STAR+ Codes: Triple-Fault-Tolerant Codes with Asymptotically Optimal Updates and Efficient Encoding/Decoding

Hanxu Hou[†§*], Patrick P. C. Lee[§]
[†] School of Electrical Engineering & Intelligentization, Dongguan University of Technology
[§] Department of Computer Science and Engineering, The Chinese University of Hong Kong

***Abstract*— STAR codes are well-known binary Maximum Distance Separable (MDS) array codes with triple fault tolerance and low encoding/decoding complexity, yet the update complexity of STAR codes is sub-optimal. We propose STAR+ codes, which extend STAR codes to achieve asymptotically optimal update complexity. We show that STAR+ codes are the generalized version of STAR codes with triple fault tolerance, and additionally have strictly less complexity in encoding, decoding, and updates than STAR codes for most parameters.**

I. Introduction

Binary maximum distance separable (MDS) array codes have been widely employed in storage systems, such as Redundant Arrays of Inexpensive Disks (RAID) [1], for fault tolerance with the two main advantages: (i) the storage redundancy is minimized subject to a given level of fault tolerance; and (ii) only exclusive-OR (XOR) operations are involved in encoding and decoding. Specifically, an (n,k,m) binary MDS array code is an array of size $m \times n$ that encodes km information bits to obtain $(n-k)m$ parity bits, where the nm bits (including km information bits and $(n-k)m$ parity bits) are stored in the $m \times n$ array. It satisfies the MDS property, meaning that the system can tolerate any $r = n-k$ out of n column failures with the minimum storage redundancy. To support storage applications with update-intensive workloads (e.g., databases), it is desirable to construct binary MDS array codes that have small *encoding/decoding complexity*, in terms of the number of XORs incurred in encoding/decoding, as well as small *update complexity*, in terms of the average number of parity bits affected by a change of a single information bit.

Binary MDS array codes have been well studied in the literature. For example, EVENODD [2], X-code [3], and RDP [4] are well-known binary MDS array codes that can tolerate any $r = 2$ column failures. In particular, EVENODD codes have a well-designed algebraic structure that motivates many follow-up studies [5]–[8]. Examples are STAR codes [5] and the generalized EVENODD codes [6], which extend the EVENODD code construction with three parity columns and more than two parity columns, respectively. Some other follow-up studies focus on the efficient decoding [7], [8] of the generalized EVENODD codes or the efficient repair of binary MDS array codes [9]–[11]. Although EVENODD codes have efficient encoding/decoding complexity, their update complexity is sub-optimal. It is shown in [12] that the minimum update complexities of systematic MDS array codes with $r = 2$ and $r = 3$ parity columns are $2 + \frac{1}{m}(1 - \frac{1}{k})$ and $3 + \frac{3}{m}(\frac{2}{3} - \frac{1}{k})$, respectively. Recently, EVENODD+ codes [13] have been proposed that can achieve the asymptotically minimum update complexity, but they only have $r = 2$ parity columns. TIP codes [14] have $r = 3$ parity columns and achieve the optimal update complexity, but have much higher decoding complexity than existing binary MDS array codes (e.g., STAR codes).

We propose STAR+ codes, a generalized construction for STAR codes with $r = 3$ parity columns. STAR+ codes have three properties: (i) MDS property; (ii) lower encoding/decoding complexity than STAR codes; (iii) asymptotically minimum update complexity (i.e., much lower update complexity than STAR codes). STAR+ codes build on the observation that STAR codes add an *adjuster bit* to each parity bit in the computation of the last two parity columns, thereby leading to high update complexity. In contrast, STAR+ codes add an adjuster bit to a subset of parity bits in the computation of the last two parity columns, so as to reduce the update complexity.

II. Construction of STAR+ Codes

We present the construction of STAR+ codes with $r = 3$ parity columns. Given an odd integer $m \geq k$ with $\gcd(m,\ell) = 1$ for $\ell = 1,2,\ldots,k-1$, we define an $(m-1) \times (k+3)$ array code as follows. For $j = 0,1,\ldots,k-1$, column j is called an *information column* that stores the information bits $b_{0,j}, b_{1,j}, \ldots, b_{m-2,j}$; for $j = k, k+1, k+2$, column j is called a *parity column* that stores the parity bits $b_{0,j}, b_{1,j}, \ldots, b_{m-2,j}$. In this paper, the subscripts are taken modulo m unless otherwise specified.

Given the $(m-1) \times k$ information array of bits $b_{i,j}$ for $i = 0,1,\ldots,m-2$ and $j = 0,1,\ldots,k-1$, we define an *imaginary* row of bits $b_{m-1,j} = 0$ for $j = 0,1,\ldots,k-1$. The bits in column k are:

$$b_{i,k} = \sum_{j=0}^{k-1} b_{i,j} \text{ for } 0 \leq i \leq m-2. \tag{1}$$

The bits in columns $k+1$ and $k+2$ are respectively:

$$b_{i,k+1} = \begin{cases} b_{m-1,k+1} + \sum_{j=0}^{k-1} b_{i-j,j} & \text{for } 0 \leq i \leq 2\lfloor \frac{k}{2} \rfloor - 1, \\ \sum_{j=0}^{k-1} b_{i-j,j} & \text{for } 2\lfloor \frac{k}{2} \rfloor \leq i \leq m-2, \end{cases} \tag{2}$$

This work was partially supported by the National Key R&D Program of China (No. 2020YFA0712300), the National Natural Science Foundation of China (No. 62071121), Research Grants Council of HKSAR (AoE/P-404/18) and Innovation and Technology Fund (ITS/315/18FX).
* Corresponding author.

