Efficient Lowest Density MDS Array Codes of Column Distance 4

Zhijie Huang*, Hong Jiang† and Nong Xiao*
*School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, 510006, P. R. China
†Department of Computer Science & Engineering, University of Texas at Arlington, TX 76010, USA
*{huangzhj35,xiaon6}@mail.sysu.edu.cn,†hong.jiang@uta.edu

***Abstract*—The extremely strict code length constraint is the main drawback of lowest density, maximum-distance separable (MDS) array codes of distance greater than 3. To break away from the status quo, we proposed in [5] a family of lowest density MDS array codes of (column) distance 4, called XI-Code. Compared with the previous alternatives, XI-Code has lower encoding and decoding complexities, and much looser constraint on the code length, thus is much more practical. In this paper, we present a new family of lowest density MDS array codes of (column) distance 4, called RΛ-Code, which is derived from XI-Code, and outperforms the latter substantially in terms of encoding complexity, decoding complexity, and memory consumption during encoding/decoding. The inherent connection between RΛ-Code and XI-Code and how the former is derived from the latter may provide inspiration for the readers to derive new codes from other existing codes in a similar way.**

I. Introduction

Array codes [2] are a special class of erasure codes, in which symbols are column vectors and hence each codeword is a two-dimensional array. The error model of array codes is that, if one component of a column is an error or erasure, then the whole column is considered an error or erasure. This makes array codes be particularly suitable for correcting burst errors in communication networks, as well as disk (node) failures in storage systems. The main advantage of array codes is use of only simple XOR and cyclic shift operations in their encoding and decoding procedures, making them quite efficient in terms of computational complexity. Maximum-distance separable (MDS) array codes are array codes that provide a certain level of fault tolerance with the minimum redundancy, hence are particularly practical.

MDS array codes are widely employed in modern storage systems to prevent data loss caused by disk/node failures and latent sector errors. In this application scenario, the data content is usually dynamic. That is, since the redundant information is constructed from the data content, to maintain the consistency, whenever a data element (e.g., a sector or a filesystem block) is changed, the coding elements to which the element contributes must be updated (read-modify-write) accordingly. In array codes, the average number of coding elements that must be updated whenever a data element is modified, is referred to as *update complexity*. Update complexity is a *key performance metric* of particular concern by array codes designed for storage systems, since it dictates the access time to the disk/node array even in the absence of failures, and it will also directly translate into costly communication overhead when array codes are employed in distributed storage systems (e.g., datacenter and cloud environments). Moreover, if array codes are employed in disk arrays composed of flash solid-state devices (SSDs), the update complexity will also affect the lifetime of the SSDs.

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 [3][11][10][9] [4][7][8][6]. These codes contain a minimal number of nonzero elements in their parity check matrices, so that they can achieve the lower bound of update complexity. Unfortunately, most of the existing lowest density array codes are of distance 3, which may be insufficient for massive storage systems with hundreds of thousands storage devices in distributed environments. Although there exist a few lowest density array codes of distance 4 or 5 [4][8][9], their parameter constraints are so strict that they can hardly be applied to practical systems. To fill the gap, in our previous work [5] we proposed XI-Code — a family of lowest density MDS array codes of column distance 4, which has several attractive properties and much looser constraints on the code length. So far, XI-Code remains the most practical and efficient class of lowest density MDS array codes of column distance 4.

In this paper, we present a new family of lowest density MDS array codes of column distance 4, called RΛ-Code, which is derived from XI-Code. In general, RΛ-Code retains the following properties of XI-Code: a) capable of correcting both triple erasures and a single error combined with one erasure, b) optimal update complexity, and c) supporting code length of p or $p+1$, where p is an odd prime; and outperforms the latter in the following three aspects:

- The encoding complexity is around $16.7 \sim 22.2$ percent lower than that of XI-Code.
- The decoding complexity is up to 16.7 percent lower than that of XI-Code.
- The column size is half that of XI-Code, meaning that the memory consumption during encoding/decoding is half that of XI-Code.

