Constructing Liberation Codes Using Latin Squares*

Wang Gang, Liu Xiaoguang, Lin Sheng, Xie Guangjun, Liu Jing
Department of Computer, College of Information Technology Science, Nankai University
wgzwp@163.com

Abstract

In recent years, multi-erasure correcting coding systems have become more pervasive. RAID6 is an important 2-erasure correcting code specification. But there is no consensus on the best concrete RAID6 coding scheme. Plank developed a brand new class of RAID6 codes called the Liberation codes that achieves good encoding, updating and decoding performance. In this paper, we present a chained decoding algorithm for the Liberation codes. Its performance is comparable with the bit matrix scheduling algorithm developed by Plank, but is more intuitive and reveals the essence better. In the process, we present a new class of Liberation codes called the Latin Liberation codes. These codes are based on column-hamiltonian Latin squares, hence the name. They are superior to the Liberation codes in parameter flexibility and structure flexibility. Finally, we analyze the performance of several XOR-based RAID6 codes and give some suggestion on their application.

1. Introduction

In recent years, as hard disks have grown greatly in size and storage systems have grown in size and complexity, it is more frequent that a failure of one disk occurs in tandem with unrecovered failures of other disks or latent failures of blocks on other disks. On a system using single-erasure correcting code such as RAID5, this combination of failures leads to a permanent data loss [1]. Hence, applications of multi-erasure correcting codes have become more pervasive. RAID6 is a 2-erasure correcting code specification [2]. Several concrete RAID6 schemes have been developed, and some are applied successfully. This paper presents a chained decoding algorithm for the Liberation codes - an important RAID6 scheme. Constructing Liberation codes by Latin squares is also discussed in this paper.

The outline of this paper is as follows. In Section 2 we discuss the known RAID6 codes. Section 3 is devoted to introducing some related combinatorics knowledge. In Section 4 we present the chained decoding algorithm, and the Latin Liberation codes are presented in Section 5. A theoretical analysis is discussed in Section 6. Conclusions and future works are presented in Section 7.

	disk0	disk1	disk2	disk3	disk4
stripe0	D	D	D	P	Q
stripe1	D	P	Q	D	D
stripe2	Q	D	D	D	P
stripe3	D	D	P	Q	D
stripe4	P	Q	D	D	D
…	…				

Figure 1. **A typical RAID6 system.**

2. Current RAID6 codes

An erasure code for storage systems is a scheme that encodes the content on n data disks into m check disks so that the system is resilient to any t device failures [3]. Unfortunately, there is no consensus on the best coding technique for n, m, $t > 1$. RAID6 is a specification for $m=t=2$. A typical RAID6 system appears as depicted in Fig 1. Each disk is divided into fixed-size blocks (stripe units). All blocks with the same in-disk offset are organized into a stripe. Each stripe contains n data units and 2 check units P and Q. The identity of the data and check disks is rotated every stripe for small write load balance. A stripe is a self-contained 2-erasure correcting entity and the whole layout is just the cyclic repetition of a stripe, so we can focus only on single stripe when design a RAID6 code. Generally a RAID6 code should be a MDS code.

The best known RAID6 codes are Reed-Solomon codes [4]. They are based on Galois Field, thus the computational complexity is a serious problem though

* This paper is supported partly by the National High Technology Research and Development Program of China (2008AA01Z401), NSFC of China (90612001), RFDP of China (20070055054), and Science and Technology Development Plan of Tianjin (08JCYBJC13000)

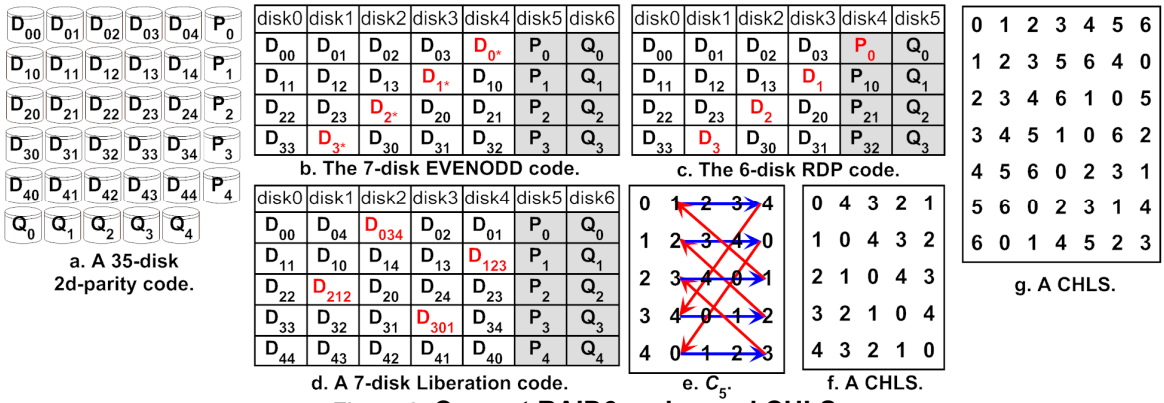


Figure 2. **Current RAID6 codes and CHLS.**

optimized algorithms have been developed [5][6].

