XI-Code: A Family of Practical Lowest Density MDS Array Codes of Distance 4

Zhijie Huang, Hong Jiang, *Fellow, IEEE*, Ke Zhou, *Member, IEEE*, Chong Wang, and Yuhong Zhao

***Abstract*— Designing the lowest density maximum-distance separable (MDS) array codes has gained much attention in recent years due to the optimal redundancy and minimum update penalty (lowest density) of such codes. However, the existing lowest density MDS array codes of distance 4 have extremely strict constraints on the code length, which makes them impractical. In particular, most of them require the code length to be p (or $p-1$), where p is a prime that satisfies: i) $3|(p-1)$ and ii) 2 is primitive in GF(p). In this paper, we propose a new family of the lowest density MDS array codes of distance 4, called XI-Code. It has the properties of: 1) being capable of correcting both triple erasures and a single error combined with one erasure; 2) having code length of either p or $p+1$ with p being an odd prime number; 3) achieving optimality in encoding and update; and 4) achieving optimality in erasure decoding for certain erasure patterns and near optimality for other erasure patterns. It is worth mentioning that XI-Code is the first discovered family of the lowest density MDS array codes of distance 4 that supports the code length of p or $p+1$ with p being an odd prime.**

***Index Terms*— MDS codes, array codes, lowest density, storage systems.**

I. Introduction

ARRAY codes [1] have been studied extensively in the past decades due to their practical utility in communication and storage systems. The main advantage of these codes is that the encoding and decoding procedures use only simple XOR and cyclic shift operations, thus are more efficient than the well-known Reed–Solomon codes [2] in terms of computational complexity [3]. In array codes, a codeword is a two-dimensional array and the error model is that errors or erasures are columns of the array, namely, if one symbol of a column is an error or erasure, then the whole column is considered an error or erasure. This error model is very suitable for characterizing burst errors in communication networks, as well as disk (node) failures in storage systems.

A typical application scenario of array codes is the Redundant Arrays of Inexpensive Disks (RAID) architectures [4], which are deployed widely in modern computing systems as a storage building block. In particular, maximum-distance separable (MDS) array codes are especially popular, since they can provide a certain level of fault tolerance with the minimum redundancy. Many such codes have been specially designed for RAID-6, such as EVENODD [5], RDP [6], Liberation [7] and MDR codes [8], and they are shown to be much more efficient than the original scheme [9] that employed Reed-Solomon codes.

However, RAID-6 does not provide sufficient protection against data loss caused by either triple disk failures or a disk failure combined with unrecoverable sector errors in other disks. These two types of failures have become increasingly pervasive in modern storage systems due to the compounding impact of a dramatic increase in single disk capacity, a fairly constant per-bit error rate and a limited transfer rate. On the other hand, to satisfy the exponential growth of storage demand, datacenters and distributed storage systems are typically constructed from a large number of commodity servers and less reliable (but more economical) disks such as SATA (vs. SCSI), which makes disk (node) failures happen more frequently than ever before. Consequently, many MDS array codes that are capable of correcting triple erasures, such as generalized EVENODD [10], STAR [11] and generalized RDP [12] codes, have been proposed to enhance the data reliability in large scale storage systems.

Although the abovementioned codes can provide protection against data loss caused by triple disk (node) failures, all of them have a common drawback in their high *update complexity* when small changes are applied to the stored content. Update complexity is a key performance metric of particular concern by array codes designed for storage systems, which refers to the average number of coding symbols that must be updated (read-modify-write) whenever a data symbol is modified. It is quite clear that for triple-erasure-correcting codes, the lowest update complexity is 3. However, generalized EVENODD, STAR and generalized RDP codes all have a update complexity of about 5. Since the update complexity dictates the access time to the disk array (storage node), even in the absence of failures, it is the primary factor to consider when

Manuscript received April 28, 2015; revised December 26, 2015 and April 30, 2016; accepted May 5, 2016. Date of publication May 13, 2016; date of current version July 12, 2016. This work is supported in part by the National Natural Science Foundation of China under Grants 61232004 and 61502189, and by the U.S. National Science Foundation within the Division of Computer and Network Systems under Grants CNS-1016609 and CNS-1116606. The associate editor coordinating the review of this paper and approving it for publication was A. Thangaraj. *(Corresponding author: Ke Zhou.)*

Z. Huang is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, and also with GaoDig Information Technology, Ltd., Beijing 100000, China (e-mail: jayzy_huang@hust.edu.cn).

H. Jiang is with the Department of Computer Science and Engineering, The University of Texas at Arlington, Arlington, TX 76010 USA (e-mail: hong.jiang@uta.edu).

K. Zhou and C. Wang are with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: k.zhou@hust.edu.cn; c_wang@hust.edu.cn).

Y. Zhao is with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China (e-mail: zhaoyuhong@iie.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCOMM.2016.2568205

choosing codes for dynamic storage applications. Moreover, when array codes are employed in distributed storage systems, the update complexity will also directly translate into costly communication overhead.

In order to minimize the update complexity, an interesting subclass of array codes, called *lowest density* array codes, have gained much attention in recent years [13]–[20]. Lowest density array codes contain a minimal number of nonzero elements in their parity check matrices, so that they can achieve the lower bound of update complexity. Thus, they are particularly suitable for storage applications that need *frequent updates* of information. Unfortunately, most of the existing lowest density array codes are of distance 3, meaning that they can only correct double erasures. Although there are some exceptions where the codes can correct triple erasures in theory, their parameter constraints are so strict that they can be hardly applied to practical systems. For instance, cyclic lowest density codes [18] require the code length to be $p-1$, while T-Code [19] and the codes in [17] both require the code length to be p or $p-1$, where p is a prime number that satisfies: i) $3|(p-1)$ and ii) 2 is primitive in GF(p). In particular, the typical size of RAID usually ranges roughly from 5 to 30, but existing lowest density codes only support a certain subset of $\{6, 12, 13, 18, 19\}$ in this range. In addition to the strict parameter constraints, another shortcoming of thest codes is that they do not provide explicit decoding algorithms, meaning that it is difficult for storage practitioners to implement such codes.

In this paper, we present a new family of lowest density MDS array codes over GF(2), called XI-Code, which has a column distance of 4 and fewer constraints on the code length. In XI-Code, the parity sets in the codeword are along rows, diagonals and anti-diagonals. This regular geometric construction makes the decoding very efficient, for both triple erasures and a single error combined with one erasure. In general, our codes have the following attractive properties:

- Capable of correcting both triple erasures and a single error combined with one erasure
- Optimal encoding complexity.
- Lowest density, thus optimal update complexity.
- Supporting code length of p or $p+1$, where p is an odd prime.
- The decoding complexity is exactly optimal for some erasure patterns, and near optimal for other erasure patterns.

It is worth mentioning that both the encoding and decoding processes in XI-codes do not involve any matrix operations. Instead, we present iterative decoding algorithms for both triple erasures and a single error combined with one erasure.

The rest of this paper is organized as follows. In the next section we give a geometric presentation of the new codes and prove their MDS property. Then, we present the corresponding decoding procedures for both triple erasures and a single error combined with one erasure in Section III. In Section IV, we analyze the encoding and decoding complexities of XI-Code. Then in Section V we compare XI-Code with other practical MDS array codes of distance 4 in terms of decoding complexity. Finally, we summarize the contributions of this work and remark on the directions of our future work in Section VI.

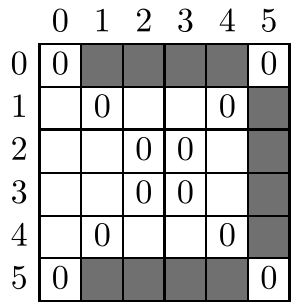

Fig. 1. The extended codeword of XI-code with $p=5$.

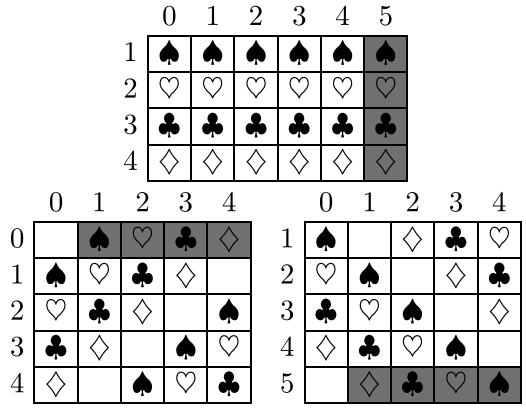

Fig. 2. The encoding rules of XI-code with $p=5$.

II. XI-Code Construction

In XI-Code, a codeword is a $(p-1)\times(p+1)$ array of binary bits, where p is an odd prime. The first column is a pure data column, while the last column is a pure parity column. Except for these two columns, each of the other columns contains $(p-3)$ data bits and two parity bits. Parity bits are constructed from the data bits along several diagonals of certain slopes with the exclusive-or (XOR) operation. Since the first column does not contain any parity bits, XI-Code can be shortened by assuming that the first column is an imaginary column that holds nothing but zeros. In other words, the code length of XI-Code can be $p+1$ or p, where p is an odd prime.

A. Encoding Procedure

To facilitate the following description, we use $b_{i,j}$ to denote the ith bit in the jth column, and let $\langle x\rangle = x \bmod p$. In addition, in order to make the encoding rules more intelligible, we assume that the codeword is a $(p+1)\times(p+1)$ array with every column containing two extra imaginary 0-bits. Take the XI-code with $p=5$ for instance, Figure 1 shows the distribution of data bits, parity bits and imaginary 0-bits in the codeword, while Figure 2 shows the encoding rules.

Generally speaking, there are three types of parities, which are constructed from the data bits along rows, diagonals (slope 1) and anti-diagonals (slope -1) respectively. Specifically, the formal encoding rules of the XI-Code are

as follows:

$$b_{i,p} = \bigoplus_{t=0}^{p-1} b_{i,t}, \quad 1 \le i \le p-1 \tag{1}$$

$$b_{0,j} = \bigoplus_{t=1}^{p-1} b_{t,\langle j-t\rangle}, \quad 1 \le j \le p-1 \tag{2}$$

$$b_{p,j} = \bigoplus_{t=1}^{p-1} b_{t,\langle j+t\rangle}, \quad 1 \le j \le p-1 \tag{3}$$

From the construction of the XI-Code, it is explicit that all the parity bits are obtained independently, i.e., every parity bit is constructed from a certain collection of data bits without involving any other parity bits. Moreover, each data bit is involved in calculating exactly *three* parity bits, which implies that modifying one data bit results in updating exactly *three* parity bits. Specifically, if $b_{i,j}$ is changed, then the corresponding parity bits $b_{i,p}$, $b_{0,\langle i+j\rangle}$ and $b_{p,\langle j-i\rangle}$ should be updated. In other words, *XI-Code achieves the lower bound* 3 *of the update complexity for any triple-erasure-correcting codes*. In addition, notice that the imaginary 0-bits do not need to really participate in calculating the parities, thus each parity bit is constructed from only $p-2$ data bits in practice. In other words, XI-Code requires only $p-3$ XORs per parity bit in encoding.

B. The MDS Property

It is easy to see that there are $3(p-1)$ parity bits in every codeword of the XI-Code. In this subsection we will demonstrate that the XI-Code can recover from any triple column erasures, i.e., it is MDS. To prove the MDS property of the XI-Code, we first provide several Lemmas that will be used in the proof.