It is worth mentioning that, the encoding/decoding complexity of RΛ-Code is also *demonstrably* lower than the currently established "lower bound"[1][7], which may interest the theoretical community.

978-1-5090-4096-4/17/$31.00 ©2017 IEEE

II. RΛ-CODE

Since RΛ-Code is derived from XI-Code, it is necessary to briefly review the construction and properties of the latter first.

In XI-Code, each codeword is a $(p-1)\times(p+1)$ array of 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. To facilitate the following description, we let $\langle x\rangle = x \bmod p$, and use $b_{i,j}$ to denote the ith bit in the jth column, where i and j both count from 0.

Assume that each codeword is a $(p+1)\times(p+1)$ array with every column containing two extra imaginary 0-bits. Specifically, in each extended codeword, $b_{i,j}$ is an imaginary 0-bit if $i+j=p$ or $i=j$. The parity bits are stored in the first row, last row and last column, and are computed as follows:

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

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

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

It can be seen that, there are three types of parity bits, which are constructed by XORing data bits along lines of slopes 0, 1, and -1 respectively. Note that the imaginary 0-bits do not really participate in the calculations, i.e., each parity bit is constructed from only $p-2$ data bits in practice.

It was proven in [5] that XI-Code is a class of lowest density MDS array codes of (column) distance 4, i.e., XI-Code can recover up to three erasures or one erasure combined with a single error, and its update complexity is 3. Moreover, since the first column is a pure data column, the code can be shortened by assuming that the first column is an imaginary 0-column, i.e., the code length of XI-Code can be $p+1$ or p.

A. From XI-Code to RΛ-Code

Given a $(p+1)\times(p+1)$ array defined above, we only use $b_{i,j}(1\le i\le (p-1)/2, 0\le j\le p-1)$ to store user data, and for $(p+1)/2\le i\le p, 0\le j\le p-1$, let $b_{i,j}\leftarrow b_{p-i,j}$, where $\leftarrow$ is an assignment operator. Then, compute the parity bits according to (1)–(3), and we will find that $b_{i,p}=b_{p-i,p}$ and $b_{0,j}=b_{p,j}$ hold due to the symmetry of the information part. Since $b_{i,j}=b_{p-i,j}$ holds for $0\le i,j\le p$, we can delete the last $(p+1)/2$ rows of the array. Observe that for $1\le j\le p-1$, we have $\bigoplus_{t=(p+1)/2}^{p-1} b_{t,\langle j-t\rangle} = \bigoplus_{t=1}^{(p-1)/2} b_{t,\langle j+t\rangle}$ due to symmetry, thus $b_{0,j}=\bigoplus_{t=1}^{(p-1)/2}(b_{t,\langle j-t\rangle}\oplus b_{t,\langle j+t\rangle})$ holds. From the above, the following equations define a new array code of size $(p+1)/2\times(p+1)$, which we call RΛ-Code:

$$b_{i,p} = \bigoplus_{t=0}^{p-1} b_{i,t}, \quad 1\le i\le (p-1)/2 \tag{4}$$

$$b_{0,j} = \bigoplus_{t=1}^{(p-1)/2} (b_{t,\langle j-t\rangle}\oplus b_{t,\langle j+t\rangle}), \quad 1\le j\le p-1 \tag{5}$$

Now let us look into the minimal working example, i.e., the RΛ-Code with $p=5$. Figure 1 shows the distribution of data bits, parity bits, and imaginary 0-bits in a codeword, as well as the parity sets of the code.

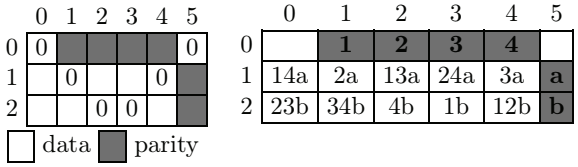

Figure 1. The structure of the codeword and the parity sets of the RΛ-Code with $p=5$, where each parity set is labeled with an integer/letter.