Another category is so-called array codes that design a stripe as an array of data and parity symbols. In this paper, we focus on the horizontal codes, such as EVENODD [7], RDP [1] and Liberation [8]. "*Horizontal*" means that some disks contain nothing but data symbols and the others contain only parity symbols. "*Vertical*" means that the data and parity symbols are stored together. Horizontal codes fit RAID6 specification well, but vertical codes don't. So some important vertical codes, such as X-Code [9], B-Code [10], are not within the scope of this paper. STAR Code [11] is a 3-erasure horizontal code, it boils down to EVENODD when applied to RAID6 scenarios; Pyramid Code [12] is not MDS; WEAVER Codes [13] and HoVer Codes [14] are not MDS too and are not horizontal. So they are all out of our sight.

Fig 2.b shows the 7-disk EVENODD code. The standard EVENODD code with p+2 disks consists of a (p-1)*p data array and a (p-1)*2 parity array. Note that the whole array instead of each row corresponds to a stripe in Fig 1, and each column instead of each symbol corresponds to a stripe unit. The "disk P" stores horizontal parity symbols and the "disk Q" stores skew diagonal parity symbols. D_{ij} denotes the data symbol that participates in P_i and Q_j. D_{i*} participates in P_i and all Qs.

Array codes can be regarded as layouts of binary linear codes. For example, the symbols in the 7-disk EVENODD code are just the symbols (except P_4, Q_4 and D_{40}~D_{44}) of the 35-disk 2d-parity code [15] shown in Fig 2.a (D_{i*}=D_{i4}). P_4 and D_{40}~D_{44} are deleted for 2-erasure correcting. Q_4 is deleted and the sum (over GF[2], namely XOR operation, the same hereinafter) S of its sons is added to every Q for MDS. Therefore, the computational performance is not optimal and each column instead of each symbol must be implemented as a stripe unit to achieve optimal update penalty. These properties are the inherent limitations of the MDS horizontal codes [16][17].

Fig 2.c shows the 6-disk RDP code. The standard p+1-disk RDP code can be described by a (p-1)*(p+1) code array. The disk P stores horizontal parity symbols and the disk Q stores skew diagonal parity symbols, too. RDP codes also can be constructed by clipped 2d-parity codes. Its strategy for the deleted Q parity symbol is "parity dependent" - some P symbols have the second role as a data member of some Q.

Fig 2.d shows the 7-disk Liberation code. It is also based on the 2d-parity code shown in Fig 2.a. A standard Liberation code can be described by a p*(p+2) code array. Unlike EVENODD and RDP, it adopts "addition" instead of "deletion". The original 2d-parity code is well-preserved and some data symbols (D_{ijk}) are arranged to participate in one extra Q group (Q_k). Plank's construction method lets the i^{th} symbol in the j^{th} disk participate in P_i and $Q_{(i-j) \bmod p}$. When j is odd, the $(\frac{p+j}{2}-1)^{th}$ data symbol is the "*3-group*" symbol and its 2nd Q group is the $(\frac{p-j}{2})^{th}$ Q group. When j is even (>0), the 3-group symbol is in row $\frac{j}{2}-1$ and it joins the $(p-\frac{j}{2})^{th}$ Q group.

Note that the parameter p must be a prime for EVENODD, RDP and Liberation. This leads to bad parameter flexibility and implementation problems. We have tried to generalize EVENODD and RDP using Latin squares [18][19]. In this paper, we apply this technique to the Liberation codes.

3. Related combinatorics knowledge

3.1. Graph representation of 2-erasure codes

Some literature refers to simple graph representation of parity independent 2-erasure linear

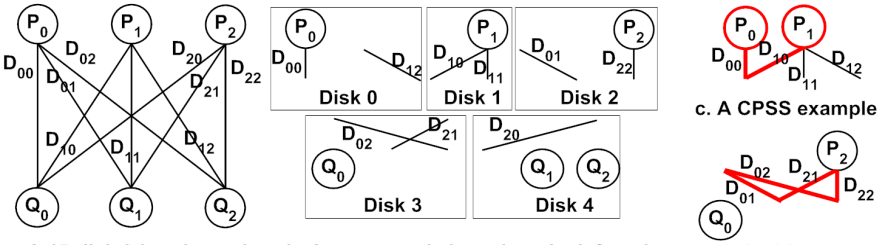


Figure 3. **Simple graph representation.**

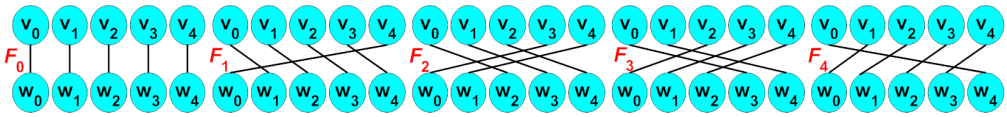


Figure 4. **A P1F of $K_{5,5}$.**

codes in which each data symbol participates in exactly two parity groups [10][15][20]: let each vertex denote a parity symbol (group) and each edge denote a data symbol - the two endpoints of an edge are just the two parity symbols of the data symbol. So an array code can be described by a graph partition if the underlying linear code can be described by a simple graph. We have proven the following theorem [20]:

Theorem 1. If an array code can be described by a partition of a simple graph, it is 2-erasure correcting iff the union of any pair of sub-graphs of the partition doesn't contain the following two types of structures:

1. A path and its two endpoints. We call this kind of unrecoverable erasure ***C**losed **P**arity **S**ymbols **S**ubset*, CPSS for short.
2. A cycle. We call it CDSS - ***C**losed **D**ata **S**ymbols **S**ubset*.

