
$$a_{1,3} + b_{1,2}, a_{4,3} + b_{4,2}, a_{5,3} + b_{5,2}, b_{0,0}, b_{1,0}, b_{3,0}, b_{4,0}, b_{5,0}, b_{7,0},$$
$$b_{0,2} + a_{8,3}, b_{1,2} + a_{9,3}, b_{4,2} + a_{0,3}, b_{5,2} + a_{1,3}, b_{0,3}, b_{1,3}, b_{4,3}, b_{5,3},$$

which are four bits more than the optimal repair access.

According to Theorem 3, the repair access of each of column 2 and column 3 is optimal. We can repair column 2 by downloading 24 bits

$$a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0}, a_{4,0}, a_{5,0}, a_{6,0}, a_{7,0},$$
$$a_{0,1}, a_{1,1}, a_{2,1}, a_{3,1}, a_{4,1}, a_{5,1}, a_{6,1}, a_{7,1},$$
$$a_{0,3} + b_{0,2}, a_{1,3} + b_{1,2}, a_{2,3} + b_{2,2}, a_{3,3} + b_{3,2},$$
$$a_{4,3} + b_{4,2}, a_{5,3} + b_{5,2}, a_{6,3} + b_{6,2}, a_{7,3} + b_{7,2},$$

from columns 0, 1 and 3, and repair column 3 by downloading 24 bits

$$b_{0,0}, b_{1,0}, b_{2,0}, b_{3,0}, b_{4,0}, b_{5,0}, b_{6,0}, b_{7,0},$$
$$b_{0,1}, b_{1,1}, b_{2,1}, b_{3,1}, b_{4,1}, b_{5,1}, b_{6,1}, b_{7,1},$$
$$b_{0,2} + a_{8,3}, b_{1,2} + a_{9,3}, b_{2,2} + a_{10,3}, b_{3,2} + a_{11,3},$$
$$b_{4,2} + a_{0,3}, b_{5,2} + a_{1,3}, b_{6,2} + a_{2,3}, b_{7,2} + a_{3,3},$$

from columns 0, 1 and 2.

For the MDS array codes [31]–[33] with general parameters k and r, each column has $(p-1)\tau$ bits, we can choose p to make the array codes satisfy the MDS

TABLE VIII: The transformed array code with $k=2$, $r=2$, $d=3$, $p=3$ and $\tau=4$.

Column 0	Column 1	Column 2	Column 3
$a_{0,0}$ $b_{0,0}$	$a_{0,1}$ $b_{0,1}$	$a_{0,2}=a_{0,0}+a_{0,1}$ $b_{0,2}+a_{8,3}=b_{0,0}+b_{0,1}+a_{7,0}+a_{6,1}$	$a_{0,3}+b_{0,2}=a_{11,0}+a_{10,1}+b_{0,0}+b_{0,1}$ $b_{0,3}=b_{11,0}+b_{10,1}$
$a_{1,0}$ $b_{1,0}$	$a_{1,1}$ $b_{1,1}$	$a_{1,2}=a_{1,0}+a_{1,1}$ $b_{1,2}+a_{9,3}=b_{1,0}+b_{1,1}+a_{8,0}+a_{7,1}$	$a_{1,3}+b_{1,2}=a_{0,0}+a_{11,1}+b_{1,0}+b_{1,1}$ $b_{1,3}=b_{0,0}+b_{11,1}$
$a_{2,0}$ $b_{2,0}$	$a_{2,1}$ $b_{2,1}$	$a_{2,2}=a_{2,0}+a_{2,1}$ $b_{2,2}+a_{10,3}=b_{2,0}+b_{2,1}+a_{9,0}+a_{8,1}$	$a_{2,3}+b_{2,2}=a_{1,0}+a_{0,1}+b_{2,0}+b_{2,1}$ $b_{2,3}=b_{1,0}+b_{0,1}$
$a_{3,0}$ $b_{3,0}$	$a_{3,1}$ $b_{3,1}$	$a_{3,2}=a_{3,0}+a_{3,1}$ $b_{3,2}+a_{11,3}=b_{3,0}+b_{3,1}+a_{10,0}+a_{9,1}$	$a_{3,3}+b_{3,2}=a_{2,0}+a_{1,1}+b_{3,0}+b_{3,1}$ $b_{3,3}=b_{2,0}+b_{1,1}$
$a_{4,0}$ $b_{4,0}$	$a_{4,1}$ $b_{4,1}$	$a_{4,2}=a_{4,0}+a_{4,1}$ $b_{4,2}+a_{0,3}=b_{4,0}+b_{4,1}+a_{11,0}+a_{10,1}$	$a_{4,3}+b_{4,2}=a_{3,0}+a_{2,1}+b_{4,0}+b_{4,1}$ $b_{4,3}=b_{3,0}+b_{2,1}$
$a_{5,0}$ $b_{5,0}$	$a_{5,1}$ $b_{5,1}$	$a_{5,2}=a_{5,0}+a_{5,1}$ $b_{5,2}+a_{1,3}=b_{5,0}+b_{5,1}+a_{0,0}+a_{11,1}$	$a_{5,3}+b_{5,2}=a_{4,0}+a_{3,1}+b_{5,0}+b_{5,1}$ $b_{5,3}=b_{4,0}+b_{3,1}$
$a_{6,0}$ $b_{6,0}$	$a_{6,1}$ $b_{6,1}$	$a_{6,2}=a_{6,0}+a_{6,1}$ $b_{6,2}+a_{2,3}=b_{6,0}+b_{6,1}+a_{1,0}+a_{0,1}$	$a_{6,3}+b_{6,2}=a_{5,0}+a_{4,1}+b_{6,0}+b_{6,1}$ $b_{6,3}=b_{5,0}+b_{4,1}$
$a_{7,0}$ $b_{7,0}$	$a_{7,1}$ $b_{7,1}$	$a_{7,2}=a_{7,0}+a_{7,1}$ $b_{7,2}+a_{3,3}=b_{7,0}+b_{7,1}+a_{2,0}+a_{1,1}$	$a_{7,3}+b_{7,2}=a_{6,0}+a_{5,1}+b_{7,0}+b_{7,1}$ $b_{7,3}=b_{6,0}+b_{5,1}$