TABLE I: STAR+$(9,3)$, with the adjuster bits $b_{8,4}=b_{7,1}+b_{6,2}$ and $b_{8,5}=b_{0,1}+b_{1,2}$.

0	1	2	3	4	5
$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,0}+b_{0,1}+b_{0,2}$	$b_{0,0}+b_{7,2}+b_{8,4}$	$b_{0,0}+b_{1,1}+b_{2,2}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,0}+b_{1,1}+b_{1,2}$	$b_{1,0}+b_{0,1}+b_{8,4}$	$b_{1,0}+b_{2,1}+b_{3,2}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,0}+b_{2,1}+b_{2,2}$	$b_{2,0}+b_{1,1}+b_{0,2}$	$b_{2,0}+b_{3,1}+b_{4,2}$
$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,0}+b_{3,1}+b_{3,2}$	$b_{3,0}+b_{2,1}+b_{1,2}$	$b_{3,0}+b_{4,1}+b_{5,2}$
$b_{4,0}$	$b_{4,1}$	$b_{4,2}$	$b_{4,0}+b_{4,1}+b_{4,2}$	$b_{4,0}+b_{3,1}+b_{2,2}$	$b_{4,0}+b_{5,1}+b_{6,2}$
$b_{5,0}$	$b_{5,1}$	$b_{5,2}$	$b_{5,0}+b_{5,1}+b_{5,2}$	$b_{5,0}+b_{4,1}+b_{3,2}$	$b_{5,0}+b_{6,1}+b_{7,2}$
$b_{6,0}$	$b_{6,1}$	$b_{6,2}$	$b_{6,0}+b_{6,1}+b_{6,2}$	$b_{6,0}+b_{5,1}+b_{4,2}$	$b_{6,0}+b_{7,1}+b_{8,5}$
$b_{7,0}$	$b_{7,1}$	$b_{7,2}$	$b_{7,0}+b_{7,1}+b_{7,2}$	$b_{7,0}+b_{6,1}+b_{5,2}$	$b_{7,0}+b_{0,2}+b_{8,5}$

and

$$b_{i,k+2}=\begin{cases} b_{m-1,k+2}+\sum_{j=0}^{k-1} b_{i+j,j} & \text{for } m-1-2\lfloor\frac{k}{2}\rfloor \le i \le m-2, \\ \sum_{j=0}^{k-1} b_{i+j,j} & \text{for } 0 \le i \le m-2-2\lfloor\frac{k}{2}\rfloor, \end{cases} \tag{3}$$

where the two *adjuster bits* are

$$b_{m-1,k+1}=\sum_{j=1}^{k-1} b_{m-1-j,j}, \text{ and } b_{m-1,k+2}=\sum_{j=1}^{k-1} b_{m-1+j,j}.$$

Let STAR+(m,k) denote the array defined from the above equations. As a special case, EVENODD+ codes [13] are an $(m-1)\times(k+2)$ array with two parity columns defined in Eq. (1) and Eq. (2).

In STAR+(m,k), we only add two adjuster bits $b_{m-1,k+1}$ and $b_{m-1,k+2}$ to $2\lfloor\frac{k}{2}\rfloor$ bits in column $k+1$ and column $k+2$, respectively, while STAR codes [5] add the two adjuster bits $b_{m-1,k+1}$ and $b_{m-1,k+2}$ to all parity bits in the second and third parity columns, respectively. This distinction makes the update complexity of STAR+(m,k) asymptotically optimal. Table I depicts an example of STAR+$(9,3)$, in which the two adjuster bits $b_{8,4}$ and $b_{8,5}$ are added to the first two parity bits in column 4 and the last two parity bits in column 5, respectively.

Note that the idea of achieving asymptotically minimum update complexity in our STAR+(m,k) is similar to that of EVENODD+ codes [13]. Here we generalize the method for more parity columns in STAR+(m,k). The key point is that we should not only find efficient decoding algorithms for all the erasure patterns, but also show that the update complexity is asymptotically optimal.

III. Decoding Algorithm of Three Failures

We present the decoding algorithm of STAR+(m,k) to recover the erasures of any three columns, thereby also justifying its MDS property. Theorem 1 shows a key property of STAR+(m,k) needed for the decoding algorithm.

