Summer 2014

S-Code: Lowest Density MDS Array Codes for RAID-6

Zhijie Huang
Huazhong University of Science and Technology, jayzy_huang@hust.edu.cn

Hong Jiang
University of Nebraska-Lincoln, jiang@cse.unl.edu

Ke Zhou
Huazhong University of Science and Technology, k.zhou@hust.edu.cn

Yuhong Zhao
Huazhong University of Science and Technology, yuhongzhao@hust.edu.cn

Chong Wang
Huazhong University of Science and Technology, C_Wang@hust.edu.cn

S-Code: Lowest Density MDS Array Codes for RAID-6

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

Abstract—RAID, a storage architecture designed to exploit I/O parallelism and provide data reliability, has been deployed widely in computing systems as a storage building block. In large scale storage systems, in particular, RAID-6 is gradually replacing RAID-5 as the dominant form of disk arrays due to its capability of tolerating concurrent failures of any two disks. MDS (maximum distance separable) array codes are the most popular erasure codes that can be used for implementing RAID-6, since they enable optimal storage efficiency and efficient encoding and decoding algorithms.

In this paper, we propose a new class of MDS array codes called S(ymmetry)-code, aiming to optimize every metric of coding. Specifically, S-code has the following properties: (a) optimality in encoding, decoding and update, (b) code length of either p or $p-1$ with p being a prime number, and (c) the least I/O cost for single-disk failure recovery among current representative RAID-6 codes. Our comprehensive evaluation shows that compared with other codes, S-code achieves the best trade-off among all the metrics of coding.

Index Terms—RAID-6, MDS array codes, reliability, storage system

1 Introduction

It is a known fact that RAID-5 [1] does not provide sufficient protection against data loss caused by either concurrent disk failures or a disk failure combined with unrecoverable sector errors in other disks. However, 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. Therefore, RAID-6 [2] has become increasingly popular due to its unique capability of tolerating both of these two types of failures, especially in large scale storage systems (e.g., cloud storage systems) that consist of a large number of less reliable (but more economical) disks such as SATA (vs. SCSI).

There are various erasure codes that can be used for implementing RAID-6, among which MDS (maximum distance separable) codes are most widely used due to their optimal storage efficiency. The original implementation of RAID-6 [3] used Reed-Solomon codes [4] that are *all-purpose* MDS erasure codes providing arbitrarily high fault tolerance. However, Reed-Solomon codes require specialized hardware to enable efficient computation of the finite field arithmetic on which the codes are based. Consequently, in order to implement RAID-6 efficiently, a series of MDS array codes that involve only XOR (exclusive-OR) computations were proposed in the past few years.

In MDS array codes, a codeword is a two-dimensional array containing both data and redundancy (parity) information. Logically, each column of the codeword corresponds to a disk of a RAID-6 system. According to the distribution of the parity information in a codeword, MDS array codes can be categorized into *horizontal codes*, such as EVENODD codes [5], RDP codes [6], and Liberation codes [7], and *vertical codes*, such as X-code [8], P-code [9], and HDP codes [10]. In horizontal codes, the parity information is placed in dedicated columns, while in vertical codes it is evenly distributed among almost all the columns. Both horizontal codes and vertical codes have their own advantages and disadvantages. In particular, horizontal codes are more flexible but suffer from unbalanced I/O, while vertical codes are more elegant but have more limitations in code length.

Although there are many codes that can be used to implement RAID-6, none of the existing codes is perfect. In particular, horizontal codes, in addition to unbalanced I/O, are unable to reach the lower bound of update complexity [11], factors that are not conducive to high performance and scalability. On the other hand, existing vertical codes have either too strict parameter limitations [8] or irregular geometrical constructions [9] in their codewords, making their implementation and deployment difficult and inflexible.

In this paper, we propose a new class of MDS

- *Z. Huang, C. Wang, K. Zhou and Y. Zhao are with the Wuhan National Laboratory for Optoelectronics, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China.*
  E-mail: {jayzy_huang, k.zhou, yuhongzhao, C_Wang}@hust.edu.cn.
- *H. Jiang is with the Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, NE 68588-0150.*
  E-mail: jiang@cse.unl.edu.

array codes called Symmetry-code (or S-code), which take all metrics of coding into consideration. S-code is a class of vertical codes that have the inherent capability to balance I/O load. In general, S-code has the following attractive properties:

- Optimal encoding, decoding and update complexity.
- The code length of p or $p-1$, where p is a prime number.
- Lower I/O cost of single erasure (disk failure) recovery than most of other representative codes.

We will in this paper present a detailed description of how to construct S-code and how to efficiently retrieve the lost data when no more than two columns (disks) are erased (failed). In addition, in order to protect the system from data loss caused by silent data corruptions, we will also provide an efficient algorithm for correcting a single column error.

The rest of this paper is organized as follows. In the next section we first briefly introduce the necessary background on RAID-6 and MDS array codes, then review the strengths and weaknesses of the representative existing codes, which serves as the key motivation for this work. Then, in Section 3, we describe the encoding procedure of our new codes and prove their MDS property. Section 4 presents the corresponding decoding procedure for two erasures and shows how to efficiently correct a single error. In addition, an efficient algorithm for reducing the I/O cost of the single-erasure recovery is also included. In Section 5, we compare S-code with the most representative and relevant codes in terms of computational complexity, code length limit, and I/O cost of the single-erasure recovery. Finally, in Section 6, we summarize the contributions of this work and remark on the directions of our future work.

2 RAID-6 AND MDS ARRAY CODES

In general, the term "RAID-6" refers to any form of RAID that can tolerate any two concurrent disk failures [2]. In order to implement RAID-6, we need to employ some kind of erasure codes to provide the required fault-tolerant capability. *MDS array codes* are the most popular erasure codes that can be used for implementing RAID-6, since they enable optimal storage efficiency and efficient encoding and decoding algorithms.

Array codes are a kind of erasure codes that involve only simple XOR and cyclic shift operations in both the encoding and decoding procedures, thus are much more efficient than the Reed-Solomon codes in terms of computation complexity [12]. *MDS array codes* refer to those array codes that provide a certain level of fault tolerance with the minimum storage overhead. In what follows we will focus on MDS array codes over $\mathrm{GF}(2)$ with distance 3, since they are the most representative RAID-6 codes. In these codes, each codeword is a two-dimensional array of binary bits, containing both data bits and parity bits. Each parity bit is calculated to be the even parity of a certain collection of the data bits, and the calculations must be such that any two erasures can be tolerated without data loss.

When implementing a RAID-6 system with above codes, every disk is first partitioned into a number of equal-size segments, called *strips*. Then, a *stripe* is defined as a maximal set of strips that are dependently related by an erasure code. Each disk contributes exactly one strip to a certain stripe. Finally, each strip is divided into a certain number of *elements*, of which each generally consists of one or more machine words. Now, a stripe appears as a two-dimensional array of elements. Actually, a stripe consists of an exact number of codewords that are interleaved for efficient computing [13]. For instance, if an element is defined to be a 32-bit machine word, then a stripe consists of 32 interleaved codewords. In this way, the XORs are performed on *machine words* rather than *bits*, improving the efficiency. Thus, the element size is restricted to be a multiple of the machine's word size. In the most common case, an element is implemented as a whole sector, which is the smallest unit of disk access.