property and choose τ to achieve asymptotically optimal repair bandwidth for each of the k information columns. For $j=0,1,\ldots,k+r-1$, we can represent the $(p-1)\tau$ bits $a_{0,j},a_{1,j},\ldots,a_{(p-1)\tau-1,j}$ in column j and τ extra bits $a_{(p-1)\tau,j},a_{(p-1)\tau+1,j},\ldots,a_{p\tau-1,j}$ associated with column j by polynomial $a_j(x)=\sum_{i=0}^{p\tau-1}a_{i,j}x^i\in\mathbb{F}_2[x]/(1+x^{p\tau})$, where the extra bit $a_{(p-1)\tau+\mu,j}$ with $\mu=0,1,\ldots,\tau-1$ is computed by

$$a_{(p-1)\tau+\mu,j}=a_{\mu,j}+a_{\tau+\mu,j}+\ldots+a_{(p-2)\tau+\mu,j}.$$

If we apply the first transformation with encoding coefficient being x^e for the columns from k to d of the MDS array codes, we can obtain the transformed codes with each column containing $d-k+1$ polynomials. We should carefully choose the encoding coefficient in the transformation in order to make sure that the efficient repair property of any information column of original MDS array codes is preserved in the transformed MDS array codes. In the example with $k=2$, $r=2$ and $d=3$, we choose the encoding coefficient of the transformation to be x^4. In fact, the efficient repair property of any information column is also maintained if the encoding coefficient is any polynomial of $\{1+x^4,x^8,1+x^8,x^4+x^8,1+x^4+x^8\}$. However, the efficient repair property of any information column is not maintained if the encoding coefficient is other polynomial in $\mathbb{F}_2[x]/(1+x^{12})$.

Note that the following two properties are the essential reasons to preserve the efficient repair property. First, there is a cyclic structure in the ring $\mathbb{F}_2[x]/(1+x^{12})$. The multiplication of x^4 and the polynomial $a_3(x)$ in $\mathbb{F}_2[x]/(1+x^{12})$ can be implemented by cyclicly shifting 4 positions of $a_3(x)$. Second, the exponent of the encoding coefficient of the transformation, $e=4$ is a multiple of two. Otherwise, the efficient repair property of original array codes is not maintained. In the example, we have $d=k+r-1$, i.e., all the surviving columns are connected to recover a failure column. By applying one transformation for the r parity columns, the transformed array codes will have asymptotically or exactly optimal repair for any single column. However, if $d<k+r-1$, then we may need to employ the transformation for many times, as like the transformation for EVENODD codes. When we apply multiple transformations for the array codes in [31]–[33], we should not only carefully choose the encoding coefficient but also the transformed columns in each transformation, in order to preserve the efficient repair property of the information column.

B. Transformation for EVENODD to Preserve the Efficient Repair Property of Any Information Column

The number of symbols stored in each column or node is also referred to as the *sub-packetization level*. It is important to have a low sub-packetization level for practical consideration. It is shown in [43] that the lower bound of sub-packetization of optimal access MDS codes over finite field with $d=n-1$ is $r^{(k-1)/r}$. The sub-packetization of MDS code constructions over finite field with optimal repair access presented in [16], [17] is $(n-k)^{\lceil\frac{n}{n-k}\rceil}$, for $d=n-1$. ϵ-minimum storage regenerating (ϵ-MSR) codes are proposed in [44] to reduce the sub-packetization at a cost of slightly more repair bandwidth. Existing constructions [33] of binary MDS array codes with $d=k+1$ and asymptotically optimal repair access for any single information column show that the sub-packetization is strictly less than $p\cdot 2^{\frac{k}{r-1}+r-1}$ [33, Theorem 2], where p is a prime and constant number. The existing constructions of MDS codes with asymptotically or exactly optimal repair access have an exponential sub-packetization level. The construction of MDS codes with efficient repair for any column with lower sub-packetization level is attractive. In the following, we take EVENODD codes with $r=2$ as an example to show how to design new transformation to enable optimal repair for any single parity column and the repair access of any single information column is roughly $3/4$ of all the information bits, and thus the sub-packetization level is low.

Consider the example of EVENODD codes with $k=3$, $r=2$ and $p=5$. We have $k=3$ information columns and $r=2$ parity columns. Let $a_{0,j},a_{1,j},a_{2,j},a_{3,j}$ be the four bits in column j, where $j=0,1,2,3,4$. Table IX shows the example.