Theorem 1. *For $\ell=1,2$, $b_{m-1,k+\ell}=\sum_{i=0}^{m-2}(b_{i,k}+b_{i,k+\ell})$.*

Proof. We first consider $\sum_{i=0}^{m-2}(b_{i,k}+b_{i,k+1})$. From Eq. (1) and Eq. (2), it can be expressed as

$$\begin{aligned}\sum_{i=0}^{m-2}(b_{i,k}+b_{i,k+1}) &= \sum_{i=0}^{m-2}\sum_{j=0}^{k-1} b_{i,j}+ \\ &\quad \sum_{j=0}^{k-1}\left(\sum_{i=0}^{2\lfloor\frac{k}{2}\rfloor-1}(b_{i-j,j}+b_{m-1,k+1})+\sum_{i=2\lfloor\frac{k}{2}\rfloor}^{m-2} b_{i-j,j}\right) \\ &= \sum_{i=0}^{m-2}\sum_{j=0}^{k-1} b_{i,j}+\sum_{j=0}^{k-1}\sum_{i=0}^{m-2} b_{i-j,j}.\end{aligned}$$

From the definition of $b_{m-1,k+1}$, as well as $b_{m-1,j}=0$ for $j=0,1,\ldots,k-1$, the above equation becomes

$$\sum_{i=0}^{m-1}\sum_{j=0}^{k-1} b_{i,j}+\sum_{i=0}^{m-1}\sum_{j=0}^{k-1} b_{i-j,j}+\sum_{j=0}^{k-1} b_{m-1-j,j}=b_{m-1,k+1}.$$

With the same argument, we can also obtain $b_{m-1,k+2}=\sum_{i=0}^{m-2}(b_{i,k}+b_{i,k+2})$. The proof is completed. □

Since the double-erasure decoding of STAR+(m,k) can be viewed as a special case of the triple-erasure decoding, we focus on the decoding algorithm for three erased columns. Suppose that three columns f,g,h are erased, where $0\le f<g<h\le k+2$. We want to present a decoding algorithm to reconstruct all the erased bits from the remaining k columns. The reconstruction can be divided into four cases based on different erasure patterns: (i) three parity erasures, i.e., $f=k, g=k+1, h=k+2$; (ii) two parity erasures, i.e., $0\le f\le k-1$ and $k\le g<h\le k+2$; (iii) one parity erasure, i.e., $0\le f<g\le k-1$ and $k\le h\le k+2$; and (iv) no parity erasure, i.e., $0\le f<g<h\le k-1$.

For three parity erasures, the decoding algorithm is the same as the definition shown in Eq. (1), Eq. (2), and Eq. (3).

For two parity erasures, we should consider three patterns: (i) $g=k+1,h=k+2$, (ii) $g=k,h=k+2$, and (iii) $g=k,h=k+1$. For the first two patterns, we can recover the failed information column f with the same method as in EVENODD+ codes [13]. The decoding procedure of $g=k,h=k+1$ is similar to the decoding method of $g=k,h=k+2$.

In the following, we consider the remaining two cases of erasure patterns.

A. Decoding Algorithm with One Parity Erasure

We consider three patterns: (i) $h=k+2$, (ii) $h=k+1$, and (iii) $h=k$. If the third parity column is erased (i.e., $h=k+2$), we recover the two erased information columns using the EVENODD+ decoding algorithm [13] and recover the third parity column by Eq. (3). If the second parity column is erased (i.e., $h=k+1$), the decoding algorithm is also similar to the EVENODD+ decoding algorithm [13].

We now consider the decoding algorithm when the first parity column is erased (i.e., $h=k$ and $0\le f<g\le k-1$). According to Theorem 1, we can directly obtain $b_{m-1,k+1}+b_{m-1,k+2}$ in the following lemma.

Lemma 2. $b_{m-1,k+1}+b_{m-1,k+2}=\sum_{i=0}^{m-2}(b_{i,k+1}+b_{i,k+2})$.

By subtracting all the information bits in $k-2$ surviving information columns from bits $b_{i,k+1}$ and $b_{i,k+2}$ for $i = 0,1,\ldots,m-2$, we obtain the following $2m-2$ *syndrome bits*

$$p_{i,1} = \begin{cases} b_{-f+i,f} + b_{-g+i,g} + (b_{-1-f,f} + b_{-1-g,g}) \\ \quad \text{for } i = 0,1,\ldots,2\lfloor \frac{k}{2} \rfloor - 1, \\ b_{-f+i,f} + b_{-g+i,g} \\ \quad \text{for } i = 2\lfloor \frac{k}{2} \rfloor, 2\lfloor \frac{k}{2} \rfloor + 1, \ldots, m-2, \end{cases} \tag{4}$$

$$p_{i,2} = \begin{cases} b_{f+i,f} + b_{g+i,g} \\ \quad \text{for } i = 0,1,\ldots,m-2-2\lfloor \frac{k}{2} \rfloor, \\ b_{f+i,f} + b_{g+i,g} + (b_{-1+f,f} + b_{-1+g,g}) \\ \quad \text{for } i = m-1-2\lfloor \frac{k}{2} \rfloor, \ldots, m-2. \end{cases} \tag{5}$$

According to Lemma 2, we can compute

$$(b_{-1-f,f} + b_{-1-g,g}) + (b_{-1+f,f} + b_{-1+g,g})$$

by summing all the bits in Eq. (4) and Eq. (5).