2.1 Key Performance Metrics of Coding

According to the operation characteristics of RAID-6 systems, array codes can be evaluated in the following metrics:

- *Storage overhead* refers to the redundancy needed to provide A certain level of fault tolerance. Theoretically, in order to protect the system ag ainst the loss of any r disks, we need at least r redundant disks. In coding theory, this is known as the Singleton bound [14], and the codes that attain this bound are classified as the *Maximum Distance Separable (MDS)* codes.
- *Encoding complexity* refers to the number of XOR operations needed to compute the parity information. *For MDS array codes with distance* 3, *if the data bits occupy* k *columns in the codeword, then optimal encoding needs* $k-1$ *XOR operations per parity bit* [6].
- *Decoding complexity* refers to the number of XOR operations needed to reconstruct the lost information. As with encoding, *optimal decoding needs* $k-1$ *XOR operations per missing bit*.
- *Update complexity* refers to the average number of parity bits that must be updated when a data bit is modified. *The theoretical lower bound is* 2 *for two-erasure-correcting codes*.
- *Code length limit* refers to the restriction on the number of columns in the codeword.
- *I/O cost of the single-erasure recovery* refers to the amount of data that is required to reconstruct a single erasure.

Now let us briefly review the representative existing MDS array codes for RAID-6.

2.2 Horizontal Codes

There are many horizontal codes that can be applied to RAID-6. Nevertheless, the most popular and representative ones are known as follows.

EVENODD codes [5] are the first class of MDS array codes that are specially designed for RAID-6 and perform significantly better than all variants of the Reed-Solomon codes. In EVENODD codes, a codeword is a $(p-1)\times(p+2)$ array, where p is a prime number. When there are an arbitrary k data disks in the RAID-6 system, the prime p must be selected such that $p \geq k$, then assume that there are additional $p-k$ imaginary disks that hold nothing but zeros in the system. This method is referred to as "code shortening" and can be used by any other horizontal codes. EVENODD codes need roughly $k-1/2$ XOR operations per parity bit for encoding, and roughly k XOR operations per erased bit for decoding, which are both slightly higher than the theoretical optimums. However, they need to update roughly three parity bits per modified data bit, which is 1.5 times over the theoretical optimum 2.

RDP Codes [6] are a class of MDS array codes that are quite similar to EVENODD codes but have better performance. In RDP codes, a codeword is a $(p-1)\times(p+1)$ array, where p is a prime number. When the number of data disks k conform to $k=p-1$ or $k=p-2$, then RDP codes achieve optimal performance in both encoding and decoding. Otherwise, RDP codes are not optimal but still outperform EVENODD codes. However, like EVENODD codes, RDP codes need to update roughly three parity bits per modified data bit.

Liberation codes [7] are a class of lowest-density horizontal MDS array codes, whose parity check matrices are systematic and have the minimal number of 1s. Their update complexity is very close to the optimal value, i.e., two updated parity bits per modified data bit, achieving the lowest update complexity of all horizontal codes. In Liberation codes, a codeword is a $p\times(p+2)$ array, where p is a prime number. When there are k data disks in the RAID-6 system, Liberation codes need $k-1+\frac{k-1}{2p}$ XORs per parity bit for encoding, which is asymptotically optimal as $p\to\infty$. The main drawback of Liberation codes is their relatively poor decoding performance compared to EVENODD codes and RDP codes. Specifically, the decoding complexity of Liberation codes is about 15% higher than the optimal value.

2.3 Vertical Codes

Similarly, we only focus on the most representative vertical MDS array codes for RAID-6.

X-Code [8] are the most well-known vertical MDS array codes for RAID-6. In X-code, a codeword is a $p\times p$ array with the first $(p-2)$ rows containing data bits and the last two rows containing parity bits, where p is a prime number. The two rows of parity bits are calculated from the data bits along diagonals and anti-diagonals, respectively. X-code achieves the optimal encoding, decoding and update complexities. However, since every column contains two parity bits, X-code cannot be shortened, i.e., it requires the number of disks in the RAID-6 system to be a prime number. This strict limit renders X-code impractical in the production environment.

P-Code [9] are a class of vertical MDS array codes that are derived from B-Code [15] but have simpler construction and reconstruction algorithms. In P-code, a codeword is a $\frac{p-1}{2}\times p$ array with one column containing only data bits and the $(p-1)$ parity bits distributed evenly in other columns, where p is a prime number. Like X-code, P-code achieves the optimal encoding, decoding and update complexities. In addition, the code length, i.e., number of columns in the codeword, can be p or $p-1$, which makes P-code more practical than X-code. However, unlike X-code, the parity sets in P-code do not have regular geometric constructions. Thus, it needs lookup operations to determine which bits belong to a certain parity set in both encoding and decoding, which will degrade the real performance in the implementations. Furthermore, irregular geometric construction also makes it difficult to reduce the I/O cost of the single-erasure recovery.

HDP codes [10] are specially designed to optimize I/O load balancing. In HDP codes, a codeword is a $(p-1)\times(p-1)$ array with the parity bits placed in two diagonals of slopes 1 and -1 respectively, where p is a prime number. HDP codes have better load balancing than horizontal codes and P-code, and the authors claimed that they can reduce more I/O cost during the single-erasure recovery. However, since the anti-diagonal parity bits are involved in calculating the horizontal-diagonal parity bits, the encoding, decoding and update complexities of HDP codes are all suboptimal. In addition, the code length must be $p-1$, which makes HDP codes less appealing.

From the above, we can find that all of the existing codes have their own limitations — none of them can truly represent the *de facto* standard of RAID-6 codes. On the other hand, all of the above metrics of array codes for RAID-6 are important in the practical systems. In particular, storage overhead directly translates into the cost for fault tolerance, thus MDS codes are preferred. Encoding complexity represents the performance of full-stripe writes, while update complexity directly affects the performance of small writes, which are the dominant write operations in database systems and many big-data and data-intensive storage systems. Moreover, when applied to SSD arrays, update complexity can also affect the SSDs' service lifetime. Decoding complexity deter-

mines the performance of recovery and degraded reads. Note that the reconstruction time is inversely proportional to the system's availability. Code length limit implies the supported sizes of RAID-6 systems.

The metric of I/O cost of the single-erasure recovery has received a great deal of attention in recent years due to the emergence of large-scale distributed storage systems. Recent research reveals that, while erasure codes can tolerate multiple simultaneous failures, single failures represent 99.75 % of the actual recoveries [16]. Thus, the performance of the Single-erasure recovery, which is mainly determined by the I/O cost for reconstructing the lost information, is of paramount importance, especially in distributed environments. However, most of the existing RAID-6 codes fail to provide specially optimized decoding algorithm for single erasures. A few related solutions for EVENODD and RDP codes have been proposed recently [17] [18], but none for vertical codes.

As has been discussed above, it is clearly desirable to have RAID-6 codes that take all the performance metrics described in Section 2.1 into consideration. To this end, we present a new class of vertical MDS array codes, aiming to optimize every metric of coding. We describe the new codes and analyze their performance next.

3 SYMMETRIC-CODE (S-CODE)

In S-Code, a codeword is a $(p-1) \times p$ array of binary bits, where p is a prime number. Each codeword contains $(p-1) \times (p-2)$ data bits and $2(p-1)$ parity bits. Except for the first column, each column contains two parity bits. But instead of being placed in separate columns or rows, the parity bits of the S-Code are placed along two symmetric diagonals, as we will see below. Like the X-Code, parity bits are constructed from the data bits along several diagonals of certain slopes with the exclusive-or (XOR) operation. Notice that the first column does not contain any parity bits, thus the S-Code can be shortened by assuming that the first column is an imaginary column that holds nothing but zeros. In other words, the S-Code can be practically used in disk arrays with p or $p-1$ disks, where p is a prime number.

3.1 Encoding Procedure

Before formally describing the encoding procedure, we first define some notations. We use $b_{i,j}$ to denote the ith bit in the jth column, and let $\langle x \rangle = x \bmod p$. To facilitate the description, we also assume that there is an imaginary 0-row after the last row of the codeword. Then, the parity bits of the S-Code are constructed according to the following encoding rules:

$$b_{j-1,j} = \bigoplus_{\substack{t=0 \\ t \neq j}}^{p-1} b_{\langle 2j-1-t \rangle, t} \quad (1)$$

$$b_{p-1-j,j} = \bigoplus_{\substack{t=0 \\ t \neq j}}^{p-1} b_{\langle p-1-2j+t \rangle, t} \quad (2)$$

where $j = 1, 2, \cdots, p-1$.