When we say one information bit is repaired by a parity column (the first parity column or the second parity column), it means that we repair the bit by downloading the parity bit in the parity column that contains the failed information bits and all

TABLE IX: The EVENODD code with $k = 3$, $r = 2$ and $p = 5$.

Column 0	Column 1	Column 2	Column 3	Column 4
$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3} = a_{0,0} + a_{0,1} + a_{0,2}$	$a_{0,4} = a_{0,0} + a_{3,2} + (a_{3,1} + a_{2,2})$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3} = a_{1,0} + a_{1,1} + a_{1,2}$	$a_{1,4} = a_{1,0} + a_{0,1} + (a_{3,1} + a_{2,2})$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3} = a_{2,0} + a_{2,1} + a_{2,2}$	$a_{2,4} = a_{2,0} + a_{1,1} + a_{0,2} + (a_{3,1} + a_{2,2})$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3} = a_{3,0} + a_{3,1} + a_{3,2}$	$a_{3,4} = a_{3,0} + a_{2,1} + a_{1,2} + (a_{3,1} + a_{2,2})$

the information bits that are used to compute the downloaded parity bit except the failed information bit. For example, the bit $a_{0,0}$ is repaired by the first parity column, which means that we download the parity bit $a_{0,3} = a_{0,0} + a_{0,1} + a_{0,2}$ in the first parity column and two information bits $a_{0,1}, a_{0,2}$ to recover the information bit $a_{0,0}$. According to the repair method given in [27], we can repair two information bits of the failed information column by the first parity column and the other information bits by the second parity column. Consider column 1. We can repair $a_{0,1}, a_{1,1}$ by

$$\begin{aligned} a_{0,1} =& a_{0,0} + a_{0,2} + a_{0,3}, \text{ where } a_{0,3} = a_{0,0} + a_{0,1} + a_{0,2}, \\ a_{1,1} =& a_{1,0} + a_{1,2} + a_{1,3}, \text{ where } a_{1,3} = a_{1,0} + a_{1,1} + a_{1,2}, \end{aligned}$$

and repair $a_{2,1}, a_{3,1}$ by

$$\begin{aligned} a_{3,1} =& a_{1,0} + a_{0,1} + a_{2,2} + a_{1,4}, \\ & \text{where } a_{1,4} = a_{1,0} + a_{0,1} + a_{3,1} + a_{2,2}, \\ a_{2,1} =& a_{3,0} + a_{1,2} + a_{3,1} + a_{2,2} + a_{3,4}, \\ & \text{where } a_{3,4} = a_{3,0} + a_{2,1} + a_{1,2} + a_{3,1} + a_{2,2}. \end{aligned}$$

We need to download 10 bits to recover column 1, i.e., the repair bandwidth of column 1 is roughly $3/4$ of all 12 information bits.

Next, we present the transformation for general parameters k and p of EVENODD codes with $r = 2$ and $d = k + 1$. Each column of the transformed EVENODD codes has $2(p-1)$ bits. The transformed EVENODD codes have optimal repair bandwidth for each parity column and the repair bandwidth of each information column is roughly $3/4$ of all the information bits.

Create two instances of EVENODD codes $a_0(x)$, $a_1(x), \ldots, a_{k+1}(x)$ and $b_0(x), b_1(x), \ldots, b_{k+1}(x)$, where $a_j(x) = \sum_{i=0}^{p-2} a_{i,j} x^i$ and $b_j(x) = \sum_{i=0}^{p-2} b_{i,j} x^i$. The information polynomials are $a_0(x), a_1(x), \ldots, a_{k-1}(x)$ and $b_0(x), b_1(x), \ldots, b_{k-1}(x)$, and the parity polynomials are computed by

$$\begin{aligned} &\begin{bmatrix} a_k(x) & a_{k+1}(x) \\ b_k(x) & b_{k+1}(x) \end{bmatrix} \\ &= \begin{bmatrix} a_0(x) & a_1(x) & \cdots & a_{k-1}(x) \\ b_0(x) & b_1(x) & \cdots & b_{k-1}(x) \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & x \\ \vdots & \vdots \\ 1 & x^{k-1} \end{bmatrix}. \end{aligned}$$

Let $\mathbf{a_j} = [a_{0,j}, a_{1,j}, \ldots, a_{p-2,j}]^T$ and $\mathbf{b_j} = [b_{0,j}, b_{1,j}, \ldots, b_{p-2,j}]^T$ be the coefficients of polynomials $a_j(x)$ and $b_j(x)$, respectively, where $j = 0, 1, \ldots, k+1$. Given a column vector $\mathbf{a_0}$, we define

$$\begin{aligned} \mathbf{a_0^*} =& [a_{1,0}, a_{0,0}, a_{3,0}, a_{2,0}, \ldots, a_{p-2,0}, a_{p-3,0}]^T, \\ \bar{\mathbf{a}}_\mathbf{0} =& [a_{0,0}, 0, a_{2,0}, 0, \ldots, a_{p-3,0}, 0]^T. \end{aligned}$$