Geometrically speaking, there are two types of parity sets, one formed along rows and the other formed by (broken) lines along diagonals of slopes 1 and -1 that resembles the letter "Λ". We refer to the two types of parity sets as "row parity sets" and "Λ parity sets" respectively. Each row parity bit is obtained by XORing the data bits along the same row, and each Λ parity bit is obtained by XORing the data bits along the diagonal of slope 1 and the diagonal of slope -1 that traverse the parity bit itself.

Essentially, RΛ-Code is the XI-Code with the following constraint: for $0\le i,j\le p$, equation $b_{i,j}=b_{p-i,j}$ holds. Therefore, we have the following theorem:

Theorem 1: For any odd prime p, the RΛ-code defined above is a lowest density MDS array code of column distance 4.

B. Encoding of RΛ-Code

From (4) and (5) we can find that, the expression $b_{i,\langle j-i\rangle}\oplus b_{i,\langle j+i\rangle}$ is involved in calculating both $b_{i,p}$ and $b_{0,j}$, thus the encoding complexity can be reduced if we evaluate the common expression first and reuse the result in encoding. Note that only the common expressions that do not involve any imaginary 0-bits are useful. As an example, let us look into Figure 1. There are four *useful* common expressions in total, however, this does not imply that we can save 4 XORs during encoding. The reason is that some common expressions cannot be used simultaneously, e.g., $b_{1,0}\oplus b_{1,2}$ and $b_{1,0}\oplus b_{1,3}$, similarly $b_{2,0}\oplus b_{2,1}$ and $b_{2,0}\oplus b_{2,4}$. In this example, an alternative encoding scheme is to first evaluate common expressions $b_{1,0}\oplus b_{1,2}$ and $b_{2,0}\oplus b_{2,1}$, then reuse the results in the calculations of the corresponding parity bits, which can save 2 XORs.

From the above, the key to the encoding algorithm is using as many common expressions as possible. To achieve this goal, we present the respective schemes for the following two cases: $4|(p-3)$ and $4\nmid(p-3)$. For the first case, it is possible to avoid using the common expressions that involve data bits of column 0, so that *all* the selected common expressions can be used by the shortened RΛ-Code. Let us take the RΛ-Code with $p=7$ for instance, whose parity sets are shown in Figure 2. The best scheme is to use the following common expressions: $b_{1,2}\oplus b_{1,4}$, $b_{1,3}\oplus b_{1,5}$, $b_{2,1}\oplus b_{2,4}$, $b_{2,3}\oplus b_{2,6}$, $b_{3,1}\oplus b_{3,2}$ and $b_{3,5}\oplus b_{3,6}$, since none of them involves data bits of column 0. For the second case, however, we cannot maximize the number

of compatible common expressions without using the common expressions that involve data bits of column 0.

	0	1	2	3	4	5	6	7
0		**1**	**2**	**3**	**4**	**5**	**6**	
1	16a	2a	13a	24a	35a	46a	5a	**a**
2	25b	36b	4b	15b	26b	3b	14b	**b**
3	34c	45c	56c	6c	1c	12c	23c	**c**

Figure 2. The parity sets of the RΛ-Code with $p = 7$

In what follows we present a formal encoding algorithm that is optimized for reusing common expressions.

Algorithm 1 (*Optimized Encoding Algorithm*):

1. For $1 \le i \le (p-1)/2$, set $b_{i,p} \leftarrow 0$ and $C(i) \leftarrow \{0, 1, \cdots, p-1\}$;
2. For $1 \le j \le p-1$, set $b_{0,j} \leftarrow 0$ and $R(j) \leftarrow \{1, 2, \cdots, (p-1)/2\}$;
3. Set $A \leftarrow \{3, 4\}$ and $a \leftarrow 3$.
4. Let $a \leftarrow a + 4$;
5. If $a < p-1$, then let $A \leftarrow A \cup \{a, a+1\}$ and goto Step 4.
6. If $4 \nmid (p-3)$, then remove $p-2$ from A.
7. For $i = 1, 2, \cdots, (p-1)/2$ do
8. for $a \in A$ do
9. $j \leftarrow \langle ia \rangle$;
10. $v \leftarrow b_{i,\langle j-i \rangle} \oplus b_{i,\langle j+i \rangle}$;
11. $b_{i,p} \leftarrow b_{i,p} \oplus v$;
12. remove $\langle j-i \rangle$ and $\langle j+i \rangle$ from $C(i)$;
13. $b_{0,j} \leftarrow b_{0,j} \oplus v$;
14. remove i from $R(j)$.
15. Let $b_{i,p} \leftarrow b_{i,p} \oplus \left(\bigoplus_{t \in C(i)} b_{i,t} \right)$.
16. For $j = 1, 2, \cdots, p-1$ do
17. $b_{0,j} \leftarrow b_{0,j} \oplus \left(\bigoplus_{t \in R(j)} (b_{t,\langle j-t \rangle} \oplus b_{t,\langle j+t \rangle}) \right)$.

