Triple Fault-Tolerance Binary MDS Array Codes with Asymptotically Optimal Repair

***Abstract*—Binary maximal-distance separable (MDS) array code is one class of erasure codes, which is widely employed in distributed storage systems, due to its low computational complexity. However, one drawback of existing binary MDS array codes, such as RDP and X-code, is the high repair bandwidth. In this paper, we present an explicit construction of triple fault-tolerance binary MDS array codes with asymptotically optimal repair bandwidth.**

I. Introduction

Array codes are error-correcting codes with application to storage systems such as Redundant Arrays of Inexpensive Disks (RAID) architectures [?]. A binary array code consists of arrays of size $L \times n$, with each element of an array storing one bit. We focus on binary array code, as its low computational complexity. Among the n columns, k *information columns* store the information bits and $r = n - k$ *parity columns* store the redundant bits. The L bits in a column are stored in the same disk, or storage node. We refer to a disk as a column or a storage node interchangeably, and an entry in the array as a bit. When a storage node fails, the corresponding column of the array code is considered to be an *erasure*. If the array code can tolerate any r disk erasures, then it is called MDS array code. Examples of existing binary MDS array codes include X-code [?] and RDP [?] with $r = 2$, STAR [?], generalized RDP [?] and TIP [?] can tolerate any three erasures.

When a disk fails in a distributed storage system, we will rebuild the erased bits in the failed disk by downloading some bits from the surviving disks. The amount of the downloaded bits in a repair process is called the *repair bandwidth*. As the repair bandwidth plays a crucial role in the overall recovery time [?] and influences the system service performance [?]. It is important to minimize repair bandwidth.

The repair problem was first formulated and studied by Dimakis *et al.* in [?] using the concept of information flow graph. In the framework in [?], a data file of size B symbols is encoded and distributed to n storage nodes, with each node storing L symbols such that the file can be decoded from any k of them. Furthermore, upon the failure of a storage node, a new node replaces the failed node by downloading β symbols from each of d surviving nodes, and the repair bandwidth is $d\beta$. It is shown in [?] that the optimal repair bandwidth is

$$\frac{(k+1)L}{2}, \tag{1}$$

when $d = k + 1$. Although the optimal repair bandwidth can be achieved in [?], [?] over a large enough finite field, how to design binary MDS array codes to obtain the optimal repair bandwidth is less clear.

It is shown in [?], [?] that the repair bandwidth of X-code and RDP is 50% larger than the optimal value in (**??**). For the case of two parity columns, butterfly code is proposed in [?] with optimal repair. Wang *et al.* [?], [?] constructed the MDR codes over binary field and achieved the optimal repair. However, the optimal repair of binary MDS array codes with $r \geq 3$ parity columns is still an open problem.

In this paper, we give a new construction of binary MDS array codes with three parity columns. We show that the proposed binary MDS array code has comparable encoding complexity, compared to the existing binary MDS array codes. More importantly, the optimal repair bandwidth in (**??**) can be achieved asymptotically when k is large enough, for any one information failure. The repair bandwidth reduction is made by a cyclic structure and a well-chosen encoding matrix, where the bits accessed in a repair intersect as much as possible. A ring with cyclic structure in reducing computational complexity in regenerating codes can be found in [?], [?].

This paper is organized as follows. We first give the construction of binary MDS array codes and then present the MDS property condition in Section **??**. In Section **??**, we give the repair algorithm for one single information erasure. We conclude in Section **??**.

II. Binary MDS Array Codes

In this section, we show how our proposed new binary MDS array codes are constructed.

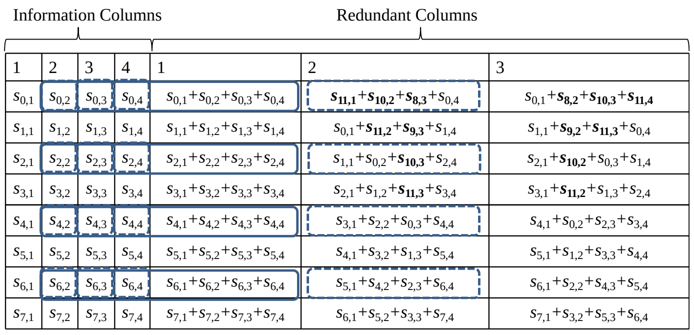
A. Construction of Binary MDS Array Codes