We first consider the case of $f > 0$. When $f > 0$, we have

$$p_{f-1,1} = b_{-g+f-1,g} + (b_{-1-f,f} + b_{-1-g,g}) \tag{6}$$

in Eq. (4) with $i = f-1$ (as $0 \le f-1 \le k-3$) and

$$p_{g-1,1} = b_{-f+g-1,f} + (b_{-1-f,f} + b_{-1-g,g}) \tag{7}$$

in Eq. (4) with $i = g-1$ (as $1 \le g-1 \le k-2$). Similarly, we have

$$p_{m-f-1,2} = b_{g-f-1,g} + (b_{-1+f,f} + b_{-1+g,g}) \tag{8}$$

in Eq. (5) with $i = m-f-1$ (as $m-k+1 \le m-f-1 \le m-2$) and

$$p_{m-g-1,2} = b_{f-g-1,f} + (b_{-1+f,f} + b_{-1+g,g}) \tag{9}$$

in Eq. (5) with $i = m-g-1$ (as $m-k \le m-g-1 \le m-3$).

We call the bits in Eq. (6), Eq. (7), Eq. (8), and Eq. (9) *starting bits*. Starting from the starting bit in Eq. (6), we can compute one information bit as follows. First, we compute the summation of the bits $p_{f-1,1}$ and $p_{-2g+f-1,2}$ to obtain

$$b_{-2g+2f-1,f} + (b_{-1-f,f} + b_{-1-g,g}) + \varepsilon(b_{-1+f,f} + b_{-1+g,g}),$$

where $\varepsilon \in \{0,1\}$. Then, we find the bit in Eq. (4) that contains $b_{-2g+2f-1,f}$ and compute the summation of the bit in Eq. (4) with $i = 3f-2g-1$ and the above bit to obtain

$$\eta(b_{-1-f,f} + b_{-1-g,g}) + \varepsilon(b_{-1+f,f} + b_{-1+g,g}) + b_{-3g+3f-1,g},$$

where $\eta \in \{0,1\}$. By repeating the above procedure for ℓ times, we can obtain

$$\begin{aligned} & p_{f-1,1} + p_{f-2g-1,2} + \cdots + p_{2\ell(f-g)-f-1,2} \\ = \; & \eta(b_{-1-f,f} + b_{-1-g,g}) + \varepsilon(b_{f-1,f} + b_{g-1,g}) \\ & + b_{2\ell(f-g)-1,f}, \text{ or} \end{aligned} \tag{10}$$

$$\begin{aligned} & p_{f-1,1} + p_{f-2g-1,2} + \cdots + p_{2\ell(f-g)+f-1,1} \\ = \; & \eta(b_{-1-f,f} + b_{-1-g,g}) + \varepsilon(b_{f-1,f} + b_{g-1,g}) \\ & + b_{(2\ell+1)(f-g)-1,g}, \end{aligned} \tag{11}$$

where ℓ is a non-negative integer. Recall that there does not exist the term $b_{-f-1,f} + b_{-g-1,g}$ and $b_{f-1,f} + b_{g-1,g}$ according to Eq. (4) and Eq. (5), respectively. If

$$\begin{aligned} & b_{2\ell(f-g)-1,f} = b_{-f-1,f} \text{ or} \\ & b_{2\ell(f-g)-1,f} = b_{f-1,f} \text{ or} \\ & b_{(2\ell+1)(f-g)-1,g} = b_{-g-1,g} \text{ or} \\ & b_{(2\ell+1)(f-g)-1,g} = b_{g-1,g}, \end{aligned}$$

then the above procedure is stopped. Similarly, we can obtain

$$\eta(b_{-1-f,f} + b_{-1-g,g}) + \varepsilon(b_{f-1,f} + b_{g-1,g}) + b_{2\ell(g-f)-1,g}, \text{ or} \tag{12}$$

$$\eta(b_{-1-f,f} + b_{-1-g,g}) + \varepsilon(b_{f-1,f} + b_{g-1,g}) + b_{(2\ell+1)(g-f)-1,f}, \tag{13}$$

where ℓ is a non-negative integer, by summing the chosen bits in Eq. (4) and Eq. (5) with starting bit $p_{g-1,1}$ until

$$\begin{aligned} & b_{(2\ell+1)(g-f)-1,f} = b_{-f-1,f} \text{ or} \\ & b_{(2\ell+1)(g-f)-1,f} = b_{f-1,f} \text{ or} \\ & b_{2\ell(g-f)-1,g} = b_{-g-1,g} \text{ or} \\ & b_{2\ell(g-f)-1,g} = b_{g-1,g}. \end{aligned}$$

For the starting bit $p_{-f-1,2}$ in Eq. (8), we can obtain

$$\eta(b_{-1-f,f} + b_{-1-g,g}) + \varepsilon(b_{f-1,f} + b_{g-1,g}) + b_{2\ell(g-f)-1,f}, \text{ or} \tag{14}$$

$$\eta(b_{-1-f,f} + b_{-1-g,g}) + \varepsilon(b_{f-1,f} + b_{g-1,g}) + b_{(2\ell+1)(g-f)-1,g}, \tag{15}$$

where ℓ is a non-negative integer,

$$\begin{aligned} & b_{2\ell(g-f)-1,f} = b_{-f-1,f} \text{ or} \\ & b_{2\ell(g-f)-1,f} = b_{f-1,f} \text{ or} \\ & b_{(2\ell+1)(g-f)-1,g} = b_{-g-1,g} \text{ or} \\ & b_{(2\ell+1)(g-f)-1,g} = b_{g-1,g}. \end{aligned}$$