The summation of two column vectors $\mathbf{a_0}, \mathbf{a_1}$ is define by

$$\mathbf{a_0} \oplus \mathbf{a_1} = [a_{0,0} + a_{0,1}, a_{1,0} + a_{1,1}, \ldots, a_{p-2,0} + a_{p-2,1}]^T.$$

For example, when $p = 5$, we have

$$\mathbf{a_0} \oplus \mathbf{a_1} = [a_{0,0} + a_{0,1}, a_{1,0} + a_{1,1}, a_{2,0} + a_{2,1}, a_{3,0} + a_{3,1}]^T,$$

and

$$\begin{aligned} \mathbf{a_0^*} =& [a_{1,0}, a_{0,0}, a_{3,0}, a_{2,0}]^T, \\ \bar{\mathbf{a}}_\mathbf{0} =& [a_{0,0}, 0, a_{2,0}, 0]^T. \end{aligned}$$

For $j = 0, 1, \ldots, k-1$, column j stores $2(p-1)$ information bits $\mathbf{a_j}, \mathbf{b_j}$. The first parity column stores $2(p-1)$ parity bits

$$\begin{aligned} &\mathbf{a_k} \oplus \mathbf{b_k} = [a_{0,k} + b_{0,k}, a_{1,k} + b_{1,k}, \ldots, a_{p-2,k} + b_{p-2,k}]^T, \\ &\mathbf{b_{k+1}} = [b_{0,k+1}, b_{1,k+1}, \ldots, b_{p-2,k+1}]^T, \end{aligned}$$

and the second parity column stores $2(p-1)$ parity bits

$$\begin{aligned} \mathbf{a_k} \oplus \bar{\mathbf{b}}_\mathbf{k} \oplus \mathbf{b_k^*} =& [a_{0,k} + b_{0,k} + b_{1,k}, a_{1,k} + b_{0,k}, \ldots, \\ & a_{p-3,k} + b_{p-3,k} + b_{p-2,k}, a_{p-2,k} + b_{p-3,k}]^T, \\ \mathbf{a_{k+1}} =& [a_{0,k+1}, a_{1,k+1}, \ldots, a_{p-2,k+1}]^T. \end{aligned}$$

We show that the transformed EVENODD codes satisfy MDS property, i.e., we can retrieve all $2k(p-1)$ information bits from any k columns. Consider the k columns from columns 2 to $k+1$. First, we can compute $(p-1)/2$ bits $b_{i,k}$ by $(a_{i-1,k} + b_{i-1,k}) + (a_{i-1,k} + b_{i-1,k} + b_{i,k})$ for $i = 1, 3, \ldots, p-2$ and compute $a_{i,k}$ by $b_{i,k} + (a_{i,k} + b_{i,k})$ for $i = 1, 3, \ldots, p-2$. Then, we can compute $b_{i,k}$ by $a_{i+1,k} + (a_{i+1,k} + b_{i,k})$ for $i = 0, 2, \ldots, p-3$ and compute $a_{i,k}$ by $b_{i,k} + (a_{i,k} + b_{i,k})$ for $i = 0, 2, \ldots, p-3$. Finally, we can obtain the information bits $b_{0,0}, b_{1,0}, \ldots, b_{p-2,0}$ and $b_{0,1}, b_{1,1}, \ldots, b_{p-2,1}$ from $b_{0,j}, b_{1,j}, \ldots, b_{p-2,j}$ for $j = 2, 3, \ldots, k+1$, as the EVENODD code is MDS code. The information bits $a_{0,0}, a_{1,0}, \ldots, a_{p-2,0}$ and $a_{0,1}, a_{1,1}, \ldots, a_{p-2,1}$ can be computed similarly. We can also retrieve all information bits from any $k-1$ information columns and any one parity column. Consider the k columns from column 1 to k. We can obtain $k(p-1)$ bits $b_{0,j}, b_{1,j}, \ldots, b_{p-2,j}$ for $j = 1, 2, \ldots, k$ from column 1 to k, and compute the information bits $b_{0,0}, b_{1,0}, \ldots, b_{p-2,0}$. Then, we can compute $b_{i,k}$ from the information bits $b_{i,0}, b_{i,1}, \ldots, b_{i,k-1}$, and compute $a_{i,k}$ by $a_{i,k} = b_{i,k} + (a_{i,k} + b_{i,k})$ for $i = 0, 1, \ldots, p-2$. Together with $(k-1)(p-1)$ bits $a_{0,j}, a_{1,j}, \ldots, a_{p-2,j}$ with $j = 1, 2, \ldots, k$ from column 1 to k, we can compute $p-1$ information bits $a_{0,0}, a_{1,0}, \ldots, a_{p-2,0}$. The decoding method from any $k-1$ information columns plus any one parity column is similar.

Each parity column of the transformed EVENODD codes has optimal repair access. We can repair column k by downloading $\mathbf{b_j}$ from column j for $j = 0, 1, \ldots, k-1$ and $\mathbf{a_k} \oplus \bar{\mathbf{b}}_\mathbf{k} \oplus \mathbf{b_k^*}$ from column $k+1$. Specifically, we can compute $\mathbf{b_k}, \mathbf{b_{k+1}}$

from $\mathbf{b_0}, \mathbf{b_1}, \ldots, \mathbf{b_{k-1}}$, and $\mathbf{a_k} \oplus \mathbf{b_k}$ by $(\mathbf{a_k} \oplus \bar{\mathbf{b}}_\mathbf{k} \oplus \mathbf{b}^*_\mathbf{k}) \oplus \mathbf{b}^*_\mathbf{k} \oplus \bar{\mathbf{b}}_\mathbf{k} \oplus \mathbf{b_k}$. Similarly, we can repair column $k+1$ by downloading $\mathbf{a_j}$ from column j for $j = 0, 1, \ldots, k-1$ and $\mathbf{a_k} \oplus \mathbf{b_k}$ from column k. In the next theorem, we show that the efficient repair property of any single information column of EVENODD codes is preserved in the transformed EVENODD codes.

