Candidate MDS Array Codes for Tolerating Three Disk Failures in RAID-7 Architectures

Mayur Punekar
Dept. of Computer Science and Eng.
Qatar University
Doha, Qatar
mayur.punekar@ieee.org

Qutaibah Malluhi
Dept. of Computer Science and Eng.
Qatar University
Doha, Qatar
qmalluhi@qu.edu.qa

Yongge Wang
Dept. SIS, UNC Charlotte
NC, USA
yongge.wang@uncc.edu

Yvo Desmedt
The University of Texas at Dallas
Richardson, TX, USA
yvo.desmedt@utdallas.edu

ABSTRACT

Current storage systems use RAID-5 and RAID-6 architectures to provide protection against one and two disk failures, respectively. However, as the size of storage system grows rapidly three concurrent disk failures are becoming more frequent. To cope up with three disk failure, we propose a new RAID level, i.e., RAID-7, for which three-column-erasure tolerating MDS array codes are needed. However, it is an open question as to which MDS array codes should be used for RAID-7. In this paper, we compare different array codes, which can be used in RAID-7 systems that require storage efficiency (the ratio of number of information symbols to encoding (or codeword) symbols) ≤ 0.5. The paper discusses three-column-erasure tolerating MDS array codes proposed in the literature namely, $[5,2]$ 2×5 BP-XOR code, $[6,3]$ 4×6 lowest-density array code, $[6,3]$ 2×6 STAR code, $[6,3]$ 4×6 generalized RDP code. The paper introduces a new three-column-erasure tolerating $[6,3]$ 2×6 almost BP-XOR codes. We analyze annual failure rate, storage efficiency, worst case normalized encoding/update/repairing/read complexity, repair bandwidth, and number of buffers required for these codes. We also provide experimental results to understand the average case encoding and repairing complexity of BP-XOR, STAR, GRDP, and almost BP-XOR codes by implementing them in software. From our analysis and experimental results, we conclude that $[6,3]$ 2×6 almost BP-XOR are best suited for RAID-7 systems with storage efficiency ≤ 0.5.

CCS CONCEPTS

• **Information systems → RAID**;

KEYWORDS

RAID-7, MDS Array codes, Three-Column-Erasure Tolerating Codes

BDCAT'17, December 5–8, 2017, Austin, Texas, USA

ACM ISBN 978-1-4503-5549-0/17/12…$15.00
https://doi.org/10.1145/3148055.3148056

1 INTRODUCTION

Redundant arrays of inexpensive (or independent) disks (RAID) has been widely used as an important building block for developing reliable storage systems. Popular RAID architectures such as those based on RAID-5, e.g., [1], use simple parity schemes to provide protection against a single disk failure. However, it was already observed in early 90's that as disk arrays grow, the chances of double disk failures increase significantly [2]. Hence, RAID-6 architecture was proposed, which uses two parity blocks to provide protection against two concurrent disk failures. Several MDS (maximum distance separable) array codes, e.g., EVENODD [3], Row-diagonal parity (RDP) [4], etc., have been proposed for RAID-6 architecture. One of the most important features of such codes is that they in general require only XOR operations to encode and decode the data. Hence, the encoding and decoding complexity is significantly less compared to the RAID systems which use MDS codes over finite fields, e.g., Reed-Solomon codes, to protect data against disk failure. In [5], authors propose $[n,2]$ and $[n,n-2]$ MDS BP-XOR codes which are also suitable for use in RAID-6 architecture. BP-XOR codes utilize ideas from efficient belief propagation (BP) decoding process used in LT codes [6] and have substantially lower decoding complexity compared to other two-column-erasure tolerating array codes.

In 2005, Kryder's law was reported in [7] which states that the hard drive density will double annually. It has been observed in [2] that this rate of doubling has not been maintained but it has been close. On the other hand, hard disk throughput has been growing rather slowly [2]. As Kryder's law continues to hold and the hard disk throughput not able to match its pace, RAID reconstruction times factor more into reliability calculations than ever before [2]. Due to this, burden of providing reliability is increasingly shifting from the hard drive manufacturers to the RAID systems that integrate them [2]. It was predicted by Leventhal in [2] that by 2020, RAID-6 will provide only the level of protection that was given by RAID-5 system in previous decade.

Cloud data storage has received extensive interest in the past few years from both business entities and individuals. More and more business organizations are beginning to move their business critical data to cloud data storage systems. Consequently, the capacity of cloud storage systems is also increasing rapidly. However, according to a Google report [8] 37% of failures in Google cloud storage are

part of correlated burst failures of more than 2 nodes. Hence, if such large scale cloud storage systems are protected using RAID-6 architecture then the risk of data loss is very high.

As the RAID-6 is increasingly unable meet the reliability requirements, a new RAID level which can tolerate three disk failures is needed. A natural extension of RAID-6 would be RAID-7 however, to the best of the authors knowledge such a system was discussed only in [2]. The author in [2] did not propose any specific coding scheme for RAID-7 and which array codes are suitable RAID-7 architecture remains an open problem. Some work has already been done to develop codes to tolerate three disk failure, e.g., a generalization of EVENODD is proposed in [9] and another generalization known as STAR codes is given in [10], and generalized RDP codes are proposed in [11]. Also the lowest density array codes were proposed in [12] and proved to be MDS for appropriately selected parameters in [13]. BP-XOR codes of [5] can also be designed to provide protection against three disk failures. The authors in [27] have analyzed some three-column-erasure tolerating codes including STAR code. However, they also consider other codes which can not be decoded using XOR operation only and half of their selected codes are non-MDS which are not interesting from practical perspective. Also, they do not use optimized decoding algorithm for STAR code to reduce repair complexity.

In this paper, we analyse and compare three-column-erasures tolerating array codes which can be considered for future RAID-7 architectures. The rest of the paper is structured as follows. First, reliability of RAID-6 and RAID-7 systems is compared in Sec. 2. Notations and background is given in Sec. 3. Then, we discuss the state-of-the art array codes, namely BP-XOR codes, lowest density array codes, generalized EVENODD codes and generalized RDP codes, which can tolerate three disk failures in Sec. 4. In Sec. 5, we propose almost BP-XOR codes for tolerating three disk failures which can be decoded using combination of pre-processing and BP-XOR decoding. In Sec. 6, we compare various parameters, such as, encoding/update/repair complexity, repair bandwidth etc. for the array codes discussed in previous sections. The experimental results for average encoding and repairing complexity for different codes are provided in Sec. 7. We conclude the paper in Sec. 8.

2 MOTIVATION FOR NEW RAID LEVEL

As mentioned above, RAID-7 architecture needs to recover data even when three simultaneous disk failures occur. However, it is natural to ask how much reliability gain can be achieved using such a system when compared to RAID-6? To address this question, we give an example in which a fixed amount of data is stored using RAID-6 and RAID-7 systems and compare their *Mean Time To Data Loss* (MTTDL). MTTDL is one of the most important metric used to assess RAID system's reliability.

Let us analyse RAID-6 storage system with total capacity of 1 petabytes (PB) and 10 PB. MTTDL for RAID-6 and RAID-7 systems can be calculated using equation (1) which has been adapted from [14, eq. (6)] for t-column-erasure tolerating codes.

$$MTTDL = \frac{MTTF^{(t+1)}}{N * (N-1) * \cdots * (N-t) * MTTR^{t}}, \qquad (1)$$

where, N = total number of disks in a RAID array, t = maximum number of column erasures that can be corrected, $MTTF$ = mean time to failure for a disk, and $MTTR$ = mean time to repair a disk.

Now, we calculate $MTTDL$ for RAID-6 storage systems using (1). First, assume that RAID-6 array stores 1 PB of data using 1000 units of 1 terabyte (TB) hard disks. We assume that the parity to stored-data ratio is $P/(D+P) = 0.5$ where D = number of information disks, P = number of parity disks (e.g., if [4, 2] EVENODD code is used then the total number of information disks are 500 for 1 PB RAID-6 system). A reasonable $MTTF$ value for 1 TB hard disk is $1,000,000$ hours. For write speeds of 30 MB/s, $MTTR \approx 9.71$ hours. For RAID-6 systems $t = 2$. Using these parameters we get $MTTDL = 1214.4$ years. However, if we now store 10 PB of data using 10000 units of 1 terabyte (TB) hard disks, $MTTDL = 1.21$ years! As can be observed, a 10 fold increase in data size reduces $MTTDL$ dramatically and hence, risk of data loss significantly increases.