Fig 3.a shows the graph that corresponds to a 15-disk 2d-parity code. Fig 3.b shows an array code based on it. Fig 3.c gives a CPSS that corresponds to the unrecoverable 2-erasure (disk0, disk1), and Fig 3.d shows a CDSS that corresponds to the unrecoverable 2-erasure (disk2, disk3).

3.2. Perfect one-factorizations

A *one-factor* of a graph G is a set of edges in which every vertex appears exactly once. A *one-factorization* of G is a partition of the edge-set of G into one-factors. A *perfect one-factorization* (P1F) is a one-factorization in which every pair of distinct one-factors forms a Hamiltonian cycle. There is a widely believed conjecture in graph theory: every complete graph with an even number of vertices has a P1F [21]. Fig 4 shows a P1F of $K_{5,5}$.

3.3. Latin squares

An $n*n$ *Latin square* L is an $n*n$ matrix of entries chosen from some set of symbols of cardinality n, so that no symbol is duplicated within any row or any column. We select $Z_n=\{0, 1, \ldots, n\text{-}1\}$ as the symbol set, it is also can be used as the row and column number set. The symbol in row r, column c of L is denoted by L_{rc}. A Latin square of order n can be described by a set of n^2 triples of the form (*row*, *column*, *symbol*).

Each row r of a Latin square L is the image of some permutation σ_r of Z_n, namely $L_{ri}=\sigma_r(i)$. Each pair of rows (r; s) defines a permutation by $\sigma_{r,s}=\sigma_r\sigma_s^{-1}$. If $\sigma_{r,s}$ consists of a single cycle for each pair of rows (r, s) in a Latin square L, we say L is *row-hamiltonian*. Similar concepts can be defined in terms of the *column* and *symbol*. In this paper, we are concerned with column-hamiltonian Latin squares, CHLS for short. Fig 2.e shows a CHLS of order 5, and $\sigma_{1,4}$ of it. It is just the Cayley table C_5 of the cyclic group of order 5. In fact, C_p is a CHLS when p is a prime number.

There is a CHLS L of order n iff $K_{n,n}=(V,W,E)$ has a P1F $F=\{F_0, \ldots, F_{n-1}\}$ [21]. To show this, we create three one-to-one correspondence: between the row set and V, between the symbol set and W, and between the column set and F. Namely, (i, j, k) $(\in L)$ corresponds to the edge (v_i, w_k) in F_j. Obviously, the cycle pattern in $\sigma_{r,s}$ in L corresponds to that in $F_r \cup F_s$. The P1F shown in Fig 4 corresponds to C_5. There is another conclusion [21]: if K_{n+1} has a P1F, then so does $K_{n,n}$. Thus we have a conjecture: $K_{n,n}$ has a P1F (CHLS of order n exists) for n=2 and all odd positive integers n. Graph theorists have proven that all even(odd) numbers less than 54(53) are "K_n P1F numbers" (CHLS/$K_{n,n}$ P1F numbers) and have found many larger K_n P1F numbers (CHLS/$K_{n,n}$ P1F numbers).

4. The chained decoding algorithm

An 2-erasure array code can be described by a partition, a P1F is just a partition, and there is a bijection between CHLS and P1F of $K_{n,n}$. Thus a

natural idea is constructing 2-erasure array codes by CHLS. In [18] and [19], we have tried this idea. Two algorithms are developed to construct EVENODD-like codes and RDP-like codes respectively by CHLS. We call the first kind of codes PIHLatin codes (Parity Independent Horizontal Latin codes), and the second kind PDHLatin codes (Parity Dependent Horizontal Latin codes). The key ideas of the two algorithms are similar: each column of the CHLS is used to construct a disk; one row is deleted to break the Hamiltonian cycles induced by disk pairs - CDSS are avoided; finally, parity symbols are arranged properly to avoid CPSS. Therefore, 2-erasure correcting is guaranteed.

PIHLatin and PDHLatin are superior to EVENODD and RDP in parameter flexibility because the distribution of P1F numbers is far denser than that of prime numbers. Although horizontal shortening (deleting some data disks - assuming they contain nothing but zeros) can alleviate EVENODD and RDP's problem, we have shown that it is harmful to encoding/decoding/updating performance [18][19]. The PIHLatin and PDHLatin codes constructed by C_5 are respectively the 7-disk EVENODD code and the 6-disk RDP code shown in Fig 2. In fact, PIHLatin and PDHLatin are the supersets of EVENODD and RDP respectively. We have shown that the relationship is proper superset [18][19].