Theorem 5. *In the $(k+2, k)$ EVENODD codes, suppose that $p-1$ is a multiple of four and we can download the bits $a_{i,j}$ for all $i \in S_j$ and $j = 0, 1, \ldots, f-1, f+1, \ldots, k+1$ to recover column f, where $0 \leq f \leq k-1$, S_j denotes the set of indices of the downloaded bits from column j and $S_k = \{0, 1, \ldots, (p-1)/2-1\}$. Then, column f of the transformed EVENODD codes can be recovered by downloading $a_{i,j}, b_{i,j}$ for all $i \in S_j$ from column j for $j = 0, 1, \ldots, f-1, f+1, \ldots, k-1$, $a_{i,k} + b_{i,k}$ for all $i \in S_k = \{0, 1, \ldots, (p-1)/2-1\}$ and $b_{i,k+1}$ for all $i \in S_{k+1}$ from column k, $a_{i,k} + b_{i,k} + b_{i+1,k}$ and $a_{i+1,k} + b_{i,k}$ for all $i \in \{0, 2, \ldots, (p-1)/2-2\}$ and $a_{i,k+1}$ for all $i \in S_{k+1}$ from column $k+1$.*

Proof. Consider the repair of column f for the transformed EVENODD codes. We have received the following bits

Column 0	$a_{i,0}\ \forall i \in S_0$ and $b_{i,0}\ \forall i \in S_0$
$\ldots$	$\ldots$
Column $k-1$	$a_{i,k-1}\ \forall i \in S_{k-1}$ and $b_{i,k-1}\ \forall i \in S_{k-1}$
Column k	$a_{i,k} + b_{i,k}\ \forall i \in S_k$ and $b_{i,k+1}\ \forall i \in S_{k+1}$
Column $k+1$	$a_{i,k} + b_{i,k} + b_{i+1,k}, a_{i+1,k} + b_{i,k}$ $\forall i \in \{0, 2, \ldots, \frac{p-5}{2}\}$ and $a_{i,k+1}\ \forall i \in S_{k+1}$

We can calculate $b_{i,k}$ by $(a_{i-1,k} + b_{i-1,k}) + (a_{i-1,k} + b_{i-1,k} + b_{i,k})$ for $i = 1, 3, \ldots, (p-1)/2-1$, and $a_{i,k}$ by $b_{i,k} + (a_{i,k} + b_{i,k})$ for $i = 1, 3, \ldots, (p-1)/2-1$. Then, we can compute $b_{i,k}$ by $a_{i+1,k} + (a_{i+1,k} + b_{i,k})$ for $i = 0, 2, \ldots, (p-5)/2$ and $a_{i,k}$ by $b_{i,k} + (a_{i,k} + b_{i,k})$ for $i = 0, 2, \ldots, (p-5)/2$. We thus obtain $a_{i,k}$ and $b_{i,k}$ for all $i \in S_k = \{0, 1, \ldots, (p-1)/2-1\}$. Recall that we can recover $a_{0,f}, a_{1,f}, \ldots, a_{p-2,f}$ by downloading the bits $a_{i,j}$ for all $i \in S_j$ and $j = 0, 1, \ldots, f-1, f+1, \ldots, k+1$. Therefore, we obtain the bits $a_{i,j}, b_{i,j}$ for all $i \in S_j$ and $j = 0, 1, \ldots, f-, f+1, \ldots, k+1$, and the bits $a_{0,f}, a_{1,f}, \ldots, a_{p-2,f}$ and $b_{0,f}, b_{1,f}, \ldots, b_{p-2,f}$ in column f of the transformed EVENODD codes can be recovered. □

By Theorem 5, the efficient repair property of any information column of EVENODD codes with $r = 2$ is preserved after the transformation, if $p-1$ is a multiple of four. When $r \geq 3$, the repair method of information column of EVENODD codes is different from that of EVENODD codes with $r = 2$. We need to design new transformation carefully to preserve the efficient repair property of information column and that will be our future work.