However, if we use a three-column-erasure tolerating ($t = 3$) $[6, 3]$ 2×6 array code for RAID-7 architecture (which has the same parity to stored-data ratio of 0.5) then for storage of 10 PB of data using 10000 units of 1 terabyte (TB) hard disks, we get $MTTDL = 12.48$ years which is 10 times higher compared to RAID-6 system. As can be observed, three-column-erasure tolerating RAID-7 system improves reliability of the system significantly compared to RAID-6 system even when the parity to stored-data ratio is the same.

We remark that the above calculation of $MTTDL$ is presented to give an idea of improvement in reliability that can be expected by moving from RAID-6 to RAID-7 architecture. It may be possible to improve $MTTDL$ for RAID-6 using other techniques, e.g., declustering, etc., but the effects of such improvements on $MTTDL$ are beyond the scope of this paper and hence not considered here.

3 BACKGROUND

Array codes have been used in storage systems that utilizes RAID-5 and RAID-6 architectures. Array codes are a type of linear code in which a codeword is placed in a two dimensional matrix array. Let n, k, t, and b be fixed numbers such that $n > \max\{k, t\}$ and $v_1, v_2, \ldots, v_{bk}$ be variables taking values from the set $M = \{0, 1\}$. $v_1, v_2, \ldots, v_{bk}$ are referred to as information symbols. A t-erasure tolerating $[n, k]$ array code is a $b \times n$ matrix $\mathbf{C} = [\alpha_{i,j}]_{1 \le i \le b, 1 \le j \le n}$ such that each encoding symbol $\alpha_{i,j} \in M$ is the exclusive-or (XOR) of one or more information symbols from $v_1, \ldots, v_{bk}$ such that the symbols $v_1, \ldots, v_{bk}$ can be recovered from any $n - t$ columns of the matrix. For an encoding symbol $\alpha_{i,j} = v_{i_1} \oplus \cdots \oplus v_{i_\sigma}$, we call $v_{i_j} (1 \le j \le \sigma)$ a neighbor of $\alpha_{i,j}$ and call σ the degree of $\alpha_{i,j}$. If an $[n, k]$ $b \times n$ array code $\mathbf{C}$ can tolerate t column erasures such that $k = n - t$ hold then $\mathbf{C}$ is said to be maximum distance separable (MDS) code.

An $[n, k]$ array code $\mathbf{C}$ over the alphabet M can be considered as a linear code over the extension alphabet M^b of length n or a linear code over the alphabet M of length bn. A $bk \times bn$ binary matrix is said to be a generator matrix of a $b \times n$ array code $\mathbf{C}$ if it is a generator matrix of $\mathbf{C}$ when $\mathbf{C}$ is considered as a length bn linear code over the alphabet M. A $bt \times bn$ parity-check matrix for array code $\mathbf{C}$ can be defined in a similar manner. The matrix H is a parity-check matrix of the array code $\mathbf{C}$ if we have $H\mathbf{y}^T = 0$ where $\mathbf{y} = (\alpha_{1,1}, \ldots, \alpha_{b,1}, \ldots, \alpha_{1,n}, \ldots, \alpha_{b,n}), x = (v_1, \ldots, v_{bk})$, and the

addition is defined as the XOR on bits. For the generator matrix G we have $\mathbf{y} = \mathbf{x}G$.

We define *annual failure rate* (*AFR*) as the average number of disks that fails in a year in a RAID system which is designed to protect 1 PB data (we assume that a single disk has capacity of 1 TB). Formally,

$$AFR = 365 * 24/MTTFR, \tag{2}$$

where $MTTFR = MTTF/N$, N = number of disks in RAID system. We use $MTTF = 1,000,000$ hours throughout this paper. We define storage efficiency as the ratio of number of information symbols to encoding symbols for a given code. The *Normalized encoding complexity* is defined as the ratio of number of XOR operations required to generate all encoding symbols to the number of information symbols. Similarly, *normalized repairing complexity* is defined as the ratio of number of XOR operations required to decode all information symbols and to reconstruct all missing encoding symbols to the number of information symbols when three-column-erasure occurs. *The update complexity* is the maximum number of encoding symbols updated when an information symbol is changed. Similarly, we define the *read complexity* as the maximum number of encoding symbols that need to be read in order to retrieve an information symbol. Please note that the read complexity is 1 for systematic codes. Repair bandwidth [15] is given as βd where β is the number of encoding symbols needed to reconstruct single column erasure and d is the number of columns accessed to obtain them.

4 STATE-OF-THE ART THREE-COLUMN-ERASURE TOLERATING MDS ARRAY CODES

In this section, we discuss three-column-erasure tolerating MDS array codes proposed in the literature which can be considered for RAID-7 architecture. Two of these codes, namely, the $[6,3]$ 2×6 STAR code and $[6,3]$ 4×6 generalized RDP code, are selected as they are generalization of popular two-column-erasure tolerating EVENODD and RDP codes used in RAID-6 architectures. We also discuss lowest density array codes as they have systematic generator and parity-check matrices with the smallest possible number of nonzero entries [12]. $[n,2]$ BP-XOR codes have been considered here as their encoding and decoding complexity is claimed to be significantly lower compared to other array codes [5].

As we show in the following, for $t = 3$ the BP-XOR code must have $n = 5$. On the other hand, the STAR code, lowest density array code and generalized RDP code can have multiple k and n values for $t = 3$. However, as we compare these three codes with BP-XOR code with $n = 5$ and the almost BP-XOR codes (proposed in Sec. 5) with $n = 6$, we select $k = 3$ and $n = 6$ for these three codes.

Apart from XOR array codes, Reed-Solomon codes (*RS codes*) [24] are also considered for RAID-6 architecture. It is also possible to design three-column-erasure tolerating RS codes however, they can not be decoded using XOR operations only. Due to this, it is difficult to compare read/write and repair complexity of RS codes with that of XOR array codes. Hence, we are not considering RS codes in this paper.

We have selected three codes with parity to stored-data ratio of 0.5 which is same as data replication and higher than data triplication (i.e., each data symbol is repeated three times on 3 different disks) 0.33 used in many modern storage arrays [25]. Since three-column-erasure tolerating BP-XOR code can not have parity to stored-data ratio of 0.5, we are using BP-XOR codes with the ratio of 0.4 which is still higher than that of data triplication.

4.1 $[5,2]$ 2×5 BP-XOR Code

A type of array codes, known as BP-XOR codes, were proposed by Wang in [5]. A t-erasure tolerating $[n,k]$ BP-XOR code $\mathbf{C} = [\alpha_{i,j}]_{1 \le i \le b, 1 \le j \le n}$ can recover all information symbols $v_1, \ldots, v_{bk}$ from any $n-t$ columns of encoding symbols using the BP-decoding process on the binary erasure channel.

It has been shown in [5] that, if each encoding symbol in $\mathbf{C} = [\alpha_{i,j}]_{1 \le i \le b, 1 \le j \le n}$ is restricted to degree $\sigma = 2$ then array BP-XOR codes are equivalent to edge-colored graphs introduced by Wang *et al.* in [16] for tolerating network homogeneous failures. In the following we briefly discuss edge-colored graphs and their link to $[n,2]$ BP-XOR codes.

An edge-colored graph [16] is a tuple $G(V; E; C; f)$, with V the node set, E the edge set, C the color set, and f a map from E onto C. The structure $\mathcal{Z}_{C,t} = \{Z : Z \subseteq E \text{ and } |f(Z)| \le t\}$ is called a *t-color* adversary structure. Let $A, B \in V$ be distinct nodes of G. A, B are called *$(t+1)$-color connected* for $t+1$ if for any color set $C_t \subseteq C$ of size t, there is a path p from A to B in G such that the edges on p do not contain any color in C_t. An edge-colored graph G is *$(t+1)$-color connected* if and only if for any two nodes A and B in G, they are $(t+1)$-color connected.

A general construction of $(t+1)$-color connected edge-colored graphs using perfect one-factorizations of complete graphs has been proposed in [5], which is then used to construct $[n,2]$ BP-XOR codes. A one-factor of complete graph $K_n = (V, E)$ with n nodes (n is even) is a spanning 1-regular subgraph of K_n. A one-factorization of K_n is a set of one-factors that partition the set of edges E. If the union of every two distinct one-factors is a Hamiltonian circuit then such an one-factorization is knowns as perfect (or P1F). If p is a prime number then it is known (see [17]) that perfect one-factorizations for K_{p+1}, K_{2p}, and certain K_{2n} exist and it has been conjectured that P1F exist for all K_{2n}. P1F for K_{p+1} and K_{2p} is given in [5, Example 2.2].