C. Decoding of RΛ-Code

Since RΛ-Code is derived from the XI-Code, naturally the decoding algorithms of the latter can be adjusted and then applied to the former. Note that by utilizing the symmetry of the extended RΛ-Code, the decoding algorithms of XI-Code can be simplified while being applied to RΛ-Code. Moreover, as with encoding, the decoding complexity of RΛ-Code can be reduced by reusing certain common expressions. Before discussing the specific decoding algorithms, we first give a proposition and two theorems that have been proven in [5].

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 $\langle x\delta \rangle + \langle y\delta \rangle = p$.

Theorem 2: For $x_i, y_i \in \mathrm{GF}(2)$, $i = 1, 2, \cdots, p-1$, 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

$$x_i \oplus y_i = a_i, i \ne p-d_1, p-d_2 \quad (6)$$

$$x_{\langle i-d_1 \rangle} \oplus y_{\langle i-d_2 \rangle} = c_i, i \ne d_1, d_2 \quad (7)$$

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

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

$$x_i \oplus x_{\langle i+\delta \rangle} = a_i, i \ne p-\delta, \langle p - \delta/2 \rangle \quad (8)$$

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

1) 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 $S_i^{(0)}$ and $S_j^{(\Lambda)}$ to denote the ith row syndrome and the jth Λ syndrome respectively, where $1 \le i \le (p-1)/2$ and $1 \le j \le p-1$. Set the three erased columns to be zeros, then according to (4) and (5), the row and Λ syndromes can be calculated as follows:

$$S_i^{(0)} = \bigoplus_{t=0}^{p} b_{i,t} \quad (9)$$

$$S_j^{(\Lambda)} = b_{0,j} \oplus \left(\bigoplus_{t=1}^{(p-1)/2} (b_{t,\langle j-t \rangle} \oplus b_{t,\langle j+t \rangle}) \right) \quad (10)$$

Note that all the 0-bits (including the erased bits) do not really participate in the calculations. As with encoding, common expressions can be first evaluated and reused during the syndrome calculations. The specific procedure is quite similar to the encoding algorithm, thus we do not give the pseudocode again. It should be noted that, the available common expressions during syndrome calculations are fewer than those of the encoding procedure, since the common expressions involving erased bits are useless.

Now let us consider the extended RΛ-Code of size $(p+1) \times (p+1)$. As with [5], we use $S_j^{(1)}$ and $S_j^{(-1)}$ to denote the jth diagonal and anti-diagonal syndromes respectively, where $1 \le j \le p-1$. Then according to the symmetry of the extended RΛ-Code, we can let $S_i^{(0)} \leftarrow S_{p-i}^{(0)}$ for $(p+1)/2 \le i \le p$ and let $S_j^{(1)} \leftarrow S_j^{(\Lambda)}$ and $S_j^{(-1)} \leftarrow S_{p-j}^{(\Lambda)}$ for $1 \le j \le p-1$, where $\leftarrow$ is an assignment operator. After obtaining all the syndromes, we can recover the three erased columns using the Algorithm 1 described in [5]. However, owing to the symmetry of the extended RΛ-Code, some steps of the decoding procedure are unnecessary or redundant. To make the difference clear, we describe the simplified decoding algorithm in the following. To facilitate the description, we let $d_1 = m - l$, $d_2 = r - l$, and $\delta = d_2 - d_1 = r - m$.

