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

Hanxu Hou[†§], Patrick P. C. Lee[§], Yunghsiang S. Han[†], Yuchong Hu[‡]
[†] School of Electrical Engineering & Intelligentization, Dongguan University of Technology
[§] Department of Computer Science and Engineering, The Chinese University of Hong Kong
[‡] School of Computer Science and Technology, Huazhong University of Science and Technology

***Abstract*—Binary maximum distance separable (MDS) array codes are a special class of erasure codes for distributed storage that not only provide fault tolerance with minimum storage redundancy, but also achieve low computational complexity. They are constructed by encoding k information columns into r parity columns, in which each element in a column is a bit, such that any k out of the $k+r$ columns suffice to recover all information bits. In addition to providing fault tolerance, it is critical to improve repair performance. Specifically, if a single column fails, our goal is to minimize the repair bandwidth by downloading the least amount of bits from d non-failed columns, where $k \leq d \leq k+r-1$. However, existing binary MDS codes that achieve high data rates (i.e., $k/(k+r) > 1/2$) and minimum repair bandwidth only support double fault tolerance (i.e., $r = 2$), which is insufficient for failure-prone distributed storage environments in practice. This paper fills the void by proposing an explicit construction of triple-fault-tolerant (i.e., $r = 3$) binary MDS array codes that achieve asymptotically minimum repair bandwidth for $d = k+1$.**

I. Introduction

Modern distributed storage systems deploy erasure codes to maintain data availability against failures of storage nodes. Binary maximum distance separable (MDS) array codes are a special class of erasure codes that achieve fault tolerance with minimum storage redundancy and low computational complexity. Specifically, a binary array code is composed of an array of $k+r$ columns with L bits each. Among the $k+r$ columns, k *information columns* store information bits and r *parity columns* store redundant bits. The L bits in each column are stored in the same storage node. The code is said to be MDS if any k out of the $k+r$ columns suffice to reconstruct all k information columns (i.e., it can tolerate any r failed columns). Examples of binary MDS array codes include X-code [1] and RDP codes [2], both of which are double-fault-tolerant (i.e., $r = 2$), as well as STAR codes [3], generalized RDP codes [4], and TIP codes, all of which are triple-fault-tolerant (i.e., $r = 3$).

When a node fails in a distributed storage system, we should repair the failed column by downloading bits from d non-failed nodes, where $k \leq d \leq k+r-1$. Minimizing the *repair bandwidth*, defined as the amount of bits downloaded in the repair operation, is critical to speed up the repair operation and minimize the window of vulnerability, especially in distributed storage in which network transfer is the bottleneck. The repair problem was first formulated by Dimakis *et al.* [5] based on the concept of information flow graph. It is shown in [5] that the minimum repair bandwidth subject to the minimum storage redundancy, also known as the *minimum storage regenerating (MSR)* point, is given by:

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

when $d = k+1$. Although the minimum repair bandwidth is achievable [5], [6] over a sufficiently large finite field, how to construct binary MDS array codes that achieve the minimum repair bandwidth remains a challenging issue.

There have been studies on reducing the repair bandwidth for a single failed column in binary MDS array codes. Some approaches minimize disk reads for RDP codes [7] and X-code [8], but their repair bandwidth is sub-optimal and 50% larger than the minimum value in (1). MDR codes [9], [10] and ButterFly codes [11] are binary MDS array codes that achieve optimal repair, but they only provide double fault tolerance (i.e., $r = 2$). How to construct binary MDS array codes with both optimal repair and higher fault tolerance (i.e., $r > 2$) is still an open problem. Such constructions will be beneficial for maintaining data availability in failure-prone distributed storage systems in practice.

In this paper, we present a new construction of triple-fault-tolerant binary MDS array codes with three parity columns (i.e., $r = 3$). We show that our proposed binary MDS array codes have comparable encoding complexity compared to the existing binary MDS array codes. More importantly, the minimum repair bandwidth in (1) for any single information column failure can be achieved asymptotically when k is sufficiently large. Our construction minimizes the repair bandwidth by exploiting a quotient ring with cyclic structure and a well-chosen encoding matrix, such that the bits accessed in a repair operation intersect as much as possible. Previous studies [12]–[14] also exploit similar techniques to reduce computational complexity of regenerating codes and array codes.