$(n-2)$-erasure tolerating MDS $[n,2]$ $b \times n$ array BP-XOR codes can be designed with the help of a $(n-2+1)$-color connected edge-colored graphs with n colors. For this, the smallest p (or $2p$) such that $n \le p$ (or $n \le 2p-1$), where p is an odd prime is selected. Then, as mentioned above, P1F of the complete graph K_{p+1} can be found. This edge-colored graph is then converted to an MDS $[n,2]$ $b \times n$, $b = (p-1)/2$ BP-XOR code using the process described in [5]. Using the same process we can design three-column-erasures tolerating $[5,2]$ BP-XOR code. For this, we select $p = 5$ and hence we have $b = 2$ rows per codeword column. The resulting code is given in Table 1.

As can be observed from Table 1, $[5,2]$, 2×5 BP-XOR code is systematic. Hence, its read complexity is 1 and if there is no disk failure then the stored-data can be retrieved without any XOR operations. A RAID-7 system designed using this code would require 2500 disks of 1 TB capacity to protect 1 PB of data. Hence, for such a system $AFR \approx 22$ disks/year. The storage efficiency for this code is $2/5 = 0.4$. The encoding process requires 6 XOR operations which

$v_1 \oplus v_4$	v_2	$v_3 \oplus v_1$	$v_4 \oplus v_2$	v_3
$v_2 \oplus v_3$	$v_3 \oplus v_4$	v_4	v_1	$v_1 \oplus v_2$

Table 1: $[5,2]$ 2×5 **BP-XOR Code Generated from P1F of** K_5**.**

gives normalized encoding complexity of $6/4 = 1.5$. If one information symbol changes then 4 encoding symbols stored on 3 disks are updated due to which the update complexity is 4. For one and two disk failures, 2 and 3 XOR operations are required, respectively. Since $\beta = 4$ encoding symbols from $d = 2$ columns need to be accessed to rebuild a single column, the repair bandwidth is 8. In case of three disk failures, the decoder needs to calculate (in worst case) 3 XOR operations to retrieve 4 information symbols and 4 XOR operations are needed to rebuild the parity symbols on three disks. Hence, normalized repair complexity is $7/4 = 1.75$.

4.2 $[6,3]$ 2×6 STAR Code

A type of three-column-erasure tolerating MDS codes known as the *STAR codes* were proposed by Huang *et al.* in [10]. STAR codes are a generalization of the two-column-erasure tolerating EVENODD codes [3] by adding a third parity column to it. In the following we briefly describe the process of constructing a STAR code.

A $[p+3, p]$ $(p-1)\times(p+3)$ STAR code consists of $p+3$ columns (p is a prime number) and each column has $p-1$ rows. The first p columns contain information symbols (referred to as *information columns*) whereas the last three contain parity symbols (*parity columns*). Let us assume that $a_{i,j} (0 \leq i \leq p-2, 0 \leq j \leq p+2)$ represent symbol i in column j. The first two parity columns of STAR codes are same as EVENODD codes. So first, we describe construction of the these columns. A parity symbol in column p is computed as the XOR of all information symbols in the same row. The computation of column $(p+1)$ takes the following steps. First, the array is augmented with an imaginary row $p-1$, where all symbols are assigned zero values. The XOR of all information symbols along the same diagonal (a diagonal with slope 1) is then computed and assigned to their corresponding parity symbol. Symbol $a_{p-1,p+1}$ now becomes nonzero and is called the EVENODD *adjuster* S_1. To remove this symbol from the array, adjuster complement is performed, which adds (XOR) the adjuster to all symbols in column $p+1$.

Now the third column of a STAR code $p+2$ is computed very similarly to the second column $p+1$ however, the XOR operations are along diagonals of slope -1 instead of slope 1 as in column $p+1$. Due to this, the third parity column is also referred to as *antidiagonal* parity. Like the second parity column, the generation of the third parity column also involves an adjuster S_2 which is the symbol $a_{p-1,p+2}$ in the imaginary row $p-1$. The adjuster complement operation is used to remove the adjuster symbol from the final code.

The algebraic description of the encoding process for STAR codes is given below. We assume $0 \leq i \leq p-2$ and $\langle x \rangle_p$ denotes x mod p.

$$a_{i,p} = \bigoplus_{j=0}^{p-1} a_{i,j}, \tag{3}$$

$$a_{i,p+1} = S_1 \oplus \left(\bigoplus_{j=0}^{p-1} a_{\langle i-j \rangle_p, j} \right), \; S_1 = \bigoplus_{j=0}^{p-1} a_{\langle p-1-j \rangle_p, j}, \tag{4}$$

$$a_{i,p+2} = S_2 \oplus \left(\bigoplus_{j=0}^{p-1} a_{\langle i+j \rangle_p, j} \right), \; S_2 = \bigoplus_{j=1}^{p-1} a_{\langle j-1 \rangle_p, j}. \tag{5}$$

We select $p = 3$ and derive $[6,3]$ 2×6 STAR code using the above mentioned procedure. The resultant code is given in Table 2. This code, like the EVENODD code, is systematic. Similar to $[6,3]$ 4×6 lowest density code, a RAID-7 system designed to protect 1 PB of data using this code would also have $AFR \approx 18$ disks/year. The storage efficiency for this code is again 0.5 and since its a systematic code, the read complexity is 1. It requires 14 XOR operations to encode 6 information symbols which results in the normalized encoding complexity of $14/6 = 2.33$. Since 4 encoding symbols are updated when a single information symbol changes, the update complexity is 4. To rebuild a single column, 6 encoding symbols from 3 columns needs to be accessed which results in repair bandwidth of 18.

The decoding procedure for this code is quite complex and complete description of it is beyond the scope of this paper. For more details reader is referred to [10, Sec. 4].

As explained in [10, Sec. 6.1], the total number of XOR operations required to decode first three (i.e., information) column erasures is given by $(3k + 2l_d + l_h)(p-1) = 22$ where $l_d = 1, l_h = 0$ are chosen according to [10, Sec. 4.3]. Since the first three columns are information columns, no XOR operation is required to rebuild them. With this, the normalized repair complexity for the worst case is $22/6 = 3.67$. Also, the decoder would require 13 buffers to store intermediate values of syndromes and crosses.

However, we remark that with the optimization of decoding algorithm for the first three-column-erasures, the maximum number of XOR operations required to decode all information symbols is upper bounded by 13. On the other hand, since we are considering repair complexity instead of decoding complexity, the worst case now occurs when columns 4 and 5 (or 4 and 6) are erased along with an information column. Though only 6 XOR operations are required to decode 2 missing information symbols for such erasure patterns, 9 XOR operations are needed to rebuild columns 4 and 5 (or 4 and 6). Hence, even with optimized decoder the maximum number of XOR operations required to rebuild three columns is 15. Due to this, the normalized repair complexity for this code is $15/6 = 2.5$. Also, the optimized decoder requires only 4 buffers instead of 13.

The STAR code as discussed above is very similar to the $[6,3]$ 2×6 generalized EVENODD codes proposed earlier by Blaum *et al.* in [9]. Both differ only in the slop of the diagonal used to calculate parity column $p+2$. The generalized EVENODD code uses the slop of 2 to calculate XOR of information symbols for $p+2$ and adjuster S_2 instead of -1 used by the STAR code. However, as explained in [10, Sec. 7], the worst case normalized repairing complexity for this code is 10. Hence, we do not use this code in our comparison.

Blaum *et al.* also proposed a generalization of EVENODD code in [20] however, these code are not systematic and hence not discussed in this paper.

v_1	v_3	v_5	$v_1 \oplus v_3 \oplus v_5$	$S_1 \oplus v_1 \oplus v_6$	$S_2 \oplus v_1 \oplus v_4$
v_2	v_4	v_6	$v_2 \oplus v_4 \oplus v_6$	$S_1 \oplus v_2 \oplus v_3$	$S_2 \oplus v_2 \oplus v_5$

Table 2: $[6,3]$ 2×6 **STAR Code** $\left(S_1 = v_4 \oplus v_5,\ S_2 = v_3 \oplus v_6\right)$.