Geometrically speaking, the parity bits are divided into two groups that are placed along two diagonals of slopes -1 and 1, called diagonal parity-bit group and anti-diagonal parity-bit group respectively. Specifically, the first group of parity bits are placed along the diagonal $\{(i,j)|j-i=1, j=1,2,\cdots,p-1\}$, and each parity bit is calculated to be the even parity of the data bits along the diagonal that traverses the parity bit itself and has a slope of 1. Similarly, the second group of parity bits are placed along the diagonal $\{(i,j)|j+i=p-1, j=1,2,\cdots,p-1\}$, and each parity bit is calculated to be the even parity of the data bits along the diagonal that traverses the parity bit itself and has a slope of -1. As an example, Fig. 1 shows the encoding rules of the S-code with $p=7$.

	D0	D1	D2	D3	D4	D5	D6
0	0a	**1**	2f	3e	4d	5c	**b**
1	1b	2a	**3**	4f	5e	**d**	0c
2	2c	3b	4a	**5**	**f**	0e	1d
3	3d	4c	5b	**a**	**0**	1f	2e
4	4e	5d	**c**	0b	1a	**2**	3f
5	5f	**e**	0d	1c	2b	3a	**4**
6		0f	1e	2d	3c	4b	5a

Fig. 1. Encoding rules of the S-code with $p=7$, where a parity bit in the diagonal parity group (or the anti-diagonal parity group) is labeled by a single numeric letter, e.g., "1", (or a single alphabet letter, e.g., "a"); while a data bit is labeled by a concatenated numeric and alphabet letter, e.g., "1a", to indicate the particular diagonal parity group (e.g., "1") and anti-diagonal parity group it belongs to.

From the construction of the S-code, it is easy to see that all the parity bits are obtained independently, i.e., each 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 two parity bits, which are placed in different parity diagonals. This implies that updating one data bit results in updating exactly two parity bits. In other words, *S-code achieves the lower bound* 2 *of the update complexity for any codes of distance* 3. In addition, notice that the last row is just an imaginary all-0-row, thus each parity bit is constructed from only $p-2$ data bits in practice. Therefore, *S-code requires only* $p-3$ *XORs per parity bit in encoding, which achieves the lower bound*

of the encoding complexity for any codes of distance 3 with code length p.

3.2 The MDS Property

To prove the MDS property of the S-code, we first provide two Lemmas that will be used in the proof.

Lemma 1: In the sequence of numbers $\{(p-1+k\delta) \bmod p, k = 0, 1, \cdots, p\}$, with p being prime and $0 < \delta < p$, the endpoints are both equal to $p-1$, and all numbers $0, 1, \cdots, p-2$ occur exactly once in the sequence [6].

Lemma 2: Let x and y be the m-th and n-th numbers in the sequence $\{(p-1+k\delta) \bmod p, k = 0, 1, \cdots, p\}$, with p being prime and $0 < \delta < p$, if $0 \le x, y \le p-2$ and $x + y = p-2$, then $m + n = p$.

Proof: From $x \equiv p-1+m\delta (\bmod p)$ and $y \equiv p-1+n\delta (\bmod p)$, we have $x + y \equiv 2p-2+(m+n)\delta \equiv (m+n)\delta - 2 (\bmod p)$. If $x+y = p-2$, the above equation can be reduced to $p-2 \equiv (m+n)\delta - 2 (\bmod p)$, i.e., $(m+n)\delta \equiv 0 (\bmod p)$. Since p is a prime number and $0 < \delta < p$, we have $m + n \equiv 0 (\bmod p)$. On the other hand, from $0 \le x, y \le p-2$ and Lemma 1, we have $1 \le m, n \le p-1$, thus $2 \le m+n \le 2p-2$. From the above, it can be easily deduced that $m + n = p$. □

Theorem 1: For any odd prime p, the S-code has column distance of 3, i.e., it is MDS.

Proof: Observe that the S-code is a linear code, thus its column distance is equal to its minimum column weight. Thus we only need to prove that the code has a minimum column weight of 3, i.e., a valid codeword of the S-code has at least three nonzero columns. Now we prove it by contradiction.

First, from the construction of the S-code, each parity bit is obtained along a certain diagonal, so it is clearly impossible to have only one nonzero column.

Now suppose that there is a valid codeword that contains exactly two nonzero columns, then there are two possible cases:

a) The nonzero columns are the 0th and lth columns, where $1 \le l \le p-1$. According to Lemma 1, $b_{0,0}, b_{1,0}, \cdots, b_{p-2,0}$ occur exactly once in the sequence $b_{p-1,0}$, $b_{\langle p-1+l\rangle,0}$, $b_{\langle p-1+2l\rangle,0}$, $\cdots$, $b_{\langle p-1+(p-1)l\rangle,0}$, $b_{\langle p-1+p\cdot l\rangle,0}$. Similarly, $b_{0,l}, b_{1,l}, \cdots, b_{p-2,l}$ occur exactly once in the sequence $b_{p-1,l}, b_{\langle p-1+l\rangle,l}, b_{\langle p-1+2l\rangle,l}, \cdots$, $b_{\langle p-1+(p-1)l\rangle,l}$, $b_{\langle p-1+p\cdot l\rangle,l}$. From the construction of the S-code, the 0th column is an all-data column and the lth column contains two parity bits $b_{l-1,l}$ and $b_{p-1-l,l}$. Note that $\langle p-1+l\rangle = l-1$ and $\langle p-1+(p-1)l\rangle = p-1-l$, thus the two parity bits in the corresponding sequence are $b_{\langle p-1+l\rangle,l}$ and $b_{\langle p-1+(p-1)l\rangle,l}$, respectively. Observe that $b_{\langle p-1+l\rangle,0}$ and $b_{p-1,l}$ are in the same diagonal of slope 1, and that $b_{p-1,l} = 0$, we have $b_{\langle p-1+l\rangle,0} = 0$. Since $b_{\langle p-1+l\rangle,0}$ and $b_{\langle p-1+2l\rangle,l}$ are in the same diagonal of slope -1, we can further deduce that $b_{\langle p-1+2l\rangle,l} = 0$. By parity of reasoning, we have $b_{\langle p-1+l\rangle,0} = b_{\langle p-1+2l\rangle,l} = \cdots = b_{\langle p-1+(p-2)l\rangle,0} = b_{\langle p-1+(p-1)l\rangle,l} = 0$. This zigzag recursion stops at $b_{\langle p-1+(p-1)l\rangle,l}$, which is a parity bit that only lies in one diagonal. Similarly, since $b_{p-1-l,0}$ and $b_{p-1,l}$ are in the same diagonal of slope -1, we have $b_{p-1-l,0} = 0$, i.e., $b_{\langle p-1+(p-1)l\rangle,0} = 0$. Then, by a similar deduction, we can get $b_{\langle p-1+(p-1)l\rangle,0} = b_{\langle p-1+(p-2)l\rangle,l} = \cdots = b_{\langle p-1+2l\rangle,0} = b_{\langle p-1+l\rangle,l} = 0$. From the above, we have $b_{\langle p-1+l\rangle,0} = b_{\langle p-1+2l\rangle,0} = \cdots = b_{\langle p-1+(p-2)l\rangle,0} = b_{\langle p-1+(p-1)l\rangle,0} = 0$ and $b_{\langle p-1+l\rangle,l} = b_{\langle p-1+2l\rangle,l} = \cdots = b_{\langle p-1+(p-2)l\rangle,l} = b_{\langle p-1+(p-1)l\rangle,l} = 0$, i.e., $b_{0,0} = b_{1,0} = \cdots = b_{p-2,0} = 0$ and $b_{0,l} = b_{1,l} = \cdots = b_{p-2,l} = 0$. This contradicts the assumption.