Liberation codes can be described by Latin squares, too. Writing down the (first) Q index of every data symbol, we get a CHLS. For example, the 7-disk Liberation code shown in Fig 2.d corresponds to the CHLS shown in Fig 2.f. This CHLS is an isotopy of C_5 (constructed by performing column swapping on C_5). Examining Plank's construction method [8], we can see that all "Liberation CHLS" are of this kind.

The Q index matrix of a Liberation code is a complete CHLS. This means that any pair of disks induces a Hamiltonian cycle (CDSS). So how do Liberation codes tolerant all 2-erasures? The answer is the 3-group data symbols. Plank has proven that the Liberation codes are 2-erasure correcting by matrix description. He also presented a decoding algorithm named bit matrix scheduling algorithm [8]. But it's hard to understand the interior mechanism through matrix description. There is a simple and intuitive decoding algorithm for PIHLatin and PDHLatin codes. Suppose that a 2-erasure (disk1, disk2) occurs in a 6-disk RDP coding system. All horizontal parity groups and the 2nd and 3th skew diagonal parity groups lose exactly two symbols, we can't recover them directly. But Q_0 and Q_1 lose only one symbol, we can start decoding from them. D_{01} can be reconstructed by summing all surviving symbols in Q_1 first, then D_{02} is reconstructed using the horizontal parity group P_0, then D_{12} is reconstructed through Q_2, and then D_{13} is recovered by P_1, and so on, until D_2 is recovered by P_2. Similarly, D_{30} and D_3 are reconstructed along another chain. Graph representation describes this algorithm visually. A reconstruction chain corresponds to an open path (the opposite of CPSS and CDSS) in the union of the failed disks (sub-graphs), and its starting point (D_{01}, D_{30}) corresponds to the end edge of the open path. Now we present a chained decoding algorithm for Liberation codes. We illustrate its correctness by several examples instead of a strict formal proof.

The reconstruction of single-erasures is trivial, so we focus on four kinds of 2-erasures.

The Two Parity Disks Fail

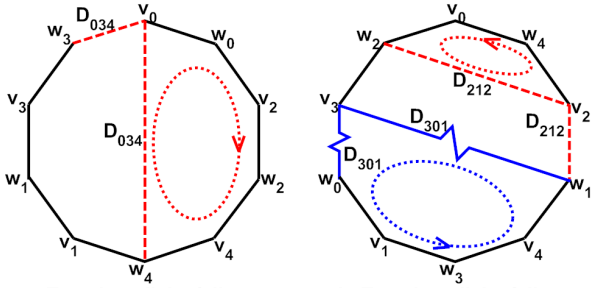
In this case, decoding equals to encoding.

The Disk *Q* and a Data Disk Fail

Decodes the data disk using horizontal parity groups, and then encodes the Disk Q.

The Disk *P* and a Data Disk Fail

We can decode the data disk using Q parity groups. There is at most one 3-group data symbol in a data disk. Thus if the failed data disk contains a D_{ijk}, we recover it through Q_j first, and then recover $D_{i'k}$ through Q_k (all other symbols in Q_k including D_{ijk} are prepared now). And then we can encode the disk P.



a. Two data disks failure including the first disk

b. Two data disks failure excluding the first disk

Figure 5. **Chained decoding algorithm.**

The First Data Disk and another Data Disk Fail

The union of the two failed disks composes a cycle with a chord induced by the 3-group symbol. For example, Fig 5.a shows the sub-graph that corresponds to the 2-erasure (disk0, disk2) in a 7-disk Liberation coding system. The chord increases the degrees of v_0 and w_4 to 3. But v_0 touches only 2 edges because the two dashed lines both denote D_{034}. w_4 touches really 3 edges. We call the former *fake 3-vertex* and the latter *real 3-vertex*. Examining the right sub-cycle, every involved parity groups (vertices) loses (touches) exactly 2 symbols (edges) except Q_4 (w_4) that loses 3 symbols: D_{14}, D_{44} and D_{034}. So, we can reconstruct the "tail" D_{14} by summing all surviving symbols in these

groups. This process can be described formally as follow. $\widetilde{P}_i$ is defined as the sum of the surviving symbols in P_i and $\widetilde{Q}_i$ as the sum of the surviving symbols in Q_i. $\widetilde{P}_i$ ($\widetilde{Q}_i$) also equals to the sum of the lost symbols in the group. Huang et al gave them the name *syndromes* [11]. So we have:

$$\begin{aligned}
&\widetilde{P}_0 = D_{034} \oplus D_{00} \quad \widetilde{Q}_0 = D_{00} \oplus D_{20} \quad \widetilde{P}_2 = D_{20} \oplus D_{22} \\
&\widetilde{Q}_2 = D_{22} \oplus D_{42} \quad \widetilde{P}_4 = D_{42} \oplus D_{44} \\
&\widetilde{Q}_4 = D_{44} \oplus D_{14} \oplus D_{034} \\
&\Rightarrow D_{14} = \widetilde{P}_0 \oplus \widetilde{Q}_0 \oplus \widetilde{P}_2 \oplus \widetilde{Q}_2 \oplus \widetilde{P}_4 \oplus \widetilde{Q}_4
\end{aligned} \tag{1}$$

We call this process *cycle resolving* and the traversal path *resolving path*. The dashed oval in Fig 5.a shows the resolving path described by formula 1. After the resolving process, we can recover all lost symbols clockwise from D_{14}.