4.3 $[6,3]$ 4×6 Generalized RDP Code

Two-column-erasure tolerating RDP codes were introduced by Corbett *et al.* in [4]. Like generalized EVENODD code, each codeword of RDP code contains k information columns and two parity columns. The encoding/decoding for RDP code requires only $k+1$ XOR operations for each encoding symbol when either $k+1$ or $k+2$ is a prime number. There are two generalizations of the RDP codes proposed in the literature, first by Blaum [21] and the other by Goel *et al.* [11]. We describe construction of both codes in the following.

Let us assume that $a_{i,j} (0 \leq i \leq p-2, 0 \leq j \leq p+1)$ represent symbol i in column j. Then, the three parity columns for $[p+2, p-1]$ $p-1 \times p+2$ generalized RDP array codes (p is prime) are constructed according to the following equations [11][21][22].

$$a_{i,p-1} = \bigoplus_{j=0}^{p-2} a_{i,j}, \tag{6}$$

$$a_{i,p} = \bigoplus_{j=0}^{p-1} a_{\langle i-j \rangle_p, j}, \tag{7}$$

$$a_{i,p+1} = \bigoplus_{j=0}^{p-1} a_{\langle i-s \cdot j \rangle_p, j}. \tag{8}$$

The third parity column for the generalized RDP code of Blaum [21] uses slop $s = 2$ in (8). On the other hand, for the generalized RDP code of [11], $s = -1$. For the rest of the paper we use generalized RDP code with $s = -1$ from [11]. If we select $p = 5$ and set all information symbols in the first column to 0 then from the above equations we get $[6,3]$ 4×6 generalized RDP code shown in Table 3.

Similar to the previous two codes, $AFR \approx 18$ disks/year for this code too. The storage efficiency for this code, like the previous two codes, is 0.5 and the read complexity is 1. The normalized encoding complexity for the code in Table 3 is $24/12 = 2$. The update complexiy is again 4 for the code. Since 12 encoding symbols from 3 columns needs to be read to rebuild a single parity column, the repair bandwidth is 36. The decoding algorithm for the RDP code from Table 3 is given in [11] however, an improved decoding algorithm for the same code is proposed in [22]. This improved decoding algorithm requires 6 XOR operations to calculate syndromes. The code has only 3 information columns and hence the decoding algorithm given in [22, Sec. III-B] can be used. The number of XOR operations required to retrieve all 12 information variables with this algorithm is [22, Sec. IV] $7p + k - 15 = 23$. In this case, no XOR operations are required to rebuild information columns due to which, the normalized repair complexity for this code is $(23+6)/12 = 2.42$. The decoder requires 12 buffers for syndromes and 5 additional buffers for intermediate variables ([22, eq. (16),(17)]). So the total number of buffers used are 17.

5 $[6,3]$ 2×6 ALMOST BP-XOR CODES

$[n,2]$ BP-XOR codes of [5] discussed in Sec. 4.1 are restricted to $k = 2$. It is natural to think about their extension to the higher value of k, e.g., $k = 3$, which can tolerate more than two-column-erasures.

As discussed previously, $[n,2]$ BP-XOR codes requires 1-factors of regular graph K_p. However, as explained in [23, Sec. 4], for $k = 3, t = 3$ at least some encoding symbols must have degree $\sigma \geq 3$. Hence, $[6,3]$ $b \times 6$ BP-XOR codes can not be constructed using 1-factors of a regular graph and instead 1-factors of regular hypergraph K_p^3 need to be used. However, as we show in the following, there is no systematic MDS $[6,3]$ 2×6 BP-XOR code. It is an open question as to whether non-systematic $[6,3]$ 2×6 or systematic/non-systematic $[6,3]$ $b \times 6, b > 2$ BP-XOR codes exist but we conjecture here that there is no $[6,3]$ $b \times 6$ BP-XOR code in general.

Fact 1. *There is no systematic MDS* $[6,3]$ 2×6 *BP-XOR code for* $\sigma = 3$.

Proof. See Appendix A. □

We now introduce *almost BP-XOR codes* for the same parameters as in Fact 1. Most of the three-column-erasure patterns for these codes can be decoded using BP-XOR decoding. On the other hand when BP-XOR decoding fails, the decoder for these codes can utilize the inverse of a 6×6 submatrix $\tilde{G}$ derived from a generator matrix G by deleting matrix columns related to erased codeword columns. Due to this requirement, each 6×6 submatrix $\tilde{G}$ (in which 3 groups of two columns related to 3 codeword columns are selected) of the code must be invertible.

An algorithm to generate almost BP-XOR codes is given in Algorithm 1. An example $[6,3]$ 2×6 almost BP XOR code generated using this algorithm is given in Table 4. We now show that these codes are able to correct all $\binom{6}{3} = 20$ combination of three-column-erasure patterns. It can be observed that, for the almost BP-XOR codes generated using Algorithm 1 (e.g., almost BP-XOR code given in Table 4), BP-XOR decoding can correct up to 14 out of 20 three-column-erasure patterns. For the other three-column-erasure patterns, we need to invert the submatrix $\tilde{G}$ to decode missing three information symbols (three information symbols are directly available from degree one encoding symbols of the available three columns). However, the decoder need not necessarily have to invert the submatrix on-the-fly as the inverse of $\tilde{G}$ can be precomputed and buffered. With this arrangement the decoder needs to buffer $\tilde{G}^{-1}$ for several erasure patterns. However, instead of storing all submatrices in buffer, the decoder can store the solution for one of the information symbol which is derived from the $\tilde{G}$. As an example, let us assume that for a given three-column-erasure pattern, encoding symbols available to the decoder are $y_1, y_2, \ldots, y_6$ and information symbols v_4, v_5, v_6 need to be determined then a solution $v_4 = y_1 \oplus y_2 \oplus y_4$,

v_1	v_5	v_9	$v_1 \oplus v_5 \oplus v_9$	$v_8 \oplus v_{11} \oplus (v_2 \oplus v_6 \oplus v_{10})$	$v_2 \oplus v_7 \oplus v_{12}$
v_2	v_6	v_{10}	$v_2 \oplus v_6 \oplus v_{10}$	$v_1 \oplus v_{12} \oplus (v_3 \oplus v_7 \oplus v_{11})$	$v_3 \oplus v_8 \oplus (v_1 \oplus v_5 \oplus v_9)$
v_3	v_7	v_{11}	$v_3 \oplus v_7 \oplus v_{11}$	$v_2 \oplus v_5 \oplus (v_4 \oplus v_8 \oplus v_{12})$	$v_4 \oplus v_9 \oplus (v_2 \oplus v_6 \oplus v_{10})$
v_4	v_8	v_{12}	$v_4 \oplus v_8 \oplus v_{12}$	$v_3 \oplus v_6 \oplus v_9$	$v_5 \oplus v_{10} \oplus (v_3 \oplus v_7 \oplus v_{11})$

Table 3: $[6,3]$ 4×6 **Generalized RDP Code.**

v_1	v_2	v_3	v_4	v_5	v_6
$v_2 \oplus v_3 \oplus v_5$	$v_1 \oplus v_4 \oplus v_6$	$v_1 \oplus v_5 \oplus v_6$	$v_2 \oplus v_5 \oplus v_6$	$v_1 \oplus v_3 \oplus v_4$	$v_2 \oplus v_3 \oplus v_4$

Table 4: An Example $[6,3]$ 2×6 **Almost BP-XOR Code.**

Algorithm 1 Create a generator matrix G for systematic MDS $[6,3]$ 2×6 almost BP-XOR codes

1: Create 6×6 identity matrix I (each column in I corresponds to a degree one encoding symbol).
2: FALG $\leftarrow 0$
3: **while** FLAG $\neq 1$ **do**
4: Randomly generate 6×6 matrix P such that row degree and column degree is 3 (each column in P corresponds to a degree three encoding symbol).
5: **for** c $\leftarrow 1 \ldots 6$ **do** {Check if a codeword column contains same information symbol twice.}
6: **if** $P_{c,c} \neq 0$ **then** {$P_{i,j}$ is the i-th element in j-th column}
7: **goto** 4.
8: **end if**
9: **end for**
10: Generate 6×12 generator matrix G by combining columns of I and P. Odd columns $G_i = I_j, i \in \{1,3,\ldots,11\}, j = (i+1)/2$ and even columns $G_i = P_j, i \in \{2,4,\ldots,12\}, j = i/2$.
11: FLAG_INVERSE $\leftarrow 0$.
12: **for** erasure_counter $\leftarrow 1 \ldots 20$ **do** {//there are $\binom{6}{3} = 20$ erasure patterns for three-column-erasures}
13: Build submatrix $\tilde{G}$ from G by deleting columns related to three-column-erasure pattern.
14: **if** rank of $\tilde{G} < 6$ **then** {//if $\tilde{G}$ is not invertible}
15: FLAG_INVERSE $\leftarrow 1$.
16: **end if**
17: **end for**
18: **if** FLAG_INVERSE $\neq 1$ **then** {//if all $\tilde{G}$ are invertible}
19: FLAG $\leftarrow 1$.
20: **end if**
21: **end while**
22: **return** G.

which is derived from the inverse of an appropriate submatrix $\tilde{G}$, needs to be stored in a buffer. Such solutions can be precomputed using $\tilde{G}$ for all three-column-erasure patterns where BP-XOR fails and buffered by the decoder. The decoder can use this solution to decode one of the information symbol and then the other two information symbols can be decoded using BP-XOR decoder. We note that, five out of six information symbols can be retrieved using BP-XOR decoding and hence we refer to these codes as almost BP-XOR codes. The proof for this fact is given below.