Proposition 1: For any prime number p and integer $0 < \delta < p$, all numbers $1, 2, \cdots, p-1$ occur exactly once in the sequence $\delta, \langle 2\delta\rangle, \cdots, \langle (p-1)\delta\rangle$. Moreover, for any two integers $1 \le x, y \le p-1$, $x + y = p$ iff $\langle x\delta\rangle + \langle y\delta\rangle = p$.

Lemma 1: A valid codeword of XI-code cannot have exactly two nonzero columns.

Proof: Suppose that there is a valid codeword that contains exactly two nonzero columns — the lth and rth columns, where $0 \le l < r \le p$.

First, if $r = p$ and $0 \le l \le p-1$, then according to (2) and (3), it is easy to deduce that every bit in the lth column is 0, which contradicts the assumption.

Next, let us consider the case $0 \le l < r \le p-1$. Let $\delta = r - l$, obviously $1 \le \delta \le p-1$. Then, according to Proposition 1, $b_{0,j}, b_{1,j}, \cdots, b_{p,j}$ can be rearranged as $b_{0,j}, b_{\delta,j}, b_{\langle 2\delta\rangle,j}, \cdots, b_{\langle (p-1)\delta\rangle,j}, b_{p,j}$ where $j = l, r$. And these two sequences can be further rearranged as

$$b_{p,l}, b_{\delta,r}, b_{\langle 2\delta\rangle,l}, b_{\langle 3\delta\rangle,r}, \cdots, b_{\langle (p-1)\delta\rangle,l}, b_{p,r} \quad (\text{Seq}.A)$$

$$b_{0,r}, b_{\delta,l}, b_{\langle 2\delta\rangle,r}, b_{\langle 3\delta\rangle,l}, \cdots, b_{\langle (p-1)\delta\rangle,r}, b_{0,l} \quad (\text{Seq}.B)$$

Now we demonstrate that Seq. A and Seq. B each contains two imaginary 0s. The lth column contains two imaginary 0-bits $b_{l,l}$ and $b_{p-l,l}$, and the rth column contains two imaginary 0-bits $b_{r,r}$ and $b_{p-r,r}$. According to Proposition 1, there must be x and y such that $\langle x\delta\rangle = l$ and $\langle y\delta\rangle = p-l$, clearly $x + y = p$. Then, notice that $l + \delta = r$ and $p-l-\delta = p-r$, we have $\langle (x+1)\delta\rangle = r$ and $\langle (y-1)\delta\rangle = p-r$. If x is odd, then both $x+1$ and y are necessarily even, hence $y-1$ is odd. In this case, the two imaginary 0-bits: $b_{\langle y\delta\rangle,l}$ and $b_{\langle (y-1)\delta\rangle,r}$ fall into Seq. A, while $b_{\langle x\delta\rangle,l}$ and $b_{\langle (x+1)\delta\rangle,r}$ fall into Seq. B. If x is even, then both y and $x+1$ are necessarily odd, hence $y-1$ is even. In this case, $b_{\langle y\delta\rangle,l}$ and $b_{\langle (y-1)\delta\rangle,r}$ fall into Seq. B, while $b_{\langle x\delta\rangle,l}$ and $b_{\langle (x+1)\delta\rangle,r}$ fall into Seq. A.

From the above, the four imaginary 0-bits divide Seq. A and Seq. B into four subsequences. Every subsequence has the following two important properties: (a) one of the endpoints is an imaginary 0-bit while the other is a parity bit, and (b) any two adjacent elements either belong to the same diagonal parity set or belong to the same anti-diagonal parity set. From these two properties and the encoding rules (2) and (3), we can easily deduce that *all the elements of* the four subsequences are zeros, i.e., the lth and rth columns are zero columns, which contradicts the assumption. □

Lemma 2: For $x_i, y_i \in \text{GF}(2)$, $i = 1, 2, \cdots, p-1$, if both $x_{\langle (p-d_1)/2\rangle}$ and $y_{\langle (p-d_2)/2\rangle}$ are known, then other $x_i (i \ne p-d_1)$ and $y_i (i \ne p-d_2)$ can be obtained from

$$x_i \oplus y_i = a_i, \quad i \ne p-d_1, p-d_2 \tag{4}$$

$$x_{\langle i-d_1\rangle} \oplus y_{\langle i-d_2\rangle} = c_i, \quad i \ne d_1, d_2 \tag{5}$$

where $a_i, c_i \in \text{GF}(2)$ and $0 < d_1, d_2, d_2 - d_1 < p$.

Proof: Let $\delta = d_2 - d_1$, then (5) is equivalent to

$$y_i \oplus x_{\langle i+\delta\rangle} = c_{\langle i+d_2\rangle}, \quad i \ne p-d_2, p-\delta \tag{6}$$

where $1 \le i \le p-1$. According to Proposition 1, there must be $2 \le u \le p-1$ such that $\langle u\delta\rangle = p-d_1$ and $\langle (u-1)\delta\rangle = p-d_2$. Then, $x_1, x_2, \cdots, x_{p-1}$ and $y_1, y_2, \cdots, y_{p-1}$ can be rearranged respectively as follows

$$x_\delta, x_{\langle 2\delta\rangle}, \cdots, x_{\langle (u-1)\delta\rangle}, x_{\langle u\delta\rangle}, \cdots, x_{\langle (p-1)\delta\rangle} \quad (\text{Seq}.1)$$

$$y_\delta, y_{\langle 2\delta\rangle}, \cdots, y_{\langle (u-1)\delta\rangle}, y_{\langle u\delta\rangle}, \cdots, y_{\langle (p-1)\delta\rangle} \quad (\text{Seq}.2)$$

All the elements in Seq. 1 and Seq. 2, *except* $x_{\langle u\delta\rangle}$ and $y_{\langle (u-1)\delta\rangle}$, can form the following two sequences

$$x_\delta, y_\delta, x_{\langle 2\delta\rangle}, \cdots, y_{\langle (u-2)\delta\rangle}, x_{\langle (u-1)\delta\rangle} \quad (\text{Seq}.3)$$

$$y_{\langle u\delta\rangle}, x_{\langle (u+1)\delta\rangle}, y_{\langle (u+1)\delta\rangle}, \cdots, x_{\langle (p-1)\delta\rangle}, y_{\langle (p-1)\delta\rangle} \quad (\text{Seq}.4)$$

Clearly, if *any* element of Seq. 3 (Seq. 4) is known, then we can iteratively evaluate all the other elements of the sequence by using (4) and (6) alternately.

Next, we demonstrate that $x_{\langle (p-d_1)/2\rangle}$ and $y_{\langle (p-d_2)/2\rangle}$ necessarily fall into different sequences. From $\langle u\delta\rangle = p-d_1$ and $\langle (u-1)\delta\rangle = p-d_2$, we have $\left\langle \frac{u}{2}\delta\right\rangle = \langle (p-d_1)/2\rangle$ and $\left\langle \frac{u-1}{2}\delta\right\rangle = \langle (p-d_2)/2\rangle$. If u is even, then $u-1$ is odd and hence $u-1+p$ is even. In this case, we have $\left\langle \frac{u-1+p}{2}\delta\right\rangle = \langle (p-d_2)/2\rangle$. Observe that $1 \le \frac{u}{2} < u$ and $u \le \frac{u-1+p}{2} < p$, thus $x_{\langle (p-d_1)/2\rangle}$ appears in Seq. 3 while $y_{\langle (p-d_2)/2\rangle}$ appears in Seq. 4. If u is odd, then $u-1$ and $u+p$ are both even. In this case, we have $\left\langle \frac{u+p}{2}\delta\right\rangle = \langle (p-d_1)/2\rangle$. Observe that $1 \le \frac{u-1}{2} < u-1$ and $u < \frac{u+p}{2} < p$, thus $x_{\langle (p-d_1)/2\rangle}$ appears in Seq. 4 while $y_{\langle (p-d_2)/2\rangle}$ appears in Seq. 3. □

Lemma 3: For $x_1, x_2, \cdots, x_{p-1} \in$ GF(2), if both x_u and x_{p-u} are known, then other x_i can be obtained from

$$x_i \oplus x_{\langle i+\delta \rangle} = a_i, \quad i \neq p-\delta, \langle p-\delta/2 \rangle \tag{7}$$

where $a_i \in$ GF(2), $i = 1, 2, \cdots, p-1$ and $0 < \delta, u < p$.

Proof: According to Proposition 1, $x_1, x_2, \cdots, x_{p-1}$ can be rearranged as $x_\delta, x_{\langle 2\delta \rangle}, \cdots, x_{\langle (p-1)\delta \rangle}$(Seq. C). Observe that $\langle (p-1)\delta \rangle = p-\delta$ and $\left\langle \frac{p-1}{2}\delta \right\rangle = \langle p-\delta/2 \rangle$, thus (7) is equivalent to

$$x_{\langle i\delta \rangle} \oplus x_{\langle (i+1)\delta \rangle} = a_{\langle i\delta \rangle}, \quad i \neq \frac{p-1}{2} \tag{8}$$

where $i = 1, 2, \cdots, p-2$. Seq. C can be divided into two subsequences: $x_\delta, x_{\langle 2\delta \rangle}, \cdots, x_{\langle (p-1)\delta/2 \rangle}$ and $x_{\langle (p+1)\delta/2 \rangle}, \cdots, x_{\langle (p-2)\delta \rangle}, x_{\langle (p-1)\delta \rangle}$. Clearly, if each subsequence contains a known element, then other x_i can be iteratively evaluated using (8). According to Proposition 1, there must be $1 \leq n \leq p-1$ such that $\langle n\delta \rangle = u$ and $\langle (p-n)\delta \rangle = p-u$. Obviously, $x_{\langle n\delta \rangle}$ and $x_{\langle (p-n)\delta \rangle}$ necessarily fall into different subsequences. From the above, if both x_u and x_{p-u} are known, then other $x_i (1 \leq i \leq p-1)$ can be obtained from (7). □

Lemma 4: A valid codeword of the XI-code cannot have exactly three nonzero columns.

Proof: Suppose that there is a valid codeword that contains exactly three nonzero columns — the lth, mth and rth columns, where $0 \leq l < m < r \leq p$.

First, if $r = p$, then $0 \leq l < m \leq p-1$. According to Lemma 1, we can deduce that every bit in the lth and mth columns is 0, which contradicts the assumption.

Next, let us consider the case $0 \leq l < m < r \leq p-1$. In this case, let $d_1 = m-l$, $d_2 = r-l$, and $\delta = d_2 - d_1 = r-m$, then according to the parity constraints of XI-code, we have

$$b_{i,l} \oplus b_{i,m} \oplus b_{i,r} = 0 \tag{9}$$

$$b_{i,l} \oplus b_{\langle i-d_1 \rangle, m} \oplus b_{\langle i-d_2 \rangle, r} = 0, \quad i \neq d_1, d_2 \tag{10}$$

$$b_{i,l} \oplus b_{\langle i+d_1 \rangle, m} \oplus b_{\langle i+d_2 \rangle, r} = 0, \quad i \neq p-d_1, p-d_2 \tag{11}$$

where $1 \leq i \leq p-1$. Adding (9) to (10) and (11) respectively, we get

$$M_{\langle i-d_1 \rangle} \oplus R_{\langle i-d_2 \rangle} = 0, \quad i \neq d_1, d_2 \tag{12}$$

$$M_i \oplus R_i = 0, \quad i \neq p-d_1, p-d_2 \tag{13}$$

where $M_i = b_{i,m} \oplus b_{\langle i+d_1 \rangle, m}$ and $R_i = b_{i,r} \oplus b_{\langle i+d_2 \rangle, r}$, $1 \leq i \leq p-1$.