Let $k \geq 3$ and $L = (p-1)\tau$ be positive integers, where $\tau = 2^{k-2}$ and p is a prime number such that $2^i \not\equiv 1 \bmod p$ for $i = 1, 2, \ldots, p-2$. Assume that a file of size $k(p-1)\tau$ denoted by information bits $s_{0,i}, s_{1,i}, \ldots, s_{(p-1)\tau-1,i} \in \mathbb{F}_2^{(p-1)\tau}$ for $i = 1, 2, \ldots, k$, which are employed to generate $3(p-1)\tau$ redundant bits $s_{0,j}, s_{1,j}, \ldots, s_{(p-1)\tau-1,j} \in \mathbb{F}_2^{(p-1)\tau}$ for $j = k+1, k+2, k+3$. For $\ell = 1, 2, \ldots, k+3$ and $\mu = 0, 1, \ldots, \tau-1$, we define the following short-hand notations

$$s_{(p-1)\tau+\mu,\ell} := \sum_{j=0}^{p-2} s_{j\tau+\mu,\ell}. \tag{2}$$

We call $s_{(p-1)\tau+\mu,\ell}$ as the *parity-check bit* associated with $s_{\mu,\ell}, s_{\tau+\mu,\ell}, \ldots, s_{(p-2)\tau+\mu,\ell}$. For example, when $p = 3$, $k = 4$ and $\tau = 4$, we have the parity-check bit of $s_{0+\mu,\ell}, s_{4+\mu,\ell}$ is

$$s_{8+\mu,\ell} = s_{0+\mu,\ell} + s_{4+\mu,\ell}.$$



Information Columns				Redundant Columns		
1	2	3	4	1	2	3
$s_{0,1}$	$s_{0,2}$	$s_{0,3}$	$s_{0,4}$	$s_{0,1}+s_{0,2}+s_{0,3}+s_{0,4}$	$\mathbf{s_{11,1}}+\mathbf{s_{10,2}}+\mathbf{s_{8,3}}+s_{0,4}$	$s_{0,1}+\mathbf{s_{8,2}}+\mathbf{s_{10,3}}+\mathbf{s_{11,4}}$
$s_{1,1}$	$s_{1,2}$	$s_{1,3}$	$s_{1,4}$	$s_{1,1}+s_{1,2}+s_{1,3}+s_{1,4}$	$s_{0,1}+\mathbf{s_{11,2}}+\mathbf{s_{9,3}}+s_{1,4}$	$s_{1,1}+\mathbf{s_{9,2}}+\mathbf{s_{11,3}}+s_{0,4}$
$s_{2,1}$	$s_{2,2}$	$s_{2,3}$	$s_{2,4}$	$s_{2,1}+s_{2,2}+s_{2,3}+s_{2,4}$	$s_{1,1}+s_{0,2}+\mathbf{s_{10,3}}+s_{2,4}$	$s_{2,1}+\mathbf{s_{10,2}}+s_{0,3}+s_{1,4}$
$s_{3,1}$	$s_{3,2}$	$s_{3,3}$	$s_{3,4}$	$s_{3,1}+s_{3,2}+s_{3,3}+s_{3,4}$	$s_{2,1}+s_{1,2}+\mathbf{s_{11,3}}+s_{3,4}$	$s_{3,1}+\mathbf{s_{11,2}}+s_{1,3}+s_{2,4}$
$s_{4,1}$	$s_{4,2}$	$s_{4,3}$	$s_{4,4}$	$s_{4,1}+s_{4,2}+s_{4,3}+s_{4,4}$	$s_{3,1}+s_{2,2}+s_{0,3}+s_{4,4}$	$s_{4,1}+s_{0,2}+s_{2,3}+s_{3,4}$
$s_{5,1}$	$s_{5,2}$	$s_{5,3}$	$s_{5,4}$	$s_{5,1}+s_{5,2}+s_{5,3}+s_{5,4}$	$s_{4,1}+s_{3,2}+s_{1,3}+s_{5,4}$	$s_{5,1}+s_{1,2}+s_{3,3}+s_{4,4}$
$s_{6,1}$	$s_{6,2}$	$s_{6,3}$	$s_{6,4}$	$s_{6,1}+s_{6,2}+s_{6,3}+s_{6,4}$	$s_{5,1}+s_{4,2}+s_{2,3}+s_{6,4}$	$s_{6,1}+s_{2,2}+s_{4,3}+s_{5,4}$
$s_{7,1}$	$s_{7,2}$	$s_{7,3}$	$s_{7,4}$	$s_{7,1}+s_{7,2}+s_{7,3}+s_{7,4}$	$s_{6,1}+s_{5,2}+s_{3,3}+s_{7,4}$	$s_{7,1}+s_{3,2}+s_{5,3}+s_{6,4}$