Algorithm 2 (All-Erasure Decoding): First, according to (1) and the definition of $S_i^{(0)}$, we have

$$b_{i,l} \oplus b_{i,m} \oplus b_{i,r} = S_i^{(0)}, 1 \le i \le p-1 \quad (11)$$

Next, we distinguish between the following two cases:

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

In this case, from (2), (3) and the definitions of diagonal and anti-diagonal syndromes, we have

$$b_{i,l} \oplus b_{\langle i-d_1 \rangle,m} = S_{\langle i+l \rangle}^{(1)} \quad (12)$$

$$b_{i,l} \oplus b_{\langle i+d_1 \rangle,m} = S_{\langle i-l \rangle}^{(-1)} \quad (13)$$

where $0 \le i \le p-1$. The above equations are simpler than that of [5], since $b_{p,j} = b_{0,j}$ holds in the extended RΛ-Code.

According to the proof of Lemma 1 in [5], the erased bits in the lth and mth columns can be rearranged as

$$b_{p,l}, b_{d_1,m}, b_{\langle 2d_1\rangle,l}, b_{\langle 3d_1\rangle,m}, \cdots, b_{\langle (p-1)d_1\rangle,l}, b_{p,m} \quad (\text{Seq}.A)$$
$$b_{0,m}, b_{d_1,l}, b_{\langle 2d_1\rangle,m}, b_{\langle 3d_1\rangle,l}, \cdots, b_{\langle (p-1)d_1\rangle,m}, b_{0,l} \quad (\text{Seq}.B)$$

Every two adjcent elements are associated either by (12) or by (13), and each sequence contains two imaginary 0s. Therefore, the erased bits in the lth and mth columns can be easily retrieved by using (12) and (13) alternately. It should be noted that, for RΛ-Code, $b_{\langle id_1\rangle,m} = b_{\langle (p-i)d_1\rangle,m}$ and $b_{\langle id_1\rangle,l} = b_{\langle (p-i)d_1\rangle,l}$ hold since $\langle id_1\rangle + \langle (p-i)d_1\rangle = p$ (according to Proposition 1). Consequently, using either of the two sequences is sufficient to recover all the erased bits in the lth and mth columns. Finally, the row parity bits can be calculated as $b_{i,p} = b_{i,l} \oplus b_{i,m} \oplus S_i^{(0)}$, where $1 \le i \le (p-1)/2$.

Case 2: The row parity column is available.

In this case, from (2), (3) and the definitions of diagonal and anti-diagonal syndromes, we have

$$b_{i,l} \oplus b_{\langle i-d_1\rangle,m} \oplus b_{\langle i-d_2\rangle,r} = S^{(1)}_{\langle i+l\rangle}, i \ne d_1, d_2 \quad (14)$$
$$b_{i,l} \oplus b_{\langle i+d_1\rangle,m} \oplus b_{\langle i+d_2\rangle,r} = S^{(-1)}_{\langle i-l\rangle}, i \ne p-d_1, p-d_2 \quad (15)$$

where $1 \le i \le p-1$. Let $M_i = b_{i,m} \oplus b_{\langle i+d_1\rangle,m}$ and $R_i = b_{i,r} \oplus b_{\langle i+d_2\rangle,r}$, then adding (11) to (14) results in

$$M_{\langle i-d_1\rangle} \oplus R_{\langle i-d_2\rangle} = S_i^{(0)} \oplus S^{(1)}_{\langle i+l\rangle}, i \ne d_1, d_2 \quad (16)$$

According to the symmetry of the extended RΛ-Code, we have $b_{i,m} = b_{p-i,m}$ and $b_{\langle i-d_1\rangle,m} = b_{\langle p-i+d_1\rangle,m}$, thus $M_{\langle i-d_1\rangle} = M_{p-i}$. Similarly, we have $R_{\langle i-d_2\rangle} = R_{p-i}$, $S_i^{(0)} = S_{p-i}^{(0)}$ and $S^{(1)}_{\langle i+l\rangle} = S^{(1)}_{\langle p-i-l\rangle}$. Then, from (16) we can get

$$M_{p-i} \oplus R_{p-i} = S_{p-i}^{(0)} \oplus S^{(1)}_{\langle p-i-l\rangle}, i \ne d_1, d_2 \quad (17)$$

which is equivalent to

$$M_i \oplus R_i = S_i^{(0)} \oplus S^{(-1)}_{\langle i-l\rangle}, i \ne p-d_1, p-d_2 \quad (18)$$