For the starting bit $p_{-g-1,2}$ in Eq. (9), we can obtain

$$\eta(b_{-1-f,f} + b_{-1-g,g}) + \varepsilon(b_{f-1,f} + b_{g-1,g}) + b_{2\ell(f-g)-1,g}, \text{ or} \tag{16}$$

$$\eta(b_{-1-f,f} + b_{-1-g,g}) + \varepsilon(b_{f-1,f} + b_{g-1,g}) + b_{(2\ell+1)(f-g)-1,f}, \tag{17}$$

where ℓ is a non-negative integer,

$$\begin{aligned} & b_{(2\ell+1)(f-g)-1,f} = b_{-f-1,f} \text{ or} \\ & b_{(2\ell+1)(f-g)-1,f} = b_{f-1,f} \text{ or} \\ & b_{2\ell(f-g)-1,g} = b_{-g-1,g} \text{ or} \\ & b_{2\ell(f-g)-1,g} = b_{g-1,g}. \end{aligned}$$

Recall that there is no $b_{-f+i,f} + b_{-g+i,g}$ for $i = m-1$ in Eq. (4) and $b_{f+i,f} + b_{g+i,g}$ for $i = m-1$ in Eq. (5). We put the bit $b_{-f-1,f} + b_{-g-1,g}$ and $m-1$ bits in Eq. (4) into the first set and put the bit $b_{f-1,f} + b_{g-1,g}$ and $m-1$ bits in Eq. (5) into the second set. Given an integer t with $0 \le t \le m-1$, we can always find a bit in the first set that contains $b_{(-f+i) \bmod m,f} = b_{t,f}$ or

$b_{(-g+i) \bmod m,g} = b_{t,g}$, and a bit in the second set that contains $b_{(f+i) \bmod m,f} = b_{t,f}$ or $b_{(g+i) \bmod m,g} = b_{t,g}$. For each starting bit, we recursively choose one bit in Eq. (4) or Eq. (5) that can cancel out one bit until the existing bit is $b_{-f-1,f}$ or $b_{-g-1,g}$ or $b_{f-1,f}$ or $b_{g-1,g}$. Thus, all $2(m-1)$ syndrome bits in Eq. (4) and Eq. (5) are involved in computing a bit with starting bit in Eq. (6), Eq. (7), Eq. (8), and Eq. (9), and each bit in Eq. (4) and Eq. (5) is only involved once in computing the bit with starting bit in Eq. (6), Eq. (7), Eq. (8) and Eq. (9). We can divide $2(m-1)$ bits in Eq. (4) and Eq. (5) into four groups. The bits in each group are used to compute the bit starting with a starting bit. We denote the group associated with starting bits $p_{f-1,1}$, $p_{g-1,1}$, $p_{-f-1,2}$ and $p_{-g-1,2}$ by S_1, S_2, S_3, and S_4, respectively.

Lemma 3. *Let the number of bits in groups S_1, S_2, S_3 and S_4 be $|S_1|$, $|S_2|$, $|S_3|$ and $|S_4|$, respectively. We have $|S_1| = |S_3|$, and $|S_2| = |S_4|$.*

Proof. We first want to show that $|S_1| = |S_3|$. The summation in Eq. (10) ends with $2\ell(f-g)-1 = -f-1 \bmod m$ or $2\ell(f-g)-1 = f-1 \bmod m$, and the summation in Eq. (11) ends with $(2\ell+1)(f-g)-1 = -g-1 \bmod m$ or $(2\ell+1)(f-g)-1 = g-1 \bmod m$. If $2\ell_1(f-g)-1 = -f-1 \bmod m$, we have $\ell_1 = f \cdot (2g-2f)^{-1} \bmod m$. Otherwise, if $2\ell_2(f-g)-1 = f-1 \bmod m$, we have $\ell_2 = (m-f) \cdot (2g-2f)^{-1} \bmod m$. On the other hand, if $(2\ell_3+1)(f-g)-1 = -g-1 \bmod m$, we have $\ell_3 = f \cdot (2g-2f)^{-1} \bmod m$; if $(2\ell_4+1)(f-g)-1 = g-1 \bmod m$, we have $\ell_4 = (m+f-2g) \cdot (2g-2f)^{-1} \bmod m$. The number of syndrome bits in Eq. (10) and Eq. (11) is

$$2\min\{\ell_1,\ell_2\} = 2\min\{f \cdot (2g-2f)^{-1} \bmod m, (m-f) \cdot (2g-2f)^{-1} \bmod m\},$$

and

$$2\min\{\ell_3,\ell_4\}+1 = 2\min\{f(2g-2f)^{-1} \bmod m, (f-2g)(2g-2f)^{-1} \bmod m\}+1,$$

respectively. Thus, we have

$$|S_1| = \min\{2\ell_1, 2\ell_2, 2\ell_3+1, 2\ell_4+1\} = \min\{2f(2g-2f)^{-1} \bmod m, 2(m-f)(2g-2f)^{-1} \bmod m, 2(f-2g)(2g-2f)^{-1} \bmod m+1\}.$$