Fig. 1: An example of storage code for three redundant columns. When information column 1 fails, the bits in the solid line box are downloaded to repair the information bits $s_{0,1}, s_{2,1}, s_{4,1}, s_{6,1}$ and the bits in the dashed box are used to repair the information bits $s_{1,1}, s_{3,1}, s_{5,1}, s_{7,1}$.

For $\ell = 1, 2, \ldots, k+3$, we present the bits $s_{0,\ell}, s_{1,\ell}, \ldots, s_{(p-1)\tau-1,\ell}$ in column ℓ, together with τ parity-check bits $s_{(p-1)\tau,\ell}, s_{(p-1)\tau+1,\ell}, \ldots, s_{p\tau-1,\ell}$, by a polynomial $s_\ell(x)$ over the ring $\mathbb{F}_2[x]$,

$$s_\ell(x) = s_{0,\ell} + s_{1,\ell}x + s_{2,\ell}x^2 + \ldots + s_{p\tau-1,\ell}x^{p\tau-1}.$$

The polynomial $s_i(x)$, corresponds to the i information column for $i = 1, 2, \ldots, k$, is called *data polynomial*. While the polynomial $s_j(x)$, corresponds to the $j-k$ parity column for $j = k+1, k+2, k+3$, is called *coded polynomial*.

We write the k data polynomials and 3 coded polynomials as the row vector

$$[s_1(x), s_2(x), \cdots, s_{k+3}(x)], \tag{3}$$

which can be computed by taking the product

$$[s_1(x), s_2(x), \cdots, s_{k+3}(x)] = [s_1(x), s_2(x), \cdots, s_k(x)] \cdot \mathbf{G}$$

with arithmetic performed in $\mathbb{F}_2[x]/(1+x^{p\tau})$, where the $k \times (k+3)$ *generator matrix* $\mathbf{G}$ is composed by the $k \times k$ identity matrix $\mathbf{I}$ and a $k \times 3$ *encoding matrix* $\mathbf{P}$,

$$\mathbf{P} := \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ x & x^2 & x^4 & \cdots & x^{2^{k-2}} & 1 \\ 1 & x^{2^{k-2}} & x^{2^{k-3}} & \cdots & x^2 & x \end{bmatrix}^T.$$

The proposed code is denoted as $\mathcal{C}(k, 3, p)$. Note that we do not store the parity-check bits in the disk. It is present only for notational convenience. Consider an example of $k = 4$ and $p = 3$, the 32 information bits are represented by $s_{0,i}$, $s_{1,i}, \ldots, s_{7,i}$, for $i = 1, 2, 3, 4$. The encoding matrix of this example is

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ x & x^2 & x^4 & 1 \\ 1 & x^4 & x^2 & x \end{bmatrix}^T.$$

The example is illustrated in Fig. **??**, where the bit with bold type is parity-check bit.

The encoding procedure can be described in terms of polynomials as follows. Given $k(p-1)\tau$ information bits, we append τ parity-check bits for each of $(p-1)\tau$ information bits and form the message vector $[s_1(x), s_2(x), \cdots, s_k(x)]$. After obtaining the vector in (**??**), we store the coefficients of the terms in the polynomials of degrees 0 to $(p-1)\tau - 1$. The proposed array code can be considered as puncturing a systematic linear code over $\mathbb{F}_2[x]/(1+x^{p\tau})$.