A feasible resolving path should be a "Q shape" - a cycle with a tail edge. Moreover, it must pass one and only one real 3-vertex that is just the connection point between the cycle and the tail. This structure guarantees that all edges are touched precisely twice except the tail only once. Therefore summing all the syndromes decodes the tail symbol. In addition, chained decoding should be performed along the tail direction after cycle resolving. The decoding path must pass a fake 3-verrtex before pass its corresponding real 3-vertex. Otherwise, the decoding process will have to stop at the real 3-vertex. Thus, the left sub-cycle in Fig 5.a is also a feasible resolving path, but the decoding direction is anticlockwise. We can see that a resolving path always exists in this case, and chained decoding always complete.

The syndromes seem to be calculated twice, one during cycle resolving and another during chained decoding. Therefore the decoding cost is far away from the optimal cost. But in fact, we can store the syndromes in the memory for the lost symbols once they are calculated during cycle resolving. Then they can be fetched and put into calculation immediately during chained decoding. Therefore, near-optimal performance is still guaranteed.

Two Data Disks (Excluding the First Disk) Fail

Fig 5.b shows the sub-graph that corresponds to the 2-erasure (disk1, disk3) in a 7-disk Liberation coding system. The dashed chord denotes the 3-group symbol D_{212} and the wavy chord denotes D_{145}. The decoding process is similar to the last case. Formula 2 shows a resolving path that decodes D_{32}. Another path decodes D_{212} as Formula 3 shows.

Fig 5.b is the only recoverable structure. The real and fake 3-vertices must be interleaved on the cycle. Otherwise, the two tails point to each other inevitably. Thus the decoding process from any real 3-vertex must quit halfway at another real 3-vertex. We will show that Plank's method guarantees this structure.

$$\begin{aligned}
&\widetilde{P}_2 = D_{212} \oplus D_{24} \quad \widetilde{Q}_4 = D_{24} \oplus D_{04} \quad \widetilde{P}_0 = D_{04} \oplus D_{02} \\
&\widetilde{Q}_2 = D_{02} \oplus D_{212} \oplus D_{32} \Rightarrow \quad D_{32} = \widetilde{P}_2 \oplus \widetilde{Q}_4 \oplus \widetilde{P}_0 \oplus \widetilde{Q}_2
\end{aligned} \tag{2}$$

$$\begin{aligned}
&\widetilde{Q}_0 = D_{301} \oplus D_{10} \quad \widetilde{P}_1 = D_{10} \oplus D_{13} \quad \widetilde{Q}_3 = D_{13} \oplus D_{43} \\
&\widetilde{P}_4 = D_{43} \oplus D_{41} \quad \widetilde{Q}_1 = D_{41} \oplus D_{301} \oplus D_{212} \\
&\Rightarrow \quad D_{212} = \widetilde{Q}_0 \oplus \widetilde{P}_1 \oplus \widetilde{Q}_3 \oplus \widetilde{P}_4 \oplus \widetilde{Q}_1
\end{aligned} \tag{3}$$

Let a 2-erasure be described by an index pair (i, j) $(0 < i < j < p)$. All erasures can be divided into four classes by the parity of i and j. Because the code structure is symmetric, we can focus only on the erasures involving disk1 or/and disk2.

Now we show that a 2-erasure $(1, j)$ (j>1 is an odd number) leads to the recoverable structure. All operations are mod p arithmetic. An erasure $(1, j)$ induces a Hamiltonian cycle v_0-w_{-1}-v_d-w_{d-1}-v_{2d}-w_{2d-1}-...-$w_{(p-1)d-1}$-v_{pd}(v_0) (d=j-1). The two fake 3-vertices are $v_{\frac{p-1}{2}}$ and $v_{\frac{p+j}{2}-1}$. The two real 3-vertices are $w_{\frac{p-1}{2}}$ and $2_{\frac{p-j}{2}}$. We replace the two real 3-vertices with their neighbors $v_{\frac{p+1}{2}}$ and $v_{\frac{p-j}{2}+1}$. This substitution doesn't affect the relative order of the two kinds of vertices.

We translate the original proposition into an equivalent problem: p is a prime; d is an even number between 2 and p-2; 0, d, …, (p-1)d is a permutation of {0, 1, …, p-1}; mark $\frac{p-1}{2}$ and $\frac{p+d+1}{2}-1$ as red, and $\frac{p+1}{2}$ and $\frac{p-d-1}{2}+1$ as black; then the red and black numbers are interleaved in the permutation.