For the starting bit $p_{-f-1,2}$, we have the bit in Eq. (14) or Eq. (15). If the index in Eq. (14) is $-2\ell f+2\ell g-1 = f-1 \bmod m$ or $-2\ell f+2\ell g-1 = -f-1 \bmod m$, the summation in Eq. (14) is ended. Similarly, if the index in Eq. (15) is $2\ell g-2\ell f+g-f-1 = g-1 \bmod m$ or $2\ell g-2\ell f+g-f-1 = -g-1 \bmod m$, the summation in Eq. (14) is ended. If $-2\ell_1' f+2\ell_1' g-1 = f-1 \bmod m$, we have $\ell_1' = f \cdot (2g-2f)^{-1} \bmod m$. Otherwise, if $-2\ell_2' f+2\ell_2' g-1 = -f-1 \bmod m$, we have $\ell_2' = (m-f) \cdot (2g-2f)^{-1} \bmod m$. On the other hand, if $2\ell_3' g-2\ell_3' f+g-f-1 = g-1 \bmod m$, we have $\ell_3' = f \cdot (2g-2f)^{-1} \bmod m$; if $2\ell_4' g-2\ell_4' f+g-f-1 = -g-1 \bmod m$, we have $\ell_4' = (f-2g) \cdot (2g-2f)^{-1} \bmod m$. We have

$$|S_3| = \min\{2\ell_1', 2\ell_2', 2\ell_3'+1, 2\ell_4'+1\} = \min\{2f(2g-2f)^{-1} \bmod m, 2(m-f)(2g-2f)^{-1} \bmod m, 2(f-2g)(2g-2f)^{-1} \bmod m+1\} = |S_1|.$$

With the same argument for the two starting bits $p_{g-1,1}$ and $p_{-g-1,2}$, we can also show that $|S_2| = |S_4|$. □

By Lemma 3, we have $|S_1| = |S_3|$ and $|S_2| = |S_4|$. As there are $2(m-1)$ syndrome bits and each syndrome bit is in one group, we obtain that $|S_1|+|S_2| = |S_3|+|S_4| = m-1$.

From the starting bit $p_{f-1,1}$, we can obtain the bit

$$\eta(b_{-1-f,f}+b_{-1-g,g})+\varepsilon(b_{f-1,f}+b_{g-1,g})+b_{-1-f,f}, \quad \text{if } |S_1| = 2f(2g-2f)^{-1} \bmod m, \tag{18}$$

$$\eta(b_{-1-f,f}+b_{-1-g,g})+\varepsilon(b_{f-1,f}+b_{g-1,g})+b_{f-1,f}, \quad \text{if } |S_1| = 2(m-f)(2g-2f)^{-1} \bmod m. \tag{19}$$

$$\eta(b_{-1-f,f}+b_{-1-g,g})+\varepsilon(b_{f-1,f}+b_{g-1,g})+b_{g-1,g}, \quad \text{if } |S_1| = 2(f-2g)(2g-2f)^{-1} \bmod m+1, \tag{20}$$

and further obtain

$$\begin{cases} b_{m-1-f,f} & \text{if } (\eta,\varepsilon) \in \{(0,0),(1,1)\}, \\ & |S_1| = 2f \cdot (2g-2f)^{-1} \bmod m, \\ b_{m-1-g,g} & \text{if } (\eta,\varepsilon) \in \{(1,0),(0,1)\}, \\ & |S_1| = 2f \cdot (2g-2f)^{-1} \bmod m, \\ b_{f-1,f} & \text{if } (\eta,\varepsilon) \in \{(0,0),(1,1)\}, \\ & |S_1| = 2(m-f) \cdot (2g-2f)^{-1} \bmod m, \\ b_{g-1,g} & \text{if } (\eta,\varepsilon) \in \{(1,0),(0,1)\}, \\ & |S_1| = 2(m-f) \cdot (2g-2f)^{-1} \bmod m, \\ b_{g-1,g} & \text{if } (\eta,\varepsilon) \in \{(0,0),(1,1)\}, \\ & |S_1| = 2(f-2g)(2g-2f)^{-1} \bmod m+1, \\ b_{f-1,f} & \text{if } (\eta,\varepsilon) \in \{(1,0),(0,1)\}, \\ & |S_1| = 2(f-2g)(2g-2f)^{-1} \bmod m+1. \end{cases} \tag{21}$$

Similarly, from the starting bit $p_{-f-1,1}$, we can obtain

$$\begin{cases} b_{f-1,f} & \text{if } (\eta,\varepsilon) \in \{(0,0),(1,1)\}, \\ & |S_3| = 2f \cdot (2g-2f)^{-1} \bmod m, \\ b_{g-1,g} & \text{if } (\eta,\varepsilon) \in \{(1,0),(0,1)\}, \\ & |S_3| = 2f \cdot (2g-2f)^{-1} \bmod m, \\ b_{m-f-1,f} & \text{if } (\eta,\varepsilon) \in \{(0,0),(1,1)\}, \\ & |S_3| = 2(m-f) \cdot (2g-2f)^{-1} \bmod m, \\ b_{m-g-1,g} & \text{if } (\eta,\varepsilon) \in \{(1,0),(0,1)\}, \\ & |S_3| = 2(m-f) \cdot (2g-2f)^{-1} \bmod m, \\ b_{m-g-1,g} & \text{if } (\eta,\varepsilon) \in \{(0,0),(1,1)\}, \\ & |S_3| = 2(f-2g) \cdot (2g-2f)^{-1} \bmod m+1, \\ b_{m-f-1,f} & \text{if } (\eta,\varepsilon) \in \{(1,0),(0,1)\}, \\ & |S_3| = 2(f-2g) \cdot (2g-2f)^{-1} \bmod m+1. \end{cases} \tag{22}$$