B. *Proof of the MDS Property*

As we choose the prime p such that the multiplication order of 2 mod p is equal to $p-1$, we have $x^{p\tau}+1$ can be factorized as a product of two co-prime factors $x^\tau + 1$ and

$$M_p^\tau(x) := x^{(p-1)\tau} + x^{(p-2)\tau} + \cdots + x^\tau + 1.$$

By the Chinese Remainder Theorem, the ring $R_{p\tau} := \mathbb{F}_2[x]/(x^{p\tau}+1)$ is isomorphic to the direct sum of $\mathbb{F}_2[x]/(x^\tau + 1)$ and $\mathbb{F}_2[x]/(M_p^\tau(x))$. Indeed, we can set up an isomorphism

$$\theta : R_{p\tau} \to \mathbb{F}_2[x]/(x^\tau + 1) \oplus \mathbb{F}_2[x]/(M_p^\tau(x))$$

by defining

$$\theta(f) := (f \bmod x^\tau + 1, f \bmod M_p^\tau(x)).$$

The mapping θ is a ring homomorphism and a bijection, because it has an inverse function $\phi(a, b)$ given by

$$\phi(a, b) := a \cdot (x^\tau + 1) + b \cdot e(x) \bmod x^{p\tau} + 1,$$

where $e(x) = x^\tau + x^{2\tau} + \cdots + x^{(p-1)\tau}$. It can be checked that the composition $\phi \circ \theta$ is the identity map of $R_{p\tau}$.

By construction, $s_\ell(x) \equiv 0 \bmod x^\tau + 1$ for all $\ell = 1, 2, \ldots, k+3$. Hence, the first components of $\theta(s_\ell(x))$'s are all equal to zero. So, we are effectively working over the ring $\mathbb{F}_2[x]/(M_p^\tau(x))$. Recall that $\tau = 2^{k-2}$, we have

$$\begin{aligned} M_p^\tau(x) =& x^{(p-1)\tau} + x^{(p-2)\tau} + \cdots + x^\tau + 1 \\ =& (x^{p-1} + x^{p-2} + \cdots + x + 1)^\tau := (M_p(x))^\tau. \end{aligned}$$

As $M_p(x)$ is irreducible in $\mathbb{F}_2[x]$ [**?**], $\mathbb{F}_2[x]/(M_p^\tau(x))$ is isomorphic to the direct sum of τ finite fields $\mathbb{F}_2[x]/(M_p(x))$. With the same discussion in [**?**], we have the following result.

Theorem 1. *$\mathcal{C}(k, 3, p)$ is MDS if for $t = 1, 2, 3$, the determinant of each $t \times t$ sub-matrix of $\mathbf{P}$ is not divisible by $x^p + 1$.*

We need the following lemma in the proof of MDS property.

Lemma 2. *Let p be a prime such that the multiplicative order of 2 mod p is equal to $p-1$. For $i = 1, 2, \ldots, p-2$, the following equation holds*

$$2^i + 2^{i+\frac{p-1}{2}} \equiv 0 \ mod\ p.$$

Proof. By Fermat's little theorem, we have $2^{p-1} \equiv 1 \mod p$. As $2^{(p-1)/2}$ is a root of $x^2 - 1 \mod p$, so $2^{(p-1)/2}$ is equal to either 1 or $-1 \mod p$. Recall that the multiplicative order of 2 mod p is equal to $p-1$, we have $2^{(p-1)/2} \not\equiv 1 \mod p$ and $2^{(p-1)/2} + 1 \equiv 0 \mod p$. Multiply both sides of the above equation by 2^i and we get the equation in Lemma **??** holds. $\square$

The next theorem gives a sufficient MDS property condition.

Theorem 3. *If $p \geq \max\{2k-8, k\}$ is a prime such that the multiplicative order of 2 mod p is equal to $p-1$, then the code $\mathcal{C}(k,3,p)$ satisfies the MDS property.*

Proof. By Theorem **??**, we need to prove that for $t = 1,2,3$, the determinant of each sub-matrix of $\mathbf{P}$ of size $t \times t$ is not divisible by x^p+1 in $\mathbb{F}_2[x]$. When $t=1$, the determinant is equal to a power of x, and hence cannot be divisible by x^p+1. When $t=2$, the determinant can be classified as x^i+1 with $i=1,2$, $x^{2^i}+x^{2^j}$ with $0 \leq i < j \leq k-2$, and $x^{2^i+2^{k-j-1}}+x^{2^j+2^{k-i-1}}$ with $1 \leq i < j \leq k-2$.