II. Our Proposed Codes

A. Construction

Let $k \geq 3$ and $L = (p-1)\tau$ be positive integers, where $\tau = 2^{k-2}$ and p is a prime such that 2 is a primitive element

This work was supported by the National Natural Science Foundation of China (Grant No. 61502191, 61502190, 61671007), the Hubei Provincial Natural Science Foundation of China (Grant No. 2016CFB226), the Research Grants Council of Hong Kong (Grant No. GRF 14216316), and the University Grants Committee of Hong Kong (Grant No. AoE/E-02/08).

in the field $\mathbb{Z}_p$. Consider 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 used 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 notation:

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

We call $s_{(p-1)\tau+\mu,\ell}$ the *extra 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 $\tau = 4$, the extra bit of $s_{0+\mu,\ell}, s_{4+\mu,\ell}$ is

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

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 extra 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 $s_i(x)$, which corresponds to the ith information column for $i = 1, 2, \ldots, k$, is called a *data polynomial*; the polynomial $s_j(x)$, which corresponds to the $j-k$ parity column for $j = k+1, k+2, k+3$, is called a *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 $R_{p\tau} := \mathbb{F}_2[x]/(1+x^{p\tau})$. The $k \times (k+3)$ *generator matrix* $\mathbf{G}$ is composed of the $k \times k$ identity matrix $\mathbf{I}$ and a $k \times 3$ *encoding matrix* $\mathbf{P}$,

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

In the ring $R_{p\tau}$, the variable x represents the *cyclic-right-shift* operator on a polynomial. This is crucial to reduce repair bandwidth for one information column failure. Our proposed code is denoted as $\mathcal{C}(k, 3, p)$. Note that we do not store the extra bits on disk; they are 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. 1, where the bits with bold type are extra bits.

The encoding procedure can be described in terms of polynomials as follows. Given $k(p-1)\tau$ information bits, we append τ extra bits for each of $(p-1)\tau$ information

Information Columns | Parity Columns

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

Fig. 1. An example of storage code for three parity 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 $s_{1,1}, s_{3,1}, s_{5,1}, s_{7,1}$.

bits and form the message vector $[s_1(x), s_2(x), \cdots, s_k(x)]$. After obtaining the vector in (3), 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 $R_{p\tau}$.

B. Proof of the MDS Property

When we say an $L \times n$ array code is MDS, it means any k columns suffice to recover all the information bits. As each column of $\mathcal{C}(k, 3, p)$ is presented as a polynomial in $R_{p\tau}$, we should first introduce $R_{p\tau}$, before giving the MDS property condition. As we choose the prime p such that 2 is a primitive element in $\mathbb{Z}_p$, $x^{p\tau}+1$ 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 Chinese Remainder Theorem, the ring $R_{p\tau}$ is isomorphic to the product ring $\mathbb{F}_2[x]/(x^\tau+1) \times \mathbb{F}_2[x]/(M_p^\tau(x))$. 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

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

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

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

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, we have $s_\ell(x) \equiv 0 \bmod x^\tau + 1$ for all $\ell = 1, 2, \ldots, k+3$ according to (2). Hence, the first components of $\theta(s_\ell(x))$'s are all 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

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

Therefore, the code $\mathcal{C}(k, 3, p)$ is MDS if every $k \times k$ submatrix of the generator matrix $\mathbf{G}$ is invertible in $\mathbb{F}_2[x]/(M_p^\tau(x))$. This is equivalent to the condition that the determinant of any $\ell \times \ell$

submatrix of $\mathbf{P}$ is indivisible by $M_p(x) := 1+x+\cdots+x^{p-1}$, for $1 \le \ell \le 3$ since $M_p(x)$ is irreducible [14, Theorem 2].

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

Lemma 1. *Let p be a prime such that 2 is a primitive element in $\mathbb{Z}_p$. For $0 \le i \le p-2$,*

$$2^i \equiv -1 \text{ mod } p \text{ if and only if } i = \frac{p-1}{2}. \tag{4}$$

Proof. First we prove that $2^i \not\equiv 2^j \bmod p$ for $0 \le i \ne j \le p-2$. Assume that $2^i \equiv 2^j \bmod p$, then $2^{\min(i,j)}(2^{|i-j|}-1) \equiv 0 \bmod p$. This is impossible since the multiplicative order of 2 mod p is equal to $p-1$, i.e., $2^{|i-j|} \not\equiv 1 \mod p$.

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 $2^{(p-1)/2}+1 \equiv 0 \mod p$. □

The next theorem gives a sufficient MDS property condition.

Theorem 2. *If $p \ge 2k-1$ is a prime such that 2 is a primitive element in $\mathbb{Z}_p$, then the code $\mathcal{C}(k,3,p)$ satisfies the MDS property.*

Proof. We need to prove that for $t = 1, 2, 3$, the determinant of each submatrix of $\mathbf{P}$ of size $t \times t$ is not divisible by $M_p(x)$ in $\mathbb{F}_2[x]$. When $t = 1$, the determinant is equal to a power of x, and hence cannot be divisible by $M_p(x)$. Note that there are even number of terms of the determinant for $t \ge 2$, it is sufficient to show that the determinant is not divisible by $1+x^p$ for $t = 2, 3$. When $t = 2$, the determinant can be classified as x^i+1 with $i = 1, 2$, $x^{2^i}+x^{2^j}$ with $0 \le i < j \le k-2$, and $x^{2^i+2^{k-j-1}}+x^{2^j+2^{k-i-1}}$ with $1 \le i < j \le k-2$.