By Lemma 3, we have $|S_1| = |S_3|$ and together with Eq. (21) and Eq. (22), we can always compute two different information bits from two starting bits $p_{f-1,1}$ and $p_{-f-1,2}$. Similarly, we can show that we can also obtain two different information bits from two starting bits $p_{g-1,1}$ and $p_{-g-1,2}$. Thus, we can obtain the four information bits $b_{g-1,g}, b_{f-1,f}, b_{-1-g,g}, b_{-1-f,f}$ by the

four starting bits and can further compute $b_{-1-f,f}+b_{-1-g,g}$ and $b_{f-1,f}+b_{g-1,g}$. The other information bits in columns f and g can be computed as in STAR codes [5, Section. 4.3.3].

When $f=0$, we have two starting bits $b_{g-1,0}+b_{-1-g,g}$ and $b_{-g-1,0}+b_{g-1,g}$ and only need to compute two bits $b_{-1-g,g}$ and $b_{g-1,g}$ by the same method of $f>0$ as shown above.

Consider the example in Table I. Suppose that columns $f=1$, $g=2$ and $h=3$ are failed. By subtracting the information bits in column 0 from the parity bits in columns 4 and 5, we can obtain the following bits

$$b_{7,2}+b_{8,4}, b_{0,1}+b_{8,4}, b_{1,1}+b_{0,2}, b_{2,1}+b_{1,2},$$
$$b_{3,1}+b_{2,2}, b_{4,1}+b_{3,2}, b_{5,1}+b_{4,2}, b_{6,1}+b_{5,2},$$
$$b_{1,1}+b_{2,2}, b_{2,1}+b_{3,2}, b_{3,1}+b_{4,2}, b_{4,1}+b_{5,2},$$
$$b_{5,1}+b_{6,2}, b_{6,1}+b_{7,2}, b_{7,1}+b_{8,5}, b_{0,2}+b_{8,5}.$$

By summing all the above bits, we obtain $b_{8,4}+b_{8,5}=b_{7,1}+b_{6,2}+b_{0,1}+b_{1,2}$. As $f>0$, we have four starting bits $b_{7,2}+b_{8,4}$, $b_{0,1}+b_{8,4}$, $b_{7,1}+b_{8,5}$ and $b_{0,2}+b_{8,5}$. Starting from the starting bit $b_{7,2}+b_{8,4}$, we can compute $b_{0,1}$ by

$$(b_{7,2}+b_{8,4})+(b_{6,1}+b_{7,2})+(b_{6,1}+b_{5,2})+(b_{4,1}+b_{5,2})+$$
$$(b_{4,1}+b_{3,2})+(b_{2,1}+b_{3,2})+(b_{2,1}+b_{1,2})+(b_{8,4}+b_{8,5})=b_{0,1},$$

and compute $b_{8,4}$ by $b_{0,1}+(b_{0,1}+b_{8,4})$. Once $b_{8,4}$ is known, we can compute $b_{8,5}$. All the other information bits can be decoded iteratively.

B. Decoding Algorithm without Parity Erasures

Consider that three information columns f, g and h are erased, where $0\le f<g<h\le k-1$, and we want to recover the information bits in columns f, g and h. One decoding algorithm can be summarized as follows.

Step 1. Calculate $b_{m-1,k+1}$ and $b_{m-1,k+2}$ by Theorem 1.

Step 2. Calculate the following $3p-1$ syndromes

$$b_{i,f}+b_{i,g}+b_{i,h} \text{ for } 0\le i\le p-2,$$
$$b_{i-f,f}+b_{i-g,g}+b_{i-h,h} \text{ for } 0\le i\le p-1,$$
$$b_{i+f,f}+b_{i+g,g}+b_{i+h,h} \text{ for } 0\le i\le p-1,$$

by subtracting the information bits in $k-3$ surviving information columns from $b_{m-1,k+1}$, $b_{m-1,k+2}$ and the $3(p-1)$ parity bits.

Step 3. Find a starting point in column g and recover the column g.

Step 4. Recover columns f and h by the decoding algorithm of EVENODD+ in [13].

We can recover the erased three information columns after executing the above four steps. The detailed decoding algorithm of recovering two information columns is described in [13]. The method of finding a starting point in column g is similar to that of RTP [15], which is omitted due to space limitation.

