Generalizing RDP Codes Using the Combinatorial Method*

Wang Gang, Liu Xiaoguang, Lin Sheng, Xie Guangjun, Liu Jing
Dept. of Computer, College of Information Technical Science,
Nankai University, 300071, Tianjin, China
wgzwp@163.com

Abstract

In this paper, we present PDHLatin - a new class of 2-erasure horizontal codes with dependent parity symbols based on column-hamiltonian Latin squares (CHLS). We prove that PDHLatin codes are MDS codes. We also present a new class of 2-erasure parity independent mixed codes based on CHLS - PIMLatin. We show that the performance of the new codes is comparable to or better than other codes of this kind. They have perfect parameter flexibility and structure variety that benefit performance. We also discuss code shortening technologies that can improve parameter flexibility, structure variety and reliability. Borrowing ideas from vertical shortening, we develop a 2-erasure array code construction method using non-hamiltonian Latin squares.

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 unreconstructed 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. But all of the known multi-erasure coding techniques have limitations [2]. This paper offers a class of 2-erasure codes based on column-hamiltonian Latin squares. The new codes outperform anything else of its kind in terms of parameter flexibility and computational performance.

* 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)

The outline of this paper is as follows. In Section 2 we discuss related works. Section 3 is devoted to introducing a graph representation of 2-erasure correcting codes, Latin squares and perfect one-factorizations. In Section 4 we present PDHLatin and PIMLatin. Code shortening and constructing codes by non-hamiltonian Latin squares are also discussed in this section. A theoretical analysis is discussed in Section 5. Conclusions and future works are presented in Section 6.

2. Current multi-erasure codes

Plank's tutorial on FAST'05 gives a great introduction of erasure codes for storage applications [2]. 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. Unfortunately, there is no consensus on the best coding technique for n, m, $t > 1$.

The known multi-erasure codes typically fall into one of three categories: Reed-Solomon codes, binary linear codes or array codes. RS codes [3] are the only known MDS codes for arbitrary n, m ($=t$). This means optimal storage efficiency and optimal update penalty. But the computational complexity is a serious problem because Galois Field computation is used though optimized algorithms have been developed [4].

Binary linear codes [5] are XOR-based, hence have perfect computational complexity, but bad storage efficiency is their inherent drawback. Fig 1.a shows a 2d-parity code [5], where D_{ij} denotes a data symbol that participates in parity symbols P_i and Q_j. This example illustrates the key idea of linear codes - divide data symbols into several overlapping parity groups; namely each data symbol participates in multiple groups, so that multi erasures are tolerated.

Array codes arrange data/parity symbols into an array, hence the name. EVENODD [6] is the first MDS array code, perhaps also the most important one -

many subsequent array codes are similar to it and its generalization [7], such as X-code [8], RDP [1], STAR-code [9], etc. EVENODD is 2-erasure, horizontal (some disks contain nothing but data symbols, and the others contain only parity symbols. The opposite is vertical codes in which the parity symbols and the data symbols are stored together) and parity independent (none of the parity symbols participate in other parity groups). Fig 1.b shows the 7-disk EVENODD code. A standard EVENDODD code with p+2 disks consists of a (p-1)*p data array and a (p-1)*2 parity array (where p must be a prime number). The first check disk is a horizontal parity disk, and the second is a skew diagonal parity disk. D_{i*} denotes a data symbol that participates in P_i and all Qs. Namely, the sum S (over GF[2]) of all of these kind of symbols is added into every diagonal parity symbol. Thus the computational performance of EVENODD is non-optimal. Moreover, in order to get optimal update penalty (the number of parity units needs to be modified when a data unit is modified), we must implement each column instead of each symbol as a stripe unit. Some literature has shown that these two properties are the inherent drawbacks of horizontal MDS array codes [7][10]. The p+2-disk EVENODD code can be transformed from the p*p+2p 2d-parity code. For example, the EVENODD code showed in Fig 1.b can be constructed by deleting P_4, Q_4 and D_{40}~D_{44} from the 2d-parity code shown at the left. The contributors of S - members of Q_4 are dealt with distinctly in order to gain MDS property.