Let $\xi_i = M_i \oplus M_{\langle p-d_1-i \rangle}$ and $\eta_i = R_i \oplus R_{\langle p-d_2-i \rangle}$, then from (12) and (13) we have

$$\xi_i \oplus \eta_i = 0, \quad i \neq p-d_1, p-d_2 \tag{14}$$

$$\xi_{\langle i-d_1 \rangle} \oplus \eta_{\langle i-d_2 \rangle} = 0, \quad i \neq d_1, d_2 \tag{15}$$

Observe that $\xi_{\langle (p-d_1)/2 \rangle} = \eta_{\langle (p-d_2)/2 \rangle} = 0$, thus according to Lemma 2, from (14) and (15) we can easily deduce that $\xi_i = 0 (1 \leq i \leq p-1 \text{ and } i \neq p-d_1)$ and $\eta_i = 0 (1 \leq i \leq p-1 \text{ and } i \neq p-d_2)$.

Let $\mathcal{X}_i = b_{i,m} \oplus b_{p-i,m}$ and $\mathcal{Y}_i = b_{i,r} \oplus b_{p-i,r}$, then we have $\mathcal{X}_i \oplus \mathcal{X}_{\langle i+d_1 \rangle} = b_{i,m} \oplus b_{p-i,m} \oplus b_{\langle i+d_1 \rangle, m} \oplus b_{p-\langle i+d_1 \rangle, m} = M_i \oplus M_{\langle p-d_1-i \rangle} = \xi_i$ and $\mathcal{Y}_i \oplus \mathcal{Y}_{\langle i+d_2 \rangle} = b_{i,r} \oplus b_{p-i,r} \oplus b_{\langle i+d_2 \rangle, r} \oplus b_{p-\langle i+d_2 \rangle, r} = R_i \oplus R_{\langle p-d_2-i \rangle} = \eta_i$. Therefore, the following equations hold

$$\mathcal{X}_i \oplus \mathcal{X}_{\langle i+d_1 \rangle} = 0, \quad i \neq p-d_1 \tag{16}$$

$$\mathcal{Y}_i \oplus \mathcal{Y}_{\langle i+d_2 \rangle} = 0, \quad i \neq p-d_2 \tag{17}$$

where $1 \leq i \leq p-1$. Recall that $b_{m,m}$, $b_{p-m,m}$, $b_{r,r}$ and $b_{p-r,r}$ are imaginary 0-bits, thus $\mathcal{X}_m = \mathcal{X}_{p-m} = \mathcal{Y}_r = \mathcal{Y}_{p-r} = 0$. Then, according to Lemma 3, from (16) and (17) we can easily deduce that $\mathcal{X}_i = \mathcal{Y}_i = 0 (1 \leq i \leq p-1)$.

According to the definitions of M_i, R_i, $\mathcal{X}_i$ and $\mathcal{Y}_i$, we have $M_{\langle (p-d_1)/2 \rangle} = \mathcal{X}_{\langle (p-d_1)/2 \rangle} = 0$ and $R_{\langle (p-d_2)/2 \rangle} = \mathcal{Y}_{\langle (p-d_2)/2 \rangle} = 0$. Once $M_{\langle (p-d_1)/2 \rangle}$ and $R_{\langle (p-d_2)/2 \rangle}$ are determined, according to Lemma 2 we can easily deduce from (12) and (13) that $M_i = 0 (1 \leq i \leq p-1 \text{ and } i \neq p-d_1)$ and $R_i = 0 (1 \leq i \leq p-1 \text{ and } i \neq p-d_2)$. According to the definitions of M_i and R_i, we have

$$b_{i,m} \oplus b_{\langle i+d_1 \rangle, m} = 0, \quad i \neq p-d_1 \tag{18}$$

$$b_{i,r} \oplus b_{\langle i+d_2 \rangle, r} = 0, \quad i \neq p-d_2 \tag{19}$$

where $1 \leq i \leq p-1$. Recall that $b_{m,m}$, $b_{p-m,m}$, $b_{r,r}$ and $b_{p-r,r}$ are imaginary 0-bits, thus according to Lemma 3, from (18) and (19) we can get $b_{i,m} = b_{i,r} = 0 (1 \leq i \leq p-1)$.

Finally, from (9) we have $b_{i,l} = b_{i,m} \oplus b_{i,r} = 0 (1 \leq i \leq p-1)$, then according to the encoding rules we can get $b_{0,j} = b_{p,j} = 0 (j = l, m, r)$. From the above, we have $b_{i,j} = 0 (0 \leq i \leq p, j = l, m, r)$, which contradicts the assumption. □

Theorem 1: For any odd prime p, the XI-code has column distance of 4, i.e., it is MDS.

Proof: Observe that XI-code is a linear code, thus its column distance is equal to its minimum column weight. Then, we only need to prove that the code has a minimum column weight of 4, i.e., a nonzero codeword of XI-code has at least four nonzero columns.

First, from the construction of XI-code, parity bits are obtained along diagonals of slope 0, 1, or -1, so a valid codeword is clearly impossible to have exactly one nonzero column. Next, according to Lemma 1 and Lemma 4 we can further deduce that the minimum column weight of XI-code is at least 4. On the other hand, it is easy to see there is a codeword of column weight 4. For instance, take the all-zero codeword and set data bit $b_{i,j}$ to be 1, then the corresponding parity bits $b_{i,p}$, $b_{0,\langle i+j \rangle}$ and $b_{p,\langle j-i \rangle}$ should be updated to 1 accordingly. The new codeword has 4 nonzero columns, i.e., the jth, pth, $\langle i+j \rangle$th and $\langle j-i \rangle$th columns.

From the above, the column distance of XI-Code is 4. □

C. Algebraic Presentation of XI-Code

Let R_p denote the quotient ring $\mathbb{F}_2[x]/(x^p-1)$. An arbitrary polynomial $g \in R_p$ can be written in the form

$$g(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{p-1} x^{p-1} \tag{20}$$

where $a_i \in$ GF(2). In R_p, addition is the usual term-wise addition, while multiplication is performed modulo $x^p - 1$. Thus, for any integer t, $x^t g(x)$ refers to the polynomial formed by cyclicly shifting the coefficients of $g(x)$ to the right by t steps.

Now let us consider the extended $(p+1)\times(p+1)$ array defined in Section 2.1. For $j=0,1,\cdots,p-1$, the $p-1$ data bits (including imaginary 0-bits) stored in the jth column can be represented by a polynomial in R_p, i.e.,

$$c_j(x) = b_{1,j}x + b_{2,j}x^2 + \cdots + b_{p-1,j}x^{p-1} \tag{21}$$

Then, XI-Code can also be defined by the following parity-check equations:

$$\sum_{j=0}^{p-1} c_j(x) = c_p(x) \tag{22}$$

$$\sum_{j=0}^{p-1} x^j(b_{0,j} + c_j(x)) = 0 \tag{23}$$

$$\sum_{j=0}^{p-1} x^{p-j}(b_{p,j} + c_j(x)) = 0 \tag{24}$$

where $c_p(x)$ corresponds to the row parity column, while $b_{0,j}$ and $b_{p,j}$ correspond to the diagonal and anti-diagonal parity bits respectively.

III. Efficient Decoding Algorithms

Since XI-code has a column distance of 4, any codeword with three column erasures or one column erasure combined with one column error can be corrected. In this section, we present efficient decoding algorithms for these two error patterns.

A. Correcting Three Erasures

Suppose that the lth, mth and rth columns of a codeword are erased, where $0 \le l < m < r \le p$. We use $S_i^{(0)}$, $S_i^{(1)}$ and $S_i^{(-1)}$ to denote the ith row, diagonal and anti-diagonal syndromes respectively, where $1 \le i \le p-1$. According to (1), (2) and (3), the syndromes can be calculated as follows:

$$S_i^{(0)} = \bigoplus_{\substack{t=0\\ t\neq l,m,r}}^{p} b_{i,t} \tag{25}$$

$$S_i^{(1)} = \bigoplus_{\substack{t=0\\ t\neq l,m,r}}^{p-1} b_{\langle i-t\rangle,t} \tag{26}$$

$$S_i^{(-1)} = \bigoplus_{\substack{t=1\\ t\neq \langle l+i\rangle,\\ \langle m+i\rangle,\langle r+i\rangle}}^{p} b_{t,\langle t-i\rangle} \tag{27}$$

To facilitate the following discussion, we let $d_1 = m-l$, $d_2 = r-l$, and $\delta = d_2 - d_1 = r-m$. Then a decoding algorithm for correcting any three erasures can be described as follows:

Algorithm 1 (All-Erasure Decoding): First, we compute the row, diagonal and anti-diagonal syndromes according to (25)-(27). Then we distinguish between the following two cases:

Case 1: The row parity column is erased, i.e., $r = p$.

In this case, from (1)–(3) and (25)–(27) we have

$$b_{i,l} \oplus b_{i,m} \oplus b_{i,r} = S_i^{(0)}, \quad i \neq 0, p \tag{28}$$

$$b_{i,l} \oplus b_{\langle i-d_1\rangle,m} = S_{\langle i+l\rangle}^{(1)}, \quad i \neq p \tag{29}$$

$$b_{i,l} \oplus b_{\langle i+d_1\rangle,m} = S_{\langle i-l\rangle}^{(-1)}, \quad i \neq 0, p-d_1 \tag{30}$$

$$b_{p-d_1,l} \oplus b_{p,m} = S_{\langle p-d_1-l\rangle}^{(-1)}$$

where $0 \le i \le p$. According to the proof of Lemma 1, we can easily reconstruct the lth and mth columns using (29) and (30). Then, the row parity bits can be calculated as $b_{i,p} = b_{i,l} \oplus b_{i,m} \oplus S_i^{(0)}$, where $1 \le i \le p-1$. Instead of writing the formalized decoding procedure, we will illustrate it with an example later.

Case 2: The row parity column is available, i.e., $0 \le l < m < r \le p-1$.