Fact 2. *Almost BP-XOR codes built using Algorithm 1 can decode five information symbols using BP-XOR decoding.*

Proof. See Appendix B. □

From the above analysis it is clear that the $[6,3]$ 2×6 almost BP-XOR codes can correct any three-column-erasure pattern and hence they are MDS codes.

As discussed, for some three-column-erasure patterns the decoder for almost BP-XOR codes need to buffer the solution for a selected information symbol. Such a solution involves XOR of at least three of the available encoding symbols but in the worst case it may require XOR of all six available encoding symbols. However, decoder can buffer a solution which requires least number of XOR operations. With this, in worst case 5 XOR operations are required to decode an information symbol using corresponding solution.

Like most of the codes discussed in previous section, $AFR \approx 18$ disks/year, storage efficiency is 0.5 and read complexity is 1 for the almost BP-XOR code. Encoding of almost BP-XOR codes require 12 XOR operations. Thus, the normalized encoding complexity is $12/6 = 2$. When an information symbol is updated, 4 encoding symbols are updated due to which the update complexity is 4. To rebuild a column 6 encoding symbols from 3 disks are accessed which gives repair bandwidth of 18.

Since almost BP-XOR codes are systematic codes, their read complexity is 1 and also no XOR operations are required to retrieve data if there is no disk failure. On the other hand, for one disk failure, 4 XOR operations are required each to regenerate degree three and degree one encoding symbols using BP-XOR. Similarly, for 2 disk failures, minimum of 8 XOR operations are needed using BP-XOR to regenerate two columns.

The decoding of almost BP-XOR codes require just 6 XOR operations when for a given three-column-erasure pattern BP-XOR decoding succeeds. However, when BP-XOR decoder fails, it has to use the solution for a selected information symbol. Such a solution may require up to 5 XOR operations. For other two information symbol 4 XOR operations are needed using BP-XOR. Hence, total 9 XOR operations are needed to retrieve 6 information symbols

	$[5,2]$ 2×5 BP-XOR code	$[6,3]$ 2×6 almost BP-XOR code	$[6,3]$ 4×6 Lowest density array code	$[6,3]$ 2×6 STAR code	$[6,3]$ 4×6 Generalized RDP code
AFR (disks/year)	22	18	18	18	18
Storage efficiency	0.4	0.5	0.5	0.5	0.5
Read complexity	1	1	3	1	1
Normalized encoding complexity	1.5	2	3	2.33	2
Update complexity	4	4	6	4	4
Repair bandwidth	8	18	36	18	36
Normalized repair complexity	1.75	2.5	3	2.5	2.42
Additional buffers	0	6	2	4	17

Table 5: Comparison of Three-Column-Erasures Tolerating MDS Array Codes

in worst case. Further, 6 additional XOR operations are needed to recalculate degree three encoding symbols. Consequently, 15 XOR operations in total are needed to rebuild three columns with almost BP-XOR codes. The normalized repairing complexity is $15/6 = 2.5$. The number of buffers required by the decoder is 6, which is used to store solutions for three-column-erasure patterns for which BP-XOR fails.

6 DISCUSSION

In this section we compare MDS array codes discussed in Sec. 4 and Sec. 5. Table 5 lists different parameters and their values for different codes.

In almost all parameters $[5,2]$ 2×5 BP-XOR code outperforms the other codes. However, since this code require higher number of storage disks to protect the same amount of data as other codes, its AFR is higher compared to all other codes. Another problem is with storage efficiency which is 0.4 for this code whereas for other codes it is 0.5. Hence, in terms of AFR and storage efficiency the other array codes are better. Between these four codes, almost BP-XOR code and generalized RDP code are better in terms of encoding complexity than other two codes. The update complexity is same for almost BP-XOR, STAR and Generalized RDP codes. Generalized RDP code has marginally lower (3.5%) normalized repair complexity compared to almost BP-XOR code and STAR code. The STAR and almost BP-XOR codes have lowest repair bandwidth where as lowest density array code and generalized RDP code have the worst. In terms of additional buffers, STAR code is the most efficient where as the generalized RDP code is the worst.

Generally, array codes are divided in to two types depending on how information and parity symbols are organized : horizontal and vertical. If all symbols in all columns are either information or parity symbols then such a code is referred to as the horizontal array code. On the other hand, if all columns contain a mix of information and parity symbols then such a code is known as the vertical array code. It is easy to observe that BP-XOR, almost BP-XOR and lowest density array codes are vertical array codes whereas STAR and GRDP codes are horizontal array codes. Traditionally horizontal codes preferred by RAID system designers due to ease of access of information symbols. However, Jin *et al.* in [26] implemented RAID-6 horizontal (RDP) and vertical (P-code) codes in software and their test results show that both types of codes have similar read and write performance. We leave the detailed study to compare read and write performance of horizontal and vertical three-column-erasure tolerating array codes as future work. However, we expect the horizontal and vertical codes for RAID-7 to have similar read and write performance.

From our analysis, we conclude that if a RAID-7 system has to be efficient in terms of worst case encoding/update/repair complexity and repair bandwidth, then $[5,2]$ 2×5 BP-XOR codes are most appropriate array codes. However, such a RAID-7 system has to accept the lower storage efficiency of 0.4 and somewhat higher AFR. On the other hand, if a RAID-7 system requires better storage efficiency then $[6,3]$ 2×6 almost BP-XOR code or $[6,3]$ 2×6 STAR code are more suitable for such a system.

7 EXPERIMENTAL RESULTS

In this section, we present simulation results obtained from the software implementation of the XOR array codes discussed in Sec. 4 and Sec. 5. We implemented XOR array codes in software using C++ under Ubuntu distribution. We carry out simulations on PC with Intel Xeon E5-2640 CPU clocked at 2.5GHz and 8GB of RAM. As we are interested in write and repair complexity of the codes, we

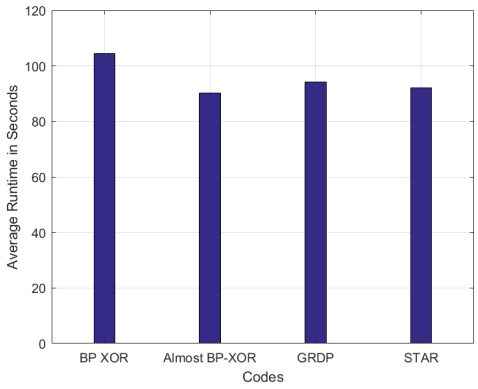


Figure 1: Average Runtime required to Encode 240GB of Data for Three-Column-Erasure Tolerating Codes.

do not use file input-output operations and instead generate large random data on the fly. This data is given as an input to the encoder and output from the decoder is compared with the original random data to test whether decoder is working or not. The random data generated is of 240 Gigabyte. The decoder also performs repairing of the failed columns. Repairing for each erasure pattern is repeated 5 times in order to obtain reliable results. Since 240GB of data can not be processed at once, it is split into blocks of 180 Megabytes (MB) and then each block is processed at a time by encoder and decoder. Our software implementation use the double words of 64-bits as a single information symbol. Hence, all XOR and data read/write operations are performed on 64-bit double words.