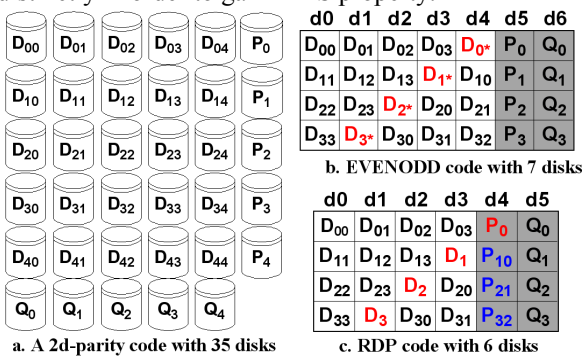


Figure 1. **Binary Linear Codes and Array Codes.**

RDP is another important 2-erasure horizontal code. Fig 1.c shows the 6-disk RDP code. A standard (p+1)-disk RDP code can be described by a (p-1)*(p+1) code array (where p must be a prime number). RDP can be constructed from 2d-parity codes too. Its strategy for the absence of the last Q is parity dependent. As Fig 1.c shows, p-2 horizontal parity symbols also participate in diagonal parity groups, and p-2 data symbols participate in only horizontal parity groups. Such strategy leads to better computational performance than EVENODD. One important advantage of EVENODD and RDP is that they meet the RAID6 specification. Moreover their coding schemes are simple. Thus they are easy to implement.

Most array codes require a prime-related size. Horizontal codes can alleviate this problem by horizontal shortening, but such a transformation is harmful to performance.

B-Code [11] is an interesting 2-erasure vertical code constructed via perfect one-factorizations (P1F) of complete graphs. It has no prime-size limitation because of denseness of "P1F numbers." PIHLatin codes [12] are parity independent horizontal codes based on column-hamiltonian Latin squares (CHLS). It also has no prime size limitation.

In this paper, we mainly focus on RDP-like codes, namely, horizontal codes with dependent parity symbols. We call this kind of codes PDH codes (**P**arity **D**ependent **H**orizontal codes). We present a new class of PDH codes based on CHLS. We call it PDHLatin. It is the superset of RDP and is superior to RDP in parameter flexibility (applicability for different system sizes) and structure variety (how many different structures exist for a given size).

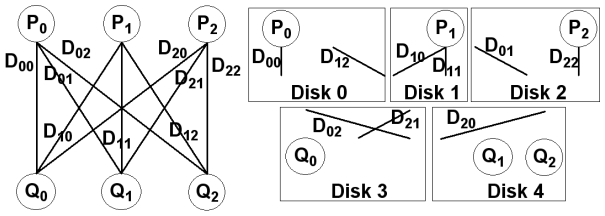
3. Related combinatorics knowledge

3.1. Graph representation of 2-erasure codes

Some literature refers to simple graph representation of parity independent 2-erasure linear codes in which each data symbol participates in exactly two parity groups [5][11][13]: let each vertex denote a parity symbol and each edge denote a data symbol - the two endpoints of an edge are just the two parity symbols of the data symbol. Then an array code can be described by a graph partition if its underlying linear code can be described by a simple graph. So, we can study the construction of array codes through graph partition. We have proven the following theorem [13]:

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

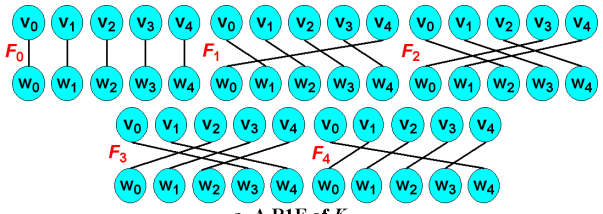
1. A path and its two vertices. (Containing an edge doesn't mean containing its two vertices necessarily because an edge and its two vertices are separate objects - a data symbol and its two parity symbols) We call this kind of unrecoverable erasure **C**losed **P**arity **S**ymbols **S**ubset, CPSS for short.
2. A cycle. We call it **C**losed **D**ata **S**ymbols **S**ubset, CDSS for short.



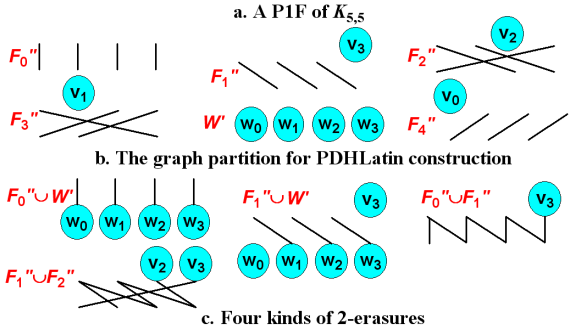
a. A 15-disk 2d-parity code b. An array code based on the left code

c. A CPSS example d. A CDSS example

Figure 2. **Graph Representation.**



a. A P1F of $K_{5,5}$



b. The graph partition for PDHLatin construction

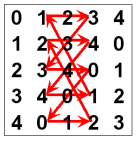
c. Four kinds of 2-erasures

Figure 3. **Constructing PDHLatin Using P1F.**

Fig 2.a shows the graph that corresponds to a 15-disk 2d-parity code. Fig 2.b shows an array code based on it. Fig 2.c and 2.d give a CPSS example and a CDSS example respectively. They correspond to two unrecoverable 2-erasures (disk0, disk1) and (disk2, disk3) of the array code respectively. Theorem 1 can interpret almost all of 2-erasure array codes.

