Constructions of Binary MDS Array Codes with Optimal Repair/Access Bandwidth

Lei Li, Xinchun Yu, Liang Chen, Yuanyuan Dong, and Yuan Luo

***Abstract*—Maximum distance separable (MDS) codes are commonly deployed in distributed storage systems as they provide the maximum failure tolerance for some given redundancy. The repair problem of MDS codes has drawn much attention and various constructions of MDS array codes with optimal repair bandwidth have been proposed in the last decade. However, few of the existing codes are constructed over the binary field. In this paper, we propose new constructions of binary MDS array codes with optimal repair (or access) bandwidth for single-node failure. Specifically, by stacking multiple Blaum-Roth code instances of which the parity-check matrices are judiciously designed, we obtain three families of binary MDS array codes with optimal repair bandwidth; using the permutation matrices as building blocks, we also construct two families of binary MDS array codes with optimal access bandwidth. Moreover, error-resilient capability while achieving the lower bound on repair (or access) bandwidth is obtained when the number of helper nodes $d < n - 1$. All the codes in this paper are constructed over a particular ring of binary polynomials. Consequently, computation operations involved in the encoding, decoding and node repair procedures for these codes are only XORs and cyclic shifts, avoiding complex multiplications and divisions over large finite fields.**

***Index Terms*—Distributed storage system, repair bandwidth, optimal access, MSR codes, binary MDS codes.**

I. Introduction

DISTRIBUTED storage systems (DSS) are built upon a large number of individually unreliable nodes to store and analyze massive amount of data, where transient and permanent node failures may occur as daily events. To provide reliability and availability in the face of node failures, the system need to store some redundant data. The traditional mechanism for introducing redundancy in DSS, such as the Google File System [2], Hadoop Distributed File System [3], and Microsoft Azure [4], is replication. Clearly, the replication scheme will be very costly as the amount of data is increasing exponentially. Error-correcting codes (ECC) have been introduced into DSS as a viable alternative to replication since they can achieve higher reliability for some given redundancy. An important class of ECC that are widely deployed in DSS are maximum distance separable (MDS) codes, which achieve the optimal trade-off between storage efficiency and fault tolerance. For a file of size $\mathcal{M}$, the system using an (n, k) MDS code first divides the file into k equal-size packets, and then encodes them into n packets which are distributed over n distinct storage nodes. The MDS property guarantees reconstruction of the original file as long as any k out of these n packets are accessible. For systems using an (n, k) MDS code, the failed node can be repair by accessing and communicating an amount of data equal to the size of the original file with the help of any k out of the surviving nodes. However, this is not "efficient" for single-node failure in that the amount of data communicated, named repair bandwidth, is k times of the amount of data lost.

The repair problem was first formulated in the seminal work [5], wherein a trade-off between storage and repair bandwidth was derived. The two extremal points on the optimal trade-off curve, corresponding to the best storage efficiency and the minimum repair bandwidth, are called minimum storage regenerating (MSR) codes and minimum bandwidth regenerating (MBR) codes, respectively. MSR codes have drawn much attraction due to their optimal storage efficiency and various constructions of MSR codes have been proposed in the last decade. The reader can refer to [6]–[20] for details.

Among the failures, single-node failure is the most common scenario in real-world storage systems [21], and it is important to construct codes that can repair any node efficiently. Most of the existing MSR codes are constructed over some finite field whose size can not be too small. Thus, a lot of multiplications and divisions over the field are needed during node repair and file reconstruction procedures. In the present paper, we focus on the single-node repair and construct binary MDS array codes with optimal repair bandwidth, i.e., the field used is $\mathbb{F}_2$.

Few binary MDS array codes with optimal repair bandwidth have been proposed in the literature [22], [23]. In [22], the authors proposed a general transformation framework to construct binary MDS array codes with optimal repair bandwidth for $k + 1 \leq d \leq n - 1$, where some of the helper nodes are specific. In [23], another generic transformation that can convert any (n, k) binary MDS array code into a new one with $r = n - k$ chosen nodes having optimal repair bandwidth. After multiple transformations, the original binary MDS array code becomes a binary MDS array code with optimal repair bandwidth for all nodes where the number of helper nodes d is $n - 1$. Note that the codes in [22] and [23] also have the optimal-access property, i.e., the amount of data accessed during node repair is equal to the minimum amount of data that need to be communicated.

This work was presented in part at the 2023 IEEE International Symposium on Information Theory [1].

Lei Li and Yuan Luo are with Department of Computer Science and Engineering, Shanghai Jiao Tong University, 200240 Shanghai, China (e-mail: {staroverseas,yuanluo}@sjtu.edu.cn). Yuan Luo is the corresponding author.

Xinchun Yu is with Institute of Data and Informtion, Shenzhen International Graduate School of Tsinghua University, 518055 Shenzhen,China (email: yuxinchun@sz.tsinghua.edu.cn)

Liang Chen and Yuanyuan Dong are with Alibaba Group, 310030 Hangzhou, China (e-mail:{cl304641,yuanyuan.dyy}@alibaba-inc.com).

Motivated by the code constructions over nonbinary fields in [24], we propose new constructions of binary MDS array codes with optimal repair/access bandwidth. Specifically, by stacking multiple Blaum-Roth code instances, we construct three families of binary MDS array codes, named $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$, with optimal repair bandwidth. Code $\mathcal{C}_1$ achieves the optimal repair bandwidth for $d = n-1$, and with a slight modification to $\mathcal{C}_1$, we obtain code $\mathcal{C}_2$ which has optimal repair bandwidth for any arbitrary fixed $d \in [k+1, n-1]$. Through some further modification to $\mathcal{C}_2$, we arrive at the construction of $\mathcal{C}_3$ which has optimal repair bandwidth for multiple values of $d \in [k+1, n-1]$ simultaneously. Using the permutation matrices, we construct two families of codes $\mathcal{C}_4$ and $\mathcal{C}_5$ with optimal access bandwidth, respectively, for arbitrary fixed $d \in [k+1, n-1]$ and for multiple values of $d \in [k+1, n-1]$ simultaneously. We note that codes $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3$ were presented at the 2023 IEEE International Symposium on Information Theory [1].

The remainder of the paper is organized as follows. Section II presents some necessary preliminaries. Binary MDS array codes with optimal repair bandwidth and binary MDS array codes with optimal access bandwidth are constructed in Section III and Section IV, respectively. Evaluations and comparisons are made in Section V. Finally, Section VI draws the conclusion.

II. Preliminaries

Given two integers i and j with $i < j$, denote by $[i]$ and $[i, j]$ two ordered sets $\{1, 2, \cdots, i\}$ and $\{i, i+1, \cdots, j\}$, respectively. For an (n, k) code, denote by $r := n - k$ the number of parity nodes. Following the literature of codes for distributed storage, we use the words coordinate and node interchangeably. As a result, repairing failed nodes in a distributed storage system can be viewed as correcting erasures of a codeword. In this paper, the boldface $\mathbf{0}$ is to denote a zero vector, and the plain 0 is to denote a scalar 0. Given M distinct positive integers $i_1, i_2, \ldots, i_M$, denote by $\mathrm{lcm}(i_1, i_2, \ldots, i_M)$ and $\gcd(i_1, i_2, \ldots, i_M)$, respectively, the least common multiple and the greatest common divisor of these integers.

A. A binary polynomial ring

For a prime p and the binary field $\mathbb{F}_2$, let $\mathcal{R}$ be the ring of polynomials of degree $< p-1$ over $\mathbb{F}_2$ with multiplication taken modulo $1 + x + x^2 + \cdots + x^{p-1}$, i.e., $\mathcal{R} = \mathbb{F}_2[x]/(1 + x + x^2 + \cdots + x^{p-1})$. Let $\mathcal{R}^*$ denote the multiplicative group of polynomials in $\mathcal{R}$, which are relatively prime to $1 + x + x^2 + \cdots + x^{p-1}$. Clearly, the multiplication operation in $\mathcal{R}$ is commutative.

In this paper, we use special elements in $\mathcal{R}^*$ to construct the codes with desired properties. So, it is beneficial to introduce some of the elements in $\mathcal{R}^*$ here. First observe that $x \in \mathcal{R}^*$ since $\gcd(x, 1 + x + x^2 + \cdots + x^{p-1}) = 1$. Consequently, for any $i \in [p-2]$, we have $x^i \in \mathcal{R}^*$. Note that $x^p - 1 = (x-1)(1 + x + x^2 + \cdots + x^{p-1})$ and for $i \in [p-2]$, we have

$$\gcd(x^i - 1, x^p - 1) = x^{\gcd(i,p)} - 1 = x - 1.$$

Since p is not the characteristic of $\mathbb{F}_2$, we have that 1 is not a root of $1 + x + x^2 + \cdots + x^{p-1} = 0$, i.e., $\gcd(x-1, 1 + x + x^2 + \cdots + x^{p-1}) = 1$. As a result, we have $\gcd(x^i - 1, 1 + x + x^2 + \cdots + x^{p-1}) = 1$, meaning $x^i - 1 \in \mathcal{R}^*$ and $x^i - x^j$ is also in $\mathcal{R}^*$ with $i \neq j \in [p-2]$.

B. Blaum-Roth codes

An (n, k, l) binary array code can be viewed as a set of matrices of size $l \times n$ over $\mathbb{F}_2$. Let p be a prime number, a Blaum-Roth code $\mathcal{C}_{BR}$ is a code over the polynomial ring $\mathcal{R}$. Each codeword in $\mathcal{C}_{BR}$ is a $(p-1) \times n$ array (matrix) $c[i, a]$ where $i \in [n]$ and $a \in [0, p-1]$. The $p-1$ bits $c_{i,0}, c_{i,1}, \ldots, c_{i,p-2}$ in the i-th column of the array represent a polynomial in $\mathcal{R}$, which can be written as $c_i(x) = c_{i,0} + c_{i,1} \cdot x + c_{i,2} \cdot x^2 + \cdots + c_{i,p-2} \cdot x^{p-2}$. For $r \leq n \leq p$, an $(n, n-r, p-1)$ Blaum-Roth code is defined by its parity-check matrix

$$H_{BR} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & x & x^2 & \vdots & x^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x^{r-1} & x^{2(r-1)} & \cdots & x^{(n-1)(r-1)} \end{bmatrix}, \quad (1)$$

where $1, x, x^2, \ldots, x^{n-1}$ are n distinct nonzero elements that have multiplicative inverse in $\mathcal{R}$. Note that the matrix H_{BR} can be defined by $1, x, x^2, \ldots, x^{n-1}$ and we call these n elements the "evaluation points" of the (n, k, l) Blaum-Roth code.

Lemma 1: Every r columns of H_{BR} are linearly independent over the ring $\mathcal{R}$.

Proof of *Lemma 1* is omitted and the reader can refer to [25] for more details.

Since every r columns in H_{BR} are linearly independent over $\mathcal{R}$, we have that code $\mathcal{C}_{BR}$ is a code of length n, dimension $n-r$ and minimum distance $r+1$ over $\mathcal{R}$. Thus, code $\mathcal{C}_{BR}$ is MDS code.