b) The nonzero columns are the lth and rth columns, where $1 \le l < r \le p-1$. Let $\delta = r - l$, obviously $1 \le \delta \le p-2$. According to Lemma 1, $b_{0,l}, b_{1,l}, \cdots, b_{p-2,l}$ occur exactly once in the sequence $b_{p-1,l}, b_{\langle p-1+\delta\rangle,l}, b_{\langle p-1+2\delta\rangle,l}$, $\cdots, b_{\langle p-1+(p-1)\delta\rangle,l}$, $b_{\langle p-1+p\cdot\delta\rangle,l}$ (Seq. 1), and $b_{0,r}, b_{1,r}, \cdots, b_{p-2,r}$ occur exactly once in the sequence $b_{p-1,r}, b_{\langle p-1+\delta\rangle,r}, b_{\langle p-1+2\delta\rangle,r}$, $\cdots, b_{\langle p-1+(p-1)\delta\rangle,r}$, $b_{\langle p-1+p\cdot\delta\rangle,r}$ (Seq. 2). The lth column contains two parity bits $b_{l-1,l}$ and $b_{p-1-l,l}$, and the rth column contains two parity bits $b_{r-1,r}$ and $b_{p-1-r,r}$. According to Lemma 1, there must be m and n such that $\langle p-1+m\delta\rangle = l-1$ and $\langle p-1+n\delta\rangle = p-1-l$. Since $l-1+(p-1-l) = p-2$, according to Lemma 2, we have $m+n = p$. Additionally, notice that $l-1+\delta = r-1$ and $p-1-l-\delta = p-1-r$, we have $\langle p-1+(m+1)\delta\rangle = r-1$ and $\langle p-1+(n-1)\delta\rangle = p-1-r$. Now consider the sequence $b_{p-1,l}$, $b_{\langle p-1+\delta\rangle,r}$, $b_{\langle p-1+2\delta\rangle,l}$, $b_{\langle p-1+3\delta\rangle,r}$, $\cdots$, $b_{\langle p-1+(p-1)\delta\rangle,l}$, $b_{\langle p-1+p\cdot\delta\rangle,r}$ (Seq. 3). If m is odd, then n must be even (since $m+n = p$), thus $m+1$ is even and $n-1$ is odd. Therefore, in Seq. 3 there are exactly two parity bits: $b_{\langle p-1+n\delta\rangle,l}$ and $b_{\langle p-1+(n-1)\delta\rangle,r}$. Then, observe that $b_{p-1,l}$ and $b_{\langle p-1+\delta\rangle,r}$ are in the same diagonal of slope -1 and $b_{p-1,l} = 0$, we have $b_{\langle p-1+\delta\rangle,r} = 0$. If $b_{\langle p-1+\delta\rangle,r}$ is not a parity bit, i.e., $n \ne 2$, then $b_{\langle p-1+\delta\rangle,r}$ and $b_{\langle p-1+2\delta\rangle,l}$ are in the same diagonal of slope 1, thus we can further deduce that $b_{\langle p-1+2\delta\rangle,l} = 0$. By parity of reasoning, we have $b_{p-1,l} = b_{\langle p-1+\delta\rangle,r} = \cdots = b_{\langle p-1+(n-1)\delta\rangle,r} = 0$. This zigzag recursion stops at the first parity bit in Seq. 3, since a parity bit only lies in one diagonal. Similarly, starting from $b_{\langle p-1+p\cdot\delta\rangle,r} = 0$, we have $b_{\langle p-1+p\cdot\delta\rangle,r} = b_{\langle p-1+(p-1)\delta\rangle,l} = \cdots = b_{\langle p-1+n\delta\rangle,l} = 0$. Since $b_{\langle p-1+(n-1)\delta\rangle,r}$ and $b_{\langle p-1+n\delta\rangle,l}$ are adjacent elements in Seq. 3, we have $b_{p-1,l} = b_{\langle p-1+\delta\rangle,r} = b_{\langle p-1+2\delta\rangle,l} = \cdots = b_{\langle p-1+(p-1)\delta\rangle,l} = b_{\langle p-1+p\cdot\delta\rangle,r} = 0$. If m is even, then n and $m+1$ must be odd, thus $n-1$ is even. Therefore, Seq. 3 contains exactly two parity bits: $b_{\langle p-1+m\delta\rangle,l}$ and $b_{\langle p-1+(m+1)\delta\rangle,r}$. Similarly,

we can easily deduce that every element of Seq. 3 equals zero. Next, let us consider the sequence $b_{p-1,r}, b_{\langle p-1+\delta\rangle,l}, b_{\langle p-1+2\delta\rangle,r}, b_{\langle p-1+3\delta\rangle,l}, \cdots, b_{\langle p-1+(p-1)\delta\rangle,r}, b_{\langle p-1+p\cdot\delta\rangle,l}$ (Seq. 4). According to the symmetry of the S-code, it can be easily deduced that every element of Seq. 4 also equals zero. From the definitions of Seq. 3 and Seq. 4, we actually have proved that all the elements in Seq. 1 and Seq. 2 are zeros, i.e., $b_{0,l} = b_{1,l} = \cdots = b_{p-2,l} = 0$ and $b_{0,r} = b_{1,r} = \cdots = b_{p-2,r} = 0$. Again, this contradicts the assumption.

From the above, the minimum column weight of the S-code is at least 3. But it is easy to see there is a codeword of column weight 3, thus the column distance of S-code is 3. □

4 Efficient Decoding Algorithms

Since S-code has a column distance of 3, any codeword with two erased columns or one error column can be corrected. In this section, we present decoding algorithms of S-code for correcting two erasures or one error. In addition, we provide an efficient decoding algorithm for correcting one erasure, which is able to greatly reduce the I/O overhead during recovery.

4.1 Correcting Two Erasures

In the proof of Theorem 1, when showing that the claim of exactly two nonzero columns in a codeword is contradictory, we demonstrate that we can always deduce that all the bits in the two columns are zeros, starting from their imaginary 0-bit(s). Specifically, we can always find a diagonal of one of the two slopes that traverses one of the two columns at its imaginary 0-bit, thus the bit at which this diagonal traverses the other column must be zero. Using the diagonal of the other slope crossing this zero bit, we can determine that the crossed bit by the diagonal in the other column must also be zero. In this way, the zigzag recursive procedure can proceed until it hits a parity bit at one of the two columns, since a parity bit only lies in one diagonal. We call this zigzag recursion a *decoding chain*, since it indicates the reconstruction sequence of the missing bits in decoding. Given the positions of the two nonzero columns, we can determine the values of all the bits in the two columns, traversing along several such decoding chains. Obviously, if one of the two nonzero columns is the 0th column, then we have two decoding chains since there are two parity bits in the two nonzero columns. Otherwise, we have four decoding chains since there are four parity bits in the two nonzero columns. From the above, we can find that the proof of Theorem 1 also provides an efficient erasure-decoding algorithm.