It is easy to check that x^i+1 cannot be divisible by x^p+1 for $i=1,2$. If $x^{2^i}+x^{2^j}=x^{2^i}(1+x^{2^j-2^i})$ is divisible by x^p+1, then $2^j-2^i \equiv 0 \bmod p$. We have $j \equiv i \bmod p$, which contradicts the fact that $0 \leq i < j < p$. Suppose $x^{2^i+2^{k-j-1}}+x^{2^j+2^{k-i-1}}$ is divisible by x^p+1, we have

$$2^j-2^i+2^{k-i-1}-2^{k-j-1} \equiv (2^{j-i}-1)(2^i+2^{k-j-1}) \equiv 0 \bmod p.$$

Since p is a prime and $2^{j-i}-1 \not\equiv 0 \bmod p$, this implies that $2^i+2^{k-j-1} \equiv 0 \bmod p$. By Lemma **??**, the equation $p=2k-2i-2j-1$ holds. As $1 \leq i < j \leq k-2$, we have $p=2k-2i-2j-1 \leq 2k-7$, which contradicts $p \geq 2k-8$.

For $t=3$, we need to consider the following 3 determinants

$$\begin{vmatrix} 1 & 1 & 1 \\ x & x^{2^i} & 1 \\ 1 & x^{2^{k-i-1}} & x \end{vmatrix} \tag{4}$$

with $1 \leq i \leq k-2$,

$$\begin{vmatrix} 1 & 1 & 1 \\ x & x^{2^i} & x^{2^j} \\ 1 & x^{2^{k-i-1}} & x^{2^{k-j-1}} \end{vmatrix} \tag{5}$$

with $1 \leq i < j \leq k-2$, and

$$\begin{vmatrix} 1 & 1 & 1 \\ x^{2^i} & x^{2^j} & x^{2^\ell} \\ x^{2^{k-i-1}} & x^{2^{k-j-1}} & x^{2^{k-\ell-1}} \end{vmatrix} \tag{6}$$

with $1 \leq i < j < \ell \leq k-2$.

If the determinant in (**??**) is equal to zero in $\mathbb{F}_2[x]/(x^p+1)$, then the following six terms

$$x^{2^i+1},\ x^{2^{k-i-1}+1},\ x^{2^{k-i-1}},\ x^{2^i},\ x^2 \text{ and } 1$$

can be divided into 3 pairs such that the exponents in each pair are congruent modulo p. Consider the exponent of the last term. As 2^{k-i-1}, 2^i and 2 are not congruent to 0 modulo p, we only need to consider the case of 0 congruent to 2^i+1 or $2^{k-i-1}+1$. If 0 is congruent to 2^i+1, then we have $i=\frac{p-1}{2}$ by Lemma **??** and the exponents of the remaining four terms $x^{2^{k-\frac{p-1}{2}-1}+1}$, $x^{2^{k-\frac{p-1}{2}-1}}$, x^{p-1}, x^2 are not congruent modulo p with each other. If 0 is congruent to $2^{k-i-1}+1$, then we have $k-i-1=\frac{p-1}{2}$ by Lemma **??**, it indicates that

$$i = k-\frac{p+1}{2} < \frac{p-7}{2}-\frac{p+1}{2} = -4,$$

which contradicts the fact that $i \geq 1$.

The determinant in (**??**) is

$$(x^{2^i+2^{k-j-1}}+x^{2^j}+x^{2^{k-i-1}+1})+(x^{2^j+2^{k-i-1}}+x^{2^i}+x^{2^{k-j-1}+1}).$$

None of the terms in the first parenthesis is equal to any term in the second parenthesis if the exponents are reduced modulo p, and *vice versa*. Otherwise, we can deduce the contradiction of $1 \leq i < j \leq k-2$.