We note that in the proof of Lemma 2, the calculation of $b_{m-1,k+1}+b_{m-1,k+2}$ by summing all $2(m-1)$ parity bits in columns $k+1$ and $k+2$ is a key point in the decoding algorithm with one parity erasure. By Lemma 2, we can always obtain $b_{m-1,k+1}+b_{m-1,k+2}$ if the number of parity bits in column $k+1$ (resp. column $k+2$) that contains $b_{m-1,k+1}$ (resp. $b_{m-1,k+2}$) is an even number. This is one of the reasons we add $b_{m-1,k+1}$ to the first $2\lfloor\frac{k}{2}\rfloor$ parity bits in column $k+1$ and add $b_{m-1,k+2}$ to the last $2\lfloor\frac{k}{2}\rfloor$ parity bits in column $k+2$. However, the number of parity bits in column $k+1$ (resp. column $k+2$) that contain $b_{m-1,k+1}$ (resp. $b_{m-1,k+2}$) should be no less than $2\lfloor\frac{k}{2}\rfloor$, as we need to ensure that any $k-1$ information columns and column $k+1$ can recover all the information bits. In the decoding algorithm of any three information column erasures, the computation of $b_{m-1,k+1}$ and $b_{m-1,k+2}$ given in Theorem 1 is critical important, as the decoding algorithm can be reduced to that of RTP [15].

Note also that in STAR codes [5], the number of information columns should be a prime number and the number of rows is equal to the number of information columns minus one. However, we relax this constraint in STAR+(m,k). When $k=m$, STAR+(m,k) reduces to STAR codes.

TABLE II: Update complexities of STAR+$(m,7)$ and STAR codes.

m	STAR	STAR+(m,k)	m	STAR	STAR+(m,k)
7	4.4286	4.4286	31	4.6571	3.2857
11	4.5429	3.8571	37	4.6667	3.2381
13	4.4571	3.7143	41	4.6714	3.2143
17	4.6071	3.5357	43	4.6735	3.2041
19	4.6190	3.4762	47	4.6770	3.1863
23	4.6364	3.3896	49	n/a	3.1786
29	4.6531	3.3061	53	4.6813	3.1648

IV. Comparison

In this section, we evaluate the update complexity for STAR+(m,k).

If an information bit is changed, we need to update one parity bit in column k and $1+(2\lfloor\frac{k}{2}\rfloor-1)\frac{k-1}{k(m-1)}$ parity bits in each of column $k+1$ and column $k+2$ on average. Thus, the update complexity is $3+2(2\lfloor\frac{k}{2}\rfloor-1)\frac{k-1}{k(m-1)}$. If $m\gg k$, then the update complexity approaches the optimal value $3+\frac{2k-3}{k(m-1)}$ [12, Proposition 5.5]. Therefore, the update complexity is asymptotically optimal when m is much larger than k. The update complexity of STAR codes is $5-\frac{2(m+k-2)}{k(m-1)}$, which is strictly larger than that of STAR+(m,k) if $m>k$. Table II shows the update complexity of STAR codes and STAR+(m,k) when $k=7$ and m ranges from 7 to 53. We observe that STAR+(m,k) has less update complexity than STAR codes when $m>7$, and this advantage increases with m. As $m=49$ is not prime, we do not add the result for STAR codes in Table II.

Note that we only add the adjuster bit to some parity bits in the last two parity columns in our STAR+(m,k), while the adjuster bit is added to all the parity bits in the last two parity columns in STAR codes. STAR+(m,k) have lower encoding/decoding complexities compared to STAR codes.

V. Conclusion

In this paper, we present a new construction of STAR codes such that the update complexity of the newly constructed STAR codes is asymptotically optimal and the encoding/decoding complexity is slightly less than that of original STAR codes.

REFERENCES

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

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

[3] 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.

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

[5] 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.

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

[7] H. Jiang, M. Fan, Y. Xiao, X. Wang, and W. Yu, "Improved decoding algorithm for the generalized evenodd array code," in *Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on*, 2012.

[8] H. Hou, Y. S. Han, K. W. Shum, and H. Li, "A Unified Form of EVENODD and RDP Codes and Their Efficient Decoding," *IEEE Trans. Communications*, vol. 66, no. 11, pp. 5053–5066, 2018.

[9] H. Hou, P. P. C. Lee, Y. S. Han, and Y. Hu, "Triple-Fault-Tolerant Binary MDS Array Codes with Asymptotically Optimal Repair," in *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 839–843.

[10] H. Hou, Y. S. Han, P. P. C. Lee, Y. Hu, and H. Li, "A New Design of Binary MDS Array Codes with Asymptotically Weak-Optimal Repair," *IEEE Transactions on Information Theory*, vol. 65, no. 11, pp. 7095–7113, 2019.

[11] H. Hou and P. P. Lee, "Binary mds array codes with optimal repair," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1405–1422, 2020.

[12] M. Blaum and R. M. Roth, "On lowest density MDS codes," *IEEE Trans. on Information Theory*, vol. 45, no. 1, pp. 46–59, 1999.

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

[14] Y. Zhang, C. Wu, L. Jie, and M. Guo, "TIP-Code: A Three Independent Parity Code to Tolerate Triple Disk Failures with Optimal Update Complextiy," in *IEEE/IFIP International Conference on Dependable Systems & Networks*, 2015.

[15] A. Goel and P. Corbett, "RAID Triple Parity," *ACM Sigops Operating Systems Review*, vol. 46, no. 3, pp. 41–49, 2012.