Table X shows an example of the transformed code with $k = 3$, $r = 2$ and $p = 5$. When $f = 1$, we have $S_0 = \{0, 1, 3\}$, $S_2 = \{0, 1, 2\}$, $S_3 = \{0, 1\}$ and $S_4 = \{1, 3\}$ according to the repair method of the EVENODD code in Table IX. According to Theorem 5, we can recover column 1 of the transformed EVENODD code by downloading the following 20 bits.

$$a_{0,0}, a_{1,0}, a_{3,0}, a_{0,2}, a_{1,2}, a_{2,2}, b_{0,0}, b_{1,0}, b_{3,0}, b_{0,2}, b_{1,2}, b_{2,2}, a_{3,4},$$
$$a_{0,3} + b_{0,3}, a_{1,3} + b_{1,3}, b_{1,4}, b_{3,4}, a_{0,3} + b_{0,3} + b_{1,3}, a_{1,3} + b_{0,3}, a_{1,4}.$$

Specifically, we can repair the bits $a_{0,1}, a_{1,1}$ and $b_{0,1}, b_{1,1}$ by

$$\begin{aligned}
a_{0,1} =& a_{0,0} + a_{0,2} + (a_{1,3} + b_{1,3}) + (a_{1,3} + b_{0,3}) + \\
& (a_{0,3} + b_{0,3} + b_{1,3}), \\
a_{1,1} =& a_{1,0} + a_{1,2} + (a_{0,3} + b_{0,3}) + (a_{1,3} + b_{1,3}) + \\
& (a_{0,3} + b_{0,3} + b_{1,3}), \\
b_{0,1} =& b_{0,0} + b_{0,2} + (a_{0,3} + b_{0,3}) + (a_{1,3} + b_{1,3}) + \\
& (a_{0,3} + b_{0,3} + b_{1,3}) + (a_{1,3} + b_{0,3}), \\
b_{1,1} =& b_{1,0} + b_{1,2} + (a_{0,3} + b_{0,3}) + (a_{0,3} + b_{0,3} + b_{1,3}),
\end{aligned}$$

and repair $a_{2,1}, a_{3,1}, b_{2,1}, b_{3,1}$ by

$$\begin{aligned}
a_{3,1} =& a_{1,0} + a_{0,1} + a_{2,2} + a_{1,4}, \\
a_{2,1} =& a_{3,0} + a_{1,2} + a_{3,1} + a_{2,2} + a_{3,4}, \\
b_{3,1} =& b_{1,0} + b_{0,1} + b_{2,2} + b_{1,4}, \\
b_{2,1} =& b_{3,0} + b_{1,2} + b_{3,1} + b_{2,2} + b_{3,4}.
\end{aligned}$$

Therefore, we can recover column 1 by downloading 20 bits and the efficient repair property of column 1 is preserved in our transformation. We can also show that the efficient repair property of any other information column is preserved similarly.

We can also show that any one parity column of the transformed code is optimal. We can repair column 3 by downloading 16 bits

$$b_{i,j} \text{ for } i = 0, 1, 2, 3 \text{ and } j = 0, 1, 2, \text{ and}$$
$$a_{0,3} + b_{0,3} + b_{1,3}, a_{1,3} + b_{0,3}, a_{2,3} + b_{2,3} + b_{3,3}, a_{3,3} + b_{2,3}.$$

Specifically, we can compute $b_{i,3}, b_{i,4}$ from $b_{i,0}, b_{i,1}, b_{i,2}$ for $i = 0, 1, 2, 3$, as EVENODD is MDS code. Then, we can compute the other four bits in column 3 by

$$\begin{aligned}
a_{0,3} + b_{0,3} =& (a_{0,3} + b_{0,3} + b_{1,3}) + b_{1,3}, \\
a_{1,3} + b_{1,3} =& (a_{1,3} + b_{0,3}) + b_{0,3} + b_{1,3}, \\
a_{2,3} + b_{2,3} =& (a_{2,3} + b_{2,3} + b_{3,3}) + b_{3,3}, \\
a_{3,3} + b_{3,3} =& (a_{3,3} + b_{2,3}) + b_{2,3} + b_{3,3}.
\end{aligned}$$

Therefore, the repair access of the first parity column is optimal. Similarly, we can repair column 4 by downloading 16 bits

$$a_{i,j} \text{ for } i = 0, 1, 2, 3 \text{ and } j = 0, 1, 2, \text{ and}$$
$$b_{0,3} + a_{0,3}, b_{1,3} + a_{1,3}, b_{2,3} + a_{2,3}, b_{3,3} + a_{3,3},$$

and the repair access of column 4 is optimal.

In order to obtain binary MDS array codes with low sub-packetization that have efficient repair for any column, we show in this section how to apply the transformation for the array codes in [33] and EVENODD codes. Note that the transformation given in this section can be viewed as a variant of the transformation in Section II-B. We can also apply the transformation given in this section multiple times for EVENODD codes to obtain the multi-layer transformed EVENODD codes that have optimal repair for any column, as the construction in Section III-A. The difference between two transformations is that, the efficient repair property of any information column of codes in [33] is maintained with the transformation given in this section, while not for the transformation in Section II-B. The relationship of sub-packetization and repair bandwidth of binary MDS array codes is one of our future work.

TABLE X: The transformed EVENODD code with $k = 3$, $r = 2$ and $p = 5$.