To concretely describe the two-erasure decoding algorithm, let us look into (1) and (2). Obviously, the parity groups in them are along diagonals $\{(x,y)|x+y \equiv 2j-1 (\mathrm{mod}\, p)\}$ and $\{(x,y)|x-y \equiv p-1-2j (\mathrm{mod}\, p)\}$ respectively, where $j = 1, 2, \cdots, p-1$. Since $2j-1 \equiv p-1+2j (\mathrm{mod}\, p)$ and $p-1-2j \equiv p-1+(p-2)j (\mathrm{mod}\, p)$, according to Lemma 1, both $\langle 2j-1\rangle$ and $\langle p-1-2j\rangle$ traverse all the numbers $0, 1, \cdots, p-2$. Thus, (1) and (2) are equivalent to the following equations:

$$\bigoplus_{t=0}^{p-1} b_{\langle u-t\rangle,t} = 0 \tag{3}$$

$$\bigoplus_{t=0}^{p-1} b_{\langle u+t\rangle,t} = 0 \tag{4}$$

where $u = 0, 1, \cdots, p-2$. If two columns of a codeword are erased, then there are $2(p-1)$ unknown bits in the codeword. Correcting the two erasures is actually equivalent to the problem of solving $2(p-1)$ unknowns from the $2(p-1)$ linear equations in (3) and (4). Since S-code has a column distance of 3, the $2(p-1)$ linear equations must be linearly independent. From the construction of S-code, each equation has at most two unknown bits, with some having only one unknown bit. This property drastically reduces the decoding complexity. We present the formal decoding algorithm for correcting two erasures below, noting that the correctness of the algorithm can be easily deduced from the proof of Theorem 1.

Algorithm 1 (Two-Erasure Decoding Algorithm): Assume that the lth and rth columns have been erased, where $0 \le l < r \le p-1$. First, we calculate the diagonal syndromes $S^{(1)} = S_0^{(1)}, S_1^{(1)}, \cdots, S_{p-2}^{(1)}$, and the anti-diagonal syndromes $S^{(-1)} = S_0^{(-1)}, S_1^{(-1)}, \cdots, S_{p-2}^{(-1)}$ as follows:

$$S_u^{(1)} = \bigoplus_{\substack{t=0 \\ t\neq l,r}}^{p-1} b_{\langle u-t\rangle,t} \tag{5}$$

$$S_u^{(-1)} = \bigoplus_{\substack{t=0 \\ t\neq l,r}}^{p-1} b_{\langle u+t\rangle,t} \tag{6}$$

where $u = 0, 1, \cdots, p-2$. Then, from (3)–(6) we have

$$b_{\langle u-l\rangle,l} \oplus b_{\langle u-r\rangle,r} = S_u^{(1)} \tag{7}$$

$$b_{\langle u+l\rangle,l} \oplus b_{\langle u+r\rangle,r} = S_u^{(-1)} \tag{8}$$

where $u = 0, 1, \cdots, p-2$.

Next, according to the values of l and r, we have the following two cases:

- $l = 0$ and $1 \le r \le p-1$. In this case, the imaginary 0-bit $b_{p-1,r}$ is traversed by two diagonals of slopes 1 and -1 respectively. First, we need to find the two equations that are associated with the two diagonals, i.e., the two equations that contain $b_{p-1,r}$. Let $u-r \equiv p-1 (\mathrm{mod}\, p)$, we have $u = r-1$, i.e., the equation associated with the diagonal of slope 1 is $b_{r-1,0} \oplus b_{p-1,r} = S_{r-1}^{(1)}$. Similarly, let

$u + r \equiv p - 1 (\mathrm{mod}\, p)$, we have $u = p - 1 - r$, i.e., the equation associated with the diagonal of slope -1 is $b_{p-1-r,0} \oplus b_{p-1,r} = S^{(-1)}_{p-1-r}$. Since $b_{p-1,r}$ is just an imaginary 0-bit, the two equations actually have determined the values of $b_{r-1,0}$ and $b_{p-1-r,0}$. Once $b_{r-1,0}$ and $b_{p-1-r,0}$ are determined, all the other missing bits can be retrieved along two decoding chains as discussed before, using (7) and (8) alternately.

- $1 \leq l < r \leq p - 1$. In this case, there are four decoding chains as discussed before. First, observe that the imaginary 0-bit $b_{p-1,l}$ is traversed by two diagonals of slopes 1 and -1 respectively. Like the last case, we can deduce from this that $b_{p-1-(r-l),r} = S^{(1)}_{l-1}$ and $b_{r-l-1,r} = S^{(-1)}_{p-1-l}$. Since the imaginary 0-bit $b_{p-1,r}$ is also traversed by two diagonals of slopes 1 and -1 respectively, we can deduce that $b_{p-1-(r-l),l} = S^{(-1)}_{p-1-r}$ and $b_{r-l-1,l} = S^{(1)}_{r-1}$ in the same way. From these four starting points, we can retrieve all the other missing bits along four decoding chains, using (7) and (8) alternately in each decoding chain.

Note that whenever we retrieve a new missing bit, we need to check whether it is the endpoint of the corresponding decoding chain, i.e., whether it is a parity bit. According to the construction of S-code, this can be done as follows: if $j - i = 1$ or $j + i = p - 1$, then $b_{i,j}$ is a parity bit.

As an example, let us look into Fig. 1 again. Suppose that columns D2 and D3 are erased, then the decoding procedure is as follows. First, since $b_{6,2} = 0$ (labeled with "1e") belongs to diagonal parity group "1" and anti-diagonal parity group "e", we can retrieve $b_{5,3}$ and $b_{0,3}$ as: $b_{5,3} = b_{1,0} \oplus b_{0,1} \oplus b_{4,4} \oplus b_{3,5} \oplus b_{2,6}$, $b_{0,3} = b_{4,0} \oplus b_{5,1} \oplus b_{1,4} \oplus b_{2,5} \oplus b_{3,6}$. Similarly, from $b_{6,3} = 0$ we can retrieve $b_{0,2}$ and $b_{5,2}$ according to the corresponding diagonal and anti-diagonal parity constraints. Once $b_{5,3}$ is obtained, we can immediately retrieve $b_{4,2}$ since it is the sole unknown element in the anti-diagonal parity group "c". Since $b_{4,2}$ is a parity bit, the iteration stops here. Similarly, starting from $b_{0,3}$, $b_{0,2}$ and $b_{5,2}$ we can retrieve other lost bits along decoding chains $b_{0,3} \to b_{1,2}$, $b_{0,2} \to b_{1,3} \to b_{2,2} \to b_{3,3}$ and $b_{5,2} \to b_{4,3} \to b_{3,2} \to b_{2,3}$.

4.2 Locating and Correcting A Single Error

To correct a single error, the key is to locate the column in error. This can be done by analyzing the relationship between the error vector and the syndromes. For a possibly corrupted codeword, we first compute the diagonal syndromes $S^{(1)}_i$ and the anti-diagonal syndromes $S^{(-1)}_i$ as follows:

$$S^{(1)}_i = \bigoplus_{t=0}^{p-1} b_{\langle i-t \rangle, t} \tag{9}$$

$$S^{(-1)}_i = \bigoplus_{t=0}^{p-1} b_{\langle i+t \rangle, t} \tag{10}$$

where $i = 0, 1, \cdots, p-2$. Obviously, all the syndromes must be zeros if there is no error in the codeword. If there is one 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.