3.2. Perfect one-factorizations

A factor of a graph $G=(V, E)$ is a spanning subgraph of G and a one-factor of G is a one-regular spanning subgraph of G. A factorization of G is a set of factors of G $\{F_0, F_1, \ldots, F_{k-1}\}$, which are pair-wise edge disjoints - no two have a common edge - whose union is G. A one-factorization (1F) of G is a factorization of G consisting of only one-factor. If for any distinct pair F_i, F_j of factors, $F_i \cup F_j$ induces a Hamiltonian cycle, the 1F is called a perfect one-factorization (P1F). There is a widely believed conjecture in graph theory: every complete graph with an even number of vertices has a P1F [14]. Graph theorists have proven that all even numbers less than 54 are "P1F numbers" and have found many larger P1F numbers. A P1F of K_{2p} can produce two B-Code instances with $2p$-1 disks and $2p$-2 respectively [11], thus B-Code is suitable for any size if the conjecture were proved. Fig 3.a shows a P1F of $K_{5,5}$.



a. C_5

	d0	d1	d2	d3	d4	d5	d6	d7
	D_{00}	D_{01}	D_{02}	D_{03}	D_{04}	D_{05}	P_0	Q_0
	D_{11}	D_{12}	D_{13}	D_{15}	D_1	D_{14}	P_{10}	Q_1
	D_{22}	D_{23}	D_{24}	D_2	D_{21}	D_{20}	P_{25}	Q_2
	D_{33}	D_{34}	D_{35}	D_{31}	D_{30}	D_3	P_{32}	Q_3
	D_{44}	D_{45}	D_4	D_{40}	D_{42}	D_{43}	P_{41}	Q_4
	D_{55}	D_5	D_{50}	D_{52}	D_{53}	D_{51}	P_{54}	Q_5

c. PDHLatin code based on the CHLS in b

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

b. A CHLS of order 7

	d0	d1	d2	d3	d4	d5	d6	d7
	D_{00}	D_{01}	D_{02}	D_{03}	D_{04}	D_{05}	P_0	Q_0
	D_{11}	D_{12}	D_{13}	D_{15}	P_1	D_{14}	D_{10}	Q_1
	D_{22}	D_{23}	D_{24}	P_2	D_{21}	D_{20}	D_{25}	Q_2
	D_{33}	D_{34}	D_{35}	D_{31}	D_{30}	P_3	D_{32}	Q_3
	D_{44}	D_{45}	P_4	D_{40}	D_{42}	D_{43}	D_{41}	Q_4
	D_{55}	P_5	D_{50}	D_{52}	D_{53}	D_{51}	D_{54}	Q_5

d. PIMLatin code based on b

Figure 4. **CHLS, PDHLatin and PIMLatin.**

3.3. Latin squares

For $k \le n$, a $k*n$ *Latin rectangle* is a $k*n$ 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 use $Z_n=\{0, 1, \ldots, n\text{-}1\}$ as the symbol set; it also can be used as the row (and column) number set. Let $L(k, n)$ be the set of $k*n$ Latin rectangles. Elements of $L(n, n)$ are called *Latin squares* of order n. The symbol in row r, column c of a Latin rectangle R is denoted by R_{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 rectangle R is the image of some permutation σ_r of Z_n, namely $R_{ri}=\sigma_r(i)$. Each pair of rows $(r; s)$ defines a permutation by $\sigma_{r,s}=\sigma_r\sigma_s^{-1}$. Naturally $\sigma_{r,s}=\sigma_{s,r}^{-1}$. 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 4.a shows a CHLS of order 5, and $\sigma_{1,3}$ of it. It is the Cayley table C_5 of the cyclic group of order 5 [14].