In this case, from (1)˜(3) and (25)˜(27) we have

$$b_{i,l} \oplus b_{\langle i-d_1\rangle,m} \oplus b_{\langle i-d_2\rangle,r} = S_{\langle i+l\rangle}^{(1)}, \quad i \neq d_1, d_2 \tag{31}$$

$$b_{i,l} \oplus b_{\langle i+d_1\rangle,m} \oplus b_{\langle i+d_2\rangle,r} = S_{\langle i-l\rangle}^{(-1)}, \quad i \neq p-d_1, p-d_2 \tag{32}$$

where $1 \le i \le p-1$. Then, adding (28) to (31) and (32) respectively, we get

$$M_{\langle i-d_1\rangle} \oplus R_{\langle i-d_2\rangle} = S_i^{(0)} \oplus S_{\langle i+l\rangle}^{(1)}, \quad i \neq d_1, d_2 \tag{33}$$

$$M_i \oplus R_i = S_i^{(0)} \oplus S_{\langle i-l\rangle}^{(-1)}, \quad i \neq p-d_1, p-d_2 \tag{34}$$

where $M_i = b_{i,m} \oplus b_{\langle i+d_1\rangle,m}$ and $R_i = b_{i,r} \oplus b_{\langle i+d_2\rangle,r}$. From (33) and (34) we can retrieve all the data bits in the mth and rth columns through the following steps:

Step 1: For $1 \le i \le p-1$ and $i \neq d_1, d_2$, let $\alpha_i \leftarrow S_i^{(0)} \oplus S_{\langle i+l\rangle}^{(1)}$. For $1 \le i \le p-1$ and $i \neq p-d_1, p-d_2$, let $\beta_i \leftarrow S_i^{(0)} \oplus S_{\langle i-l\rangle}^{(-1)}$.

Step 2: Let $\xi_i = M_i \oplus M_{\langle p-d_1-i\rangle}$ and $\eta_i = R_i \oplus R_{\langle p-d_2-i\rangle}$, then from (33), (34) and Step 1, we have

$$\xi_i \oplus \eta_i = \beta_i \oplus \alpha_{p-i}, \quad i \neq p-d_1, p-d_2 \tag{35}$$

$$\xi_{\langle i-d_1\rangle} \oplus \eta_{\langle i-d_2\rangle} = \alpha_i \oplus \beta_{p-i}, \quad i \neq d_1, d_2 \tag{36}$$

Observe that $\xi_{\langle (p-d_1)/2\rangle} = \eta_{\langle (p-d_2)/2\rangle} = 0$, thus according to Lemma 2, from (35) and (36) we can easily evaluate $\xi_i (i \neq p-d_1)$ and $\eta_i (i \neq p-d_2)$, where $1 \le i \le p-1$.

Step 3: Evaluate $M_{\langle (p-d_1)/2\rangle}$ and $R_{\langle (p-d_2)/2\rangle}$.

Let $\mathcal{X}_i = b_{i,m} \oplus b_{p-i,m}$ and $\mathcal{Y}_i = b_{i,r} \oplus b_{p-i,r}$, then according to the proof of Lemma 4 and Lemma 3, we have

$$\mathcal{X}_{\langle id_1\rangle} \oplus \mathcal{X}_{\langle (i+1)d_1\rangle} = \xi_{\langle id_1\rangle}$$

$$\mathcal{Y}_{\langle id_2\rangle} \oplus \mathcal{Y}_{\langle (i+1)d_2\rangle} = \eta_{\langle id_2\rangle}$$

where $1 \le i \le p-2$, $\mathcal{X}_m = \mathcal{X}_{p-m} = \mathcal{Y}_r = \mathcal{Y}_{p-r} = 0$, and $M_{\langle (p-d_1)/2\rangle} = \mathcal{X}_{\langle (p-d_1)/2\rangle}$, $R_{\langle (p-d_2)/2\rangle} = \mathcal{Y}_{\langle (p-d_2)/2\rangle}$. Observe that $\langle (p-d_1)/2\rangle = \left\langle \frac{p-1}{2} d_1 \right\rangle$ and $\mathcal{X}_i = \mathcal{X}_{p-i}$, i.e., $\mathcal{X}_{\langle id_1\rangle} = \mathcal{X}_{\langle (p-i)d_1\rangle}$, thus we can evaluate $\mathcal{X}_{\langle (p-d_1)/2\rangle}$ as follows: First, find $1 \le n_1 \le (p-1)/2$ such that $\langle n_1 d_1\rangle = m$ or $\langle n_1 d_1\rangle = p-m$, then we have $\mathcal{X}_{\langle n_1 d_1\rangle} = 0$. Next, if $n_1 = (p-1)/2$ then set $\mathcal{X}_{\langle (p-d_1)/2\rangle} \leftarrow 0$, else let $\mathcal{X}_{\langle (p-d_1)/2\rangle} \leftarrow \bigoplus_{i=n_1}^{(p-3)/2} \xi_{\langle id_1\rangle}$. Similarly, $\mathcal{Y}_{\langle (p-d_2)/2\rangle}$ can be evaluated as follows: Find $1 \le n_2 \le (p-1)/2$ such that $\langle n_2 d_2\rangle = r$ or $\langle n_2 d_2\rangle = p-r$.

If $n_2 = (p-1)/2$ then set $\mathcal{Y}_{\langle (p-d_2)/2\rangle} \leftarrow 0$, else let $\mathcal{Y}_{\langle (p-d_2)/2\rangle} \leftarrow \bigoplus_{i=n_2}^{(p-3)/2} \eta_{\langle id_2\rangle}$.

Step 4: Evaluate M_i and R_i for $1 \le i \le p-1$, except M_{p-d_1} and R_{p-d_2}.

As discussed in the proof of Lemma 2, once $M_{\langle (p-d_1)/2\rangle}$ and $R_{\langle (p-d_2)/2\rangle}$ are obtained, we can easily evaluate other M_i and R_i using (33) and (34) alternately.

Step 5: Evaluate $b_{i,m}$ and $b_{i,r}$ for $1 \le i \le p-1$.

According to Lemma 3 and the definitions of M_i and R_i, we can easily retrieve all the data bits in the mth and rth columns.

Finally, retrieve the data bits in the lth column using (28), and then recalculate the missing parity bits according to (2), (3), (26) and (27), e.g., $b_{0,m} = S_m^{(1)} \oplus b_{d_1,l} \oplus b_{p-\delta,r}$ and $b_{p,r} = b_{p-d_2,l} \oplus b_{p-\delta,m} \oplus S_{p-r}^{(-1)}$. □

Remarks:

1. For certain erasure patterns, we can skip Step 2 and Step 3, since $M_{\langle (p-d_1)/2\rangle}$ and $R_{\langle (p-d_2)/2\rangle}$ can be directly obtained from the imaginary 0-bits. Specifically, if $\langle 2m+d_1\rangle = 0$ or $\langle 2m\rangle = d_1$, then we have $M_{\langle (p-d_1)/2\rangle} = b_{m,m} \oplus b_{p-m,m} = 0$. Similarly, if $\langle 2r+d_2\rangle = 0$ or $\langle 2r\rangle = d_2$, then we have $R_{\langle (p-d_2)/2\rangle} = b_{r,r} \oplus b_{p-r,r} = 0$. Obviously, if both $M_{\langle (p-d_1)/2\rangle}$ and $R_{\langle (p-d_2)/2\rangle}$ are zeros, then Step 2 and Step 3 are both unnecessary.
2. To facilitate the discussion, we have assumed that $l < m < r$, however, this assumption is unnecessary in the real implementation. Actually, we can swap the values of l, m and r, then find the combination that needs the least amount of XORs.

To make the above decoding algorithm easier to understand, for each case, we give an specific example to illustrate how the algorithm works.

Example 1: Let us consider the XI-Code with $p = 5$, of which the extended codeword is depicted in Figure 1. Suppose columns 1, 2 and 5 are erased, then the missing bits in the columns 1 and 2 can be rearranged as $b_{5,1}, b_{1,2}, b_{2,1}, b_{3,2}, b_{4,1}, b_{5,2}$ and $b_{0,2}, b_{1,1}, b_{2,2}, b_{3,1}, b_{4,2}, b_{0,1}$. From $b_{3,2} = b_{4,1} = 0$ we can evaluate other missing bits in the first sequence as follows:

1. $b_{5,2} = S_3^{(-1)} = b_{3,0} \oplus b_{1,3} \oplus b_{2,4}$
2. $b_{2,1} = S_1^{(-1)} = b_{1,0} \oplus b_{4,3} \oplus b_{5,4}$
3. $b_{1,2} = b_{2,1} \oplus S_3^{(1)}$
4. $b_{5,1} = b_{1,2} \oplus S_4^{(-1)}$

Similarly, from $b_{2,2} = b_{1,1} = 0$ we can evaluate other missing bits in the second sequence as follows:

1. $b_{0,2} = S_2^{(1)} = b_{2,0} \oplus b_{4,3} \oplus b_{3,4}$
2. $b_{3,1} = S_4^{(1)} = b_{4,0} \oplus b_{1,3} \oplus b_{0,4}$
3. $b_{4,2} = b_{3,1} \oplus S_2^{(-1)}$
4. $b_{0,1} = b_{4,2} \oplus S_1^{(1)}$

Note that the missing parity bits are always at the endpoints of the sequences, and hence they will not break the recursive chains. Finally, reconstruct the column 5 according to (1).

For the second case, i.e., $0 \le l < m < r \le p-1$, we only focus on Step 3 in the following example, since other steps are quite easy to understand.

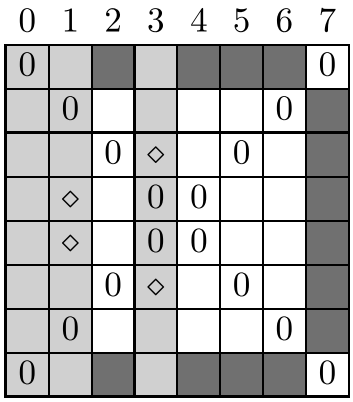

Fig. 3. The extended codeword structure of the XI-Code with $p = 7$, where columns 0, 1 and 3 are erased.

Example 2: Consider the XI-Code with $p = 7$. Figure 3 shows the extended array, assuming that columns 0, 1 and 3 are erased. Now we illustrate how to evaluate M_3 and R_2 from ξ_i and η_i respectively.

In this example, we have $l = 0, m = 1, r = 3$ and $d_1 = 1$, $d_2 = 3$. According to the definition of $\mathcal{X}_i$, we have $\mathcal{X}_1 = \mathcal{X}_6 = b_{1,1} \oplus b_{6,1}$, $\mathcal{X}_2 = \mathcal{X}_5 = b_{2,1} \oplus b_{5,1}$ and $\mathcal{X}_3 = \mathcal{X}_4 = b_{3,1} \oplus b_{4,1}$. Observe that $M_{\langle (7-1)/2\rangle} = b_{3,1} \oplus b_{4,1} = \mathcal{X}_3$, thus we only need to evaluate $\mathcal{X}_3$. Observe that $\mathcal{X}_{\langle 1\cdot 1\rangle} = \mathcal{X}_1 = 0$, i.e., $n_1 = 1$, and that $\mathcal{X}_1 \oplus \mathcal{X}_2 = \xi_1$, $\mathcal{X}_2 \oplus \mathcal{X}_3 = \xi_2$, thus we have $\mathcal{X}_3 = \xi_1 \oplus \xi_2$.