Suppose that the lth column is in error, with the error vector $\boldsymbol{e} = (e_0, e_1, \cdots, e_{p-2}, 0)^T$. Note that the $(p-1)$th row of the codeword is just an imaginary row of zeros, thus the last component of $\boldsymbol{e}$ is always 0. Then, according to (9) and (10) we have

$$S^{(1)}_i = e_{\langle i-l \rangle}$$
$$S^{(-1)}_i = e_{\langle i+l \rangle}$$

where $i = 0, 1, \cdots, p-2$. If $l = 0$, we have $S^{(1)}_i = e_i = S^{(-1)}_i$. Otherwise, let $\boldsymbol{v}(\downarrow n)$ (or $\boldsymbol{v}(\uparrow n)$) denote the vector derived from vector $\boldsymbol{v}$ by cyclically down- (or up-) shifting n positions, then $S^{(1)}_i$ (or $S^{(-1)}_i$) is equal to the ith component of $\boldsymbol{e}(\downarrow l)$ (or $\boldsymbol{e}(\uparrow l)$). Obviously, e_{p-1-l} (or e_{l-1}) does not exist in the diagonal (or anti-diagonal) syndromes. From the above, if we let

$$S^{(1)} = (S^{(1)}_0, S^{(1)}_1, \cdots, S^{(1)}_{p-2}, 0)^T$$
$$S^{(-1)} = (S^{(-1)}_0, S^{(-1)}_1, \cdots, S^{(-1)}_{p-2}, 0)^T$$

then $S^{(1)}(\uparrow l)$ (or $S^{(-1)}(\downarrow l)$) and $\boldsymbol{e}$ must only differ in the $(p-1-l)$th (or $(l-1)$th) component. Therefore, $S^{(1)}(\uparrow l)$ and $S^{(-1)}(\downarrow l)$ must only differ in the $(p-1-l)$th and $(l-1)$th components.

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

Algorithm 2 (One-Error Decoding Algorithm): First, we compute the diagonal syndrome vector $S^{(1)}$ and the anti-diagonal syndrome vector $S^{(-1)}$ from the possibly corrupted codeword, according to (9) and (10) respectively. If the two syndrome vectors are both all-zero vectors, then there is no error in the codeword. Otherwise, we have the following two cases:

- $S^{(1)} = S^{(-1)}$. In this case, the 0th column is in error, and the error vector is equal to $S^{(1)}$, i.e., $\boldsymbol{e} = S^{(1)}$.
- $S^{(1)} \neq S^{(-1)}$. In this case, we find the first index l to be $1 \leq l \leq p-1$, such that $S^{(1)}(\uparrow l)$ and $S^{(-1)}(\downarrow l)$ only differ in the $(p-1-l)$th and $(l-1)$th components. This index l corresponds to the location of the column in error. If there is no such l, then there may be more than one erroneous column in the codeword. Once we find l, we can obtain the error vector $\boldsymbol{e}$ as follows: first set $\boldsymbol{e} = S^{(1)}(\uparrow l)$, then set e_{p-1-l} to be the $(p-1-l)$th component of $S^{(-1)}(\downarrow l)$.

Once the error position and the error vector are determined, the final step is to add modulo-2 the first $p-1$ bits of $\boldsymbol{e}$ to the erroneous column of the corrupted codeword.

4.3 Optimal Recovery from A Single Erasure

If only one column is erased, then the lost parity bits (if any) can be retrieved according to the encoding rules, while every lost data bit can be retrieved using either the corresponding diagonal parity or the corresponding anti-diagonal parity. Obviously, if a missing bit is retrieved using the diagonal (or anti-diagonal) parity, then all the surviving bits along this diagonal (or anti-diagonal) need to be read. However, this does not mean that we need to read all $(p-2)\times(p-1)$ bits to reconstruct the erased column, since diagonals of different slopes necessarily intersect with each other in the codeword. In other words, there are some surviving bits that can be reused in the reconstruction. Obviously, the less surviving bits that are needed for reconstruction, the less reconstruction time will be needed. In addition, less I/O cost usually translates into less network traffic in distributed storage systems (e.g., distributed RAID).

Now the question is: how can we reconstruct the erased column with a minimum number of surviving bits? In fact, this can be done by elaborately choosing the parity type (i.e., diagonal/anti-diagonal parity) used in reconstruction for every missing bit.

Assume that m missing bits are retrieved using diagonal parities, and n missing bits are retrieved using anti-diagonal parities, where $m+n=p-1$. Obviously, there are $m\times n$ intersection points between the m diagonals and the n anti-diagonals. In order to obtain as many reusable surviving bits as possible in reconstruction, it is clearly desirable to have the maximum number of intersection points. As is well known, $m\times n\leq\left(\frac{m+n}{2}\right)^2=\left(\frac{p-1}{2}\right)^2$, and $m\times n$ attains the maximum value iff $m=n=\frac{p-1}{2}$. Thus, *it is best to reconstruct half of the missing bits using diagonal parities, and reconstruct the other half of the missing bits using anti-diagonal parities*.

However, not every intersection point above gives rise to a reusable surviving bit, since it may lie in the $(p-1)$th row, i.e., the imaginary 0-row. For example, in Fig. 1, the diagonal traversed $b_{2,2}$ and the anti-diagonal traversed $b_{3,2}$ intersect at $b_{6,5}$, which is just an imaginary 0-bit. Therefore, in order to fully utilize the intersection points, we need to try our best to avoid generating intersection points that lie in the $(p-1)$th row of the codeword. To facilitate the following discussion, we first present a useful theorem as follows.

Theorem 2: For the f-th column, where $0\leq f\leq p-1$, let $b_{x,y}$ be the intersection point of the diagonal that crosses $b_{i,f}$ and the anti-diagonal that crosses $b_{j,f}$, then

a) $b_{x,y}$ lies in the $(p-1)$th row, if $i+j=p-2$.
b) $b_{x,y}$ lies in the 0-th column, if $j-i\equiv 2f(\mathrm{mod}\,p)$.

Proof: The diagonal of slope 1 that crosses $b_{i,f}$ is $\{(x,y)|x+y\equiv i+f(\mathrm{mod}\,p)\}$, and the diagonal of slope -1 that crosses $b_{j,f}$ is $\{(x,y)|x-y\equiv j-f(\mathrm{mod}\,p)\}$. Since $b_{x,y}$ is the intersection point, the following equations hold:

$$x+y\equiv i+f\pmod p$$
$$x-y\equiv j-f\pmod p$$

For a), adding the two equations above, we have $2x\equiv i+j(\mathrm{mod}\,p)$, i.e., $2x\equiv p-2(\mathrm{mod}\,p)$. Since $p-2$ is an odd number, we have $x\equiv\frac{p-2}{2}\equiv\frac{p-2+p}{2}\equiv p-1(\mathrm{mod}\,p)$, i.e., $b_{x,y}$ lies in the $(p-1)$th row. For b), the difference of the two equations is $-2y\equiv j-i-2f(\mathrm{mod}\,p)$. If $j-i\equiv 2f(\mathrm{mod}\,p)$, we have $y\equiv 0(\mathrm{mod}\,p)$, i.e., $b_{x,y}$ lies in the 0-th column. □

According to Theorem 2, *it is best to reconstruct the ith missing bit and the $(p-2-i)$th missing bit of the erased column using the same type of parities*.

As is mentioned in Section 3, the code length of S-Code can be directly shortened to $p-1$ by assuming that the first column of the codeword is an imaginary all-0-column. In what follows we will refer to S-codes of length $p-1$ as *shortened S-codes*.

For shortened S-codes, since the 0-th column of the codeword is an imaginary 0-column, we should also reduce the number of intersection points that lie in the 0-th column as far as possible. Now suppose that the f-th column is erased, it is obvious that $1\leq f\leq p-1$. According to Lemma 1, the missing bits occur exactly once in the sequence $b_{\langle p-1+f\rangle,f},b_{\langle p-1+2f\rangle,f},\cdots,b_{\langle p-1+(p-1)f\rangle,f}$, which can be divided into $(p-1)/2$ pairs $\{(b_{\langle p-1+kf\rangle,f},b_{\langle p-1+(p-k)f\rangle,f})|k=1,3,\cdots,p-2\}$. Since $p-1+kf+p-1+(p-k)f\equiv 2p-2+pf\equiv p-2(\mathrm{mod}\,p)$, according to Theorem 2 it is best to reconstruct $b_{\langle p-1+kf\rangle,f}$ and $b_{\langle p-1+(p-k)f\rangle,f}$ using the same type of parities. Following this principle, we have another theorem.