We shift the four numbers to 0, $\frac{d}{2}$, 1 and $1-\frac{d}{2}$. This transformation doesn't change the relative order of the vertices too. Suppose the indices of the four numbers in the permutation are x, y, u and v. Namely,

$$x = 0,\ yd = \frac{d}{2} \bmod p,\ ud = 1 \bmod p,\ vd = (1-\frac{d}{2}) \bmod p.$$

Thus $2yd = d \bmod p$. Since y and d are relative prime, we get $2y = 1 \bmod p$. So we have $y = \frac{p+1}{2}$. Note that $(y+v-u)d = 0 \bmod p$, thus $y+v = u \bmod p$. Thus $y+v = u$ or $y+v = u+p$. Thus $0 < v < y(=\frac{p+1}{2}) < u$ or $0 < u < y < v$. Anyway, the red and black numbers are interleaved. Therefore any 2-erasure of two disks

with odd indices has the structure as Fig 5.b shows. Other cases can be proven in the similar way. We omit the proof due to lack of space.

By then, we have shown that any 2-erasure in a Liberation coding system can be reconstructed by the chained decoding algorithm. This algorithm provides an intuitive decoding process and helps understand the interior mechanism of Liberation codes better.

5. Constructing Liberation codes by CHLS

Studying Plank's construction method, we see that the key is that each pair of disks forms a Hamiltonian cycle. On this basis, arrange the 3-group symbols properly, 2-erasure correcting will be guaranteed. An intuitive idea is constructing Hamiltonian cycles using general CHLS instead of Cayley tables. The code in Fig 6 is constructed in this way. We call this kind of codes the *Latin Liberation codes*. Latin Liberation codes have advantage in parameter flexibility because of the dense distribution of CHLS numbers. Another advantage is good structure flexibility. The code in Fig 6 is based on the CHLS in Fig 2.g. This CHLS is not isotopic to C_7. This means that the code is not isomorphic to the 9-disk Liberation code even if regardless of the 3-group symbols. Generally, the Latin Liberation codes may have more heterogeneous instances than the Liberation codes for a given size. Moreover, the larger the system is, the more remarkable this advantage is.

disk0	disk1	disk2	disk3	disk4	disk5	disk6	disk7	disk8
D_{005}	D_{01}	D_{02}	D_{03}	D_{04}	D_{05}	D_{06}	P_0	Q_0
D_{11}	D_{12}	D_{13}	D_{15}	D_{162}	D_{14}	D_{10}	P_1	Q_1
D_{22}	D_{23}	D_{24}	D_{26}	D_{21}	D_{20}	D_{25}	P_2	Q_2
D_{33}	D_{34}	D_{353}	D_{31}	D_{30}	D_{36}	D_{32}	P_3	Q_3
D_{44}	D_{45}	D_{46}	D_{40}	D_{42}	D_{436}	D_{41}	P_4	Q_4
D_{55}	D_{56}	D_{50}	D_{524}	D_{53}	D_{51}	D_{54}	P_5	Q_5
D_{66}	D_6	D_{61}	D_{64}	D_{65}	D_{62}	D_{63}	P_6	Q_6

Figure 6. **A 9-disk Latin Liberation code.**

Another interesting of this example is the data symbol D_6 that is the reduced form of D_{600}. This kind of 3-group symbols (1-group symbols in fact) decreases computational complexity obviously. However, we can arrange only one 1-group symbol and still satisfy "interleave property". Otherwise, 2-erasure correcting ability is not guaranteed.

There still is a big problem: how to arrange the 3-group symbols to guarantee that the real and fake 3-vertices are interleaved on the cycle. We haven't found a general arranging scheme that is suitable for all CHLS. And it seems that this kind of schemes doesn't exist. But the schemes for specific families of CHLS and individual instance are possible.

6. Performance analysis

In this section, we compare the performance of the XOR-based RAID6 codes. Because these codes are neck and neck in reliability, check disk overhead, update penalty, group size, and extensibility and so on, we mainly focus on the computational performance of encoding/decoding/updating. The performance is measured in the number of XOR operations per data word. In this section, n denotes the number of the data disks, and p denotes the order of the underlying CHLS.

Because PIHLatin is the superset of EVENODD, PDHLatin is the superset of RDP, and Latin Liberation performs equivalently with Liberation when p is a prime, we select PIHLatin, PDHLatin and Latin Liberation for comparison. PIHLatin has two versions that one has the same structure as EVENODD, and the other reverses the last Q parity symbol. We call the former PIHLatin I, and the latter PIHLatin II. Although PIHLatin II codes are only near-MDS, they still can be used in a RAID6 coding system.

Besides standard codes, horizontally shortened codes are also discussed. We can see that deleting the first data disk in a PIHLatin I coding system shrinks each P and Q parity group exactly by 1. Deleting each of the others shrinks each P parity group by 1, p-1 Q parity groups by 1 and the "S" group by 1. Thus, the deleting order makes no sense to the encoding performance. But the first data disk carries the lowest updating load, so we delete the disks from right to left. In a PIHLatin II coding system, because all data disks are symmetric, the deleting order is insignificant.

Deleting the first data disk from a PDHLatin code yields a code that performs equivalently [19]. We call it a *buddy code*. Moreover, the first data disk carries higher load than all of the others. So, the disks are deleted from left to right.

In a Latin Liberation coding system, the first data disk carries the lowest load. Thus the deleting order is from right to left.

Since the system is just the cyclic repetition of a stripe, performance analysis on a stripe is enough. In addition, we focus only on the 2-erasures of two data disks which have the highest reconstruction load.