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 $2^j-2^i \equiv 0 \bmod p$ that is equivalent to $2^{j-i} \equiv 1 \bmod p$. This contradicts to the assumption $2^\ell \not\equiv 1 \bmod p$ for $\ell = 1, 2, \ldots, p-2$.

Suppose $x^{2^i+2^{k-j-1}}+x^{2^j+2^{k-i-1}}$ is divisible by x^p+1, then

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

Since p is a prime and $2^{j-i}-1 \not\equiv 0 \bmod p$, this implies that $2^i+2^{k-j-1} \equiv 0 \bmod p$. Since $2^i+2^{k-j-1} = 2^{\min(i,k-j-1)}(1+2^{|k-j-1-i|})$, we have

$$|k-j-1-i| \equiv \frac{p-1}{2} \bmod p \tag{5}$$

by Lemma 1. Since $1 \le i < j \le k-2$, we have $|k-j-1-i| \le k-4$. However by assumption, $p \ge 2k-1$, we have $\frac{p-1}{2} \ge k-1 > k-4$. Contradiction.

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

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

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

$$\begin{vmatrix} 1 & 1 & 1 \\ x & x^{2^i} & x^{2^j} \\ 1 & x^{2^{k-i-1}} & x^{2^{k-j-1}} \end{vmatrix} \tag{7}$$

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

$$\begin{vmatrix} 1 & 1 & 1 \\ x^{2^i} & x^{2^j} & x^{2^\ell} \\ x^{2^{k-i-1}} & x^{2^{k-j-1}} & x^{2^{k-\ell-1}} \end{vmatrix} \tag{8}$$

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

By determinant calculation, (6) is

$$d(x) = x^{2^i+1}+x^{2^{k-i-1}+1}+x^{2^{k-i-1}}+x^{2^i}+x^2+1,$$

which can be reduced to

$$\begin{aligned} \tilde{d}(x) &= x^{(2^i+1) \bmod p}+x^{(2^{k-i-1}+1) \bmod p} \\ &\quad +x^{(2^{k-i-1}) \bmod p}+x^{(2^i) \bmod p}+x^2+1 \end{aligned}$$

in $R_{p\tau}$ and $\tilde{d}(x)$ has degree at most $p-1$. $\tilde{d}(x)$ cannot be divided by x^p+1 with less degree unless $\tilde{d}(x) = 0$.

If $\tilde{d}(x) = 0$ in $R_{p\tau}$, then the following six terms

$$2^i+1,\ 2^{k-i-1}+1,\ 2^{k-i-1},\ 2^i,\ 2 \text{ and } 0$$

can be divided into three 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 $2^{k-i-1}+1$. By Lemma 1 and $1 \le i \le k-2$, we have $1 \le (p-1)/2 \le k-2$ or $1 \le k-(p-1)/2 \le k-2$, which contradicts to the assumption that $p \ge 2k-1$.

The determinant in (7) is

$$\begin{aligned} d_1(x) &= (x^{2^i+2^{k-j-1}}+x^{2^j}+x^{2^{k-i-1}+1}) \\ &\quad +(x^{2^j+2^{k-i-1}}+x^{2^i}+x^{2^{k-j-1}+1}). \end{aligned}$$

Similarly, if $d_1(x) = 0$ in $R_{p\tau}$, then the following six terms

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

can be divided into three pairs such that the exponents in each pair are congruent modulo p. None of the first three terms is equal to anyone in the last three terms if they are reduced modulo p, and *vice versa*. Otherwise, we can deduce the contradiction of $1 \le i < j \le k-2$. Similarly, we can prove that the determinant (8) is not divisible by x^p+1.

□

III. Asymptotically Optimal Repair of One Information Failure

We will show how to recover the bits $s_{0,f}, s_{1,f}, \ldots, s_{(p-1)\tau-1,f}$ stored in the information column f by accessing bits from $k-1$ other information columns and 2 parity columns with asymptotically optimal repair bandwidth in this section, where $1 \le f \le k$. Recall that we can compute the extra bits by (2). 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 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*