Similarly, according to the definition of $\mathcal{Y}_i$, we have $\mathcal{Y}_1 = \mathcal{Y}_6 = b_{1,3} \oplus b_{6,3}$, $\mathcal{Y}_2 = \mathcal{Y}_5 = b_{2,3} \oplus b_{5,3}$ and $\mathcal{Y}_3 = \mathcal{Y}_4 = b_{3,3} \oplus b_{4,3}$. Observe that $R_{\langle (7-3)/2\rangle} = b_{2,3} \oplus b_{5,3} = \mathcal{Y}_2$, thus we only need to evaluate $\mathcal{Y}_2$. According to Proposition 1, $\mathcal{Y}_1, \mathcal{Y}_2, \cdots, \mathcal{Y}_6$ can be rearranged as $\mathcal{Y}_{\langle 1\cdot 3\rangle}, \mathcal{Y}_{\langle 2\cdot 3\rangle}, \mathcal{Y}_{\langle 3\cdot 3\rangle}, \mathcal{Y}_{\langle 4\cdot 3\rangle}, \mathcal{Y}_{\langle 5\cdot 3\rangle}, \mathcal{Y}_{\langle 6\cdot 3\rangle}$. Observe that $\mathcal{Y}_{\langle 1\cdot 3\rangle} = \mathcal{Y}_3 = 0$, i.e., $n_2 = 1$, and that $\mathcal{Y}_{\langle 1\cdot 3\rangle} \oplus \mathcal{Y}_{\langle 2\cdot 3\rangle} = \eta_{\langle 1\cdot 3\rangle}$, $\mathcal{Y}_{\langle 2\cdot 3\rangle} \oplus \mathcal{Y}_{\langle 3\cdot 3\rangle} = \eta_{\langle 2\cdot 3\rangle}$, thus $\mathcal{Y}_2 = \mathcal{Y}_{\langle 3\cdot 3\rangle} = \eta_3 \oplus \eta_6$. □

In the case that row parity column is surviving, i.e., $0 \le l < m < r \le p-1$, if $m-l = r-m$ or $r-m = \langle l-r\rangle$ or $\langle l-r\rangle = m-l$, then we have another specially optimized decoding algorithm. To show the specific decoding procedure, we take the erasure pattern $\langle l-r\rangle = m-l$ for instance.

Algorithm 2 (Equidistant Erasures Decoding): First, let $d = m-l = \langle l-r\rangle$ and $\sigma = 2d$. According to the definitions of d_1 and d_2, we have $d_1 = d$ and $d_2 = p-d$, thus (31) and (32) are equivalent to

$$b_{i,l} \oplus b_{\langle i-d\rangle,m} \oplus b_{\langle i+d\rangle,r} = S_{\langle i+l\rangle}^{(1)} \tag{37}$$

$$b_{i,l} \oplus b_{\langle i+d\rangle,m} \oplus b_{\langle i-d\rangle,r} = S_{\langle i-l\rangle}^{(-1)} \tag{38}$$

where $1 \le i \le p-1$ and $i \ne d, p-d$. Let $\varphi_i = b_{i,m} \oplus b_{i,r}$, then the difference of (37) and (38) is

$$\varphi_{\langle i-d\rangle} \oplus \varphi_{\langle i+d\rangle} = S_{\langle i+l\rangle}^{(1)} \oplus S_{\langle i-l\rangle}^{(-1)} \tag{39}$$

where $1 \le i \le p-1$ and $i \ne d, p-d$. Note that (39) is equivalent to

$$\varphi_i \oplus \varphi_{\langle i+\sigma\rangle} = S_{\langle i+m\rangle}^{(1)} \oplus S_{\langle i-l+d\rangle}^{(-1)} \tag{40}$$

where $1 \le i \le p-1$ and $i \ne p-d, p-\sigma$. Since $l+d = m < r = l+p-d$, we have $\sigma = 2d < p$. If $l = 0$, then

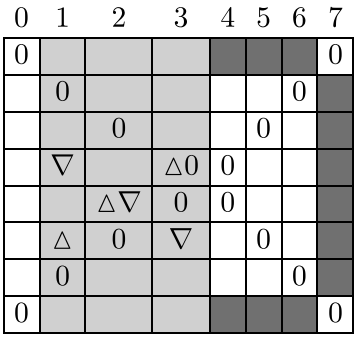

Fig. 4. The structure of the extended codeword of the XI-Code with $p = 7$.

$d = m = p - r$, hence $\varphi_d = b_{m,m} \oplus b_{p-r,r} = 0$ and $\varphi_{p-d} = b_{p-m,m} \oplus b_{r,r} = 0$. Otherwise, we have $b_{l,l} = b_{p-l,l} = 0$, then from (28) we can obtain $\varphi_l = b_{l,l} \oplus S_l^{(0)} = S_l^{(0)}$ and $\varphi_{p-l} = b_{p-l,l} \oplus S_{p-l}^{(0)} = S_{p-l}^{(0)}$. According to Lemma 3, we can iteratively evaluate other $\varphi_i (1 \le i \le p-1)$ using (40).

Once $\varphi_1, \varphi_2, \cdots, \varphi_{p-1}$ are obtained, we can easily retrieve $b_{i,l}$ using (28), i.e., $b_{i,l} = S_i^{(0)} \oplus \varphi_i$, where $1 \le i \le p-1$. Then from (37) and the definition of φ_i, we have

$$b_{i,m} \oplus b_{\langle i+\sigma\rangle,r} = S_{\langle i+m\rangle}^{(1)} \oplus b_{\langle i+d\rangle,l} \tag{41}$$

$$b_{k,m} \oplus b_{k,r} = \varphi_k \tag{42}$$

where $1 \le i, k \le p-1$ and $i \ne p-d, p-\sigma$. Using (41) and (42) we can retrieve $b_{i,m}$ and $b_{i,r}$ for $1 \le i \le p-1$. First, according to Proposition 1, $b_{1,j}, b_{2,j}, \cdots, b_{p-1,j} (j = m, r)$ can be rearranged as

$$b_{\sigma,m}, b_{\langle 2\sigma\rangle,m}, \cdots, b_{\langle (p-1)\sigma/2\rangle,m}, \cdots, b_{\langle (p-1)\sigma\rangle,m} \quad \text{(Seq .5)}$$

$$b_{\sigma,r}, b_{\langle 2\sigma\rangle,r}, \cdots, b_{\langle (p+1)\sigma/2\rangle,r}, \cdots, b_{\langle (p-1)\sigma\rangle,r} \quad \text{(Seq .6)}$$

which can be further rearranged as

$$b_{\sigma,r}, b_{\sigma,m}, b_{\langle 2\sigma\rangle,r}, b_{\langle 2\sigma\rangle,m}, \cdots, b_{\langle (p-1)\sigma/2\rangle,m} \quad \text{(Seq .7)}$$

$$b_{\langle (p+1)\sigma/2\rangle,r}, b_{\langle (p+1)\sigma/2\rangle,m}, \cdots, b_{\langle (p-1)\sigma\rangle,r}, b_{\langle (p-1)\sigma\rangle,m} \quad \text{(Seq .8)}$$

Then, observe that $\langle (p-1)\sigma\rangle = p - \sigma$ and $\left\langle \frac{p-1}{2}\sigma \right\rangle = p - d$, so if each sequence contains at lease one known element, then all the other elements can be evaluated by using (41) and (42) alternately. On the other hand, it is quite clear that imaginary 0-bits $b_{m,m}$ and $b_{p-m,m}$ are necessarily in different subsequences, and so do $b_{r,r}$ and $b_{p-r,r}$. Thus, all the data bits in the mth and rth columns can be retrieved from (41) and (42).

After retrieving all the missing data bits, we can easily recalculate the erased parity bits accoridng to (2) and (3). □

The above algorithm can be easily adapted to decode other equidistant erasure patterns.

Example 3: Consider the XI-Code with $p = 7$. Figure 4 shows the structure of the extended codeword. Suppose columns 1, 2 and 3 are erased, then the erased columns can be recovered using Algorithm 2, as follows.

First, let $l = 2, m = 3$ and $r = 1$, then we have $d = 1$ and $\sigma = 2$. From (40) we have

$$\varphi_i \oplus \varphi_{\langle i+2\rangle} = S_{\langle i+3\rangle}^{(1)} \oplus S_{\langle i-1\rangle}^{(-1)}, \quad 1 \le i \le 4 \tag{43}$$

where $\varphi_i = b_{i,1} \oplus b_{i,3}$. For example, from Figure 4 we can see that $b_{4,2}$ lies in the 6th diagonal and the 2nd anti-diagonal, hence we have $\varphi_3 \oplus \varphi_5 = S_6^{(1)} \oplus S_2^{(-1)}$. From $b_{2,2} = b_{5,2} = 0$ we can get $\varphi_2 = S_2^{(0)}$ and $\varphi_5 = S_5^{(0)}$. From φ_2, φ_5 and (43) we can evaluate $\varphi_1, \varphi_3, \varphi_4, \varphi_6$ as follows:

1. For $1 \le i \le 4$, let $\alpha_i \leftarrow S_{\langle i+3\rangle}^{(1)} \oplus S_{\langle i-1\rangle}^{(-1)}$.
2. $\varphi_4 \leftarrow \varphi_2 \oplus \alpha_2$
3. $\varphi_6 \leftarrow \varphi_4 \oplus \alpha_4$
4. $\varphi_3 \leftarrow \varphi_5 \oplus \alpha_3$
5. $\varphi_1 \leftarrow \varphi_3 \oplus \alpha_1$

Now we can easily retrieve $b_{i,2}$ for $i \in \{1, 3, 4, 6\}$, i.e., $b_{i,2} = S_i^{(0)} \oplus \varphi_i$. Next, we further retrieve the missing data bits in columns 1 and 3. From $b_{1,1} = b_{3,3} = 0$, we have $b_{1,3} = \varphi_1$ and $b_{3,1} = \varphi_3$. Then, $b_{5,1}$ and $b_{5,3}$ can be evaluated as follows: $b_{5,1} \leftarrow S_6^{(1)} \oplus b_{4,2}$, $b_{5,3} \leftarrow \varphi_5 \oplus b_{5,1}$. Similarly, from $b_{6,1} = b_{4,3} = 0$, we have $b_{6,3} = \varphi_6$ and $b_{4,1} = \varphi_4$. Then, $b_{2,1}$ and $b_{2,3}$ can be evaluated as follows: $b_{2,1} \leftarrow S_1^{(-1)} \oplus b_{3,2}$, $b_{2,3} \leftarrow \varphi_2 \oplus b_{2,1}$. Finally, retrieve the missing parity bits as follows:

1. $b_{0,1} \leftarrow S_1^{(1)} \oplus b_{6,2} \oplus b_{5,3}$
2. $b_{0,2} \leftarrow S_2^{(1)} \oplus b_{6,3}$
3. $b_{0,3} \leftarrow S_3^{(1)} \oplus b_{2,1} \oplus b_{1,2}$
4. $b_{7,1} \leftarrow S_6^{(-1)} \oplus b_{1,2} \oplus b_{2,3}$
5. $b_{7,2} \leftarrow S_5^{(-1)} \oplus b_{1,3}$
6. $b_{7,3} \leftarrow S_4^{(-1)} \oplus b_{5,1} \oplus b_{6,2}$

B. Correcting One Erasure Combined With One Error

If a single column error and up to one column erasure have occurred, then the key of decoding is to locate the column in error. This can be achieved by analyzing the relationship between the error vector and the syndromes. Suppose the lth column (denoted by $V = (b_{0,l}, b_{1,l}, b_{2,l}, \cdots, b_{p,l})^T$) is erased and the fth column is in error, with the error vector $\boldsymbol{e} = (e_0, e_1, e_2, \cdots, e_p)^T$. For a possibly corrupted codeword, we first compute the row syndromes $S_i^{(0)}$, diagonal syndromes $S_i^{(1)}$ and anti-diagonal syndromes $S_i^{(-1)}$ as follows:

$$S_i^{(0)} = \bigoplus_{\substack{t=0 \\ t \ne l}}^{p} b_{i,t} \tag{44}$$

$$S_i^{(1)} = \bigoplus_{\substack{t=0 \\ t \ne l}}^{p-1} b_{\langle i-t\rangle,t} \tag{45}$$

$$S_i^{(-1)} = \bigoplus_{\substack{t=1 \\ t \ne \langle l+i\rangle}}^{p} b_{t,\langle t-i\rangle} \tag{46}$$

where $i = 1, 2, \cdots, p-1$. Observe that $b_{i,j}$ is an imaginary 0-bit if $\langle i + j\rangle = 0$ or $\langle i - j\rangle = 0$, thus $S_0^{(1)} = S_p^{(-1)} = 0$.