C. Lower bounds on repair/access bandwidth of MDS codes

An (n, k, l) MDS array code can tolerate any $r = n - k$ erasures because any k coordinates of a codeword can reconstruct the whole codeword, meaning that the storage system is reliable even when r storage nodes are failed. However, when there is only one failed node, downloading data from k nodes to repair the failed node is not efficient in terms of repair bandwidth. Specifically, the amount of data communicated during node repair procedure is kl, which is k times the amount l of data stored on the failed node.

In [5], the authors analysed the information flow graph of storage systems and derived the lower bound on repair bandwidth γ of any (n, k, l) MDS code for single-node repair. The bound is also called cut-set bound and can be written as

$$\gamma \geq \frac{dl}{d - k + 1}, \quad (2)$$

where $d \in [k, n-1]$ is the number of helper nodes. The optimal repair bandwidth will decrease as the number of helper nodes d increases. In particular, when $d = n-1$, the optimal repair

bandwidth of an (n,k,l) MDS code gets the minimum value, which is $\frac{(n-1)l}{n-k}$.

In [26], error-resilient capability was considered in the repair process of single-node failure, i.e., there maybe erroneous or malicious nodes among the helper nodes. For $e \leq \frac{n-k}{2}$, it is proved that the lower bound on the repair bandwidth γ_e for single-node failure with e-error resilient capability can be written as

$$\gamma_e \geq \frac{dl}{d-2e-k+1}, \tag{3}$$

where d is the number of helper nodes, among which there are e errors.

III. Binary MDS array codes with optimal repair bandwidth

In this section, we present explicit constructions of three families of codes, named $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$, over the binary polynomial ring $\mathcal{R} = \mathbb{F}_2[x]/(1+x+\cdots+x^{p-1})$.

Given integers n, r and a prime p, let $\mathcal{C} \subset \mathcal{R}^{r^n \times n}$ be an $(n, k=n-r, l)$ binary MDS array code with $l=(p-1)r^n$. Node $i \in [n]$ stores coordinate $C_i \in \mathcal{R}^{r^n}$ of the codeword $(C_1, C_2, \ldots, C_n)$, where C_i is a column vector consisting of r^n polynomials in $\mathcal{R}$. In the present paper, we define a code by its parity-check equations as follows:

$$\mathcal{C} = \{(C_1, C_2, \ldots, C_n) : \sum_{i=1}^{n} H_{t,i} C_i = \mathbf{0}\}, \tag{4}$$

where $H_{t,i}, t \in [0, r-1], i \in [n]$ are $r^n \times r^n$ matrices over $\mathcal{R}$. The parity-check matrix H of code $\mathcal{C}$ can be written as

$$H = \begin{bmatrix} H_{0,1} & H_{0,2} & \cdots & H_{0,n} \\ H_{1,1} & H_{1,2} & \cdots & H_{1,n} \\ \vdots & \vdots & \vdots & \vdots \\ H_{r-1,1} & H_{r-1,2} & \cdots & H_{r-1,n} \end{bmatrix}. \tag{5}$$

We note that the sub-packetization levels of the codes in the present paper are different and are not necessarily equal to $(p-1)r^n$.

A. Binary MDS array code with optimal repair bandwidth for $d = n-1$

In this subsection, we present construction of binary MDS array codes with optimal repair bandwidth for $d=n-1$. Before presenting the general code construction, we give toy examples to highlight the main ideas behind the construction.

To illustrate how the repair bandwidth is reduced simply by stacking multiple Blaum-Roth code instances, we first give an example code with $(n=4, k=2)$. This example code is obtained by stacking two Blaum-Roth codes whose "evaluation points" are $x_{1,1}=x, x_2=x^2, x_3=x^3, x_4=x^4$ and $x_{1,2}=x^5, x_2=x^2, x_3=x^3, x_4=x^4$, respectively. These five "evaluation points" are chosen from the binary polynomial ring $\mathcal{R}=\mathbb{F}_2[x]/(1+x+\cdots+x^6)$. The parity-check equations can be written as

$$x_{1,1}^t c_{1,1} + \sum_{i=2}^{4} x_i^t c_{i,1} = 0, \tag{6}$$

and

$$x_{1,2}^t c_{1,2} + \sum_{i=2}^{4} x_i^t c_{i,2} = 0 \tag{7}$$

where $t \in [0,1]$. Summing up equations (6) and (7), we have

$$x_{1,1}^t c_{1,1} + x_{1,2}^t c_{1,2} + \sum_{i=2}^{4} x_i^t (c_{i,1}+c_{i,2}) = 0 \tag{8}$$

where $t \in [0,1]$. Obviously, equation (8) defines a new Blaum-Roth code with $(n=5, k=3)$ whose "evaluation points" are $x_{1,1}, x_{1,2}, x_2, x_3, x_4$ and we can transfer $\{c_{i,1}+c_{i,2}\}_{i\in[2,4]}$ to repair the first node. The repair bandwidth for repairing the first node is $3(p-1)$ bits, which is half of the trivial repair of Reed-Solomon code and achieves the lower bound on repair bandwidth. In the following *Example 1*, we present a code that has optimal repair bandwidth for all nodes.

Example 1. We take $n=4, k=2$ for example and the binary MDS array code $\mathcal{C}(n=4, k=2, l)$ can be defined by a set of matrices $\{H_{t,i} : t \in [0,1], i \in [4]\}$ according to equation (4). Instead of constructing $rn=8$ matrices, by making $H_{t,i} = H_i^t$, we only need $n=4$ matrices $\{H_i : i \in [4]\}$ to define the code. Let $p=11$ and $l=(p-1)r^n=160$, the code $\mathcal{C}(4,2,160)$ is obtained by stacking $r^n=16$ Blaum-Roth codes with parameter $(4,2,11)$, whose "evaluation points" are chosen from a set of $rn=8$ elements $\{x_{i,j}=x^{4j+i}\}_{i\in[4],j\in[0,1]}$ that have multiplicative inverse in $\mathcal{R}=\mathbb{F}_2[x]/(1+x+x^2+\cdots+x^{10})$. For $a \in [0,15]$, the "evaluation points" of the a-th Blaum-Roth code are $x_{1,a_1}, x_{2,a_2}, x_{3,a_3}, x_{4,a_4}$ where (a_4,a_3,a_2,a_1) is the binary expansion of a. For example, when $a=5=(0,1,0,1)$, the "evaluation points" of the corresponding Blaum-Roth code are $x_{1,1}, x_{2,0}, x_{3,1}, x_{4,0} = x^5, x^2, x^7, x^4$.

The above example shows the core structure of the codes constructed in this section. We now present the general construction and analyze the repair bandwidth in the sequel.

Construction 1 ($\mathcal{C}_1$): Given integers n and r, let $l=(p-1)r^n$ and $\{x_{i,j}=x^{jn+i}\}_{i\in[n],j\in[0,r-1]}$ be a set of rn distinct nonzero elements in the polynomial ring $\mathcal{R}=\mathbb{F}_2[x]/(1+x+\cdots+x^{p-1})$, where p is a prime and $p-2 \geq rn$. Code $\mathcal{C}_1$ defined by (4) is constructed by taking $H_{t,i}=H_i^t$ with

$$H_i = \sum_{a=0}^{r^n-1} x_{i,a_i} e_a e_a^T, \quad i \in [n].$$

Here, $e_a \in R^{r^n}$ is a column vector whose a-th entry is 1 and the other entries are 0s. Note that $a \in [0, r^n-1]$ is also represented by its r-ary expansion, i.e.,

$$a = (a_n, a_{n-1}, \ldots, a_1) = \sum_{i=1}^{n} a_i \cdot r^{i-1}$$

where $a_i \in [0, r-1]$.

It is not hard to verify that H_i is an $r^n \times r^n$ diagonal matrix whose a-th diagonal entry is $x_{i,a_i} = x^{a_i n+i} \in \mathcal{R}$. Codeword coordinate C_i in (4) is a column vector of length $l=(p-1)r^n$ over $\mathbb{F}_2$ and it can also be viewed as a column vector of length r^n over $\mathcal{R}$. Specifically, $C_i=(c_{i,0}, c_{i,1}, \ldots, c_{i,r^n-1})^T$ where $c_{i,a}$, $a \in [0, r^n-1]$, is a vector of length $p-1$ over $\mathbb{F}_2$. For

brevity of writing, we use $c_{i,a}$, the a-th entry in C_i, to denote both a vector of length $p-1$ over $\mathbb{F}_2$ and a polynomial in $\mathcal{R}$. For any $a \in [0, r^n-1]$, we can write down the corresponding parity-check equations in (4) as

$$\sum_{i=1}^{n} x_{i,a_i}^t c_{i,a} = 0, \quad t \in [0, r-1]. \tag{9}$$

The code constructed in *Construction 1* is named $\mathcal{C}_1$ in this paper. We now give the first theorem claiming the code $\mathcal{C}_1$ has optimal repair bandwidth for $d = n-1$.

Theorem 1: Code $\mathcal{C}_1$ achieves the lower bound (2) on repair bandwidth for single-node repair.

Proof: Assume node $i \in [n]$ is failed and let

$$a(i,u) = (a_n, \ldots, a_{i+1}, u, a_{i-1}, \ldots, a_1)$$

where $u \in [0, r-1]$. According to (9), for fixed u and $a(i,u)$ we have

$$x_{i,u}^t c_{i,a(i,u)} + \sum_{j \in [n]\setminus\{i\}} x_{j,a_j}^t c_{j,a(i,u)} = 0, \quad t \in [0, r-1]. \tag{10}$$

Summing up (10) over $u = 0, 1, \ldots, r-1$, we have

$$\begin{aligned} &x_{i,0}^t c_{i,a(i,0)} + x_{i,1}^t c_{i,a(i,1)} + \cdots + x_{i,r-1}^t c_{i,a(i,r-1)} \\ &+ \sum_{j \in [n]\setminus\{i\}} x_{j,a_j}^t \Big(\sum_{u=0}^{r-1} c_{j,a(i,u)}\Big) = 0, \quad t \in [0, r-1], \end{aligned} \tag{11}$$

where $\sum_{u=0}^{r-1} c_{j,a(i,u)}$ is the data communicated from helper node j and can be denoted as $D_j^{i,a}$. According to (1), we can see that formula (11) defines an $(n+r-1, n-1, p-1)$ binary MDS array code whose parity-check matrix can be written as

$$\begin{bmatrix} 1 & 1 & \cdots & 1 & 1 & \cdots & 1 \\ x_{i,0} & x_{i,1} & \cdots & x_{i,r-1} & x_{j_1,a_{j_1}} & \cdots & x_{j_{n-1},a_{j_{n-1}}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i,0}^{r-1} & x_{i,1}^{r-1} & \cdots & x_{i,r-1}^{r-1} & x_{j_1,a_{j_1}}^{r-1} & \cdots & x_{j_{n-1},a_{j_{n-1}}}^{r-1} \end{bmatrix}. \tag{12}$$

Thus, the lost data $\{c_{i,a(i,0)}, c_{i,a(i,1)}, \ldots, c_{i,a(i,r-1)}\}$ can be obtained by downloading $\{D_j^{i,a} : j \in [n]\setminus\{i\}\}$ from the $n-1$ helper nodes where $p-1$ bits are downloaded on each node. For the overall repair of C_i, note that C_i can be partitioned into r^{n-1} disjoint sets, each of which can be written as $\{c_{i,a(i,0)}, c_{i,a(i,1)}, \ldots, c_{i,a(i,r-1)}\}$ for some $a \in [0, r^n-1]$ and each set can be repaired by downloading $p-1$ bits from every helper node. As a result, a total number of $(p-1)r^{n-1}$ bits are communicated from each helper node for the overall repair of C_i, achieving the lower bound (2) on repair bandwidth. ∎

We now analyze the MDS property of code $\mathcal{C}_1$ by giving the following Property 1.