Column 0	Column 1	Column 2	Column 3	Column 4
$a_{0,0}, b_{0,0}$	$a_{0,1}, b_{0,1}$	$a_{0,2}, b_{0,2}$	$a_{0,3} + b_{0,3}, b_{0,4}$	$a_{0,4}, a_{0,3} + b_{0,3} + b_{1,3}$
$a_{1,0}, b_{1,0}$	$a_{1,1}, b_{1,1}$	$a_{1,2}, b_{1,2}$	$a_{1,3} + b_{1,3}, b_{1,4}$	$a_{1,4}, a_{1,3} + b_{0,3}$
$a_{2,0}, b_{2,0}$	$a_{2,1}, b_{2,1}$	$a_{2,2}, b_{2,2}$	$a_{2,3} + b_{2,3}, b_{2,4}$	$a_{2,4}, a_{2,3} + b_{2,3} + b_{3,3}$
$a_{3,0}, b_{3,0}$	$a_{3,1}, b_{3,1}$	$a_{3,2}, b_{3,2}$	$a_{3,3} + b_{3,3}, b_{3,4}$	$a_{3,4}, a_{3,3} + b_{2,3}$

V. Discussion and Conclusion

In this paper, we propose a generic transformation for EVENODD codes that can enable optimal repair access for the chosen $d - k + 1$ columns. Based on the proposed EVENODD transformation, we present the multi-layer transformed $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil}$ that have optimal repair access for all $k + r$ columns. In $\mathsf{EVENODD}_{\lceil \frac{k}{d-k+1} \rceil + \lceil \frac{r}{d-k+1} \rceil}$, the d helper columns can be selected from $k + 1$ and $k + r - 1$, and some of the d helper columns should be specifically selected. Moreover, we show that the proposed transformation can also be employed in other existing binary MDS array codes, such as codes in [26], [31]–[33], [35]–[37], that can enable optimal repair access. How to combine the existing binary MDS array codes with asymptotically optimal repair access by our transformation to obtain the transformed binary MDS array codes with asymptotically optimal repair access for all columns and lower sub-packetization is an interesting and practical future work. The implementation of the proposed transformed binary MDS array codes in practical storage systems is another one of our future works.

References

[1] H. Weatherspoon and J. Kubiatowicz, "Erasure Coding Vs. Replication: A Quantitative Comparison," *Lecture Notes in Computer Science*, vol. 2429, pp. 328–337, 2002.

[2] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure Coding in Windows Azure Storage," in *Proc. of USENIX ATC*, 2012.

[3] 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.

[4] I. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *Journal of the Society for Industrial & Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

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

[6] 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.

[7] 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.

[8] L. Xu and J. Bruck, "X-Code: MDS Array Codes with Optimal Encoding," *IEEE Trans. Information Theory*, vol. 45, no. 1, pp. 272–276, 1999.

[9] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-Diagonal Parity for Double Disk Failure Correction," in *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, 2004, pp. 1–14.

[10] 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.

[11] D. Ford, F. I. Popovici, M. Stokely, V. A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in Globally Distributed Storage Systems," in *Proc. of USENIX OSDI*, 2010.

[12] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network Coding for Distributed Storage Systems," *IEEE Trans. Information Theory*, vol. 56, no. 9, pp. 4539–4551, September 2010.

[13] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal Exact-Regenerating Codes for Distributed Storage at the MSR and MBR Points via a Product-Matrix Construction," *IEEE Trans. Information Theory*, vol. 57, no. 8, pp. 5227–5239, August 2011.

[14] C. Suh and K. Ramchandran, "Exact-Repair MDS Code Construction Using Interference Alignment," *IEEE Trans. Information Theory*, vol. 57, no. 3, pp. 1425–1442, March 2011.

[15] H. Hou, K. W. Shum, M. Chen, and H. Li, "BASIC Codes: Low-Complexity Regenerating Codes for Distributed Storage Systems," *IEEE Trans. Information Theory*, vol. 62, no. 6, pp. 3053–3069, 2016.

[16] J. Li, X. Tang, and C. Tian, "A Generic Transformation to Enable Optimal Repair in MDS Codes for Distributed Storage Systems," *IEEE Trans. Information Theory*, vol. 64, no. 9, pp. 6257–6267, 2018.

[17] M. Ye and A. Barg, "Explicit Constructions of High-Rate MDS Array Codes with Optimal Repair Bandwidth," *IEEE Trans. Information Theory*, vol. 63, no. 4, pp. 2001–2014, 2017.

[18] G. Atul and C. Peter, "RAID Triple Parity," in *ACM SIGOPS Operating Systems Review*, vol. 36, no. 3, December 2012, pp. 41–49.

[19] H. Hou, K. W. Shum, M. Chen, and H. Li, "New MDS Array Code Correcting Multiple Disk Failures," in *IEEE Global Communications Conference (GLOBECOM)*, 2014, pp. 2369–2374.

[20] H. Hou and P. P. C. Lee, "A New Construction of EVENODD Codes with Lower Computational Complexity," *IEEE Communications Letters*, vol. 22, no. 6, pp. 1120–1123, 2018.

[21] C. Huang and L. Xu, "STAR: An Efficient Coding Scheme for Correcting Triple Storage Node Failures," *IEEE Trans. on Computers*, vol. 57, no. 7, pp. 889–901, 2008.

[22] H. Jiang, M. Fan, Y. Xiao, X. Wang, and Y. Wu, "Improved Decoding Algorithm for the Generalized EVENODD Array Code," in *International Conference on Computer Science and Network Technology*, 2013, pp. 2216–2219.

[23] 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.

[24] Z. Huang, H. Jiang, and K. Zhou, "An Improved Decoding Algorithm for Generalized RDP Codes," *IEEE Communications Letters*, vol. 20, no. 4, pp. 632–635, 2016.