Likewise, the determinant in (**??**) can be re-arranged as

$$\begin{aligned}&(x^{2^j+2^{k-\ell-1}}+x^{2^\ell+2^{k-i-1}}+x^{2^i+2^{k-j-1}})+\\&(x^{2^\ell+2^{k-j-1}}+x^{2^i+2^{k-\ell-1}}+x^{2^j+2^{k-i-1}}).\end{aligned}$$

None of the terms in the first parenthesis is equal to any term in the second parenthesis if the exponents are reduced modulo p, and *vice versa*. This proves that the determinant in (**??**) is not divisible by x^p+1, and completes the proof. $\square$

III. Asymptotically Optimal Repair of One Information Failure

In this section, we always assume that information column f erases, f can be any value from 1 to k, we want to recover the bits $s_{0,f}, s_{1,f}, \ldots, s_{(p-1)\tau-1,f}$ stored in column f by accessing bits from $k-1$ other information columns and 2 parity columns. Recall that we can compute the parity-check bits by (**??**). For notational convenience, we refer the *bits* of column i as the $p\tau$ bits $s_{0,i}, s_{1,i}, \ldots, s_{p\tau-1,i}$. Before giving the optimal repair algorithm, we formally define the *parity set* as follows.

Definition 1. *For $0 \leq \ell \leq p\tau-1$, we define the ℓ-th parity set of the first, the second and the third parity column as*

$$P_{\ell,1} = \{s_{\ell,1}, s_{\ell,2}, \ldots, s_{\ell,k}\},$$

$$P_{\ell,2} = \{s_{\ell-2^0,1}, s_{\ell-2^1,2}, \ldots, s_{\ell-2^{k-2},k-1}, s_{\ell,k}\} \text{ and}$$

$$P_{\ell,3} = \{s_{\ell,1}, s_{\ell-2^{k-2},2}, s_{\ell-2^{k-3},3}, \ldots, s_{\ell-2^0,k}\}$$

respectively.

Note that all the indices in Definition **??** and throughout the paper are taken modulo $p\tau$. From definition **??**, we see that the parity set $P_{\ell,j}$ consists of information bits which are used to generate the redundant bit $s_{\ell,k+j}$. When we say an information bit is repaired by a parity column, it means that we access the redundant bit of the parity column, and all the information bits in this parity set, except the erased bit. Consider the example in Fig. **??**, suppose that the first column is erased. One can access the bits $s_{0,2}, s_{0,3}, s_{0,4}$ and the redundant bit $s_{0,1}+s_{0,2}+s_{0,3}+s_{0,4}$ to rebuild $s_{0,1}$ by

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

Algorithm 1 Repair of one information failure

```
1: Suppose the information column $f$ is failed.
2: if $f \in \{1, 2, \ldots, \lceil k/2 \rceil\}$. then
3:     Repair the bit $s_{\ell,f}$ by the first parity, for $\ell \mod 2^f \in \{0, 1, 2, \ldots, 2^{f-1} - 1\}$. Otherwise, repair the bit $s_{\ell,f}$ by the second parity, for $\ell \mod 2^f \in \{2^{f-1}, 2^{f-1} + 1, 2^{f-1} + 2, \ldots, 2^f - 1\}$.
4: if $f \in \{\lceil k/2 \rceil + 1, \lceil k/2 \rceil + 2, \ldots, k\}$. then
5:     Repair the bit $s_{\ell,f}$ by the first parity, for $\ell \mod 2^f \in \{0, 1, 2, \ldots, 2^{f-1} - 1\}$. Otherwise, repair the bit $s_{\ell,f}$ by the third parity, for $\ell \mod 2^f \in \{2^{f-1}, 2^{f-1} + 1, 2^{f-1} + 2, \ldots, 2^f - 1\}$.
```

The repair algorithm is stated in Algorithm **??**. Let's consider the example in Fig. **??** to illustrate the repair process more understandable. In the example, $k = 5$, $d = 5$ and $\tau = 4$. Suppose that the first information column fails, i.e., $f = 1$. By steps 2 and 3 in Algorithm **??**, we can repair the bits $s_{\ell,1}$ by the first parity column for $\ell \equiv 0 \mod 2$ and $0 \leq \ell \leq 7$. More specifically, the bits $s_{0,1}, s_{2,1}, s_{4,1}, s_{6,1}$ are rebuilt by

$$\begin{aligned}
s_{0,1} &= s_{0,2} + s_{0,3} + s_{0,4} + (s_{0,1} + s_{0,2} + s_{0,3} + s_{0,4}) \\
s_{2,1} &= s_{2,2} + s_{2,3} + s_{2,4} + (s_{2,1} + s_{2,2} + s_{2,3} + s_{2,4}) \\
s_{4,1} &= s_{4,2} + s_{4,3} + s_{4,4} + (s_{4,1} + s_{4,2} + s_{4,3} + s_{4,4}) \\
s_{6,1} &= s_{6,2} + s_{6,3} + s_{6,4} + (s_{6,1} + s_{6,2} + s_{6,3} + s_{6,4}).
\end{aligned}$$