To facilitate the following discussion, we let

$$\begin{aligned}
\boldsymbol{e}^{(0)} &= (0, e_1, e_2, \cdots, e_{p-1})^T \\
\boldsymbol{e}^{(1)} &= (e_0, e_1, e_2, \cdots, e_{p-1})^T \\
\boldsymbol{e}^{(-1)} &= (e_p, e_1, e_2, \cdots, e_{p-1})^T \\
V^{(0)} &= (0, b_{1,l}, b_{2,l}, \cdots, b_{p-1,l})^T \\
V^{(1)} &= (b_{0,l}, b_{1,l}, b_{2,l}, \cdots, b_{p-1,l})^T \\
V^{(-1)} &= (b_{p,l}, b_{1,l}, b_{2,l}, \cdots, b_{p-1,l})^T \\
S^{(i)} &= (0, S_1^{(i)}, S_2^{(i)}, \cdots, S_{p-1}^{(i)})^T
\end{aligned}$$

where $i = 0, 1, -1$. Moreover, we let $v(\downarrow n)$ (or $\boldsymbol{v}(\uparrow n)$) denote the vector derived from vector $\boldsymbol{v}$ by cyclically down- (or up-) shifting n positions. Then, we distinguish between the following two cases:

Case 1: $l = p$ and $0 \le f \le p-1$.

In this case, we just need to compute $S_i^{(1)}$ and $S_i^{(-1)}$. If there is a single column error in the codeword, then the error information, i.e., the error vector and the position of the erroneous column, will be reflected in the diagonal and anti-diagonal syndromes.

Clearly, if there is no error (i.e., $e_0 = e_1 = \cdots = e_p = 0$) in the codeword, then all the diagonal and anti-diagonal syndromes must be zeros. If the fth column is in error, then according to (45) and (46) we have $S_i^{(1)} = e_{\langle i-f \rangle}$ and $S_i^{(-1)} = e_{\langle i+f \rangle}$, hence $S^{(1)}(\uparrow f) = \boldsymbol{e}^{(1)}$ and $S^{(-1)}(\downarrow f) = \boldsymbol{e}^{(-1)}$. Observe that $\boldsymbol{e}^{(1)}$ and $\boldsymbol{e}^{(-1)}$ only differ in the first component, thus we can use enumeration to find the index $f (0 \le f \le p-1)$ such that $S^{(1)}(\uparrow f)$ and $S^{(-1)}(\downarrow f)$ only differ in the first component.

Case 2: $0 \le l \le p-1$.

This case can be further divided into two subcases: $f = p$ and $0 \le f \le p-1$. If $f = p$, then according to (45) and (46) we have $S^{(1)}(\uparrow l) = \boldsymbol{V}^{(1)}$ and $S^{(-1)}(\downarrow l) = \boldsymbol{V}^{(-1)}$. Therefore, if $S^{(1)}(\uparrow l)$ and $S^{(-1)}(\downarrow l)$ only differ in the first component, then there is no error in the first p columns. We can reconstruct the lth column using $S^{(1)}(\uparrow l)$ and $S^{(-1)}(\downarrow l)$, then re-encoding the pth column using (1).

Otherwise, one of the first p columns is in error. According to (44)~(46) we have $S_i^{(0)} = b_{i,l} \oplus e_i$, $S_i^{(1)} = b_{\langle i-l \rangle, l} \oplus e_{\langle i-f \rangle}$ and $S_i^{(-1)} = b_{\langle i+l \rangle, l} \oplus e_{\langle i+f \rangle}$, thus the following equations hold

$$V^{(0)} \oplus \boldsymbol{e}^{(0)} = S^{(0)} \tag{47}$$

$$\boldsymbol{V}^{(1)} \oplus \boldsymbol{e}^{(1)}(\downarrow \langle f-l \rangle) = S^{(1)}(\uparrow l) \tag{48}$$

$$\boldsymbol{V}^{(-1)} \oplus \boldsymbol{e}^{(-1)}(\uparrow \langle f-l \rangle) = S^{(-1)}(\downarrow l) \tag{49}$$

Let $\delta = \langle f-l \rangle$, and

$$\begin{aligned}
\boldsymbol{e}^{(s)} &= V^{(0)} \oplus \boldsymbol{e}^{(0)} \oplus \boldsymbol{V}^{(1)} = (b_{0,l}, e_1, e_2, \cdots, e_{p-1})^T \\
\boldsymbol{e}^{(-s)} &= V^{(0)} \oplus \boldsymbol{e}^{(0)} \oplus \boldsymbol{V}^{(-1)} = (b_{p,l}, e_1, e_2, \cdots, e_{p-1})^T
\end{aligned}$$

then adding (47) to (48) and (49) respectively results in

$$\boldsymbol{e}^{(s)} \oplus \boldsymbol{e}^{(1)}(\downarrow \delta) = S^{(0)} \oplus S^{(1)}(\uparrow l) \tag{50}$$

$$\boldsymbol{e}^{(-s)} \oplus \boldsymbol{e}^{(-1)}(\uparrow \delta) = S^{(0)} \oplus S^{(-1)}(\downarrow l) \tag{51}$$

Note that (51) is equivalent to

$$\boldsymbol{e}^{(-s)}(\downarrow \delta) \oplus \boldsymbol{e}^{(-1)} = (S^{(0)} \oplus S^{(-1)}(\downarrow l))(\downarrow \delta) \tag{52}$$

Compare the left sides of (50) and (52), we can find that $\boldsymbol{e}^{(s)}$ and $\boldsymbol{e}^{(-1)}$ only differ in the first component, and $\boldsymbol{e}^{(1)}(\downarrow \delta)$ and $\boldsymbol{e}^{(-s)}(\downarrow \delta)$ only differ in the δth component. Therefore, $S^{(0)} \oplus S^{(1)}(\uparrow l)$ and $(S^{(0)} \oplus S^{(-1)}(\downarrow l))(\downarrow \delta)$ must only differ in the first and δth components. As with last case, this correlation can be utilized to locate the error column, i.e., to determine the value of f. Note that both $b_{f,f}$ and $b_{p-f,f}$ are imaginary 0-bits, so $e_f = e_{p-f} = 0$. Therefore, once the error column is located, we can easily evaluate $\boldsymbol{e}^{(1)}$ and $b_{0,l}$ from (50) and then obtain $b_{p,l}$ and e_p from (51). Finally, correct the fth column by adding modulo-2 $\boldsymbol{e}$ to it, and retrieve the data bits of the lth column using (47).

From the analysis above, a formal algorithm for correcting one erasure combined with a single error can be described as follows:

Algorithm 3 (Single-Error One-Erasure Decoding): First, we compute the diagonal syndrome vector $S^{(1)}$ and the anti-diagonal syndrome vector $S^{(-1)}$ from the possibly corrupted codeword, according to (45) and (46) respectively. Next, we distinguish between the following two cases:

$l = p$. In this case, if both $S^{(1)}$ and $S^{(-1)}$ are all-zero vectors, then there is no error in the codeword. Otherwise, find the first index f to be $0 \le f \le p-1$, such that $S^{(1)}(\uparrow f) = S^{(-1)}(\downarrow f)$ or $S^{(1)}(\uparrow f)$ and $S^{(-1)}(\downarrow f)$ only differ in the first component. This index f corresponds to the location of the column in error. If there is no such f, then there may be more than one erroneous column in the codeword. Once f is determined, the error vector $\boldsymbol{e}$ can be obtained as: $\boldsymbol{e} = \begin{bmatrix} S^{(1)}(\uparrow f) \\ S_{p-f}^{(-1)} \end{bmatrix}$. Finally, add modulo-2 $\boldsymbol{e}$ to the fth column, and then reconstruct the lth column using (1).

$0 \le l \le p-1$. In this case, if $S^{(1)}(\uparrow l)$ and $S^{(-1)}(\downarrow l)$ only differ in the first component, then $f = p$. The lth column V can be obtained as $V = \begin{bmatrix} S^{(1)}(\uparrow l) \\ S_{p-l}^{(-1)} \end{bmatrix}$, then the pth column can be recalculated using (1). Otherwise, we first compute the row syndrome vector $S^{(0)}$ using (44). Then, let $u = S^{(0)} \oplus S^{(1)}(\uparrow l)$ and $v = S^{(0)} \oplus S^{(-1)}(\downarrow l)$. Next, we find the first index f to be $0 \le f \le p-1$, such that u and $v(\downarrow \langle f-l \rangle)$ only differ in the first and $\langle f-l \rangle$th components. Once we find f, let $\delta = \langle f-l \rangle$, then from (50) we have

$$\begin{aligned}
e_i \oplus e_{\langle i+\delta \rangle} &= u_{\langle i+\delta \rangle}, \quad i \ne p-\delta \\
e_{p-\delta} \oplus b_{0,l} &= u_0
\end{aligned} \tag{53}$$

where $0 \le i \le p-1$. Recall that $e_f = e_{p-f} = 0$, hence according to Lemma 3, we can easily evaluate $\boldsymbol{e}^{(1)}$ and $b_{0,l}$ using the above equations. Then, from (51) we have $b_{p,l} = e_\delta \oplus v_0$ and $e_p = e_{p-\delta} \oplus v_{p-\delta}$. Finally, add modulo-2 $\boldsymbol{e}$ to the fth column, and retrieve the data bits of the lth column using (47).

Example 4: Consider the XI-Code with $p = 7$. Assume that we receive the following corrupted codeword (Fig. 5), where column 1 is erased and column $f (0 \le f \le 7)$ is in error. Then, the codeword can be easily corrected using Algorithm 3.

First, from the corrupted codeword we can get $S^{(1)} = (0, 0, 1, 0, 1, 0, 0)^T$ and $S^{(-1)} = (0, 1, 1, 0, 1, 1, 1)^T$, so

	0	1	2	3	4	5	6	7
0	0	x	1	1	1	1	0	0
1	1	0	1	1	1	0	0	1
2	0	x	0	0	0	0	1	1
3	1	x	1	0	0	1	0	1
4	0	x	0	0	0	1	0	0
5	1	x	0	0	0	0	1	1
6	0	0	1	1	1	0	0	0
7	0	x	0	1	1	0	1	0

Fig. 5. A codeword with one erasure and one error.