Figure 1 shows the average runtime required by encoder of different codes to encode 240GB data. As can be observed, encoders for almost BP-XOR, GRDP and STAR require similar runtime however, almost BP-XOR is the best among all codes. As shown in Table 5, BP-XOR has the lowest encoding complexity. However, it should be noted that the BP-XOR code has storage efficiency of 0.4 and hence it generates more parity data compared to other codes for same amount of information. Due to this, it requires more time compared to other codes.

The average runtime required to repair columns for different code is shown in Fig. 2. For all types of column erasure patterns (i.e., one, two and three column-erasures), almost BP-XOR requires the least amount of time to decode and reconstruct erased columns. Please note that, since we consider all possible erasure patterns for one, two and three column-erasures, these results are representative of average complexity whereas the normalized repair complexity results given in Table 5 are based on worst case complexity for three-column-erasure pattern. Further, the simulation results also include time required for data read/write from/to memory which may vary from code to code. Hence, though almost-BP XOR, STAR and GRDP codes have similar normalized repair complexity in Table 5, the results in Fig. 2 show that the almost BP-XOR code is the best among them.

BP-XOR code has the lowest repair complexity among all the codes considered in this paper however, they have the worst storage efficiency. Hence, BP-XOR decoder has to process more data compared to other codes. As mentioned earlier, the results in Fig. 2 also includes the time required for data read/write from/to memory. We believe that the additional data processing required during BP-XOR decoding and reconstruction process is responsible for higher time required for repairing erased columns for BP-XOR codes compared to almost BP-XOR code.

8 CONCLUSION

In this paper, we proposed a new $[6,3]$ 2×6 almost BP-XOR code and analyzed its performance together with existing three-column-erasure tolerating MDS array codes for future RAID-7 systems. For our analysis, we selected four state-of-the art array codes : $[5,2]$ 2×5 BP-XOR code, $[6,3]$ 4×6 lowest density array code, $[6,3]$ 2×6 STAR code, $[6,3]$ 4×6 generalized RDP code. We analyzed AFR, storage efficiency, encoding/update/repair/read complexity, and repair bandwidth for these codes. Further, we implemented BP-XOR, STAR, GRDP and almost BP-XOR codes in software to obtain experimental results for average encoding and repairing complexity. Through our analysis and experimental results we conclude that the $[6,3]$ 2×6 almost BP-XOR codes are best suited for RAID-7 system that requires storage efficiency of 0.5.

A PROOF OF THE FACT 1

We observe that, since the code is systematic and $b \cdot k = 6$, at least 6 encoding symbols in $n = 6$ columns must have degree one. Let us assume without loss of generality (w.l.o.g) that the first row of the code has degree one encoding symbols whereas the encoding symbols in second row has degree three.

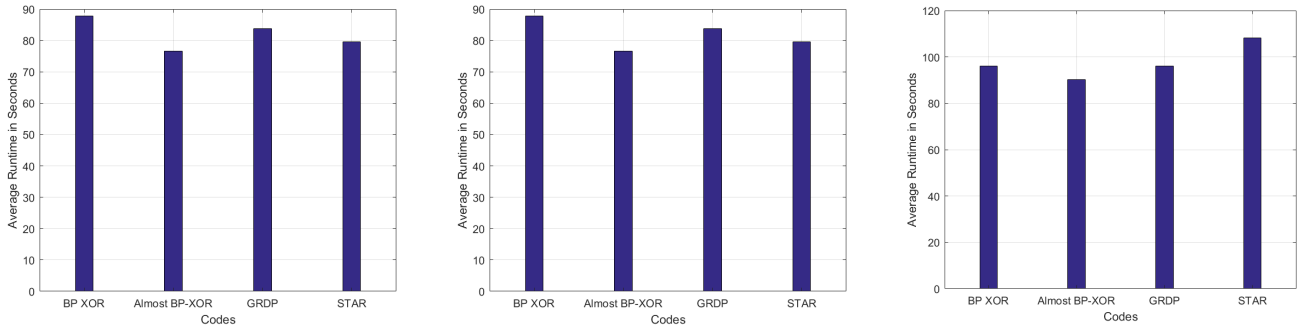
Since we want to construct an MDS array code, BP-XOR decoder must be able to recover all information symbols from any three columns of the code. Now for a given three columns, we have the following condition that must be satisfied for the BP-XOR decoder to start : at least two of the three degree one encoding symbols must occur simultaneously in at least one degree three symbol. However, as shown in the following, this necessary condition is not satisfied for the code with parameters $n = 6, k = 3, b = 2, \sigma = 3$.

We try to construct a BP-XOR code in the following such that the above mentioned necessary condition is fulfilled for all set of three columns. We start by selecting the first row of the code as follows while the rest of the entries are selected in subsequent iterations. The resulting BP-XOR code is shown in Table 6 where $\lambda_i, \mu_i, \phi_i, i \in \{1,2,\ldots,6\}$ represents the first, second and third variable in degree three encoding symbol of the i-th column, respectively.

For the ease of exposition, we represents each column of the code as a set with 4 elements where the first element corresponds to the entry from first row and the rest of the elements corresponds to the variables of second row.
$\{v_1,\{\lambda_1,\mu_1,\phi_1\}\}$, $\{v_2,\{\lambda_2,\mu_2,\phi_2\}\}$, $\{v_3,\{\lambda_3,\mu_3,\phi_3\}\}$,
$\{v_4,\{\lambda_4,\mu_4,\phi_4\}\}$, $\{v_5,\{\lambda_5,\mu_5,\phi_5\}\}$, $\{v_6,\{\lambda_6,\mu_6,\phi_6\}\}$.

W.l.o.g (by symmetry), we may put v_1 in the columns $2,3,4$ that is, the code is partially filled as:
$\{v_1,\{\lambda_1,\mu_1,\phi_1\}\}$, $\{v_2,\{v_1,\mu_2,\phi_2\}\}$, $\{v_3,\{v_1,\mu_3,\phi_3\}\}$,
$\{v_4,\{v_1,\mu_4,\phi_4\}\}$, $\{v_5,\{\lambda_5,\mu_5,\phi_5\}\}$, $\{v_6,\{\lambda_6,\mu_6,\phi_6\}\}$.

Since the first three tuple should recover v_4, v_5, v_6 the code could be further filled as (this is one candidate, for other candidates, it could be analyzed similarly)



(a) Single-Column-Erasure **(b) Two-Column-Erasure** **(c) Three-Column-Erasure**

Figure 2: Average Runtime Required to Decode and Reconstruct Erased Columns for Three-Column-Erasure Tolerating Codes.

v_1	v_2	v_3	v_4	v_5	v_6
$\lambda_1 \oplus \mu_1 \oplus \phi_1$	$\lambda_2 \oplus \mu_2 \oplus \phi_2$	$\lambda_3 \oplus \mu_3 \oplus \phi_3$	$\lambda_4 \oplus \mu_4 \oplus \phi_4$	$\lambda_5 \oplus \mu_5 \oplus \phi_5$	$\lambda_6 \oplus \mu_6 \oplus \phi_6$

Table 6: Construction of $[6,3]$, 2×6 BP-XOR Code.

$\{v_1,\{v_6,\mu_1,\phi_1\}\}$, $\{v_2,\{v_1,v_4,\phi_2\}\}$, $\{v_3,\{v_1,v_5,\phi_3\}\}$,
$\{v_4,\{v_1,\mu_4,\phi_4\}\}$, $\{v_5,\{\lambda_5,\mu_5,\phi_5\}\}$, $\{v_6,\{\lambda_6,\mu_6,\phi_6\}\}$.

Now in order to make the first 3-tuple BP-decodable, we need to fill it as (one candidate, other candidate could be analyzed similarly):
$\{v_1,\{v_6,\mu_1,\phi_1\}\}$, $\{v_2,\{v_1,v_4,v_3\}\}$, $\{v_3,\{v_1,v_5,\phi_3\}\}$,
$\{v_4,\{v_1,\mu_4,\phi_4\}\}$, $\{v_5,\{\lambda_5,\mu_5,\phi_5\}\}$, $\{v_6,\{\lambda_6,\mu_6,\phi_6\}\}$.