Theorem 3: For two adjacent pairs of missing bits in $\{(b_{\langle p-1+kf\rangle,f},b_{\langle p-1+(p-k)f\rangle,f})|k=1,3,\cdots,p-2\}$, if we reconstruct them using different types of parities, then the corresponding four diagonals have at least one intersection point that lies in the 0-th column.

Proof: Suppose that the two adjacent pairs of missing bits are $(b_{\langle p-1+kf\rangle,f},b_{\langle p-1+(p-k)f\rangle,f})$ and $(b_{\langle p-1+(k+2)f\rangle,f},b_{\langle p-1+(p-k-2)f\rangle,f})$, where k is odd and $1\leq k\leq p-4$. First, if we reconstruct the first pair using diagonal parities and reconstruct the second pair using anti-diagonal parities, then according to Theorem 2, the intersection point of the diagonal that crosses $b_{\langle p-1+kf\rangle,f}$ and the anti-diagonal that crosses $b_{\langle p-1+(k+2)f\rangle,f}$ lies in the 0-th column. Similarly, if we reconstruct the first pair using anti-diagonal parities and reconstruct the second pair using diagonal parities, then the intersection point of the diagonal that crosses $b_{\langle p-1+(p-k-2)f\rangle,f}$ and the anti-diagonal that crosses $b_{\langle p-1+(p-k)f\rangle,f}$ lies in the 0-th column. □

According to the above theorem, in order to reduce as many intersection points that lie in the 0-th column as possible, *we should try to reconstruct every two adjacent pairs of missing bits using the same type of parities*.

From the above, a formal algorithm for efficiently reconstructing a single erasure can be described as follows:

Algorithm 3 (Single-Erasure Decoding Algorithm): Assume that the f-th column is erased, where $0 \leq f \leq p-1$. First, let $n = (p-1)/2$, then we distinguish the following two cases:

$f = 0$. In this case we first divide the missing bits into n pairs $\{(b_{i,f}, b_{p-2-i,f})|i = 0, 1, \cdots, n-1\}$. If n is even, then we reconstruct the first $n/2$ pairs using diagonal parities and reconstruct the other $n/2$ pairs using anti-diagonal parities. Otherwise, the following 3-step reconstruction procedure is used: 1) reconstruct the first pair of missing bits using different types of parities, 2) reconstruct the next $(n-1)/2$ pairs of missing bits using diagonal parities, and 3) reconstruct the remaining $(n-1)/2$ pairs of missing bits using anti-diagonal parities.

$1 \leq f \leq p-1$. In this case we first divide the missing bits into n pairs $\{(b_{\langle p-1+kf\rangle,f}, b_{\langle p-1+(p-k)f\rangle,f})|k = 1, 3, \cdots, p-2\}$. Obviously, the first pair is a pair of missing parity bits, which can be retrieved according to the encoding rules. If n is even, we reconstruct the $n-1$ pairs of missing data bits as follows: 1) reconstruct the first pair of missing data bits using different types of parities, 2) reconstruct the next $(n-2)/2$ pairs of missing data bits using diagonal parities, and 3) reconstruct the remaining $(n-2)/2$ pairs of missing data bits using anti-diagonal parities. Otherwise, $n-1$ must be even. Thus, we reconstruct the first $(n-1)/2$ pairs of missing data bits using diagonal parities and reconstruct the other $(n-1)/2$ pairs of missing data bits using anti-diagonal parities.

In the next section, we will show that this algorithm indeed can reconstruct a single erased column with a minimum number of surviving bits.

5 Performance Evaluation

To demonstrate the advantages of S-code, in this section we compare them with the most representative existing codes in every metric of coding described in Section 2.

5.1 Encoding, Decoding and Update Complexities

In S-code, each codeword contains $(p-1)\times(p-2)$ data bits, which requires a space of $p-2$ columns. Thus, we can assume that there are $p-2$ data columns in the codeword. In Section 3 we have shown that both the encoding complexity and update complexity of S-code are optimal. Now let us consider the decoding complexity of S-code. From the reconstruction procedure of double erasures, we can find that every missing bit is retrieved by XORing the other $p-1$ bits along the diagonal or anti-diagonal that traverses it. Since there is an imaginary 0-bit among the $p-1$ bits, we need only $p-3$ XOR operations for reconstructing each missing bit. Therefore, *the decoding complexity of S-code is also optimal*.

We have briefly presented the encoding, decoding and update complexities of the most representative existing codes in Section 2. It is clear that only X-code and P-code are also optimal in these three metrics, which leaves all the other codes non-optimal in at lease one of the three metrics and thus less competitive. Therefore, in the following we will focus on comparing S-code with the following arguably most competitive codes, namely, RDP codes that are considered the best horizontal codes, the three vertical codes of X-code, P-code and HDP.

5.2 Code Length Limit

Since there is a pure-data column in the codeword of S-code, we can shorten S-code by assuming that the pure-data column is just an imaginary column that holds nothing but zeros. In other words, the code length of S-code can be either p or $p-1$, where p is a prime number. This constraint with S-code is the same as that with P-code, but far looser than other vertical codes such as X-code, HDP codes, etc. It is worth mentioning that the S-code of length $p-1$ retains the optimality in all of the above three metrics, i.e., encoding, decoding and update complexities. Since the reasoning behind this claim is relatively trivial and similar to that of the standard S-code, we omit the details here for brevity.

Although still inferior to horizontal codes that have no limit on code length, the supported code lengths of S-code are sufficient in most cases. In particular, in order to restrain the I/O cost during reconstructions and provide sufficient reliability, the stripe size (corresponds to code length) in the practical RAID-6 systems is generally less than 20. And it is clear that in the range of 3 to 19 the code lengths of S-code cover all except for 8, 9, 14 and 15, amounting to a coverage of 76.5%. Moreover, we can also borrow the idea of the *code shortening* technique that is employed by horizontal codes to generalize vertical codes to support arbitrary code length. For more details about this issue, the interested readers may refer to [19]. However, this method will cause the generalized codes to lose their optimality in all of the above three metrics, thus it is considered a compromise for S-code to support RAID-6 systems of arbitrary size. Therefore, in the following we will focus on the standard S-codes (code length is p or $p-1$) rather than generalized ones.

5.3 I/O Cost of Single-Erasure Recovery

In order to normalize the I/O cost of the single-erasure recovery in codes of different code lengths, we introduce the notion of *rebuilding ratio* [20]. Specifically, *rebuilding ratio refers to the smallest fraction of the surviving information that needs to be accessed in order to rebuild exactly the lost information*. Therefore, a

TABLE 1
A Qualitative Comparison among Different RAID-6 Codes

	RDP	X-code	P-code	HDP	S-code
Encoding complexity	conditionally optimal	optimal	optimal	suboptimal	optimal
Decoding complexity	conditionally optimal	optimal	optimal	suboptimal	optimal
Update complexity	about $1.5 \times$ opimal	optimal	optimal	$1.5 \times$ optimal	optimal
Code length	no limit	p	p or $p-1$	$p-1$	p or $p-1$
Rebuilding ratio	S-code≈X-code <P-code≈RDP≈HDP				

smaller value of the rebuilding ratio indicates a lower I/O cost of erasure recovery. For MDS RAID-6 codes, it is clear that when there are two erasures in the codeword, then the rebuilding ratio is 1, since in this case we need to access all the surviving information. Therefore, we only focus on the case in which there is only a single erasure in the codeword.