	0	1	2	3	4	5	6	7
0	0	1	1	0	1	1	0	0
1	1	0	1	0	1	0	0	1
2	0	1	0	1	0	0	1	1
3	1	0	1	0	0	1	0	1
4	0	1	0	0	0	1	0	0
5	1	0	0	1	0	0	1	1
6	0	0	1	0	1	0	0	0
7	0	0	0	0	1	0	1	0

Fig. 6. The corrected codeword.

$S^{(1)}(\uparrow\ 1) = (0,1,0,1,0,0,0)^T$ and $S^{(-1)}(\downarrow\ 1) = (1,0,1,1,0,1,1)^T$. Clearly, $S^{(1)}(\uparrow\ 1)$ and $S^{(-1)}(\downarrow\ 1)$ *not* only differ in the first component. Therefore, we need to compute $S^{(0)}$ from the corrupted codeword, i.e., $S^{(0)} = (0,1,0,0,1,1,1)^T$. Then, let $u = S^{(0)} \oplus S^{(1)}(\uparrow\ 1) = (0,0,0,1,1,1,1)^T$ and $v = S^{(0)} \oplus S^{(-1)}(\downarrow\ 1) = (1,1,1,1,1,0,0)^T$. Next, find the first index δ to be $1 \le \delta \le 6$, such that u and $v(\downarrow\ \delta)$ are identical except the first and δth components. It is quite easy to determine that $\delta = 2$, and hence $f = \langle 1+2 \rangle = 3$.

Observe that $e_3 = e_4 = 0$, thus from (53) and (51) we can determine the error vector $\boldsymbol{e} = (1,1,1,0,0,1,1,1)^T$ and the erased parity bits $b_{0,1} = 1$, $b_{7,1} = 0$. Finally, add modulo-2 $\boldsymbol{e}$ to column 3, and retrieve the data bits of column 1 as $b_{i,1} = e_i \oplus S_i^{(0)}$, where $2 \le i \le 5$. The corrected codeword is shown in Figure 6.

IV. Complexity Analysis

To demonstrate the advantages of XI-code, in this section we analyze its encoding, decoding and update complexities, then compare them with that of other representative triple-erasure-correcting codes.

A. Lower Bounds of Computational Complexities for MDS Codes

As a reference, we first present the lower bounds of encoding, decoding and update complexities for MDS codes. For a $(n, n-r)$ MDS code, clearly the lower bound of update complexity is r, i.e., modifying one data bit need to update at least r corresponding redundant bits. Let d denote its distance, then we have $d = r+1$. For the simple parity code $(n, n-1)$, it is quite clear that $d = 2$ and both encoding and decoding need exactly $n-2$ XOR's. For MDS codes of distance 3, it has been proven in [6] and [16] that $n-3$ is exactly the lower bound of encoding/decoding complexity. However, for MDS codes of distance greater than 3, the theoretical lower bound of encoding/decoding complexity remains unknown. Nevertheless, previous work [12], [21] still *conjecture* that, for MDS codes of distance d, $n-d$ XOR's per redundant/erased bit is the lower bound of encoding/decoding complexity. We will follow this conjecture in the following discussion. It is worth mentioning that, the lower bound of decoding complexity refers only to erasure decoding, but not error decoding.

B. Encoding and Update Complexities of XI-Code

As discussed in Section II, the code length of XI-code can be $p+1$ or p, i.e., $n \in \{p+1, p\}$. In addition, since the column distance of XI-code is 4, we have $d = 4$ and $r = 3$. We have shown in Section II that the update complexity of XI-code is 3, which equals the lower bound. Moreover, according to the encoding rules of XI-code, each parity bit is constructed from exactly $n-3$ data bits (*the imaginary 0-bits do not need to really participate in the calculations*). Therefore, XI-code requires $n-4$ XORs per parity bit in encoding, which is exactly the lower bound.

C. Decoding Complexity of XI-Code

In contrast to encoding and update, the decoding complexity is much more complicated to compute, since it depends on the specific erasure patterns. Therefore, we will discuss every possible erasure pattern respectively in the following.

1) Three Erasures: Corresponding to the decoding algorithms in Section III, we distinguish between the following two cases: a) one of the three erasures is the row parity column, b) three of the first $n-1$ columns are erased while the row parity column survives.

For the first case, from the reconstruction procedure we can find that essentially every missing data bit is retrieved by XORing the other $n-2$ bits along the diagonal or anti-diagonal that traverses it. Since there is an imaginary 0-bit among the $n-2$ bits, we need only $n-4$ XOR operations for reconstructing each missing data bit. Once the missing data bits are retrieved, reconstructing the erased parity bits is equivalent to re-encoding them, thus each erased parity bit also needs $n-4$ XORs for reconstruction. From the above, it is clear that *for this erasure pattern the decoding complexity attains the lower bound*.

For the second case, in order to simplify the calculation, we first assume that the imaginary 0-bits really participate in the decoding. Now let us consider the syndrome calculations. According to (25), calculating row syndromes needs $(p-1)(n-4)$ XORs. If one of the three erasures is the 0th column, then according to (26) and (27), calculating diagonal and anti-diagonal syndromes needs $2(p-3)(n-5)$ XORs, so we need $3pn - 14p - 7n + 34$ XORs in total for calculating all

the syndromes. If the 0th column is surviving, then calculating diagonal and anti-diagonal syndromes needs $2(p-4)(n-5)$ XORs, so we need $3pn-14p-9n+44$ XORs in total for calculating all the syndromes.

Now we further distinguish between the following two subcases: equidistant erasures and non-equidistant erasures. For equidistant erasures, let us calculate the number of XOR operations in Algorithm 2. First, consider the case of $l=0$, we can calculate the number of XORs needed by each step of the decoding procedure as follows:

1. Obtaining (39) needs $p-3$ XORs
2. Evaluating $\varphi_i (1 \le i \le p-1)$ needs $p-1$ XORs
3. Retrieving $b_{i,l} (1 \le i \le p-1)$ needs $p-1$ XORs
4. Obtaining (41) needs $2\left(\frac{p-1}{2}-1\right)=p-3$ XORs
5. Retrieving $b_{i,j} (1 \le i \le p-1, j=m,r)$ needs $2(p-3)$ XORs
6. Retrieving missing parity bits needs $4(n-3)$ XORs

The above procedure involves $6p+4n-26$ XOR operations in total. Note that each imaginary 0-bit is associated with two parity bits, thus all the $2(p-1)$ imaginary 0-bits contribute $4(p-1)$ XORs to the above results. However, as mentioned before, imaginary 0-bits do not really participate in the encoding/decoding. Therefore, the actual total number of XORs needed for decoding can be calculated as $3pn-14p-7n+34+(6p+4n-26)-4(p-1)=3(p-1)(n-4)$. In Algorithm 2, if $l \neq 0$, then Step 3 needs $p-3$ XORs, Step 4 needs $p-5$ XORs and Step 6 needs $6(n-3)$ XORs. Consequently, the total number of XORs in decoding is $3pn-14p-9n+44+(6p+6n-36)-4(p-1)=3(p-1)(n-4)$. We find that whether the three erasures contain the 0th column or not does not affect the total number of XORs needed for decoding. From the above, in this case the decoding complexity is $\frac{3(p-1)(n-4)}{3(p-1)}=n-4$ XORs per missing bit, which attains the lower bound. The reader can easily check that for other equidistant erasure patterns, the decoding complexity also attains the lower bound.

For non-equidistant erasures, we take the case of $l=0$ for instance. According to Algorithm 1, we can calculate the number of XORs needed by each step as follows:

1. Obtaining (33) and (34) needs $2(p-3)$ XORs
2. Evaluating ξ_i and η_j needs $2(p-4)$ XORs
3. Evaluating $M_{\langle (p-d_1)/2 \rangle}$ and $R_{\langle (p-d_2)/2 \rangle}$ needs at most $p-3-n_1-n_2$ XORs
4. Evaluating other M_i and R_i needs $2(p-3)$ XORs
5. Retrieving $b_{i,j} (1 \le i \le p-1, j=m,r)$ needs $2(p-3)$ XORs
6. Retrieving $b_{i,l} (1 \le i \le p-1)$ needs $2(p-1)$ XORs
7. Retrieving missing parity bits needs $4(n-3)$ XORs

At worst, the above procedure involves $11p+4n-43$ XOR operations in total. Thus, the total number of XORs needed for decoding can be calculated as $3pn-14p-7n+34+(11p+4n-43)-4(p-1)=3(p-1)(n-4)+5p-17$. So the decoding complexity is $\frac{3(p-1)(n-4)+5p-17}{3(p-1)}=n-4+5/3-\frac{4}{p-1}$ XORs per missing bit, which approaches the lower bound. Since $n \in \{p+1, p\}$, we have

$$\lim_{p\to\infty} \frac{n-4+5/3-4/(p-1)}{n-4}=1,$$

i.e., the decoding complexity is asymptotically optimal as $p \to \infty$.

2) One Erasure Combined With a Single Error: According to Algorithm 3, this error pattern contains two possible cases: a) the row parity column is erased, b) the row parity column survives.

For the first case, calculating diagonal and anti-diagonal syndromes needs $2(p-1)(n-3)$ XORs, and finding the error column needs on average $n-1$ cyclic shifts and equivalence test operations. Then, correcting the error column needs $p-1$ XORs and reconstructing the row parity column needs $(p-1)(n-4)$ XORs. In this case, the decoding procedure requires $3(p-1)(n-3)$ XOR's and $n-1$ cyclic shifts in total.

For the second case, we take the condition of $l \neq 0$ (the erased column is not the 0th column) for instance. Computing $S^{(1)}(\uparrow l)$ and $S^{(-1)}(\downarrow l)$ needs $2(p-1)(n-4)+2$ XORs and $2l$ cyclic shifts. If $S^{(1)}(\uparrow l)$ and $S^{(-1)}(\downarrow l)$ only differ in the last component, then the error column is the row parity column. The erased column can be directly obtained from $S^{(1)}(\uparrow l)$ and $S^{(-1)}(\downarrow l)$, and reconstructing the row parity column needs $(p-1)(n-4)$ XORs. Otherwise, computing $S^{(0)}$ needs $(p-1)(n-4)+2$ XORs, and evaluating $S^{(0)} \oplus S^{(1)}(\uparrow l)$ and $S^{(0)} \oplus S^{(-1)}(\downarrow l)$ requires $2(p-2)$ XORs, since every syndrome vector contains a constant zero component. Then, finding the error column needs on average $(n-1)/2$ cyclic shifts and equivalence test operations, and evaluating $\boldsymbol{e}$ needs $p-1$ XORs. Finally, correcting the error column needs $p-1$ XORs and reconstructing the erased column needs at most $p-1$ XORs. In this case, the decoding procedure requires at most $(p-1)(3n-7)+2$ XORs and $2l+(n-1)/2$ cyclic shifts in total.

V. Comparison With Other Representative Array Codes

In this section we will show that, in addition to the optimal update property, the erasure decoding complexity of XI-Code is also quite close to the lower bound and is much lower than the current best results.