Property 1: Code $\mathcal{C}_1$ is an MDS array code.

Proof: For any $a \in [0, r^n-1]$, rewrite (9) in matrix form and we have

$$\begin{bmatrix} 1 & 1 & \ldots & 1 \\ x_{1,a_1} & x_{2,a_2} & \cdots & x_{n,a_n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{1,a_1}^{r-1} & x_{2,a_2}^{r-1} & \cdots & x_{n,a_n}^{r-1} \end{bmatrix} \begin{bmatrix} c_{1,a} \\ c_{2,a} \\ \vdots \\ c_{n,a} \end{bmatrix} = \mathbf{0}. \tag{13}$$

It is clear that formula (13) defines an $(n, n-r, p-1)$ binary MDS code according to *Lemma 1*. Thus, any $k = n-r$ out of n coordinates in the codeword $(c_{1,a}, c_{2,a}, \ldots, c_{n,a})$ can reconstruct the whole codeword. The MDS property of code $\mathcal{C}_1$ follows since this holds for all $a \in [0, r^n-1]$. ∎

Remark 1: The MDS property of code $\mathcal{C}_1$ is obtained by selecting rn special polynomials of form x^i, i.e. powers of x in $\mathcal{R}$ and the optimal repair bandwidth is achieved through r^n combinations of these special polynomials. With this technique, we can construct the other two families of codes $\mathcal{C}_2$ and $\mathcal{C}_3$ by some modifications of $\mathcal{C}_1$.

B. Binary MDS array code with optimal repair bandwidth for arbitrary $d \in [k+1, n-1]$

In this subsection, we show that *Construction 1* can be slightly modified to construct code with optimal repair bandwidth for arbitrary $d \in [k+1, n-1]$.

Construction 2 ($\mathcal{C}_2$): For integers n and k, let $s = d-k+1$ where $k+1 \le d \le n-1$. Let $l = (p-1)s^n$ and $\{x_{i,j} = x^{jn+i}\}_{i\in[n], j\in[0,s-1]}$ be a set of sn distinct nonzero elements in the polynomial ring $\mathcal{R} = \mathbb{F}_2[x]/(1+x+\cdots+x^{p-1})$, where p is a prime and $p-2 \ge sn$. Code $\mathcal{C}_2$ defined by (4) is constructed by taking $H_{t,i} = H_i^t$ with

$$H_i = \sum_{a=0}^{s^n-1} x_{i,a_i} e_a e_a^T, \quad i \in [n].$$

Here, $e_a \in R^{s^n}$ is a column vector whose a-th entry is 1 and the other entries are 0s. Also, $a \in [0, s^n-1]$ is represented by its s-ary expansion, i.e., $a = (a_n, a_{n-1}, \ldots, a_1) = \sum_{i=1}^{n} a_i \cdot s^{i-1}$ where $a_i \in [0, s-1]$. The code constructed in *Construction 2* is named $\mathcal{C}_2$ in this paper.

Theorem 2: Code $\mathcal{C}_2$ achieves the lower bound (2) on repair bandwidth for arbitrary $d \in [k+1, n-1]$.

Proof: The proof follows the similar arguments to that of Theorem 1 and we present it in detail to make it more comprehensive. Assume node $i \in [n]$ is failed and let $a(i,u) = (a_n, \ldots, a_{i+1}, u, a_{i-1}, \ldots, a_1)$ where $u \in [0, s-1]$. Since the parity-check matrix is a diagonal matrix over $\mathcal{R}$, we can write down the parity-check equations for any $a \in [0, s^n-1]$ as

$$x_{i,a_i}^t c_{i,a} + \sum_{j \in [n]\setminus\{i\}} x_{j,a_j}^t c_{j,a} = 0, \quad t \in [0, r-1]. \tag{14}$$

For fixed $u \in [0, s-1]$ and $a(i,u)$, we have

$$x_{i,u}^t c_{i,a} + \sum_{j \in [n]\setminus\{i\}} x_{j,a_j}^t c_{j,a(i,u)} = 0, \quad t \in [0, r-1]. \tag{15}$$

Summing up (15) over $u = 0, 1, \ldots, s-1$, we have

$$\begin{aligned} &x_{i,0}^t c_{i,a(i,0)} + x_{i,1}^t c_{i,a(i,1)} + \cdots + x_{i,s-1}^t c_{i,a(i,s-1)} \\ &+ \sum_{j \in [n]\setminus\{i\}} x_{j,a_j}^t \Big(\sum_{u=0}^{s-1} c_{j,a(i,u)}\Big) = 0, \quad t \in [0, r-1], \end{aligned} \tag{16}$$

where $\sum_{u=0}^{s-1} c_{j,a(i,u)}$ is the data communicated from helper node j and can be denoted as $D_j^{i,a}$. According to *Lemma 1*, we can see that formula (16) defines an $(s+n-1, s+n-$

$1-r, p-1) = (d+r, d, p-1)$ binary MDS code whose parity-check matrix can be written as

$$\begin{bmatrix} 1 & 1 & \cdots & 1 & 1 & \cdots & 1 \\ x_{i,0} & x_{i,1} & \cdots & x_{i,s-1} & x_{j,a_{j_1}} & \cdots & x_{j_{n-1},a_{j_{n-1}}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i,0}^{r-1} & x_{i,1}^{r-1} & \cdots & x_{i,s-1}^{r-1} & x_{j_1,a_{j_1}}^{r-1} & \cdots & x_{j_{n-1},a_{j_{n-1}}}^{r-1} \end{bmatrix}. \tag{17}$$

Thus, the lost data $\{c_{i,a(i,0)}, c_{i,a(i,1)}, \ldots, c_{i,a(i,s-1)}\}$ can be obtained by downloading $D_j^{i,a}$ from any d out of $n-1$ helper nodes where $p-1$ bits are downloaded on each node. For the overall repair of C_i, note that C_i can be partitioned into s^{n-1} disjoint sets, each of which can be written as $\{c_{i,a(i,0)}, c_{i,a(i,1)}, \ldots, c_{i,a(i,s-1)}\}$ for some $a \in [0, s^n-1]$ and each set can be repaired by downloading $p-1$ bits from each helper node. Consequently, a total number of $(p-1)s^{n-1}$ bits are communicated from each helper node for the overall repair of C_i, achieving the lower bound (2) on repair bandwidth for arbitrary fixed $d \in [k+1, n-1]$. ■

Property 2: Code $\mathcal{C}_2$ is an MDS array code.

Proof: The proof follows the same arguments as that in the proof of Property 1 and is omitted. ■

In the above repair procedure, we notice that only d out of the $n-1$ surviving nodes are accessed to recover the lost bits and the other data in the remaining $n-1-d$ nodes are "wasted". When there are more than d nodes taking part in the repair procedure, we find that code $\mathcal{C}_2$ possesses the error-resilient capability while achieving the corresponding lower bound (3) on repair bandwidth. The following remark describes this error-resilient capability in detail.

Remark 2: By (16) and (17) we conclude that $(c_{i,a(i,0)}, c_{i,a(i,1)}, \ldots, c_{i,a(i,s-1)}, D_{j_1}^{i,a}, D_{j_2}^{i,a}, \ldots, D_{j_{n-1}}^{i,a})$ is a codeword of a $(d+r, d, p-1)$ binary MDS array code for any $a \in [0, s^n-1]$ and the minimum distance of this code is $r+1$. Thus, for an integer e such that $e \leq \frac{r}{2}$, any $d+2e$ out of the $n-1$ remaining coordinates $\{D_j^{i,a} : j \in [n]\backslash\{i\}\}$ can reconstruct the codeword as long as the number of erroneous coordinates in the $d+2e$ coordinates is not greater than e. As a result, for any $i \in [n]$, C_i can be repaired by connecting any $d+2e$ helper nodes and downloading $(p-1)s^{n-1}$ bits on each node as long as the number of erroneous nodes among the helper nodes is not greater than e. In total, there are $\frac{(d+2e)l}{d-k+1}$ bits communicated to repair a failed node. Thus, code $\mathcal{C}_2$ achieves the lower bound on repair bandwidth for any fixed $d \in [k+1, n-1]$ with e error-resilient capability where $e \leq \frac{r}{2}$.

C. Binary MDS array code with optimal repair bandwidth for multiple values of $d \in [k+1, n-1]$ simultaneously

In the previous two subsections, we constructed binary MDS array codes with optimal repair bandwidth for some fixed d, meaning that d has only one value in $[k+1, n-1]$. In the present subsection, we construct codes with optimal repair bandwidth for multiple values of d simultaneously by using previous constructions as building blocks.

Construction 3 ($\mathcal{C}_3$): For integers n, k and M distinct integers $d_1, d_2, \ldots, d_M$ such that $d_m \in [k+1, n-1], m \in [M]$, let $s = \mathrm{lcm}(d_1-k+1, d_2-k+1, \ldots, d_M-k+1)$. Let $l = (p-1)s^n$ and $\{x_{i,j} = x^{jn+i}\}_{i\in[n], j\in[0,s-1]}$ be a set of sn distinct nonzero elements in the polynomial ring $\mathcal{R} = \mathbb{F}_2[x]/(1+x+\cdots+x^{p-1})$, where p is a prime and $p-2 \geq sn$. Code $\mathcal{C}_3$ defined by (4) is constructed by taking $H_{t,i} = H_i^t$ with

$$H_i = \sum_{a=0}^{s^n-1} x_{i,a_i} e_a e_a^T, \quad i \in [n].$$

Here, $e_a \in \mathcal{R}^{s^n}$ is a column vector whose a-th entry is 1 and the other entries are 0s. Also, $a \in [0, s^n-1]$ is represented by its s-ary expansion, i.e., $a = (a_n, a_{n-1}, \ldots, a_1) = \sum_{i=1}^n a_i \cdot s^{i-1}$ where $a_i \in [0, s-1]$. The code constructed in *Construction 3* is named $\mathcal{C}_3$ in this paper.

Theorem 3: Code $\mathcal{C}_3$ achieves the lower bound (2) on repair bandwidth for $d_1, d_2, \ldots, d_M \in [k+1, n-1]$ simultaneously.