Table 1 shows the encoding/decoding/updating performance of above codes. Due to lack of space, we can only explain calculation of these results briefly.

Encoding and decoding a parity group with size of g (g data symbols and 1 parity symbol) both perform g-1 XORs. So it is not hard to calculate per data word encoding cost of the above codes.

Decoding a PIHLatin I stripe decomposes into two stages: calculating "S" by summing all parity symbols,

Table 1. **The computational performance of XOR-based RAID6 codes.**

code		encoding cost	decoding cost	updating cost
PIHLatin I (EVENODD)	standard ($n=p$)	$2-\frac{1}{n-1}$	$2-\frac{1}{n(n-1)}$	$4-\frac{2}{n}$
	shortened ($n<p$)	$2-\frac{1}{n}-\frac{1}{n(p-1)}$	$2+\frac{1}{n}-\frac{1}{p-1}$	$4-\frac{1}{n}-\frac{1}{p-1}+\frac{1}{n(p-1)}$
PIHLatin II	standard ($n=p$)	$2-\frac{2}{n}-\frac{1}{n(n-1)}$	$2-\frac{3}{n}$	3
	shortened ($n<p$)	$2-\frac{2}{n}-\frac{1}{n(p-1)}$	$2-\frac{2}{n}-\frac{1}{(p-1)}$	3
PDHLatin (RDP)	standard ($n=p$-1)	$2-\frac{2}{n}$	$2-\frac{2}{n}$	$4-\frac{2}{n}+\frac{1}{n^2}$
	buddy ($n=p$-2)	$2-\frac{2}{n}$	$2-\frac{2}{n}$	$4-\frac{2}{p-1}$
	shortened ($n<p$-2)	$2-\frac{1}{n}-\frac{1}{p-1}-\frac{1}{n(p-1)}$	$2-\frac{1}{n}-\frac{1}{p-1}-\frac{1}{n(p-1)}$	$4-\frac{2}{p-1}$
Latin Liberation	standard ($n=p$)	$2-\frac{1}{n}-\frac{1}{n^2}$	$\le(2+\frac{1}{n}-\frac{7}{n^2})$	$3+\frac{1}{n}-\frac{1}{n^2}$
	shortened ($n<p$)	$2-\frac{2}{n}+\frac{1}{p}-\frac{1}{np}$	$\le(2+\frac{1}{p}-\frac{7}{np})$	$3+\frac{1}{p}-\frac{1}{pn}$

and then recovering the lost symbols zigzag. Decoding a PDHLatin stripe performs as many XORs as encoding. As for Latin Liberation codes, compared with encoding, the decoding process induces l-1 extra XORs during resolves a cycle with size of l, and saves 2 XORs at the real 3-vertex during zigzag decoding. Because there are at most $2p$ vertices on 2 resolving paths, the decoding process performs at most $2p$-6 XORs more than encoding. Thus the decoding cost of these codes is as Table 1 shows.

Updating D_{ij} induces 3 XORs: $t = D_{ij}^{old} \oplus D_{ij}^{new}$, $P_i^{new} = P_i^{old} \oplus t$ and $Q_j^{new} = Q_j^{old} \oplus t$. Updating D_{ijk} induces 4 XORs. Updating D_{i*} needs p+1 XORs. Thus the updating cost can be calculated easily.

We learn from Table 1 that PDHLatin has the best encoding/decoding performance. In fact, it achieves the optimal performance. Thus, it is the best choice for the systems with a heavy encoding/decoding workload (for example, encoding/decoding file objects is the main workloads in a P2P publish/subscribe system). PDHLatin has the highest updating cost. Moreover, the performance of a shortened code is outstandingly lower than the standard/buddy code with the same size. Thus, if we have to use shortened codes or updating is frequent, we should not choose PDHLatin.

The encoding/decoding performance of PIHLatin II is as good as that of PDHLatin. PIHLatin II also has the optimal updating performance. Thus, if we don't mind a slight inefficiency in capacity, PIHLatin II is almost good for all storage applications.

PIHLatin I has moderate performance. Shortening is harmful to its performance, too. But the gap between the shortened codes and the standard codes with the same size is narrow.

Latin Liberation is interesting. The performance of a shortened Latin Liberation code is better than that of the standard code with the same size. Moreover, the more disks you delete, the more performance increase you gain. This means that maybe we should construct storage systems never by the standard Latin Liberation codes. In addition, we can see that the performance of the chained decoding algorithm is comparable with the bit matrix scheduling algorithm.

7. Conclusion

In this paper, we presented a chained decoding algorithm for the Liberation codes. We also show that it can recover any 2-erasure in a Liberation coding system. Compared with the bit matrix scheduling algorithm, the new algorithm provides the comparable performance. Moreover, it is more intuitive and is helpful to comprehend the interior mechanism of Liberation codes. Then we discussed the possibility of constructing Liberation codes by general CHLS. We call this kind of codes the Latin Liberation codes. The Latin Liberation codes are superiority to the Liberation

codes in parameter flexibility and structure flexibility. Finally, we compared the computational performance of the XOR-based RAID6 codes and gave some suggestion on their application.