As $f = 1 \in \{1, 2\}$, the other information bits $s_{\ell,1}$ is repaired by the second parity column for $\ell \equiv 1 \mod 2$ and $0 \leq \ell \leq 7$. Therefor, the bits $s_{1,1}, s_{3,1}, s_{5,1}, s_{7,1}$ are rebuilt by

$$\begin{aligned}
s_{1,1} &= s_{0,2} + s_{10,3} + s_{2,4} + (s_{1,1} + s_{0,2} + s_{10,3} + s_{2,4}) \\
s_{3,1} &= s_{2,2} + s_{0,3} + s_{4,4} + (s_{3,1} + s_{2,2} + s_{0,3} + s_{4,4}) \\
s_{5,1} &= s_{4,2} + s_{2,3} + s_{6,4} + (s_{5,1} + s_{4,2} + s_{2,3} + s_{6,4}) \\
s_{7,1} &= s_{6,2} + s_{4,3} + s_{8,4} + (s_{11,1} + s_{10,2} + s_{8,3} + s_{0,4}) \\
&\quad + (s_{3,1} + s_{2,2} + s_{0,3} + s_{4,4}).
\end{aligned}$$

As we can compute $s_{10,3}$ by $s_{6,3} + s_{2,3}$ and $s_{8,4}$ by $s_{4,4} + s_{0,4}$, so we do not need to download the bits $s_{10,3}$ and $s_{8,4}$. Therefore, we count that we need to download 4 bits from each of the three information columns and two parity columns. There are total 20 bits downloaded from 5 columns to repair the bits of the first information column. Namely, only half of the bits of the helping columns are accessed. In Fig. **??**, the bits in the solid line box are downloaded to repair the information bits $s_{0,1}, s_{2,1}, s_{4,1}, s_{6,1}$ and the bits in the dashed box are used to repair the information bits $s_{1,1}, s_{3,1}, s_{5,1}, s_{7,1}$.

Suppose node 2 fails, i.e,. $f = 2$. By steps 2 and 3 in Algorithm **??**, we can repair the bits $s_{0,2}, s_{1,2}, s_{4,2}, s_{5,2}$ by

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

By steps 2 and 3, we should repair the bits $s_{2,2}, s_{3,2}, s_{6,2}, s_{7,2}$ by

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

As a result, the 8 bits stored in the second information column can be recovered by downloading 6 bits from the first information column and 4 bits from each of the third information column, the fourth information column, the first parity column and the second parity column. There are total 22 bits downloaded in the repair process. It can be verified that for the code in Fig. **??**, the third information column and the last information column can be rebuilt by accessing 22 bits and 20 bits from 5 columns respectively.

Theorem 4. *When $f \in \{1, 2, \ldots, \lceil k/2 \rceil\}$, the repair bandwidth of information column f by Algorithm* **??** *is*

$$(p - 1)((k + 2)2^{k-3} - 2^{k-f-2}).$$

Proof. By Algorithm **??**, the bits $s_{\ell,f}$ are repaired by the parity sets $P_{\ell,1}$ of the first parity column for $\ell \mod 2^f \in \{0, 1, 2, \ldots, 2^{f-1} - 1\}$ and $\ell < (p - 1)\tau$. Therefore, we need to access $(p - 1)\tau/2$ information bits $s_{\ell,i}$ from each of the remaining $k - 1$ information columns for $i \in \{1, 2, \ldots, f - 1, f + 1, \ldots, k\}$ and $\ell \mod 2^f \in \{0, 1, 2, \ldots, 2^{f-1} - 1\}$, and download $(p - 1)\tau/2$ redundant bits $s_{\ell,k+1}$ for $\ell \mod 2^i \in \{0, 1, 2, \ldots, 2^{i-1} - 1\}$ from the first parity column. Thus, there are $k(p - 1)\tau/2$ bits to be downloaded.