From the symmetry of the extended RΛ-Code, it can be immediately determined that $M_{\langle (p-d_1)/2\rangle} = b_{\langle (p-d_1)/2\rangle,m} \oplus b_{\langle (p+d_1)/2\rangle,m} = 0$ and $R_{\langle (p-d_2)/2\rangle} = b_{\langle (p-d_2)/2\rangle,r} \oplus b_{\langle (p+d_2)/2\rangle,r} = 0$. Then according to Theorem 2, we can easily evaluate M_i and R_i (except M_{p-d_1} and R_{p-d_2}) for $1 \le i \le p-1$ from (16) and (18). Again, differing from XI-Code, we only need to evaluate half of the M_is and R_is due to the symmetry.

Next, according to Theorem 3 and the definitions of M_i and R_i, we can further retrieve all the data bits in the mth and rth columns. As with the last step, only half of the erased data bits need to be evaluated. Take the mth column for instance, the erased bits can be rearranged as $b_{d_1,m}$, $b_{\langle 2d_1\rangle,m}$, $\cdots$, $b_{\langle (p-1)d_1/2\rangle,m}$, $b_{\langle (p+1)d_1/2\rangle,m}$, $\cdots$, $b_{\langle (p-2)d_1\rangle,m}$, $b_{\langle (p-1)d_1\rangle,m}$, where every two adjcent elements are associated by an M_i. Observe that $b_{\langle id_1\rangle,m} = b_{\langle (p-i)d_1\rangle,m}$, thus only the erased bits $b_{d_1,m}$, $b_{\langle 2d_1\rangle,m}$, $\cdots$, $b_{\langle (p-1)d_1/2\rangle,m}$ need to be evaluated.

Finally, retrieve the data bits in the lth column using (11), and then recalculate the missing parity bits according to (2), (3) and the definitions of diagonal and anti-diagonal syndromes, e.g., $b_{0,m} = S_m^{(1)} \oplus b_{d_1,l} \oplus b_{p-\delta,r}$. □

For equidistant erasures, there is an easier way to decode. The interested readers are referred to [5] (Algorithm 2) for more details.

2) Correcting One Erasure Combined with One Error: Suppose that the lth column is erased and the fth column is in error, where $0 \le l, m \le p$. For a possibly corrupted codeword, we first set the bits in the erased column to be zeros, then obtain the row syndromes $S_i^{(0)}$, diagonal syndromes $S_i^{(1)}$ and anti-diagonal syndromes $S_i^{(-1)}$ as with the last subsection. Observe that $b_{i,j}$ is an imaginary 0-bit if $\langle i+j\rangle = 0$ or $\langle i-j\rangle = 0$, thus we also have $S_0^{(1)} = S_p^{(-1)} = 0$. Once all the syndromes are obtained, the corrupted codeword can be corrected by using the Algorithm 3 described in [5].

It is worth mentioning that, in this case most of the common expressions can be reused during the syndrome calculations, since there is at most one erased column. Therefore, even though RΛ-Code and XI-Code use the same decoding algorithm in this error pattern, the computational complexity of the former is much lower than that of the latter.

III. Complexity Analysis and Comparison with XI-Code

To demonstrate the advantages of RΛ-Code over XI-Code, in this section we analyze and compare their encoding and decoding complexities.

A. On the Lower Bound of Encoding/Decoding Complexity

In array codes, the encoding (decoding) complexity is usually defined as the average number of XOR's required for computing (recovering) a parity (erased) bit. And previous works [1][7] usually assume that the lower bound of encoding/decoding complexity for MDS array codes is $k-1$, where k is the dimension of the code. So far, to the best of our knowledge, the lower bound $k-1$ is applicable to all the MDS codes in the literature. However, as we will see below, both the encoding and decoding complexities of RΛ-Code are lower than the above lower bound. Therefore, we will only compare the encoding/decoding complexity of RΛ-Code with that of XI-Code and the currently established "lower bound".