[25] H. Hou, Y. S. Han, K. W. Shum, and H. Li, "A Unified Form of EVENODD and RDP Codes and Their Efficient Decoding," *IEEE Trans. Communications*, pp. 1–1, 2018.

[26] H. Hou and Y. S. Han, "A New Construction and an Efficient Decoding Method for Rabin-Like Codes," *IEEE Trans. Communications*, vol. 66, no. 2, pp. 521–533, 2018.

[27] Z. Wang, A. G. Dimakis, and J. Bruck, "Rebuilding for Array Codes in Distributed Storage Systems," in *2010 IEEE GLOBECOM Workshops (GC Wkshps)*, pp. 1905–1909.

[28] 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, 2010, pp. 119–130.

[29] L. Xiang, Y. Xu, J. C. S. Lui, Q. Chang, Y. Pan, and R. Li, "A Hybrid Approach of Failed Disk Recovery Using RAID-6 Codes: Algorithms and Performance Evaluation," *ACM Trans. on Storage*, vol. 7, no. 3, pp. 1–34, October 2011.

[30] Y. Zhu, P. P. C. Lee, Y. Xu, Y. Hu, and L. Xiang, "On the Speedup of Recovery in Large-Scale Erasure-Coded Storage Systems," *IEEE Transactions on Parallel & Distributed Systems*, vol. 25, no. 7, pp. 1830–1840, 2014.

[31] H. Hou, P. P. C. Lee, Y. S. Han, and Y. Hu, "Triple-Fault-Tolerant Binary MDS Array Codes with Asymptotically Optimal Repair," in *Proc. IEEE Int. Symp. Inf. Theory*, Aachen, June 2017.

[32] H. Hou, Y. S. Han, P. P. C. Lee, Y. Hu, and H. Li, "A New Design of Binary MDS Array Codes with Asymptotically Weak-Optimal Repair," *arXiv preprint https://arxiv.org/pdf/1802.07891.pdf*, 2018.

[33] H. Hou and Y. S. Han, "A Class of Binary MDS Array Codes with Asymptotically Weak-Optimal Repair," *SCIENCE CHINA Information Sciences*, vol. 61, no. 10, pp. 1–12, 2018.

[34] H. Hou, K. W. Shum, and H. Li, "On the MDS Condition of Blaum-Bruck-Vardy Codes with Large Number Parity Columns," *IEEE Communications Letters*, vol. 20, no. 4, pp. 644–647, 2016.
[35] J. Blomer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman, "An XOR-Based Erasure-Resilient Coding Scheme," *Int. Comput. Sci. Inst., Berkeley, CA, USA, Tech. Rep. TR-95-048*, 1995.
[36] 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 ($\geq$ 4) Disk Failures," *IEEE Trans. on Computers*, vol. 54, no. 12, pp. 1473–1483, 2005.
[37] 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.
[38] S. Xu, R. Li, P. P. Lee, Y. Zhu, L. Xiang, Y. Xu, and J. Lui, "Single Disk Failure Recovery for X-Code-Based Parallel Storage Systems," *IEEE Trans. on Computers*, vol. 63, no. 4, pp. 995–1007, 2014.
[39] Y. Wang, X. Yin, and X. Wang, "MDR Codes: A New Class of RAID-6 Codes with Optimal Rebuilding and Encoding," *IEEEJSAC*, vol. 32, no. 5, pp. 1008–1018, 2013.
[40] ——, "Two New Classes of Two-Parity MDS Array Codes with Optimal Repair," *IEEE Communications Letters*, vol. 20, no. 7, pp. 1293–1296, 2016.
[41] E. E. Gad, R. Mateescu, F. Blagojevic, C. Guyot, and Z. Bandic, "Repair-Optimal MDS Array Codes over GF(2)," in *Proc. IEEE Int. Symp. Inf. Theory*. IEEE, 2013, pp. 887–891.
[42] L. Pamies-Juarez, F. Blagojevic, R. Mateescu, C. Gyuot, E. E. Gad, and Z. Bandic, "Opening the Chrysalis: On the Real Repair Performance of MSR Codes," in *14th USENIX Conference on File and Storage Technologies (FAST 16)*, 2016, pp. 81–94.
[43] I. Tamo, Z. Wang, and J. Bruck, "Access Versus Bandwidth in Codes for Storage," *IEEE Trans. Information Theory*, vol. 60, no. 4, pp. 2028–2037, 2014.
[44] A. S. Rawat, I. Tamo, V.Guruswami, and K.Efremenko, "MDS Code Constructions with Small Sub-packetization and Near-Optimal Repair Bandwidth," *IEEE Trans. Information Theory*, vol. 64, no. 10, pp. 6506–6525, 2018.

Hanxu Hou 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 Honorary Postdoc of the Department of Computer Science and Engineering at the Chinese University of Hong Kong. His research interests include erasure coding and coding for distributed storage systems.

Patrick P. C. Lee received the B.Eng. degree (first class honors) in Information Engineering from the Chinese University of Hong Kong in 2001, the M.Phil. degree in Computer Science and Engineering from the Chinese University of Hong Kong in 2003, and the Ph.D. degree in Computer Science from Columbia University in 2008. He is now an Associate Professor of the Department of Computer Science and Engineering at the Chinese University of Hong Kong. His research interests are in various applied/systems topics including storage systems, distributed systems and networks, operating systems, dependability, and security.