For $\ell \mod 2^f \in \{2^{f-1}, 2^{f-1} + 1, 2^{f-1} + 2, \ldots, 2^f - 1\}$, the bits $s_{\ell,f}$ are repaired by $P_{\ell+2^{f-1},2}$. Recall that

$$P_{\ell+2^{f-1},2} = \{s_{\ell+2^{f-1}-2^0,1}, \ldots, s_{\ell+2^{f-1}-2^{k-2},k-1}, s_{\ell+2^{f-1},k}\}.$$

So we need to access $(p - 1)\tau/2$ redundant bits $s_{\ell+2^{f-1},k+2}$. For column i with $i \in \{1, 2, \ldots, f - 1\}$, we need $(p - 1)\tau/2$ bits $s_{\ell,i}$ for all the values of $\ell \mod 2^f$ in the set

$$\{0, 1, \ldots, 2^{f-1} - 2^{i-1} - 1, 2^f - 2^{i-1}, 2^f - 2^{i-1} + 1, \ldots, 2^f - 1\}.$$

While for column i with $i \in \{f + 1, f + 2, \ldots, k\}$, we need $(p - 1)\tau/2$ bits $s_{\ell,i}$ for $\ell \mod 2^f \in \{0, 1, 2, \ldots, 2^{f-1} - 1\}$.

Note that the bits $s_{\ell,i}$ for $\ell \mod 2^f \in \{0, 1, 2, \ldots, 2^{f-1} - 1\}$ and $\ell < (p - 1)\tau$ have downloaded in the repair by the first parity column. Thus, we only need to download $(p - 1)\tau/2$ redundant bits from the second parity column, and $(p - 1)2^{k+i-f-3}$ bits from column i for $i = 1, 2, \ldots, f - 1$.

We can count that the total number of bits downloaded from

$k+2$ columns to repair the information column f is

$$\underbrace{k(p-1)2^{k-3}}_{\text{the first parity column}} + \underbrace{\sum_{i=1}^{f-1}(p-1)2^{k+i-f-3}}_{\text{the second parity column}}$$
$$=(p-1)((k+2)2^{k-3}-2^{k-f-2}).$$

□

When $1 \leq f \leq \lceil k/2 \rceil$, the repair bandwidth of column $k+1-f$ is the same of that of column f according to Algorithm **??**. Therefore, we only consider the cases of $1 \leq f \leq \lceil k/2 \rceil$. By Theorem **??**, repair bandwidth increases with f increases. When $f=1$, the repair bandwidth is $(k+1)(p-1)2^{k-3}$, which achieves the optimal value in (**??**). Even for the worst case of $f=\lceil k/2 \rceil$, the repair bandwidth is

$$(p-1)((k+2)2^{k-3}-2^{k-\lceil k/2 \rceil-2}) < (p-1)(k+2)2^{k-3},$$

which is strictly less than $\frac{k+2}{k+1}$ times of the value in (**??**). Therefore, the repair bandwidth of any one information failure can achieve the optimal repair in (**??**) asymptotically when k is large enough.

It should be noted that the parity sets of the first parity column in our codes are the same of that of the first parity column in RDP and EVENODD. The key difference of our codes and the existing binary MDS array codes is the construction of the second and the third parity columns. First, the parity sets of the second and the third parity columns in our codes are not bits that correspond to straight lines in the array, but the bits that correspond to polygonal lines. Second, the row number of the array in our codes is divisible by 2^{k-2}. The two properties are essential for reducing the repair bandwidth.

IV. Conclusion

In this paper, we present new binary MDS array codes with three parity columns such that the repair bandwidth of one information column is asymptotically optimal.

We define the encoding complexity as the average number of XORs needed to generate one redundant bit. By the construction, we should first compute $k\tau$ parity-check bits by (**??**), which involves $k\tau(p-2)$ XORs. Then generate the coefficients of degree from 0 to $(p-1)\tau-1$ for three coded polynomials that take $3(p-1)\tau(k-1)$ XORs. Therefore, the encoding complexity is

$$\frac{k\tau(p-2)+3(p-1)\tau(k-1)}{3(p-1)\tau} < 4k/3-1,$$

which is comparable to the encoding complexity of the existing binary MDS array codes.

The future work includes the extension of the construction with more parity columns and efficient repair algorithm for parity column.