Proof: For any $m \in [M]$, the set $[0, s-1]$ can be partitioned into s/s_m disjoint subsets $\mathcal{I}_1 = [0, s_m-1], \mathcal{I}_2 = [s_m, 2s_m-1], \ldots, \mathcal{I}_{s/s_m} = [(s/s_m-1)s_m, s-1]$ as $s_m \mid s$, i.e., for any $\delta \in [s/s_m], |\mathcal{I}_\delta| = s_m$. Assume node $i \in [n]$ is failed and let $a(i,u) = (a_n, \ldots, a_{i+1}, u, a_{i-1}, \ldots, a_1)$ where $u \in [0, s-1]$. The parity-check matrix of code $\mathcal{C}_3$ consists of rn diagonal matrices of size $s^n \times s^n$ over $\mathcal{R}$ and for any $a \in [0, s^n-1]$ we can write down the corresponding parity-check equations as

$$x_{i,a_i}^t c_{i,a} + \sum_{j\in[n]\backslash\{i\}} x_{j,a_j}^t c_{j,a} = 0, \quad t \in [0, r-1]. \tag{18}$$

For fixed $u \in \mathcal{I}_\delta, \delta \in [s/s_m]$ and $a(i,u)$, we have

$$x_{i,u}^t c_{i,a(i,u)} + \sum_{j\in[n]\backslash\{i\}} x_{j,a_j}^t c_{j,a(i,u)} = 0, \quad t \in [0, r-1]. \tag{19}$$

Summing up (19) over $u \in \mathcal{I}_\delta$, we have

$$\sum_{u\in\mathcal{I}_\delta} x_{i,u}^t c_{i,a(i,u)} + \sum_{j\in[n]\backslash\{i\}} x_{j,a_j}^t \Big(\sum_{u\in\mathcal{I}_\delta} c_{j,a(i,u)}\Big) = 0, \quad t \in [0, r-1], \tag{20}$$

where $\sum_{u\in\mathcal{I}_\delta} c_{j,a(i,u)}$ is the data communicated from helper node j and can be denoted as $D_j^{i,a,\delta}$. According to *Lemma 1*, it is not hard to see that formula (20) defines a $(d_m+r, d_m, p-1)$ binary MDS array code. Thus, any d_m out of the $n-1$ coordinates $\{D_j^{i,a,\delta} : j \in [n]\backslash\{i\}\}$ can reconstruct the lost coordinates $\{c_{i,a(i,u)} : u \in \mathcal{I}_\delta\}$ where $p-1$ bits are downloaded on each helper node. By setting $\delta = 1, 2, \ldots, s/s_m$, we conclude that $\{c_{i,a(i,0)}, c_{i,a(i,1)}, \ldots, c_{i,a(i,s-1)}\}$ can be repaired by downloading $(p-1)s/s_m$ bits on each of the d_m helper nodes. Note that C_i can be partitioned into s^{n-1} disjoint sets, each of which can be written as $\{c_{i,a(i,0)}, c_{i,a(i,1)}, \ldots, c_{i,a(i,s-1)}\}$ for some $a \in [0, s^n-1]$. As a result, for the overall repair of C_i, a total number of $(p-1)s^n/s_m$ bits are communicated from each of the d_m helper nodes, achieving the lower bound (2) on repair bandwidth for d_m. The proof is completed since this holds for any $m \in [M]$. ■

Property 3: Code $\mathcal{C}_3$ is an MDS array code.

Proof: The proof follows the same arguments as that in the proof of Property 1 and is omitted. ■

Similar to code $\mathcal{C}_2$, code $\mathcal{C}_3$ possesses the error-resilient capability while achieving lower bound (3) on repair bandwidth

for several values of $d \in [k+1, n-1]$ simultaneously. Details about this error-resilient capability of code $\mathcal{C}_3$ are given in the following remark.

Remark 3: Since formula (20) defines a $(d_m + r, d_m, p-1)$ binary MDS array code with minimum distance equal to $r+1$, we conclude for any integer e such that $e \leq \frac{r}{2}$, the lost bits $\{c_{i,a(i,u)} : u \in \mathcal{I}_\delta\}$ can be recovered by connecting any $d_m + 2e$ out of the $n-1$ coordinates $\{D_j^{i,a,\delta} : j \in [n] \backslash \{i\}\}$ as long as the number of erroneous coordinates in the $d_m + 2e$ coordinates is not greater than e. As a result, for any $i \in [n]$ and $e \leq \frac{r}{2}$, C_i can be repaired by connecting any $d_m + 2e$ helper nodes and downloading $(p-1)s^n/s_m$ bits on each node as long as the number of erroneous nodes is not greater than e. Thus, code $\mathcal{C}_3$ achieves the lower bound (3) on repair bandwidth for all $d_m, m \in [M]$ with e-error resilient capability where $e \leq \frac{r}{2}$.

Through a further slight modification of *Construction 3*, as presented in the following remark, we obtain the binary MSR code for all values of $d \in [k+1, n-1]$ simultaneously.

Remark 4: By substituting s in *Construction 3* with $s = \mathrm{lcm}(2, \ldots, r)$, we can obtain binary MDS array code which achieves the lower bound (2). The resultant code also achieves the lower bound (3) on repair bandwidth for all values of $d \in [k+1, n-1]$ simultaneously with e-error resilient capability where $e \leq \frac{r}{2}$.

IV. Binary MDS array codes with optimal access bandwidth

In the previous section, we constructed three families of binary MDS array codes with optimal repair bandwidth. However, one can find that despite the amount of data communicated through the storage network achieves the lower bound, all the data stored on the helper nodes are accessed to repair the failed node. As a result, the total number of data accessed is $d/k > 1$ times of that in the trivial repair of an MDS code, which will consume massive disk I/Os.

In this section, we construct binary MDS array codes with optimal-access property, i.e., the amount of data accessed during node repair is equal to the minimum amount of data that need to be communicated. Specifically, two families of codes $\mathcal{C}_4$ and $\mathcal{C}_5$ are presented, which achieve the optimal access bandwidth for arbitrary fixed $d \in [k+1, n-1]$ and for multiple values of $d \in [k+1, n-1]$, respectively. The codes in this section are also constructed over the binary polynomial ring $\mathcal{R} = \mathbb{F}_2[x]/(1 + x + x^2 + \cdots + x^{p-1})$.

A. Binary MDS array code with optimal access bandwidth for arbitrary $d \in [k+1, n-1]$

In this subsection, by using some special permutation matrices over $\mathcal{R}$ as building blocks, we construct binary MDS array code with optimal access bandwidth for arbitrary fixed $d \in [k+1, n-1]$. Now we give a toy example to highlight the main ideas of the general construction.

Example 2. We take $n = 4, k = 2, d = 3$ for example and the binary MDS array code $\mathcal{C}(n = 4, k = 2, l)$ can be defined by a set of matrices $\{H_{t,i} : t \in [0,1], i \in [4]\}$ according to (4). Instead of constructing $rn = 8$ matrices, by making $H_{t,i} = H_i^t$, we only need $n = 4$ matrices $\{H_i : i \in [4]\}$ to define the code. Let $p = 7$ and $l = (p-1)(d-k+1)^n = 96$, the code $\mathcal{C}(4, 2, 96)$ is obtained by constructing 4 permutation matrices $\{H_i : i \in [4]\}$ of size 16×16 over the binary polynomial ring $\mathcal{R} = \mathbb{F}_2[x]/(1 + x + x^2 + \cdots + x^6)$. Specifically, for H_1 and $a = (a_4, a_3, a_2, a_1) \in [0, 15]$, the $a(1, a_i \oplus 1)$-th entry in the a-th row of H_1 is x^1 if $a_1 = 0$ and 1 if $a_1 \neq 0$. For example, given $a = 0 = (0,0,0,0)$, the $(0,0,0,1) = 1$-st entry in the 0-th row of H_1 is x; given $a = 1 = (0,0,0,1)$, the $(0,0,0,0) = 0$-th entry in the 1-st row of H_1 is 1. The four permutation matrices H_1, H_2, H_3, H_4 are presented as follows.

$$H_1 = \begin{bmatrix}
0 & x & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & x & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x
\end{bmatrix}$$

$$H_2 = \begin{bmatrix}
0 & 0 & x^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & x^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & x^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & x^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}$$

$$H_3 = \begin{bmatrix}
0 & 0 & 0 & 0 & x^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & x^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & x^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & x^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$H_4 = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^4 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^4 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

The above example shows the core structure of the codes constructed in this section. We present the general construction and analyze the access bandwidth in the sequel.

Construction 4 ($\mathcal{C}_4$): For integers n, k and d, let $s = d-k+1$ and $l = (p-1)s^n$ where p is a prime with $p-2 \geq n$. Code $\mathcal{C}_4$ defined by (4) is constructed by taking $H_{t,i} = H_i^t$ with

$$H_i = \sum_{a=0}^{s^n-1} x_{i,a_i} e_a e_{a(i,a_i\oplus 1)}^T, \quad i \in [n],$$

where $\oplus$ denotes the operation of addition modulo s, $\{x_{i,0} = x^i\}_{i\in[n]}$ is a set of n distinct nonzero elements in the polynomial ring $\mathcal{R}$ and $x_{i,u} = 1 \in \mathcal{R}$ for $i \in [n], u \in [s-1]$. Here, $e_a \in \mathcal{R}^{s^n}$ is a column vector whose a-th entry is 1 and the other entries are 0s. Note that $a \in [0, s^n-1]$ is also represented by its s-ary expansion, i.e., $a = (a_n, a_{n-1}, \ldots, a_1) = \sum_{i=1}^n a_i \times s^{i-1}$ where $a_i \in [0, s-1]$. The code constructed in Construction 4 is named $\mathcal{C}_4$ in this paper.

It is not hard to verify that for $\forall i \in [n]$, H_i is an $s^n \times s^n$ permutation matrix over $\mathcal{R}$ and the $a(i, a_i \oplus 1)$-th entry in the a-th row of the matrix is x^i or 1 for $a \in [0, s^n-1]$, meaning that H_i is invertible in $\mathcal{R}$. To find the structure of H_i^t where $t \in [0, s-1]$, we first compute the square of H_i as

$$\begin{aligned} H_i^2 &= (\sum_{a=0}^{s^n-1} x_{i,a_i} e_a e_{a(i,a_i\oplus 1)}^T)(\sum_{b=0}^{s^n-1} x_{i,b_i} e_b e_{b(i,b_i\oplus 1)}^T) \\ &= \sum_{a=0}^{s^n-1}\sum_{b=0}^{s^n-1} x_{i,a_i} x_{i,b_i} e_a e_{a(i,a_i\oplus 1)}^T e_b e_{b(i,b_i\oplus 1)}^T \\ &= \sum_{\substack{a=0 \\ b=a(i,a_i\oplus 1)}}^{s^n-1} x_{i,a_i} x_{i,b_i} e_a e_{b(i,b_i\oplus 1)}^T \\ &= \sum_{a=0}^{s^n-1} x_{i,a_i} x_{i,a_i\oplus 1} e_a e_{a(i,a_i\oplus 2)}^T. \end{aligned}$$

Similarly, for $i \in [n]$ and $t \in [0, s-1]$, we have

$$H_i^t = \sum_{a=0}^{s^n-1} x_{i,a_i,t} e_a e_{a(i,a_i\oplus t)}^T, \tag{21}$$

where $x_{i,u,0} = 1 \in \mathcal{R}$, and $x_{i,u,t} = \prod_{v=u}^{u\oplus(t-1)} x_{i,v}$ for $u \in [0, s-1]$ and $t \in [s-1]$. Clearly, for $\forall i \in [n]$ and $\forall t \in [0, s-1]$, H_i^t is also an $s^n \times s^n$ permutation matrix whose nonzero entries are all invertible in $\mathcal{R}$.