If we consider $\{v_2,\{v_1,v_4,v_3\}\}$,$\{v_3,\{v_1,v_5,\phi_3\}\}$, $\{v_4,\{v_1,\mu_4,\phi_4\}\}$, it is clear that $\phi_3 = v_6$ or $\mu_4 = v_6$. In the following, we show that for either case, BP decoding can not work. First assume that $\phi_3 = v_6$, i.e., we have
$\{v_1,\{v_6,\mu_1,\phi_1\}\}$, $\{v_2,\{v_1,v_4,v_3\}\}$, $\{v_3,\{v_1,v_5,v_6\}\}$,
$\{v_4,\{v_1,\mu_4,\phi_4\}\}$, $\{v_5,\{\lambda_5,\mu_5,\phi_5\}\}$, $\{v_6,\{\lambda_6,\mu_6,\phi_6\}\}$.

Now we consider $\{v_2,\{v_1,v_4,v_3\}\}$,$\{v_3,\{v_1,v_5,v_6\}\}$,$\{v_6,\{\lambda_6,\mu_6,\phi_6\}\}$. Simple analysis shows that $\{\lambda_6,\mu_6,\phi_6\} = \{v_2,v_3,v_4\}$ or $\{\lambda_6,\mu_6,\phi_6\} = \{v_2,v_3,v_5\}$.

Assume that $\{\lambda_6,\mu_6,\phi_6\} = \{v_2,v_3,v_4\}$ (the other case analysis is similar) then we have
$\{v_1,\{v_6,\mu_1,\phi_1\}\}$, $\{v_2,\{v_1,v_4,v_3\}\}$, $\{v_3,\{v_1,v_5,v_6\}\}$,
$\{v_4,\{v_1,\mu_4,\phi_4\}\}$, $\{v_5,\{\lambda_5,\mu_5,\phi_5\}\}$, $\{v_6,\{v_2,v_3,v_4\}\}$.

If we consider the tuples $\{v_1,\{v_6,\mu_1,\phi_1\}\}$, $\{v_2,\{v_1,v_4,v_3\}\}$, $\{v_6,\{v_2,v_3,v_4\}\}$, then we have $\mu_1 = v_5$. However, for any value ϕ_1, the code will not decode.

Note that for the case $\{\lambda_6,\mu_6,\phi_6\} = \{v_2,v_3,v_5\}$, we must have $\lambda_5 = v_6$, that is, we have
$\{v_1,\{v_6,\mu_1,\phi_1\}\}$, $\{v_2,\{v_1,v_4,v_3\}\}$, $\{v_3,\{v_1,v_5,v_6\}\}$,
$\{v_4,\{v_1,\mu_4,\phi_4\}\}$, $\{v_5,\{v_6,\mu_5,\phi_5\}\}$, $\{v_6,\{v_2,v_3,v_5\}\}$.

In this case, we consider the tuples:
$\{v_2,\{v_1,v_4,v_3\}\}$,$\{v_3,\{v_1,v_5,v_6\}\}$,$\{v_5,\{v_6,\mu_5,\phi_5\}\}$. Simple analysis shows that $\{\mu_5,\phi_5\} = \{v_2,v_3\}$, i.e., we have
$\{v_1,\{v_6,\mu_1,\phi_1\}\}$, $\{v_2,\{v_1,v_4,v_3\}\}$, $\{v_3,\{v_1,v_5,v_6\}\}$,
$\{v_4,\{v_1,\mu_4,\phi_4\}\}$, $\{v_5,\{v_6,v_2,v_3\}\}$, $\{v_6,\{v_2,v_3,v_5\}\}$.

Now we got a contradiction that we can not put v_4 in 4 different column which is necessary for this code to be an MDS code.

In the following, we consider the case for $\mu_4 = v_6$ for which we have
$\{v_1,\{v_6,\mu_1,\phi_1\}\}$, $\{v_2,\{v_1,v_4,v_3\}\}$, $\{v_3,\{v_1,v_5,\phi_3\}\}$,
$\{v_4,\{v_1,v_6,\phi_4\}\}$, $\{v_5,\{\lambda_5,\mu_5,\phi_5\}\}$, $\{v_6,\{\lambda_6,\mu_6,\phi_6\}\}$.

Since v_6 must appears 4 times in 4 columns, we have
$\{v_1,\{v_6,\mu_1,\phi_1\}\}$, $\{v_2,\{v_1,v_4,v_3\}\}$, $\{v_3,\{v_1,v_5,\phi_3\}\}$,
$\{v_4,\{v_1,v_6,\phi_4\}\}$, $\{v_5,\{\lambda_5,\mu_5,\phi_5\}\}$, $\{v_6,\{\lambda_6,\mu_6,\phi_6\}\}$.

Consider tuples $\{v_1,\{v_6,\mu_1,\phi_1\}\}$,$\{v_5,\{\lambda_5,\mu_5,\phi_5\}\}$,$\{v_6,\{\lambda_6,\mu_6,\phi_6\}\}$. Since v_1 cannot show up anymore, we must have $\mu_1 = v_5$, that is, the code is:
$\{v_1,\{v_6,v_5,\phi_1\}\}$, $\{v_2,\{v_1,v_4,v_3\}\}$, $\{v_3,\{v_1,v_5,\phi_3\}\}$,
$\{v_4,\{v_1,v_6,\phi_4\}\}$, $\{v_5,\{\lambda_5,\mu_5,\phi_5\}\}$, $\{v_6,\{\lambda_6,\mu_6,\phi_6\}\}$.

Now let us consider the tuple $\{v_1,\{v_6,v_5,\phi_1\}\}$, $\{v_3,\{v_1,v_5,\phi_3\}\}$, $\{v_4,\{v_1,v_6,\phi_4\}\}$ for which we must select $\phi_3 = v_4$ or $\phi_4 = v_3$.

For $\phi_3 = v_4$, we have
$\{v_1,\{v_6,v_5,\phi_1\}\}$, $\{v_2,\{v_1,v_4,v_3\}\}$, $\{v_3,\{v_1,v_5,v_4\}\}$
$\{v_4,\{v_1,v_6,\phi_4\}\}$, $\{v_5,\{\lambda_5,\mu_5,\phi_5\}\}$, $\{v_6,\{\lambda_6,\mu_6,\phi_6\}\}$.

Since v_6 must appears 4 times, we have
$\{v_1,\{v_6,v_5,\phi_1\}\}$, $\{v_2,\{v_1,v_4,v_3\}\}$, $\{v_3,\{v_1,v_5,v_4\}\}$,
$\{v_4,\{v_1,v_6,\phi_4\}\}$, $\{v_5,\{v_6,\mu_5,\phi_5\}\}$, $\{v_6,\{\lambda_6,\mu_6,\phi_6\}\}$.

Now the tuple $\{v_2,\{v_1,v_4,v_3\}\}$,$\{v_3,\{v_1,v_5,v_4\}\}$,$\{v_5,\{v_6,\mu_5,\phi_5\}\}$ have trouble to decode.

If we select $\phi_4 = v_3$ then
$\{v_1,\{v_6,v_5,\phi_1\}\}$, $\{v_2,\{v_1,v_4,v_3\}\}$, $\{v_3,\{v_1,v_5,\phi_3\}\}$,
$\{v_4,\{v_1,v_6,v_3\}\}$, $\{v_5,\{\lambda_5,\mu_5,\phi_5\}\}$, $\{v_6,\{\lambda_6,\mu_6,\phi_6\}\}$.

Since v_5 must appear 4 times, we have
$\{v_1,\{v_6,v_5,\phi_1\}\}$, $\{v_2,\{v_1,v_4,v_3\}\}$, $\{v_3,\{v_1,v_5,\phi_3\}\}$,
$\{v_4,\{v_1,v_6,v_3\}\}$, $\{v_5,\{\lambda_5,\mu_5,\phi_5\}\}$, $\{v_6,\{v_5,\mu_6,\phi_6\}\}$.

But with this placement of v_5, tuple $\{v_2,\{v_1,v_4,v_3\}\}$, $\{v_4,\{v_1,v_6,v_3\}\}$,$\{v_6,\{v_5,\mu_6,\phi_6\}\}$ can not decode.

In a summary, we observe from the above that when we try to fulfil the necessary condition for BP-XOR decoding to start for a selected group of three columns, the same condition is not fulfilled

for other set of three columns. Hence, we conclude that there is no BP-XOR code for $n = 6, k = 3, b = 2, \sigma = 3$.