Since none of the existing vertical codes provides specially optimized algorithms for single-erasure recovery, we employ a straightforward but useful method to determine their rebuilding ratios. Our method is based on exhaustive enumeration, namely, we test all the possible reconstruction schemes for a certain erased column, then find the scheme that requires the smallest amount of information for reconstruction. For most of the codes, the rebuilding ratio depends on the position of the erased column. Thus, we take the average rebuilding ratio to evaluate a given code. Fig. 2 shows the rebuilding ratios of different RAID-6 codes. Note that we mainly compare S-code with representative vertical codes. Since RDP codes are considered as the best horizontal RAID-6 codes, we take them as the representative of horizontal codes.

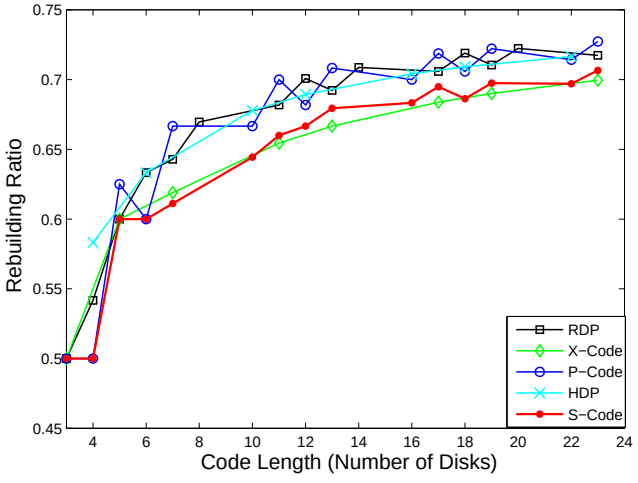

Fig. 2. Rebuilding ratios of different RAID-6 codes, where the rebuilding raito is defined to be the smallest fraction of the surviving information that needs to be accessed in order to rebuild exactly the lost information and thus the smaller the better.

As shown in Fig. 2, S-code has the smallest rebuilding ratio in most cases. Although with some code lengths, X-code has smaller rebuilding ratio than S-code, there is no existing effective algorithm for X-code to efficiently generate the best recovery scheme. For X-code, there are $p-2$ data bits in each column of the codeword, thus there are 2^{p-2} possible reconstruction schemes for any erased column. Obviously, the computation overhead of the exhaustive enumeration based method will increase explosively as p grows. In contrast, the *Single-Erasure Decoding Algorithm* that is presented in Section 4 can directly generate the optimal recovery scheme for S-code. To verify this claim, we have calculated the rebuilding ratios for S-code with both Algorithm 3 and exhaustive enumeration. As expected, for every code length the results given by the two algorithms are exactly identical.

To put it all together and provide a reasonable perspective, we compare our S-code with other popular RAID-6 codes in all key metrics of coding in Table 1 to end this section.

6 Conclusions

We have presented a new class of MDS array codes for RAID-6, called S-code, aiming to optimize every metric of coding. S-code is a class of vertical codes that combines the advantages of both X-code and P-code, thus obtaining several attractive properties. In particular, S-code has optimal encoding, decoding and update complexities, and the code length can be either p or $p-1$, where p is a prime number. Moreover, in most cases, S-code incurs less I/O cost than other representative RAID-6 codes during the single-erasure recovery. Owing to these attractive properties, we believe that S-code has a potential to become a popular class of RAID-6 codes.

Our future work includes efforts to discover new *lowest-density* MDS array codes for length of all positive integers and for correcting more than two erasures. Previous studies have shown that both of these two problems are non-trivial and challenging.

Acknowledgments

This work is supported in part by the National Basic Research Program (973 Program) of China under Grant No. 2011CB302305, the National Natural Science Foundation of China under Grant No.

61232004, and the US NSF under Grants NSF-IIS-0916859, NSF-CCF-0937993, NSF-CNS-1016609 and NSF-CNS-1116606.

REFERENCES

[1] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (raid)," in *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '88. New York, NY, USA: ACM, 1988, pp. 109–116. [Online]. Available: http://doi.acm.org/10.1145/50202.50214

[2] S. N. I. Association, "The 2014 snia dictionary." http://www.snia.org/education/dictionary/r, 2014.

[3] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "Raid: High-performance, reliable secondary storage," *ACM Computing Surveys*, vol. 26, no. 2, pp. 145–185, june 1994. [Online]. Available: http://doi.acm.org/10.1145/176979.176981

[4] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial & Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960. [Online]. Available: http://epubs.siam.org/doi/pdf/10.1137/0108018

[5] M. Blaum, J. Brady, J. Bruck, and J. Menon, "Evenodd: An efficient scheme for tolerating double disk failures in raid architectures," *Computers, IEEE Transactions on*, vol. 44, no. 2, pp. 192–202, 1995. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=364531

[6] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-diagonal parity for double disk failure correction," in *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, 2004, pp. 1–14. [Online]. Available: https://www.usenix.org/publications/library/proceedings/fast04/tech/corbett/corbett_html/

[7] J. S. Plank, "The raid-6 liberation codes," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies*. USENIX Association, 2008, p. 7.

[8] L. Xu and J. Bruck, "X-code: Mds array codes with optimal encoding," *Information Theory, IEEE Transactions on*, vol. 45, no. 1, pp. 272–276, 1999. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=746809

[9] C. Jin, H. Jiang, D. Feng, and L. Tian, "P-code: A new raid-6 code with optimal properties," in *Proceedings of the 23rd international conference on Supercomputing*. ACM, 2009, pp. 360–369. [Online]. Available: http://dl.acm.org/citation.cfm?id=1542326

[10] C. Wu, X. He, G. Wu, S. Wan, X. Liu, Q. Cao, and C. Xie, "Hdp code: A horizontal-diagonal parity code to optimize i/o load balancing in raid-6," in *Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*. IEEE, 2011, pp. 209–220. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5958220

[11] M. Blaum, J. Bruck, and A. Vardy, "Mds array codes with independent parity symbols," *Information Theory, IEEE Transactions on*, vol. 42, no. 2, pp. 529–542, 1996. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=485722

[12] M. Blaum, P. G. Farrell, and H. C. van Tilborg, "Array codes," *Handbook of Coding Theory*, vol. 2, pp. 1855–1909, 1998.

[13] J. S. Plank and C. Huang, "Tutorial: Erasure coding for storage applications," Slides presented at FAST-2013: 11th Usenix Conference on File and Storage Technologies, San Jose, February 2013.

[14] N. Sloane and F. J. MacWilliams, "The theory of error correcting codes," *North-Holland Math. Library*, vol. 16, 1977.

[15] L. Xu, V. Bohossian, J. Bruck, and D. G. Wagner, "Low-density mds codes and factors of complete graphs," *Information Theory, IEEE Transactions on*, vol. 45, no. 6, pp. 1817–1826, 1999. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=782102

[16] B. Schroeder and G. A. Gibson, "Disk failures in the real world: What does an mttf of 1, 000, 000 hours mean to you?" in *FAST*, vol. 7, 2007, p. 1.

[17] L. Xiang, Y. Xu, J. Lui, and Q. Chang, "Optimal recovery of single disk failure in rdp code storage systems," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 1. ACM, 2010, pp. 119–130. [Online]. Available: http://dl.acm.org/citation.cfm?id=1811054

[18] L. Xiang, Y. Xu, J. Lui, Q. Chang, Y. Pan, and R. Li, "A hybrid approach to failed disk recovery using raid-6 codes: Algorithms and performance evaluation," *ACM Transactions on Storage (TOS)*, vol. 7, no. 3, p. 11, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=2027071

[19] C. Jin, D. Feng, and J. Liu, "Extending and analysis of x-code," *Journal of Shanghai University (English Edition)*, vol. 15, pp. 194–200, 2011.

[20] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: Mds array codes with optimal rebuilding," *IEEE Transactions on Information Theory*, vol. 59, pp. 1597–1616, 2013. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6352912