Before unveiling the optimal access property of the code $\mathcal{C}_4$ during node repair, we introduce two properties of the permutation matrix of $H_i, i \in [n]$ and an invertible block matrix over $\mathcal{R}$, respectively, in the following two lemmas.

Lemma 2: For any $i, j \in [n]$ with $i \neq j$, $H_iH_j = H_jH_i$ and $H_i - H_j$ is invertible over $\mathcal{R}$.

Proof: According to the construction of H_i, it is easy to find that

$$\begin{aligned} H_iH_j &= (\sum_{a=0}^{s^n-1} x_{i,a_i} e_a e_{a(i,a_i\oplus 1)}^T)(\sum_{b=0}^{s^n-1} x_{j,b_j} e_b e_{b(j,b_j\oplus 1)}^T) \\ &= \sum_{a=0}^{s^n-1}\sum_{b=0}^{s^n-1} x_{i,a_i} x_{j,b_j} e_a e_{a(i,a_i\oplus 1)}^T e_b e_{b(j,b_j\oplus 1)}^T \\ &= \sum_{\substack{a=0 \\ b=a(i,a_i\oplus 1)}}^{s^n-1} x_{i,a_i} x_{j,b_j} e_a e_{b(j,b_j\oplus 1)}^T \\ &= \sum_{a=0}^{s^n-1} x_{i,a_i} x_{j,a_j} e_a e_{a(i,j,a_i\oplus 1,a_j\oplus 1)}^T = H_jH_i, \end{aligned}$$

where $a(i, j, a_i \oplus 1, a_j \oplus 1)$ is obtained by replacing a_i and a_j by $a_i \oplus 1$ and $a_j \oplus 1$, respectively. Note that the proof relies on the commutative property of $x^i \cdot x^j = x^j \cdot x^i$ in $\mathcal{R}$.

To prove the second part of the lemma that $H_i - H_j$ is invertible, we assume $H_i\mathbf{f} = H_j\mathbf{f}$, where $\mathbf{f} = (f_0, f_1, \cdots, f_{s^n-1})^T = \sum_{a=0}^{s^n-1} f_a e_a$ is a column vector of length $s^n - 1$ over $\mathcal{R}$. Clearly, we have

$$\begin{aligned} H_i\mathbf{f} &= (\sum_{a=0}^{s^n-1} x_{i,a_i} e_a e_{a(i,a_i\oplus 1)}^T)(\sum_{b=0}^{s^n-1} f_b e_b) \\ &= \sum_{a=0}^{s^n-1}\sum_{b=0}^{s^n-1} x_{i,a_i} e_a f_b e_{a(i,a_i\oplus 1)}^T e_b \\ &= \sum_{\substack{a=0 \\ b=a(i,a_i\oplus 1)}}^{s^n-1} x_{i,a_i} f_b e_a \\ &= \sum_{a=0}^{s^n-1} x_{i,a_i} f_{a(i,a_i\oplus 1)} e_a \end{aligned}$$

and similarly,

$$H_j\mathbf{f} = \sum_{a=0}^{s^n-1} x_{j,a_j} f_{a(j,a_j\oplus 1)} e_a.$$

Thus, for $\forall a \in [0, s^n-1]$ we have

$$x_{i,a_i} f_{a(i,a_i\oplus 1)} = x_{j,a_j} f_{a(j,a_j\oplus 1)}. \tag{22}$$

Since $x_{i,a_i} \in \mathcal{R}^*$ has a multiplicative inverse x_{i,a_i}^{-1} in $\mathcal{R}$ for all $i \in [n]$ and $a \in [0, s^n-1]$, multiplying (22) by x_{i,a_i}^{-1}, we obtain

$$f_{a(i,a_i\oplus 1)} = x_{i,a_i}^{-1} x_{j,a_j} f_{a(j,a_j\oplus 1)}$$

and

$$f_a = x_{i,a_i\ominus 1}^{-1} x_{j,a_j} f_{a(i,j,a_i\ominus 1,a_j\oplus 1)}.$$

Expanding $f_{a(j,a_j\oplus 1)}$ recursively, we have

$$\begin{aligned} f_a &= x_{i,a_i\ominus 1}^{-1} x_{j,a_j} x_{i,a_i\ominus 2}^{-1} x_{j,a_j\oplus 1} f_{a(i,j,a_i\ominus 2,a_j\oplus 2)} = \cdots \\ &= x_{i,a_i\ominus 1}^{-1} x_{j,a_j} \cdots x_{i,a_i\ominus s}^{-1} x_{j,a_j\oplus(s-1)} f_{a(i,j,a_i\ominus s,a_j\oplus s)} \\ &= (\prod_{u=0}^{s-1} x_{i,u})^{-1} (\prod_{u=0}^{s-1} x_{j,u}) f_a = (x^i)^{-1} x^j f_a, \end{aligned}$$

where the commutative property of multiplication over $\mathcal{R}$ is used. Note that for i, j with $1 \leq i, j \leq n \leq p$ and $i \neq j$,

$(x^i)^{-1}x^j = x^{p-i+j} \neq 1$. Thus, $H_i\mathbf{f} = H_j\mathbf{f}$ implies $\mathbf{f} = \mathbf{0}$, meaning $H_i - H_j$ is invertible over $\mathcal{R}$. ■

Lemma 3: Let $A_1, A_2, \cdots, A_m$ be $m(\geq 2)$ invertible matrices of size $l \times l$ over $\mathcal{R}$ where $A_iA_j = A_jA_i$ for all $i, j \in [m]$. The following block matrix

$$B_m = \begin{bmatrix} I & I & \cdots & I \\ A_1 & A_2 & \cdots & A_m \\ \vdots & \vdots & \vdots & \vdots \\ A_1^{m-1} & A_2^{m-1} & \cdots & A_m^{m-1} \end{bmatrix},$$

where I is an $l \times l$ diagonal matrix over $\mathcal{R}$ with the diagonal entries equal to 1, is invertible if $A_i - A_j$ is invertible for all $i, j \in [m]$.

Proof: The proof of Lemma (3) can be obtained by following the similar induction procedure to that in [24] and we omit the details for the sake of brevity. ■

Now we unveil the optimal access property of code $\mathcal{C}_4$ in the following theorem.

Theorem 4: For n, k and arbitrary fixed $d \in [k+1, n-1]$, code $\mathcal{C}_4$ achieves the lower bound (2) on repair bandwidth with optimal access property.

Proof: For $\forall i \in [n]$, H_i is an $s^n \times s^n$ permutation matrix over $\mathcal{R}$, and the parity-check equations can be written as

$$\sum_{i=1}^{n} x_{i,a_i,t}c_{i,a(i,a_i\oplus t)} = 0, \tag{23}$$

where $a \in [0, s^n - 1], t \in [0, r-1]$. Assume node i is failed, rewrite the equation (23) as

$$x_{i,a_i,t}c_{i,a(i,a_i\oplus t)} + \sum_{j\in[n]\backslash\{i\}} x_{j,a_j,t}c_{j,a(j,a_j\oplus t)} = 0. \tag{24}$$

From (24) we can see that the data $\{c_{i,a} : a_i = t\}$ can be computed once we know the data $\{c_{j,a} : j \in [n]\backslash\{i\}, a_i = 0\}$. Since (24) holds for any $t \in [0, s-1] \subseteq [0, r-1]$, knowing the set $\{c_{j,a} : j \in [n]\backslash\{i\}, a_i = 0\}$ is sufficient to repair all the data C_i stored on node i.

We now define an injection g_i from $[0, s^{n-1}-1]$ to $[0, s^n-1]$ as

$$g_i(a) = (a_{n-1}, a_{n-2}, \cdots, a_i, 0, a_{i-1}, \cdots, a_1), \tag{25}$$

where $a \in [0, s^{n-1}-1]$ can be written as $(a_{n-1}, a_{n-2}, \cdots, a_1)$ in the s-ary form.

Let $C'_j = (c_{j,g_i(0)}, c_{j,g_i(1)}, \cdots, c_{j,g_i(s^{n-1}-1)})^T \in R^{s^{n-1}}$ for $j \in [n]\backslash\{i\}$. According to (21), we have

$$H_j^s = \sum_{a=0}^{s^n-1}\Big(\prod_{v=u}^{u\oplus(s-1)} x_{j,v}\Big)e_a e_{a(j,a_j\oplus s)}^T = \sum_{a=0}^{s^n-1} x^j e_a e_a^T = x^j I$$

for $j \in [n]$. For $w \in [0, r-s-1]$, we have

$$\sum_{j=1}^{n} H_j^w C_j = \mathbf{0} \tag{26}$$

and

$$\mathbf{0} = \sum_{j=1}^{n} H_j^{w+s} C_j = \sum_{j=1}^{n} x^j H_j^w \tag{27}$$

according to (4) with $H_{w,j} = H_j^w$. Multiplying (26) by $x^i \in \mathcal{R}^*$ and substracting the result from (27), we obtain

$$\sum_{j=1}^{n}(x^j - x^i)H_j^w C_j = \sum_{j\in[n]\backslash\{i\}}(x^j - x^i)H_j^w C_j = \mathbf{0}, \tag{28}$$

where $w \in [0, r-s-1]$. For $a \in [0, s^{n-1}-1]$, let $e'_a \in \mathcal{R}^{s^{n-1}}$ be a column vector whose a-th entry is $1 \in \mathcal{R}$ and the other entries are $0 \in \mathcal{R}$. For $j \in [n]\backslash\{i\}$, let A_j be an $s^{n-1} \times s^{n-1}$ matrix over $\mathcal{R}$ which is computed as

$$A_j = \begin{cases} \sum_{a=0}^{s^{n-1}-1} x_{j,a_j} e'_a (e'_{a(j,a_j\oplus 1)})^T, & j \in [i-1], \\ \sum_{a=0}^{s^{n-1}-1} x_{j+1,a_j} e'_a (e'_{a(j,a_j\oplus 1)})^T, & j \in [i, n-1]. \end{cases}$$