B PROOF OF THE FACT 2

For any three-column-erasure pattern, three information symbols are available directly through three degree one encoding symbols. Further, due to construction of the code using Algorithm 1, all information symbols appear in three degree three encoding symbols and once in degree one encoding symbol (but no information symbol can occur in same column twice). The worst case scenario occurs when for a given three-column-erasure pattern, an information symbol occurs only once in a degree one encoding symbol out of the six available encoding symbols (such combinations, one for each information symbol and hence six in total, in general causes BP-XOR decoder to fail for such codes). For such erasure patterns, the three unknown information symbols may appear together in degree three encoding symbol (following analysis is still valid if all the available degree three encoding symbols have at least one information symbol directly available through degree one encoding symbol). Since all degree three encoding symbols are different (as otherwise generator matrix can not be full rank), the other two degree three encoding symbols must contain at least one known information symbol (which is available through a degree one encoding symbol). Hence, solving one of the two unknown information symbol in such a degree three encoding symbol using the corresponding solution (derived from the inverse of the submatrix mentioned above), allows the BP-XOR decoder to decode the other unknown information symbol. Now the remaining degree three encoding symbol can contain only one unknown information symbol which can again be decoded using BP-XOR. Hence, in any pattern of three-column-erasures, only one unknown information symbol needs to be determined through the corresponding solution and rest of the information symbols can be decoded using BP-XOR decoder.

ACKNOWLEDGMENT

This publication was made possible by the NPRP award NPRP8-2158-1-423 from the Qatar National Research Fund (a member of The Qatar Foundation). The statements made herein are solely the responsibility of the authors

REFERENCES

[1] David A. Patterson, Garth Gibson, and Randy H. Katz. 1988. A case for redundant arrays of inexpensive disks (RAID). In Proceedings of the 1988 ACM SIGMOD international conference on Management of data (SIGMOD '88), Haran Boral and Per-Ake Larson (Eds.). ACM, New York, NY, USA, 109-116. DOI=http://dx.doi.org/10.1145/50202.50214

[2] Adam Leventhal. 2009. Triple-Parity RAID and Beyond. Queue 7, 11, Pages 30 (December 2009), 10 pages. DOI: https://doi.org/10.1145/1661785.1670144

[3] Mario Blaum, Jim Brady, Jehoshua Bruck, and Jai Menon. 1995. EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures. IEEE Trans. Comput. 44, 2 (February 1995), 192-202. DOI=http://dx.doi.org/10.1109/12.364531

[4] Peter Corbett, Bob English, Atul Goel, Tomislav Grcanac, Steven Kleiman, James Leong, and Sunitha Sankar. 2004. Row-diagonal parity for double disk failure correction. In Proceedings of the 3rd USENIX conference on File and storage technologies (FAST'04). USENIX Association, Berkeley, CA, USA, 1-1.

[5] Yongge Wang. Array BP-XOR codes for reliable cloud storage systems. In *Proc. of the 2013 IEEE International Symposium on Information Theory* (ISIT), pp. 326–330, Istanbul, Turkey, July 2013.

[6] Michael Luby. 2002. LT Codes. In Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS '02). IEEE Computer Society, Washington, DC, USA, 271–280.

[7] Chip Walter. 2005. Kryder's Law. *Scientific American*. http://www.scientificamerican.com/article. cfm?id=kryders-law.

[8] Daniel Ford, François Labelle, Florentina I. Popovici, Murray Stokely, Van-Anh Truong, Luiz Barroso, Carrie Grimes, and Sean Quinlan. 2010. Availability in globally distributed storage systems. In Proceedings of the 9th USENIX conference on Operating systems design and implementation (OSDI'10). USENIX Association, Berkeley, CA, USA, 61-74.

[9] M. Blaum, J. Bruck, and A. Vardy. 2006. MDS array codes with independent parity symbols. IEEE Trans. Inf. Theor. 42, 2 (September 2006), 529-542. DOI=http://dx.doi.org/10.1109/18.485722

[10] Cheng Huang and Lihao Xu. 2005. STAR: an efficient coding scheme for correcting triple storage node failures. In Proceedings of the 4th conference on USENIX Conference on File and Storage Technologies - Volume 4 (FAST'05), Vol. 4. USENIX Association, Berkeley, CA, USA, 15-15.

[11] Atul Goel and Peter Corbett. 2012. RAID triple parity. SIGOPS Oper. Syst. Rev. 46, 3 (December 2012), 41-49. DOI=http://dx.doi.org/10.1145/2421648.2421655

[12] M. Blaum and R. M. Roth. 2006. On lowest density MDS codes. IEEE Trans. Inf. Theor. 45, 1 (September 2006), 46-59. DOI=http://dx.doi.org/10.1109/18.746771

[13] E. Louidor and R. M. Roth. 2006. Lowest density MDS codes over extension alphabets. IEEE Trans. Inf. Theor. 52, 7 (July 2006), 3186-3197. DOI=http://dx.doi.org/10.1109/TIT.2006.876235

[14] Garth A. Gibson and David A. Patterson. 1993. Designing disk arrays for high data reliability. J. Parallel Distrib. Comput. 17, 1-2 (January 1993), 4-27. DOI=http://dx.doi.org/10.1006/jpdc.1993.1002

[15] Alexandros G. Dimakis, P. Brighten Godfrey, Yunnan Wu, Martin J. Wainwright, and Kannan Ramchandran. 2010. Network coding for distributed storage systems. IEEE Trans. Inf. Theor. 56, 9 (September 2010), 4539-4551. DOI=http://dx.doi.org/10.1109/TIT.2010.2054295

[16] Yongge Wang and Yvo Desmedt. 2011. Edge-colored graphs with applications to homogeneous faults. Inf. Process. Lett. 111, 13 (July 2011), 634-641. DOI=http://dx.doi.org/10.1016/j.ipl.2011.03.017

[17] Eric Mendelsohn and Alexander Rosa. 1985. One-factorizations of the complete graph – a survey. *Journal of Graph Teory*, 9(1):43–65. DOI=http://10.1002/jgt.3190090104

[18] N. V. Semakov, G. V. Zaitsev, V. A. Zinov'ev. 1983. Minimum-check-density codes for correcting bytes of errors, erasures,or defects. *Problems Inform. Transmission*, 19(3):197–204.

[19] Yongge Wang. 2015. Privacy-Preserving Data Storage in Cloud Using Array BP-XOR Codes. *IEEE Transactions on Cloud Computing*, vol. 3, no. 4, pp. 425–435. DOI=http://10.1109/TCC.2014.2344662

[20] M. Blaum, and R. M. Roth. 1993. New Array Codes for Multiple Phased Burst Correction. *IEEE Transactions on Information Theory*, vol. 39, no. 1, pp.66–77. DOI=http://10.1109/18.179343

[21] M. Blaum. 2006. A family of MDS array codes with minimal number of encoding operations. In *Proc. IEEE International Symposium on Information Theory*, pp. 2784–2788. DOI=http://10.1109/ISIT.2006.261569

[22] Z. Huang, H. Jiang and K. Zhou. 2016. An Improved Decoding Algorithm for Generalized RDP Codes. *IEEE Communications Letters*, vol. 20, no. 4, pp. 632–635. DOI=http://10.1109/LCOMM.2016.2522414

[23] Maura B. Paterson, Douglas R. Stinson, and Yongge Wang. 2016. On encoding symbol degrees of array BP-XOR codes. *Cryptography and Communications*, vol. 8, no. 1, pp. 19–32. DOI=https://doi.org/10.1007/s12095-015-0134-9

[24] I. S. Reed, G. Solomon. 1960. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300-304. DOI=https://doi.org/10.1137/0108018

[25] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. 2003. The Google file system. SIGOPS Oper. Syst. Rev. 37, 5 (October 2003), 29-43. DOI=http://dx.doi.org/10.1145/1165389.945450

[26] Chao Jin, Dan Feng, Hong Jiang, and Lei Tian. 2011. A Comprehensive Study on RAID-6 Codes: Horizontal vs. Vertical. In Proceedings of the 2011 IEEE Sixth International Conference on Networking, Architecture, and Storage (NAS '11). IEEE Computer Society, Washington, DC, USA, 102-111. DOI=http://dx.doi.org/10.1109/NAS.2011.31

[27] Pradeep Subedi and Xubin He. 2013. A Comprehensive Analysis of XOR-Based Erasure Codes Tolerating 3 or More Concurrent Failures. In Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing Workshops and PhD Forum (IPDPSW '13). IEEE Computer Society, Washington, DC, USA, 1528-1537. DOI=http://dx.doi.org/10.1109/IPDPSW.2013.155