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

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

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

Note that all the indices in Definition 1 and throughout the paper are taken modulo $p\tau$. From Definition 1, 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 given in Fig. 1. 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 that the information column f has 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 1. Let's again consider the example given in Fig. 1 to illustrate the repair process in detail. In this example, $k = 5$, $d = 5$ and $\tau = 4$. Suppose that the first information column (i.e., node 1) fails, i.e., $f = 1$. By Steps 2 and 3 in Algorithm 1, 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 $s_{4,4} + s_{0,4}$, we do not need to download the bits $s_{10,3}$ and $s_{8,4}$. Therefore, we count that we need to download four bits from each of the three information columns and two parity columns. There are total 20 bits downloaded from five columns to repair the bits of the first information column. Namely, only half of the bits of the helping columns are accessed. In Fig. 1, 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 that the second information column (i.e., node 2) fails, i.e,. $f = 2$. By Steps 2 and 3 in Algorithm 1, we can repair the bits $s_{0,2}, s_{1,2}, s_{4,2}, s_{5,2}$ by

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

Similarly, we can 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 eight bits stored in the second information column can be recovered by downloading six bits from the first information column and four 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 given in Fig. 1, the third information column and the last information column can be rebuilt by accessing 22 bits and 20 bits from 5 columns respectively.

Theorem 3. *When $f \in \{1, 2, \ldots, \lceil k/2 \rceil\}$, the repair bandwidth of information column f by Algorithm 1 is*

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

Proof. By Algorithm 1, 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 been 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 $1 \le f \le \lceil k/2 \rceil$, the repair bandwidth of column $k+1-f$ is the same of that of column f according to Algorithm 1. Therefore, we only consider the cases of $1 \le f \le \lceil k/2 \rceil$. By Theorem 3, the repair bandwidth increases when f increases. When $f=1$, the repair bandwidth is $(k+1)(p-1)2^{k-3}$, which achieves the optimal value in (1). 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 (1). Therefore, the repair bandwidth of any one information failure can achieve the optimal repair in (1) asymptotically when k is large enough.

It should be noted that the parity sets of the first parity column in the proposed codes are the same as those of the first parity column in RDP and EVENODD. The key difference between the proposed 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 the proposed 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 the proposed codes is divisible by 2^{k-2}. The two properties are essential for reducing the repair bandwidth.

IV. Encoding Computational Complexity

We define the encoding complexity as the average number of XORs needed to generate one redundant bit. By the construction, $k\tau$ extra bits are computed by (2), which involves $k\tau(p-2)$ XORs. Then generating the coefficients of degree from 0 to $(p-1)\tau-1$ for three coded polynomials takes $3(p-1)\tau(k-1)$ XORs. Thus, 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. We summarize the comparison of binary MDS array codes in Table I.

TABLE I
Comparison of binary MDS array codes.

	r	Repair	Encoding
ButterFly code	2	optimal	$k+k/2\lfloor k/2 \rfloor$
MDR-I [9]	2	optimal	$k-1$
MDR-II [9]	2	optimal	k
Proposed code	3	Asymptotically optimal	$4k/3-1$
RDP	2	not optimal	$k-1$
X-code	2	not optimal	$k-1$
STAR	3	not optimal	$k-\frac{2}{3}+\frac{2}{3(k-1)}$

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

References

[1] L. Xu and J. Bruck, "X-code: MDS array codes with optimal encoding," *IEEE Transactions on Information Theory*, vol. 45, no. 1, pp. 272–276, 1999.

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

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

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

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

[6] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: MDS array codes with optimal rebuilding," *Information Theory, IEEE Transactions on*, vol. 59, no. 3, pp. 1597–1616, 2013.

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

[8] S. Xu, R. Li, P. P. Lee, Y. Zhu, L. Xiang, Y. Xu, and J. Lui, "Single disk failure recovery for X-code-based parallel storage systems," *Computers, IEEE Transactions on*, vol. 63, no. 4, pp. 995–1007, 2014.

[9] Y. Wang, X. Yin, and X. Wang, "MDR codes: A new class of RAID-6 codes with optimal rebuilding and encoding," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 1008–1018, 2013.

[10] ——, "Two new classes of two-parity MDS array codes with optimal repair," *IEEE Communications Letters*, vol. 20, no. 7, pp. 1293–1296, 2016.

[11] L. Pamies-Juarez, F. Blagojevic, R. Mateescu, C. Gyuot, E. E. Gad, and Z. Bandic, "Opening the chrysalis: on the real repair performance of MSR codes," in *14th USENIX Conference on File and Storage Technologies (FAST 16)*, 2016, pp. 81–94.

[12] K. W. Shum, H. Hou, M. Chen, H. Xu, and H. Li, "Regenerating codes over a binary cyclic code," in *Proc. IEEE Int. Symp. Inf. Theory*, Honolulu, July 2014, pp. 1046–1050.

[13] H. Hou, K. W. Shum, M. Chen, and H. Li, "New MDS array code correcting multiple disk failures," in *Global Communications Conference*, 2015, pp. 2369–2374.

[14] ——, "BASIC codes: Low-complexity regenerating codes for distributed storage systems," *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 3053–3069, 2016.