We have found only a few Latin Liberation code instances until now. Design of general construction methods for specific families of CHLS and searching algorithm for individual CHLS are important future works. Constructing Liberation codes of even size (especially the powers of two) is another important work. Moreover, the chained decoding algorithm needs to be refined further. Implementing fast encoding/ decoding/updating algorithm on CPU/GPU and integrating the Latin Liberation codes into a real system are also planned.

Acknowledgement

Many thanks to Dr. Ian M. Wanless for his kind help regarding the knowledge of Latin squares!

References

[1] 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 3th USENIX Conference on File and Storage Technologies, San Francisco, CA, USA, Mar, 2004, pp.1-14.

[2] 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 26(2), pp. 143-185, June 1994.

[3] J. S. Plank, "Erasure Codes for Storage Applications," Tutorial of the 4th Usenix Conference on File and Storage Technologies, San Francisco, CA, Dec, 2005.

[4] J. S. Plank, "A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems," Software - Practice & Experience 27(9), pp. 995-1012, Sep 1997.

[5] J. Blomer, M. Kalfane, M. Karpinski, R. Karp, M. Luby and D. Zuckerman, "An XOR-Based Erasure-Resilient Coding Scheme," Technical Report TR-95-048, International Computer Science Institute, August, 1995.

[6] J. S. Plank and Lihao Xu, "Optimizing Cauchy Reed-Solomon Codes for Fault-Tolerant Network Storage Applications," In Proceedings of the 5th IEEE International Symposium on Network Computing and Applications, Cambridge, MA, Jul, 2006, pp.173-180.

[7] M. Blaum, J. Brady, J. Bruck, J. Menon, "EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures", IEEE Trans. on Computers 44(2), pp. 192-202, Feb, 1995.

[8] J. S. Plank, "The RAID-6 Liberation Codes", 6th USENIX Conference on File and Storage Technologies, San Francisco, 2008, pp. 97–110.

[9] L. Xu and J. Bruck, "X-Code: MDS Array Codes with Optimal Encoding," IEEE Trans. on Information Theory 45(1), pp.272-276, Jan, 1999.

[10] L. Xu, V. Bohossian, J. Bruck, and D.G. Wagner, "Low-Density MDS Codes and Factors of Complete Graphs," IEEE Trans. on Information Theory 45(6), pp.1817-1826, Sep, 1999.

[11] C. Huang, L. Xu, "STAR: An Efficient Coding Scheme for Correcting Triple Storage Node Failures," In Proceedings of the 4th USENIX Conference on File and Storage Technologies, San Francisco, Dec, 2005, pp.197-210.

[12] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," In NCA-07: 6th IEEE International Symposium on Network Computing Applications, Cambridge, MA, USA, July, 2007, pp. 79-86.

[13] J. L. Hafner, "WEAVER Codes: Highly Fault Tolerant Erasure Codes for Storage Systems," In Proceedings of the 4th USENIX Conference on File and Storage Technologies, San Francisco, Dec, 2005, pp.211-224.

[14] J. L. Hafner, "HoVer Erasure Codes For Disk Arrays," International Conference on Dependable Systems and Networks, Philadelphia, PA, USA, Jun, 2006, pp. 217-226.

[15] Lisa Hellerstein, Garth A. Gibson, Richard M. Karp, Randy H. Katz and David A. Patterson, "Coding techniques for handling failures in large disk arrays," Algorithmica 12(2/3), pp.182-208, Aug, 1994.

[16] M. Blaum, J. Bruck, and A. Vardy, "MDS array codes with independent parity symbols," IEEE Trans. on Information Theory 42(2), pp. 529-542, Mar, 1996.

[17] Wang Gang, Dong Sha-sha, Liu Xiao-guang, Lin Sheng, Liu Jing, "Construct double-erasure-correcting Data Layout Using P1F," ACTA ELECTRONICA SINICA, 34(12A), pp.2447-2450, Mar, 2007.

[18] Gang Wang, Sheng Lin, Xiaoguang Liu, Guangjun Xie, Jing Liu, "Combinatorial Constructions of Multi-Erasure-Correcting Codes with Independent Parity Symbols for Storage Systems," In Proceedings of the 13th IEEE Pacific Rim Dependable Computing conference, Melbourne, Victoria, Austrilia, Dec, 2007, pp. 61-68.

[19] Wang Gang, Liu Xiaoguang, Lin Sheng, Xie Guangjun, Liu Jing, "Generalizing RDP Codes Using the Combinatorial Method," In NCA-08: 7th IEEE International Symposium on Network Computing Applications, Cambridge, MA, USA, July, 2008, pp.93-100.

[20] Zhou Jie,Wang Gang, Liu Xiaoguang, Liu Jing, "The Study of Graph Decompositions and Placement of Parity and Data to Tolerate Two Failures in Disk Arrays: Conditions and Existance," Chinese Journal of Computer, Vol. 26, No. 10, pp.1379-1386, Oct, 2003.

[21] I. M. Wanless, "Perfect factorisations of complete bipartite graphs and Latin squares without proper subrectangles," Electron. J. Combin, Vol. 6, 1999, R9.