As discussed in the introduction, existing lowest density codes are impractical due to their extremely strict code length constraints and absence of explicit decoding algorithms. In particular, given the size of a disk array that usually ranges roughly from 5 to 30 in the production environment, the length constraints of the existing lowest density codes result in a very small subset of sizes in this range, namely, $\{6, 12, 13, 18, 19\}$, that can be supported. In addition, all of the existing lowest density array codes of distance 4 failed to present *specially optimized* decoding algorithms due to the *irregular* geometric constructions of such codes. Although it is straight-forward to decode them according to their parity check matrices, such method usually results in a *relatively high* decoding complexity. Specifically, the codes in [17]–[19] all have a column size of $(p-1)/3$, hence they require about $(n-6)(p-1)$ XORs to calculate the syndromes, and at most $(p-1)(p-2)$ XORs to solve the corresponding system of linear equations, where $n \in \{p, p-1\}$. Therefore, the decoding complexity of these codes is about $n+p-8$ XORs per missing bit, which is about *twice* the theoretical lower bound.

TABLE I
A COMPREHENSIVE COMPARISON AMONG MDS ARRAY CODES OF DISTANCE 4

	Code length	Encoding	Update	Decoding	Constraints on p
generalized RDP	$p+1$ or $p+2$	$n-4$	5	$n-\frac{(7-x)(p-1)+1}{3(p-1)}$	—
[17] and T-Code	p or $p-1$	$n-4$	3	$n+p-8$	2 primitive in $\mathrm{GF}(p)$ and $3\|(p-1)$
[18]	$p-1$	$p-5$	3	$2p-9$	2 primitive in $\mathrm{GF}(p)$ and $3\|(p-1)$
XI-Code	$p+1$ or p	$n-4$	3	$n-\frac{7p+5}{3(p-1)}$	—

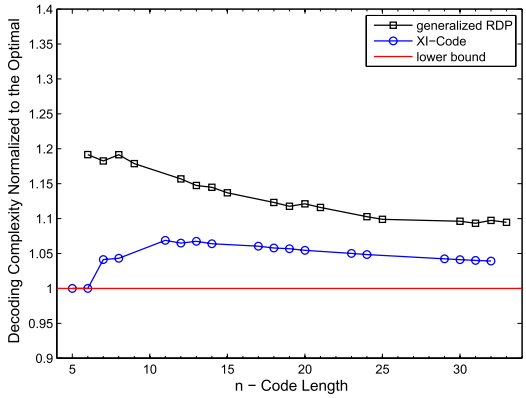

Fig. 7. Decoding complexities of XI-Code and generalized RDP.

Now let us consider those practical MDS array codes of distance 4, such as generalized EVENODD, STAR, and generalized RDP codes. Among these codes, generalized RDP codes represent the state of the art, since they can achieve the lower bound of encoding complexity and decode more efficiently than other alternatives. Thus, we only compare our XI-Code with generalized RDP codes in terms of decoding complexity. Because the decoding complexity depends on the specific erasure patterns, for a code of length n we tested all the C_n^3 possible combinations of erasures and computed the average complexity. Furthermore, in order to compare codes of different lengths, we normalize the average decoding complexity by dividing it by the lower bound of $n-4$. Figure 7 shows the normalized decoding complexities of XI-codes and generalized RDP codes for $5 \leq n \leq 33$. Note that there are several variants of generalized RDP codes (e.g., [12], [21], [22]) and we choose the one with the best decoding, namely RTP codes, as XI-Code's competitor.

It is quite clear that the decoding complexity of XI-Code is notably lower than that of generalized RDP codes, especially for small code lengths that are preferred by storage systems. Moreover, the decoding complexity of XI-Code attains the exact lower bound when $n \in \{5, 6\}$, and is very close to the lower bound for other values of n. This is because when $p = 5$, all of the erasure patterns are either equidistant erasures or erasures containing the row parity column.

To put it all together and provide a reasonable perspective, we compare our XI-Code with other competitive MDS array codes of distance 4 in all key metrics of coding in Table 1 to end this section, where n refers to the code length.

VI. CONCLUSIONS

We have presented a new class of lowest density MDS array codes of distance 4, called XI-codes, which is capable of correcting triple erasures, as well as a single error combined with one erasure. These codes have optimal encoding and update complexities, and the code length can be either p or $p+1$, provided that p is an odd prime. The erasure decoding is also optimal for some erasure patterns, such as equidistant erasures and the erasures containing the last column. For other erasure patterns, the decoding complexity is near optimal, and is asymptotically optimal as $p \to \infty$. Owing to these attractive properties, we believe that XI-codes have a potential to become a preferred choice for storage systems that need to tolerate triple failures.

Our future work includes efforts to discover new *lowest-density* MDS array codes for length of all positive integers and for correcting more than three erasures. Previous studies have shown that both of these two problems are non-trivial and challenging. In particular, our preliminary research shows that the minimum distance of XI-Code can be increased by simply forcing more pairs of diagonals to be zeros and adding corresponding parity constraints. However, the generalized XI-Code is generally *not* MDS, except for $p \in \{5, 7\}$. Further research is still ongoing.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments, which helped to improve the quality of the paper.

REFERENCES

[1] M. Blaum, P. G. Farrell, and H. C. A. van Tilborg, "Chapter on array codes," in *Handbook of Coding Theory*, vol. 2, V. S. Pless and W. C. Huffman, Eds. Amsterdam, The Netherlands: North Holland, 1998, pp. 1855–1909.

[2] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, Jun. 1960. [Online]. Available: http://epubs.siam.org/doi/pdf/10.1137/0108018

[3] 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.

[4] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, New York, NY, USA, 1988, pp. 109–116.

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

[6] P. Corbett *et al.*, "Row-diagonal parity for double disk failure correction," in *Proc. 3rd USENIX Conf. File Storage Technol.*, 2004, pp. 1–14.

[7] J. S. Plank, "The RAID-6 liberation codes," in *Proc. 6th USENIX Conf. File Storage Technol.*, 2008, Art. no. 7.

[8] Y. Wang, X. Yin, and X. Wang, "MDR codes: A new class of RAID-6 codes with optimal rebuilding and encoding," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 1008–1018, May 2014.

[9] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: High-performance, reliable secondary storage," *ACM Comput. Surv.*, vol. 26, no. 2, pp. 145–185, Jun. 1994. [Online]. Available: http://dl.acm.org/citation.cfm?id=176981

[10] M. Blaum, J. Bruck, and A. Vardy, "MDS array codes with independent parity symbols," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 529–542, Mar. 1996.
[11] C. Huang and L. Xu, "STAR: An efficient coding scheme for correcting triple storage node failures," *IEEE Trans. Commun.*, vol. 57, no. 7, pp. 889–901, Jul. 2008.
[12] M. Blaum, "A family of MDS array codes with minimal number of encoding operations," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2006, pp. 2784–2788.
[13] M. Blaum and R. M. Roth, "On lowest density MDS codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 1, pp. 46–59, Jan. 1999.
[14] L. Xu and J. Bruck, "X-code: MDS array codes with optimal encoding," *IEEE Trans. Inf. Theory*, vol. 45, no. 1, pp. 272–276, Jan. 1999.
[15] L. Xu, V. Bohossian, J. Bruck, and D. G. Wagner, "Low-density MDS codes and factors of complete graphs," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 1817–1826, Sep. 1999. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=782102
[16] C. Jin, H. Jiang, D. Feng, and L. Tian, "P-Code: A new RAID-6 code with optimal properties," in *Proc. 23rd Int. Conf. Supercomput.*, 2009, pp. 360–369.
[17] E. Louidor and R. M. Roth, "Lowest density MDS codes over extension alphabets," *IEEE Trans. Inf. Theory*, vol. 52, no. 7, pp. 3186–3197, Jul. 2006.
[18] Y. Cassuto and J. Bruck, "Cyclic lowest density MDS array codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 4, pp. 1721–1729, Apr. 2009.
[19] S. Lin, G. Wang, D. S. Stones, J. Liu, and X. Liu, "T-Code: 3-erasure longest lowest-density MDS codes," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 289–296, Feb. 2010. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5402496
[20] Z. Huang, H. Jiang, K. Zhou, Y. Zhao, and C. Wang, "Lowest density MDS array codes of distance 3," *IEEE Commun. Lett.*, vol. 19, no. 10, pp. 1670–1673, Oct. 2015.
[21] A. Goel and P. Corbett, "RAID triple parity," *ACM SIGOPS Oper. Syst. Rev.*, vol. 46, pp. 41–49, Dec. 2012.
[22] H. Fujita, "Modified low-density MDS array codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2006, pp. 2789–2793.

Zhijie Huang received the B.E. degree from Jimei University, Xiamen, China, in 2005, the M.E. degree from the University of Jiangsu, Zhenjiang, China, in 2010, and the Ph.D. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2016, all in computer science and technology. His research interests include coding theory and application, dependable and secure computing, storage systems and parallel I/O, and embedded systems.

Hong Jiang (S'87–M'91–SM'10–F'15) received the B.Sc. degree in computer engineering in 1982, from Huazhong University of Science and Technology, Wuhan, China; the M.A.Sc. degree in computer engineering in 1987, from the University of Toronto, Toronto, Canada; and the Ph.D. degree in computer science in 1991, from the Texas A&M University, College Station, TX, USA. He is currently Chair and Wendell H. Nedderman Endowed Professor of the Computer Science and Engineering Department at the University of Texas at Arlington (UTA). Prior to joining UTA, he served as a Program Director at National Science Foundation (from January 2013 to August 2015) and he at the University of Nebraska-Lincoln since 1991, where he was the Willa Cather Professor of Computer Science and Engineering. He has graduated 13 Ph.D. students who, upon their graduations, either landed academic tenure-track positions in Ph.D.-granting U.S. institutions or were employed by major U.S. IT corporations. His present research interests include computer architecture, computer storage systems and parallel I/O, high-performance computing, big data computing, cloud computing, and performance evaluation. He recently served as an Associate Editor of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. He has over 200 publications in major journals and international conferences in these areas, including IEEE-TPDS, IEEE-TC, Proceedings of IEEE, ACM-TACO, JPDC, ISCA, MICRO, USENIX ATC, FAST, EUROSYS, LISA, SIGMETRICS, ICDCS, IPDPS, MIDDLEWARE, OOPLAS, ECOOP, SC, ICS, HPDC, INFOCOM, ICPP, etc., and his research has been supported by NSF, DOD, the State of Texas, and the State of Nebraska. Dr. Jiang is a Member of ACM.

Ke Zhou (M'13) received the B.E., M.E., and Ph.D. degrees from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1996, 1999, and 2003, respectively, all in computer science and technology. He is currently a Professor with the School of Computer Science and Technology, HUST. He has authored over 50 publications in journals and international conferences, including *Performance Evaluation*, FAST, MSST, ACM MM, SYSTOR, MASCOTS, and ICC. His main research interests include computer architecture, cloud storage, parallel I/O, and storage security.

Chong Wang received the B.E. degree in computer science and technology from the University of Henan, in 2008. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Huazhong University of Science and Technology. His area of interests includes storage systems, coding theory, and cloud computing.

Yuhong Zhao received the B.E. and M.E. degrees in computer science and technology from Heilongjiang University, Harbin, China, in 2006 and 2010, respectively, and the Ph.D. degree in computer science and technology from the Huazhong University of Science and Technology, Wuhan, China, in 2015. His main research interests include computer architecture, cloud computing, computer storage system, and big data.