It can be verified that for $j \in [i-1]$, A_j is a permutation matrix and the $a(j, a_j \oplus 1)$-th entry in the a-th row of the matrix is x^j or 1; for $j \in [i, n-1]$, A_j is also a permutation matrix and the $a(j, a_j \oplus 1)$-th entry in the a-th row of the matrix is x^{j+1} or 1, where $a \in [0, s^{n-1}-1]$. According to (28), we have

$$\sum_{j=1}^{i-1}(x^j - x^i)A_j^w C'_j + \sum_{j=i}^{n-1}(x^{j+1} - x^i)A_j^w C'_j = \mathbf{0} \tag{29}$$

where $w \in [0, r-s-1]$. With the commutative property of multiplication in $\mathcal{R}$, we can rewrite (29) as

$$\sum_{j=1}^{i-1} A_j^w(x^j - x^i)C'_j + \sum_{j=i}^{n-1} A_j^w(x^{j+1} - x^i)C'_{j+1} = \mathbf{0} \tag{30}$$

for all $w \in [0, r-s-1]$. Following the similar arguments to that in the proof of Lemma 2, one can verify that for any $j_1, j_2 \in [n-1]$ with $j_1 \neq j_2$, $A_{j_1}A_{j_2} = A_{j_2}A_{j_1}$ and the matrix $A_{j_1} - A_{j_2}$ is invertible over $\mathcal{R}$. According to Lemma 3, any $r-s-1$ columns of the block matrix

$$\begin{bmatrix} I & I & \cdots & I \\ A_1 & A_2 & \cdots & A_{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ A_1^{r-s-1} & A_2^{r-s-1} & \cdots & A_{n-1}^{r-s-1} \end{bmatrix}$$

is invertible over $\mathcal{R}$. Thus,

$$\big((x-x^i)C'_1, (x^2-x^i)C'_2, \cdots, (x^{i-1}-x^i)C'_{i-1}, (x^{i+1}-x^i)C'_{i+1}, \cdots, (x^{n-1}-x^i)C'_n\big)$$

is a codeword of an $(n-1, n-1-(r-s-1), s^{n-1}) = (n-1, d, (p-1)s^{n-1})$ MDS array code. For $j \in [i-1]$, $(x^j - x^i)$ always has a multiplicative inverse in $\mathcal{R}$ and so does $(x^{j+1} - x^i)$ for $j \in [i+1, n]$. As a result, $(C'_1, C'_2, \cdots, C'_{i-1}, C'_{i+1}, \cdots, C'_n)$ is also a codeword of an $(n-1, d, (p-1)s^{n-1})$ MDS array code.

Clearly, any d out of the $n-1$ coordinates $\{C'_j : j \in [n]\backslash\{i\}\}$ are sufficient to reconstruct the whole codeword, and with (25), the set $\{c_{j,a} : j \in [n]\backslash\{i\}, a_i = 0\}$ is further obtained. Finally, C_i is recovered by (24). It is not hard to verify that in the above procedure, there are $d(p-1)s^{n-1}$ bits accessed and communicated, achieving the lower bound (2) on repair bandwidth for arbitrary fixed $d \in [k+1, n-1]$. ■

Property 4: Code $\mathcal{C}_4$ is an MDS array code.

Proof: The MDS property of $\mathcal{C}_4$ can be directly obtained according to Lemma 2 and Lemma 3. ■

During the repair of node $C_i, i \in [n]$, since $(C'_1, C'_2, \cdots, C'_{i-1}, C'_{i+1}, \cdots, C'_n)$ forms an MDS array code, we claim that code $\mathcal{C}_4$ possesses the error-resilient capability while achieving the lower bound (3) on repair bandwidth with optimal access property. The following remark presents the error-resilient capability of code $\mathcal{C}_4$ in detail.

Remark 5: Since $(C'_1, C'_2, \cdots, C'_{i-1}, C'_{i+1}, \cdots, C'_n)$ forms an $(n-1, d, (p-1)s^{n-1})$ MDS array code, any $d+2e$ out of the $n-1$ remaining coordinates $\{C'_j : j \in [n]\backslash\{i\}\}$ can reconstruct the codeword as long as the number of erroneous coordinates among the $d+2e$ coordinates is not greater than e. Thus, the recovery of the whole coordinate C_i is guaranteed by (24). In total, there are $(p-1)(d+2e)s^{n-1} = \frac{(d+2e)l}{d-k+1}$ bits are accessed and communicated during the repair of C_i, achieving the lower bound (3) on access bandwidth for any fixed $d \in [k+1, n-1]$ with e error-resilient capability where $e \leq \frac{n-1-d}{2}$.

B. Binary MDS array code with optimal access bandwidth for multiple values of $d \in [k+1, n-1]$ simultaneously

In this subsection, we present the construction of binary MDS code with optimal access property for multiple values of $d \in [k+1, n-1]$ simultaneously through a slight modification to code $\mathcal{C}_4$.

Construction 5 ($\mathcal{C}_5$): For integers n, k and M distinct integers $d_1, d_2, \ldots, d_M$ such that $d_m \in [k+1, n-1]$ for $\forall m \in [M]$. Let $s = \text{lcm}(d_1-k+1, d_2-k+1, \ldots, d_M-k+1)$ and $l = (p-1)s^n$ where p is a prime with $p-2 \geq n$. Code $\mathcal{C}_5$ defined by (4) is constructed by taking $H_{t,i} = H_i^t$ with

$$H_i = \sum_{a=0}^{s^n-1} x_{i,a_i} e_a e_{a(i,a_i\oplus 1)}^T, \quad i \in [n],$$

where $\oplus$ denotes the operation of addition modulo s, $\{x_{i,0} = x^i\}_{i\in[n]}$ is a set of n distinct elements in $\mathcal{R}$ and $x_{i,u} = 1 \in \mathcal{R}$ for $i \in [n], u \in [s-1]$. Here, $e_a \in \mathcal{R}^{s^n}$ is a column vector whose a-th entry is 1 and the other entries are 0s. Note that $a \in [0, r^n-1]$ is also represented by its s-ary expansion, i.e., $a = (a_n, a_{n-1}, \ldots, a_1) = \sum_{i=1}^n a_i \cdot s^{i-1}$ where $a_i \in [0, s-1]$. The code constructed in Construction 5 is named $\mathcal{C}_5$ in this paper.

Similar to code $\mathcal{C}_4$, for code $\mathcal{C}_5$, it can be verified that for any $i \in [n]$, H_i is an $s^n \times s^n$ permutation matrix and the $a(i, a_i \oplus 1)$-th entry in the a-th row of the matrix is x^i or 1 for $a \in [0, s^n-1]$, meaning that H_i is invertible in $\mathcal{R}$. The power of H_i, i.e., H_i^t, is also an $s^n \times s^n$ permutation matrix whose nonzero entries are all invertible in $\mathcal{R}$.

Theorem 5: Code $\mathcal{C}_5$ achieves the lower bound (2) on repair bandwidth with optimal access property for $d_1, d_2, \ldots, d_M$ simultaneously.

Proof: For any $a \in [0, s^n-1]$, the parity-check equation of code $\mathcal{C}_5$ can be written as

$$\sum_{i=1}^n x_{i,a_i,t} c_{i,a(a_i\oplus t)} = 0, \tag{31}$$

where $x_{i,u,0} = 1 \in \mathcal{R}$, and $x_{i,u,t} = \prod_{v=u}^{u\oplus(t-1)} x_{i,v}$ for $u \in [0, s-1]$ and $t \in [0, r-1]$.

For $d_m, m \in [M]$, let $s_m = d_m - k + 1$ and the set $[0, s-1]$ can be partitioned into s/s_m disjoint subsets $\mathcal{I}_1 = [0, s_m - 1], \mathcal{I}_2 = [s_m, 2s_m - 1], \ldots, \mathcal{I}_{s/s_m} = [(s/s_m - 1)s_m, s-1]$ as $s_m \mid s$, i.e., for any $\delta \in [s/s_m], |\mathcal{I}_\delta| = s_m$. Without loss of generality, we only consider the repair of node n here. Rewrite the parity-check equation (31) as

$$x_{n,a_n,t} c_{n,a(n,a_n\oplus t)} + \sum_{j=1}^{n-1} x_{j,a_j,t} c_{j,a(j,a_j\oplus t)} = 0. \tag{32}$$

For $\delta \in [s/s_m]$, with (32) one can verify that the values of $\{c_{n,a} : a_n = (\delta-1)s_m + t\}$ can be obtained from $\{c_{j,a} : j \in [n-1], a_i = (\delta-1)s_m\}$. Since this holds for any $t \in [0, s_m - 1] \subseteq [0, r-1]$, the values of $\{c_{n,a} : a_n \in \mathcal{I}_\delta\}$ can be obtained from $\{c_{j,a} : j \in [n-1], a_i = (\delta-1)s_m\}$. For the overall repair recovery of C_i, we only need to know $\{c_{j,a} : j \in [n-1], a_i = (\delta-1)s_m, \delta \in [s/s_m]\}$. Define an injection g_i from $[0, s^n/s_m - 1]$ to $[0, s^n - 1]$ as

$$g_i(a) = (s_m a_n, a_{n-1}, \ldots, a_1),$$

where $a \in [0, s^n/s_m - 1]$ can be written as $(a_n, a_{n-1}, \cdots, a_1)$ in the s-ary form.

Define $C'_j = (c_{j,g_i(0)}, c_{j,g_i(1)}, \ldots, c_{j,g_i(s^n/s_m-1)})^T$ for all $j \in [n-1]$. We want to prove that $(C'_1, C'_2, \ldots, C'_{n-1})$ forms an $(n-1, d_m, (p-1)s^n/s_m)$ binary MDS array code. Note that for any $w \in [0, r - s_m - 1]$, we have

$$\sum_{j=1}^n H_j^w C_j = \mathbf{0} \tag{33}$$

and

$$\sum_{j=1}^n H_j^{w+s_m} C_j = \mathbf{0}. \tag{34}$$

Multiplying (33) by $H_n^{s_m}$ and substracting the result from (34), we have

$$\sum_{j=1}^{n-1} (H_j^{s_m} - H_n^{s_m}) H_j^w C_j = \mathbf{0} \tag{35}$$

for all $w \in [0, r - s_m - 1]$. For $a \in [0, s^n/s_m - 1]$, let $e'_a \in \mathcal{R}^{s^n/s_m}$ be a column vector whose a-th entry is $1 \in \mathcal{R}$ and the other entries are $0 \in \mathcal{R}$. For $j \in [n-1]$, let A_j be an $s^n/s_m \times s^n/s_m$ matrix over $\mathcal{R}$ which is computed as

$$A_j = \sum_{a=0}^{s^n/s_m-1} x_{j,a_j} e'_a (e'_{a(j,a_j\oplus 1)})^T, \quad j \in [n-1],$$

and

$$A_n = \sum_{a=0}^{s^n/s_m-1} \Big(\prod_{u=s_m a_n}^{s_m a_n + s_m - 1} x_{n,u} \Big) e'_a (e'_{a(n,(s_m a_n \oplus s_m)/s_m)})^T.$$