B. Encoding Complexity of RΛ-Code

Let n denote the code length of RΛ-Code, then $n \in \{p+1, p\}$. From the encoding rules of RΛ-Code, each parity bit is constructed from exactly $n-3$ data bits, thus requires $n-4$ XORs to obtain. However, the above calculation does not take into consideration the fact that the reuse of common expressions can save some XOR's. In general, reusing one common expression saves one XOR operation. With the optimized encoding algorithm (Algorithm 1), the standard RΛ-Code can use $(p-3)(p-1)/4$ common expressions regardless of the fact that $4|(p-3)$ or not, hence always saves $(p-3)/6$ XOR's per parity bit. For the shortened RΛ-Code, the number of common expressions that can be used *simultaneously* is also $(p-3)(p-1)/4$ if $4|(p-3)$, and is $(p-5)(p-1)/4$ if $4 \nmid (p-3)$.

We summarize the encoding complexity of RΛ-Code in Table I. It can be seen that, the encoding complexity of the standard (shortened) RΛ-Code is about 16.7 (up to 22.2) percent lower than the "lower bound".

Table I
THE ENCODING COMPLEXITIES OF RΛ-CODE AND XI-CODE

code length	"lower bound"	XI-Code	RΛ-Code	
			$4\|(p-3)$	$4\nmid(p-3)$
$p+1$	$p-3$	$p-3$	$5(p-3)/6$	
p	$p-4$	$p-4$	$p-4-\frac{p-3}{6}$	$p-4-\frac{p-5}{6}$

C. Decoding Complexity of RΛ-Code

The main differences between the RΛ-Code decoding and the XI-Code decoding lie in the following two aspects: the reuse of common expressions and the evaluation of M_i and R_i. Therefore, we can determine the decoding complexity of RΛ-Code based on that of XI-Code. Note that the effect of reusing common expressions depends on the specific erasure pattern. Suppose x columns of the first p columns are erased, then $x(p-1)/2$ common expressions will lose effectiveness *in the worst case*. Now let us consider the triple-erasure decoding *without* reusing common expressions. If the row parity column is one of the erasures, or the three erased columns are equidistant, then it requires $n-4$ XORs to recover each missing bit, where n is the code length. If three of the first p columns are erased and they are non-equidistant, then it requires about $n-10/3$ XORs on average to recover each missing bit. The decoding complexities of RΛ-Code and XI-Code are shown in Table II, noting that we only consider the most complicated and common erasure pattern, i.e., three non-equidistant columns of the first p columns are erased. Clearly, the decoding complexity of RΛ-Code is about 16.7 percent lower than the "lower bound" as $p \to \infty$.

Table II
THE DECODING COMPLEXITIES OF RΛ-CODE AND XI-CODE WHILE CORRECTING TRIPLE ERASURES

code length	"lower bound"	XI-Code	RΛ-Code	
			$4\|(p-3)$	$4\nmid(p-3)$
$p+1$	$p-3$	$p-\frac{4}{3}-\frac{4}{p-1}$	$p-\frac{p+5}{6}$	
p	$p-4$	$p-\frac{7}{3}-\frac{4}{p-1}$	$p-\frac{p+11}{6}$	$p-\frac{p+9}{6}$

For the error pattern where one column is erased and another column is in error, the cost of syndrome calculations is the *only difference* between the RΛ-Code decoding and the XI-Code decoding. Therefore, here we only compare the RΛ-Code decoding and the XI-Code decoding in terms of the number of XORs required to correct the corrupted codeword. Since they have different column sizes, for the sake of fairness, we will use the number of *column-wise* (rather than bit-wise) XORs in the comparison. Table III shows the number of column-wise XORs required to correct one erasure combined with a single error, noting that we only consider the most complicated and common case, i.e., neither the erased column nor the erroneous column is the row parity column. In this error pattern, the column-wise XOR cost of RΛ-Code is about 16.7 percent lower than that of XI-Code as $p \to \infty$.

Table III
THE NUMBER OF COLUMN-WISE XORS REQUIRED TO CORRECT ONE ERASURE COMBINED WITH A SINGLE ERROR, WHERE $\mathcal{C} = \frac{2}{p-1}$.

code length	XI-Code	RΛ-Code	
		$4\|(p-3)$	$4\nmid(p-3)$
$p+1$	$3p-4+\mathcal{C}$	$3p-\frac{p+3}{2}+\mathcal{C}$	
p	$3p-7+\mathcal{C}$	$3p-\frac{p+9}{2}+\mathcal{C}$	$3p-\frac{p+7}{2}+\mathcal{C}$

IV. CONCLUSION

We have presented a new class of lowest density MDS array codes of (column) distance 4, called RΛ-Code, which is derived from XI-Code (the most practical and efficient alternative before) and hence inherits the advantages of the latter. Compared with XI-Code, RΛ-Code has three important improvements: (a) the encoding complexity is about $16.7 \sim 22.2$ percent lower than that of XI-Code, (b) the decoding complexity is up to 16.7 percent lower than that of XI-Code, and (c) the column size is half that of XI-Code, meaning that the memory consumption during encoding/decoding is half that of XI-Code. Owing to these significant improvements, we believe that RΛ-Code will replace XI-Code as the most efficient lowest density MDS array codes of distance 4.

ACKNOWLEDGMENT

This work is supported in part by the National Key R&D Program of China under Grant 2016YFB1000302, the National Natural Science Foundation of China under Grants 61433019 and U1611261, the U.S. National Science Foundation under Grant CCF-1629625, and by the NetApp Inc. under Grant 1266803710.

REFERENCES

[1] M. Blaum. A family of MDS array codes with minimal number of encoding operations. In *the IEEE International Symposium on Information Theory*, pages 2784–2788, July 2006.
[2] Mario Blaum, Patrick G Farrell, van Tilborg, and C A Henk. Chapter on array codes. *Handbook of Coding Theory*, 2:1855–1909, 1998.
[3] Mario Blaum and Ron M Roth. On lowest density MDS codes. *IEEE Transactions on Information Theory*, 45(1):46–59, 1999.
[4] Y. Cassuto and J. Bruck. Cyclic lowest density MDS array codes. *IEEE Transactions on Information Theory*, 55(4):1721–1729, 2009.
[5] Z. Huang, H. Jiang, K. Zhou, C. Wang, and Y. Zhao. XI-Code: A family of practical lowest density MDS array codes of distance 4. *IEEE Transactions on Communications*, 64(7):2707–2718, July 2016.
[6] Zhijie Huang, Hong Jiang, Ke Zhou, Yuhong Zhao, and Chong Wang. Lowest density MDS array codes of distance 3. *IEEE Communications Letters*, 19(10):1670–1673, Oct 2015.
[7] Chao Jin, Hong Jiang, Dan Feng, and Lei Tian. P-Code: A new RAID-6 code with optimal properties. In *the 23rd ACM International Conference on Supercomputing (ICS'09)*, pages 360–369. ACM, 2009.
[8] Sheng Lin, Gang Wang, D. S. Stones, Jing Liu, and Xiaoguang Liu. T-Code: 3-erasure longest lowest-density MDS codes. *IEEE Journal on Selected Areas in Communications*, 28(2):289–296, 2010.
[9] E. Louidor and R. M. Roth. Lowest density MDS codes over extension alphabets. *IEEE Trans. on Info. Theory*, 52(7):3186–3197, July 2006.
[10] Lihao Xu, Vasken Bohossian, Jehoshua Bruck, and David G Wagner. Low-density MDS codes and factors of complete graphs. *IEEE Transactions on Information Theory*, 45(6):1817–1826, 1999.
[11] Lihao Xu and Jehoshua Bruck. X-code: MDS array codes with optimal encoding. *IEEE Transactions on Information Theory*, 45:272–276, 1999.