With (35) and g_i, we have

$$\sum_{j=1}^{n-1} (A_j^{s_m} - A_n) A_j^w C'_j = \mathbf{0} \tag{36}$$

This article has been accepted for publication in IEEE Transactions on Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2023.3343421

for all $w \in [0, r-s_m-1]$. Note that for all $j \in [n-1]$, we have

$$\begin{aligned}
A_jA_n &= \left(\sum_{a=0}^{s^n/s_m-1} x_{j,a_j} e'_a (e'_{a(j,a_j+1)})^T\right) \\
&\quad \left(\sum_{b=0}^{s^n/s_m-1} \Big(\prod_{u=s_mb_n}^{s_mb_n+s_m-1} x_{n,u}\Big) e'_b (e'_{b(n,(s_mb_n\oplus s_m)/s_m)})^T\right) \\
&= \sum_{a=0}^{s^n/s_m-1}\sum_{b=0}^{s^n/s_m-1} x_{j,a_j}\left(\prod_{u=s_mb_n}^{s_mb_n+s_m-1} x_{n,u}\right) \\
&\quad e'_a \left(e_{a(j,a_j+1)}\right)^T e'_b \left(e'_{b(n,(s_mb_n\oplus s_m)/s_m)}\right)^T \\
&= \sum_{\substack{a=0\\ b=(j,a_j+1)}}^{s^n/s_m-1} x_{j,a_j}\Big(\prod_{u=s_mb_n}^{s_mb_n+s_m-1} x_{n,u}\Big) e'_a \\
&\quad (e'_{b(n,(s_mb_n\oplus s_m)/s_m)})^T \\
&= \sum_{a=0}^{s^n/s_m-1} x_{j,a_j}\Big(\prod_{u=s_mb_n}^{s_mb_n+s_m-1} x_{n,u}\Big) e'_a \\
&\quad (e'_{a(j,n,a_j+1,(s_mb_n\oplus s_m)/s_m)})^T = A_nA_j
\end{aligned}$$

where $a(j,n,a_j+1,(s_mb_n\oplus s_m)/s_m)$ is obtained by replacing a_j and a_n by a_j+1 and $(s_mb_n\oplus s_m)/s_m$, respectively. With this multiplication commutative property, equation (36) can be rewritten as

$$\sum_{j=1}^{n-1} A_j^w (A_j^{s_m} - A_n) C'_j = \mathbf{0}, \tag{37}$$

where $w \in [0, r-s_m-1]$. Following the similar arguments to that in the proof of Lemma (2), one can verify that $A_{j_1}A_{j_2} = A_{j_2}A_{j_1}$ and $A_{j_1} - A_{j_2}$ is invertible over $\mathcal{R}$ for all $j_1 \neq j_2 \in [n-1]$. According to Lemma (3), equation (37) defines an $(n-1, d_m, (p-1)s^n/s_m)$ binary MDS array code whose parity-check equations in matrix form can be written as

$$\begin{bmatrix} I & I & \cdots & I \\ A_1 & A_2 & \cdots & A_{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ A_1^{r-s_m-1} & A_2^{r-s_m-1} & \cdots & A_{n-1}^{r-s_m-1} \end{bmatrix}.$$

Specifically,

$$\left((A_1^{s_m} - A_n)C'_1, (A_2^{s_m} - A_n)C'_2, \ldots, (A_{n-1}^{s_m} - A_n)C'_{n-1}\right)$$

forms an $(n-1, d_m, (p-1)s^n/s_m)$ binary MDS array code. To further obtain that $(C'_1, C'_2, \ldots, C'_{n-1})$ forms an $(n-1, d_m, (p-1)s^n/s_m)$ MDS code, we need to prove that $A_j^{s_m} - A_n$ is invertible over $\mathcal{R}$ for all $j \in [n-1]$. We now assume that $A_j^{s_m}\mathbf{f} = A_n\mathbf{f}$ for some $j \in [n-1]$, where $\mathbf{f} = (f_0, f_1, \ldots, f_{s^n/s_m-1}) = \sum_{a=0}^{s^n/s_m-1} f_a e'_a \in \mathcal{R}^{s^n/s_m}$ is a column vector of length s^n/s_m over $\mathcal{R}$. Since

$$\begin{aligned}
A_j^{s_m}\mathbf{f} &= \Big(\sum_{a=0}^{s^n/s_m-1} x_{j,a_j} e'_a (e'_{a(j,a_j\oplus 1)})^T\Big)^{s_m} \Big(\sum_{b=0}^{s^n/s_m-1} f_b e'_b\Big) \\
&= \Big(\sum_{a=0}^{s^n/s_m-1} \Big(\prod_{u=a_j}^{a_j\oplus(s_m-1)} x_{j,u}\Big) e'_a (e'_{a(j,a_j\oplus s_m)})^T\Big) \\
&\quad \Big(\sum_{b=0}^{s^n/s_m-1} f_b e'_b\Big) \\
&= \sum_{a=0}^{s^n/s_m-1} \Big(\prod_{u=a_j}^{a_j\oplus(s_m-1)} x_{j,u}\Big) f_{a(j,a_j\oplus s_m)} e'_a
\end{aligned}$$

and

$$\begin{aligned}
A_n\mathbf{f} &= \sum_{a=0}^{s^n/s_m-1} \Big(\prod_{u=s_ma_n}^{s_ma_n+s_m-1} x_{n,u}\Big) e'_a \\
&\quad (e'_{a(n,(s_ma_n\oplus s_m)/s_m)})^T \Big(\sum_{b=0}^{s^n/s_m-1} f_b e'_b\Big) \\
&= \sum_{a=0}^{s^n/s_m-1} \Big(\prod_{u=s_ma_n}^{s_ma_n+s_m-1} x_{n,u}\Big) f_{a(n,(s_ma_n\oplus s_m)/s_m)}
\end{aligned},$$

we have

$$\begin{aligned}
&\Big(\prod_{u=a_j}^{a_j\oplus(s_m-1)} x_{j,u}\Big) f_{a(j,a_j\oplus s_m)} = \\
&\qquad \Big(\prod_{u=s_ma_n}^{s_ma_n+s_m-1} x_{n,u}\Big) f_{a(n,(s_ma_n\oplus s_m)/s_m)}
\end{aligned}$$

for all $a \in [0, s^n/s_m-1]$. Since $x_{j,u} \in \mathcal{R}^*$ has a multiplicative inverse $x_{j,u}^{-1}$ in $\mathcal{R}$ for all $j \in [n-1]$ and $u \in [0, s-1]$, we obtain

$$\begin{aligned}
f_{a(j,a_j\oplus s_m)} &= \Big(\prod_{u=a_j}^{a_j\oplus(s_m-1)} x_{j,u}\Big)^{-1} \Big(\prod_{u=s_ma_n}^{s_ma_n+s_m-1} x_{n,u}\Big) \\
&\quad f_{a(n,(s_ma_n\oplus s_m)/s_m)}
\end{aligned}$$

and

$$\begin{aligned}
f_a &= \Big(\prod_{u=a_j\ominus s_m}^{a_j\ominus 1} x_{j,u}\Big)^{-1} \Big(\prod_{u=s_ma_n}^{s_ma_n+s_m-1} x_{n,u}\Big) \\
&\quad f_{a(j,n,a_j\ominus s_m,(s_ma_n\oplus s_m)/s_m)}.
\end{aligned}$$

Expanding $f_{a(j,n,a_j\ominus s_m,(s_ma_n\oplus s_m)/s_m)}$ recursively, we have

$$\begin{aligned}
f_a &= \Big(\prod_{u=a_j\ominus s_m}^{a_j\ominus 1} x_{j,u}\Big)^{-1} \Big(\prod_{u=s_ma_n}^{s_ma_n+s_m-1} x_{n,u}\Big) \Big(\prod_{u=a_j\ominus 2s_m}^{a_j\ominus(s_m+1)} x_{j,u}\Big)^{-1} \\
&\quad \Big(\prod_{u=s_ma_n\oplus s_m}^{s_ma_n\oplus s_m+s_m-1} x_{n,u}\Big) f_{a(j,n,a_j\ominus 2s_m,(s_ma_n\oplus 2s_m)/s_m)} \\
&= \cdots = \Big(\prod_{u=0}^{s-1} x_{j,u}\Big)^{-1} \Big(\prod_{u=0}^{s-1} x_{n,u}\Big) f_a = (x^j)^{-1}x^n f_a
\end{aligned}$$

for all $a \in [0, s^n/s_m]$. Note that for all $j \in [n-1]$, $(x^j)^{-1}x^n = x^{p-j+n} \neq 1$, meaning $f_a = 0 \in \mathcal{R}$. Thus,

$A_j^{s_m}\mathbf{f} = A_n\mathbf{f}$ implies $\mathbf{f} = \mathbf{0}$, i.e., $A_j^{s_m} - A_n$ is invertible over $\mathcal{R}$ for all $j \in [n-1]$. ■

Property 5: Code $\mathcal{C}_5$ is an MDS array code.

Proof: Similar to code $\mathcal{C}_4$, the MDS property of code $\mathcal{C}_5$ can be directly obtained according to Lemma 2 and Lemma 3. ■

During the repair of node $C_i, i \in [n]$, since $(C'_1, C'_2, \ldots, C'_{i-1}, C'_{i+1}, \ldots, C'_n)$ forms an $(n-1, d_m, (p-1)s^n/s_m)$ binary MDS array code, we conclude that code $\mathcal{C}_5$ possesses the error-resilient capability while achieving the lower bound (3) with optimal access property. The details are presented in the following remark.

Remark 6: Since $(C'_1, C'_2, \ldots, C'_{i-1}, C'_{i+1}, \ldots, C'_n)$ forms an $(n-1, d_m, (p-1)s^n/s_m)$ MDS code, any $d_m + 2e$ out of the $(n-1)$ coordinates can reconstruct the whole codeword as long as the number of erroneous coordinates among the $d_m + 2e$ coordinates is not greater than e. The recovery of coordinate C_i is guaranteed by (32) where a total amount of $(p-1)(s^n/s_m) = \frac{(d_m+2e)l}{d_m-k+1}$ bits are accessed and communicated, achieving the lower bound (3) on access bandwidth for all $d_m, m \in [M]$ simultaneously.

Through a further slight modification of *Construction 5*, as presented in the following remark, we obtain the binary MSR code with optimal access property for all $d \in [k+1, n-1]$ simultaneously.

Remark 7: By substituting s in *Construction 5* with $s = \mathrm{lcm}(2, \ldots, r)$, we can obtain binary MDS array code which achieves the lower bound (2) with optimal access property. The resultant code also achieves the lower bound (3) on access bandwidth for all values of $d_m \in [k+1, n-1]$ simultaneously with e-error resilient capability as long as $d_m + 2e \leq n-1$.

V. Evaluation

In this section, we evaluate the proposed codes in terms of encoding and decoding complexity. We also make comparisons of our codes with some existing codes in the literature.

According to (9), (14) and (18), the three families of codes in Section III are constructed by stacking multiple Blaum-Roth codes whose parity-check matrices are judiciously designed. We now analyze the encoding and decoding complexity of these codes. Without loss of generality, we take code $\mathcal{C}_1$ for example and rewrite the parity-check equations for any $a \in [0, r^n - 1]$ in matrix form. Also, we assume that the first k nodes are information nodes and the last $r = n-k$ nodes are parity nodes. From (13), we can obtain

$$\begin{aligned}\begin{bmatrix} 1 & 1 & \ldots & 1 \\ x_{k+1,a_{k+1}} & x_{k+2,a_{k+2}} & \cdots & x_{n,a_n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{k+1,a_{k+1}}^{r-1} & x_{k+2,a_{k+2}}^{r-1} & \cdots & x_{n,a_n}^{r-1} \end{bmatrix} \begin{bmatrix} c_{k+1,a} \\ c_{k+2,a} \\ \vdots \\ c_{n,a} \end{bmatrix} \\ = - \begin{bmatrix} 1 & 1 & \ldots & 1 \\ x_{1,a_1} & x_{2,a_2} & \cdots & x_{k,a_k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{1,a_1}^{r-1} & x_{2,a_2}^{r-1} & \cdots & x_{k,a_k}^{r-1} \end{bmatrix} \begin{bmatrix} c_{1,a} \\ c_{2,a} \\ \vdots \\ c_{k,a} \end{bmatrix}. \end{aligned} \tag{38}$$

As a result, instead of inverting an $r^{n+1} \times r^{n+1}$ matrix over $\mathcal{R}$, we only need to invert the $r \times r$ matrix for r^n times to finish encoding. Similarly, the decoding procedure also only involves the inversion of $r \times r$ matrix over $\mathcal{R}$. Besides, the fast encoding and decoding algorithms of Blaum-Roth codes can be directly used in the implementation of these three families of codes. Note that both encoding and decoding procedures of these codes can be completed in parallel since the computation operations for different $a \in [0, r^n - 1]$ are independent.

We now analyze the encoding and decoding complexity of the codes $\mathcal{C}_4$ and $\mathcal{C}_5$ constructed in Section IV. Without loss of generality, we take code $\mathcal{C}_4$ with $d = n-1$ for example. Given any $b_1, b_2, \ldots, b_k \in [0, r-1]$, note that the r^{r+1} unknown elements $\{c_{i,a} : i \in [k+1, n], a_i = b_i\}$ appear in exactly r^{r+1} equations in (23) and these r^{r+1} equations only contain these r^{r+1} unknown elements. As a result, these r^{r+1} unknown elements can be obtained by inverting the corresponding $r^{r+1} \times r^{r+1}$ matrix over $\mathcal{R}$. Similarly, the decoding of code $\mathcal{C}_4$ with $d = n-1$ also only involves the inversion of $r^{r+1} \times r^{r+1}$ matrices for r^k times, instead of the inversion of an $r^{n+1} \times r^{n+1}$ matrix over $\mathcal{R}$. Both encoding and decoding of these codes can be completed in parallel since the r^k matrix inversion operations are independent.

The most related works to the present paper are [24], [10], [22], [23] and [27]. We now make comparisons of our codes with these codes respectively to end this section.

The codes in [24] and our codes share the same core structure and the main differences between them are in two folds. First, codes in [24] are constructed over finite fields while our codes are constructed over binary field. Consequently, multiplications and divisions over finite field are avoided, which is good for code implementation on hardware since XORs and cyclic shifts can be finished fast. Second, for codes in [24], entries in the parity-check matrices are chosen from arbitrary distinct nonzero elements in a finite field, while we choose elements of special form, i.e., powers of x, in the polynomial ring $\mathcal{R} = \mathbb{F}_2[x]/(1 + x + x^2 + \cdots + x^{p-1})$ to guarantee the MDS property. One may argue that the MSR codes in [24] constructed over the finite field $\mathbb{F}_{2^m}$ can be easily converted to binary MSR codes as the field $\mathbb{F}_{2^m}$ is isomorphism to the vector space $\mathbb{F}_2^m$. However, this conversion can not avoid operations over the field, i.e., the encoding and decoding of the resulting codes still involve multiplication and divisions over the field $\mathbb{F}_{2^m}$, which are usually implemented through look-up table. We choose the special elements (powers of x) in the $\mathcal{R}$ to construct the codes, which will facilitate the encoding and decoding procedures as the fast (advanced) encoding/decoding algorithm of Blaum-Roth code can be directly used.

The binary MDS array codes constructed in [10] achieve the optimal repair bandwidth only for information nodes while the codes in the present paper has optimal repair/access repair bandwidth for both information nodes and parity nodes. Moreover, the codes in [10] have fixed redundancy, i.e., $r = 2$. It is important to construct codes with a wider range of redundancy choices to satisfy different fault tolerance requirements and the codes in this paper works for any choice of redundancy.

The codes in [22] are binary MDS array codes with optimal access bandwidth, which are constructed over the same polynomial ring as in this paper. Compared with our work in

this paper, the codes in [22] fall short mainly in two folds. First, the codes in [22] only have one repair degree, i.e., the number of helper nodes d is fixed while our codes $\mathcal{C}_3$ and $\mathcal{C}_5$ work for multiple values of d simultaneously. Second, for some $d < n-1$, a subset of the helper nodes are specific for the codes in [22], while arbitrary d out of the $n-1$ surviving nodes can help repair the failed nodes for the codes in this paper. We want to note that the "arbitrary d" property provides more flexibility in helper node choice and may be important in situations where some nodes are too busy to help repair the failed node.

The codes in [23] are also binary MDS array codes with optimal access bandwidth, which are constructed through the similar pairwise coulping technique as used in [22]. The main difference between [22] and [23] is the coulping function. The codes in [23] only work for $d = n-1$ while our codes work for any $d \in [k+1, n-1]$. Note that the transformation in [23] can be directly used to generate codes with $d < n-1$. However, as the authors in [23] explained, the resulting codes will lack the "arbitrary d" property. Thus, the case of $d < n-1$ and the d helper nodes can be arbitrarily chosen is their future work, which is done in the present paper.

The codes in [27] are binary MDS array codes with very small sub-packetization level of $l = (p-1)r$ while the optimal repair bandwidth is sacrificed. It is worth mentioning that the sub-packetization level (column length of the array code) of our codes is much larger than that of the codes in [22] and [23]. Thus, constructing binary MDS array codes with small sub-packetization level is part of our future work.

VI. Conclusion

In this paper, we proposed new constructions of binary MDS array codes with optimal repair/access bandwidth. By stacking multiple Blaum-Roth codes whose parity-check matrices are judiciously designed, we constructed three binary MDS array codes with optimal repair bandwidth. The fast encoding and decoding algorithms of Blaum-Roth codes can be directly used in the implementation of these codes. With the help of permutation matrices, we constructed two families of binary MDS array codes with optimal access bandwidth. For $d \in [k+1, n-1]$, all the codes can achieve the lower bound on optimal repair/access bandwidth with error-resilient capability. Note that the sub-packetization level of the codes in this paper is very large, and constructions of binary MDS array codes with small sub-packetization level is part of our future work.

References

[1] L. Li, C. Ying, L. Chen, Y. Dong, and Y. Luo, "New constructions of binary mds array codes with optimal repair bandwidth," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2023, pp. 2045–2050.

[2] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proc. ACM Symposium Operating Systems Principles (SIGOPS)*, Dec. 2003, pp. 29–43.

[3] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proc. IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, Jun. 2010, pp. 1–10.

[4] 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. 2012 USENIX Annual Technical Conference (USENIX ATC 12)*, Jun. 2012, pp. 15–26.

[5] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[6] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Jan. 2010, pp. 1–5.

[7] V. R. Cadambe, C. Huang, and J. Li, "Permutation code: Optimal exact-repair of a single failed node in MDS code based distributed storage systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Oct. 2011, pp. 1225–1229.

[8] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: MDS array codes with optimal rebuilding," *IEEE Trans. Inf. Theory*, vol. 59, no. 3, pp. 1597–1616, Mar. 2012.

[9] Z. Wang, I. Tamo, and J. Bruck, "Long MDS codes for optimal repair bandwidth," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aug. 2012, pp. 1182–1186.

[10] 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 (ISIT)*, Oct. 2013, pp. 887–891.

[11] D. S. Papailiopoulos, A. G. Dimakis, and V. R. Cadambe, "Repair optimal erasure codes through hadamard designs," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 3021–3037, May 2013.

[12] Z. Wang, I. Tamo, and J. Bruck, "Explicit minimum storage regenerating codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 8, pp. 4466–4480, Aug. 2016.

[13] S. Goparaju, A. Fazeli, and A. Vardy, "Minimum storage regenerating codes for all parameters," *IEEE Trans. Inf. Theory*, vol. 63, no. 10, pp. 6318–6328, Oct. 2017.

[14] M. Ye and A. Barg, "Explicit constructions of optimal-access MDS codes with nearly optimal sub-packetization," *IEEE Trans. Inf. Theory*, vol. 63, no. 10, pp. 6307–6317, Oct. 2017.

[15] M. Vajha, V. Ramkumar, B. Puranik, G. Kini, E. Lobo, B. Sasidharan, P. V. Kumar, A. Barg, M. Ye, S. Narayanamurthy *et al.*, "Clay codes: Moulding MDS codes to yield an MSR code," in *Proc. 16th USENIX Conf. File and Storage Technol. (FAST)*, Feb. 2018, pp. 139–154.

[16] J. Li, X. Tang, and C. Tian, "A generic transformation to enable optimal repair in MDS codes for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 64, no. 9, pp. 6257–6267, Sep. 2018.

[17] Y. Liu, J. Li, and X. Tang, "Explicit constructions of high-rate MSR codes with optimal access property over small finite fields," *IEEE Trans. Commun.*, vol. 66, no. 10, pp. 4405–4413, Oct. 2018.

[18] ——, "A generic transformation to enable optimal repair/access MDS array codes with multiple repair degrees," *IEEE Trans. Inf. Theory*, vol. 69, no. 7, pp. 4407–4428, Jul. 2023.

[19] M. Vajha, S. Balaji, and P. V. Kumar, "Small-d MSR codes with optimal access, optimal sub-packetization and linear field size," *IEEE Trans. Inf. Theory*, vol. 69, no. 7, pp. 4303–4332, Jul. 2023.

[20] N. Wang, G. Li, S. Hu, and M. Ye, "Constructing MSR codes with subpacketization 2n/3 for k+ 1 helper nodes," *IEEE Trans. Inf. Theory*, vol. 69, no. 6, pp. 3775–3792, Jun. 2023.

[21] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the Facebook warehouse cluster," in *Proc. 5th USENIX Workshop Hot Topics Storage File System (HotStorage)*, Jun. 2013, p. 8.

[22] H. Hou and P. P. Lee, "Binary MDS array codes with optimal repair," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1405–1422, Mar. 2020.

[23] J. Li, X. Tang, and C. Hollanti, "A generic transformation for optimal node repair in MDS array codes over F_2," *IEEE Trans. Commun.*, vol. 70, no. 2, pp. 727–738, Feb. 2021.

[24] M. Ye and A. Barg, "Explicit constructions of high-rate MDS array codes with optimal repair bandwidth," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 2001–2014, Apr. 2017.

[25] M. Blaum and R. M. Roth, "New array codes for multiple phased burst correction," *IEEE Trans. Inf. Theory*, vol. 39, no. 1, pp. 66–77, Jan. 1993.

[26] S. Pawar, S. El Rouayheb, and K. Ramchandran, "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks," *IEEE Trans. Inf. Theory*, vol. 57, no. 10, pp. 6734–6753, Oct. 2011.

[27] H. Hou, Y. S. Han, B. Bai, and G. Zhang, "Towards efficient repair and coding of binary MDS array codes with small sub-packetization," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2022, pp. 3132–3137.