There is a close relationship between CHLS and P1F [14]. There is a CHLS of order n iff $K_{n,n}$ has a P1F. A CHLS L of order n can be transformed into a P1F F of $K_{n,n}$ = (V, W, E) [14]: let $V=\{v_i \mid 0\le i\le n\text{-}1\}$ and $W=\{w_i \mid 0\le i\le n\text{-}1\}$; let edge $(v_i, w_k) \in F_j$, for all $(i, j, k) \in L$. The reverse method converts a P1F of $K_{n,n}$

into a CHLS of order n. This method can also create a transformation between LS and 1F. We can see that the cycles in $\sigma_{r,s}$ of L correspond to cycles in $F_r \cup F_s$. The P1F shown in Fig 3.a corresponds to C_5. There is another conclusion [14]: if K_{n+1} has a P1F, then so does $K_{n,n}$, but the converse is not true. 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. PIHLatin codes [12] are based on CHLS, thus have good parameter flexibility like B-CODE.

4. New codes

4.1. PDHLatin codes

Given a CHLS L of order p, we can construct a 2-erasure PDH code C with (p+1) disks. Like RDP, the first check disk of C is the horizontal parity disk. Because the second check disk is constructed by L, we call it a "Latin parity disk." The algorithm is as follow.

Algorithm 1.
Input: L - a reduced (L_{i1}=L_{1i}=i) CHLS of order p.
Output: A 2-erasure PDHLatin code C.
Method:

1. Delete the last row of L, then we get a (p-1)*p Latin rectangle R.
2. Construct the i^{th} data symbol on the j^{th} data disk by $(i, j, k) \in R,\ 0 \le i, j \le p-2$ - let it join in the i^{th} horizontal parity symbol and the k^{th} Latin parity symbol. Namely, it can be denoted by D_{ik}.
3. Construct the i^{th} horizontal parity symbol by $(i, p-1, k) \in R,\ 1 \le i \le p-2$ - let it participate in Q_k. Thus it is denoted by P_{ik}.

Because the code height is p-1, symbols that correspond to (i, j, p-1) participate in only the i^{th} horizontal parity group but none of the Latin parity groups.

Theorem 2. PDHLatin codes constructed by algorithm 1 can tolerate any 2-erasure.

Proof: Suppose that F=$\{F_0, F_1, \ldots, F_{p-1}\}$ is the P1F of $K_{p,p}$=(V, W, E) transformed from L. C is parity dependent, so it can't be described by a simple graph directly. However, we can modify the graph representation: any "single-group" symbol is denoted by a vertex and any "double-group" symbol is denoted by an edge, whatever parity symbol or data symbol. Then C can be constructed by F as follow:

1. Let $F_j'=F_j-\{(v_{p-1}, w_k)\}$ for all $(p-1, j, k) \in L$. (Delete the last row of L) Then $F'=\{F_0', F_1', \ldots, F_{p-1}'\}$ is a P1F of $K_{p-1,p} = (V', W, E')$.
2. Let $F_j''= F_j'-\{(v_i, w_j)\}+\{v_i\}$ for all $(i, j, p-1) \in L$, and $F_0''=F_0'$. (Delete the (p-1)th Latin parity group, and let some vertices denote data symbols) Then $F''=\{F_0'', F_1'' \ldots, F_{p-1}'', W'\}$ is a partition of $K_{p-1,p-1} = (V', W', E')$
3. Construct j^{th} data disk by F_j'' for $0 \le j \le p-2$, horizontal parity disk by F_{p-1}'', and Latin parity disk by W' - let any symbol that corresponds to an edge (v_i, w_j) participate in the i^{th} horizontal parity group and the j^{th} Latin parity group, and any symbol that corresponds to a vertex v_i (w_j) participate in only the i^{th} horizontal parity group (the j^{th} Latin parity group).

Fig 3.b shows the F'' transformed from the P1F to its left. F_0'' (the first data disk) contains nonadjacent p-1 edges; W' (Latin parity disk) contains p-1 vertices; and F_j'' ($1 \le j \le p-1$, other disks) is a mixed set of p-2 nonadjacent edges and an isolated vertex from V' that is adjacent to w_{p-1} originally. Theorem 1 still holds though what CPSS and CDSS represent maybe changed. The recoverability of single-erasures is trivial. We focus on 2-erasures. Suppose the i^{th} disk and the j^{th} disk fail ($0 \le i < j \le p$), there are four cases:

1. i=0, j=p (the first data disk and the Latin parity disk): $F_0'' \cup W'$ consists of p-1 "lollipops," and contains neither CPSS nor CDSS.
2. $1 \le i \le p-1$, j=p (one of the data disks except the first and the Latin parity disk): $F_i'' \cup W'$ contains p-2 lollipops and 2 isolated vertices, and contains neither CPSS nor CDSS.
3. i=0, $1 \le j \le p-1$ (the first data disk and another disk except the Latin parity disk): Because $F_0 \cup F_j$ induces a hamiltonian cycle of $K_{p,p}$. $F_0'' \cup F_j''$ is converted from $F_0 \cup F_j$ by deleting (v_{p-1}, w_{p-1}), (v_{p-1}, w_k) and ($v_{k'}$, w_{p-1}) and adding $v_{k'}$, thus it consists of a path of $K_{p-1,p-1}$ from $v_{k'}$ to w_k with a length of 2p-3 and one of its two endpoints $v_{k'}$, contains neither CPSS nor CDSS.
4. $1 \le i, j \le p-1$: Because $F_i \cup F_j$ induces a hamiltonian cycle of $K_{p,p}$ and $F_i'' \cup F_j''$ is transformed from $F_i \cup F_j$ by deleting (v_{p-1}, w_k), (v_{p-1}, w_l), ($v_{k'}$, w_{p-1}) and ($v_{l'}$, w_{p-1}) and adding $v_{k'}$ and $v_{l'}$, it consists of two paths of $K_{p-1,p-1}$ respectively from $v_{k'}$ to w_k and from $v_{l'}$ to w_l and their endpoints $v_{k'}$ and $v_{l'}$, contain neither CPSS nor CDSS. □

Fig 3.c shows the examples of the four cases.

We can see that the PDHLatin code based on C_5 is just the 6-disk RDP code showed in Fig 1.c. The PDHLatin code based on C_p equals the (p+1)-disk RDP code when p is a prime number. PDHLatin is the superset of RDP. The PDHLatin code shown in Fig 4.c is based on the CHLS shown in Fig 4.b. It is not isomorphic to the 8-disk RDP code. This means that RDP is a proper subset of PDHLatin. This code shows the superiority of PDHLatin in structure variety. There

are two heteromorphic 8-disk PDHLatin codes because there are two main classes containing CHLS of order 7 [14]. However there is only one RDP instance for any size. Generally, as the size increases, the gap becomes wider. For example, there are at least 11 main classes containing CHLS of order 11 [15].

There are 37 heteromorphic 10-disk PDHLatin codes because there are 37 main classes containing CHLS of order 9 [14]. However, there is no 10-disk RDP code because 9 is not a prime number. This shows the advantage of PDHLatin over RDP in parameter flexibility. Although horizontally shortened RDP codes are suitable for any size, we will show that horizontal shortening is harmful to computational performance. In an EVENODD/RDP/PDHLatin code, the first data disk touches every parity group exactly once. Thus deleting it produces a code that performs equivalently. This means that standard PDHLatin code exists for any size if the P1F conjecture were proved.

4.2. PIMLatin codes

Swapping roles of each single-group data symbol with its parity symbol, we transform a PDHLatin code into a parity independent 2-erasure mixed code. We call this kind of codes PIMLatin. Fig 4.d shows an 8-disk PIMLatin code transformed from the PDHLatin code shown in Fig 4.c. In fact, there exists a bijection between PIMLatin and PDHLatin. We omit the construction algorithm and the 2-erasure theorem because they are very similar to algorithm 1 and theorem 2 respectively.

We can see that PIMLatin is MDS code. It achieves optimal computational performance. This is the most important advantage of PIMLatin over PDHLatin/RDP. PIMLain also outperforms RDP in parameter flexibility the way PDHLatin does. It can be regarded as a vertical generalization of RDP.

4.3. Code shortening

There are two general code extension strategies.

The first is horizontal shortening - deleting some data disks (assuming they contain nothing but zeros). This operation is very suitable for horizontal codes. The fault tolerance remains the same and parameter flexibility improves after horizontal shortening. As mentioned above, the first data disk should be deleted first to get a code that performs equivalently. Every other disk touches only p-2 Latin parity groups and each parity group is missed exactly once. Thus the orders of deletion make no difference.

It is difficult to horizontally shorten vertical/mixed codes, such as PIMLatin, because deleting a parity symbol makes its data symbols orphans. Bohossian et al. suggested a take-over method [16]. For PIMLatin, we can let the data symbol on the last column take over its horizontal parity group from the deleted parity symbol. This yields half PIMLatin, half PDHLatin codes. Because the partition of a shortened code is a subset of its original PIMLatin/PDHLatin code, thus the fault tolerance remains the same. Fig 5.a shows a 5-disk horizontally shortened PIMLatin code based on the PIMLatin code shown in Fig 4.d.

Horizontal shortening can be used reversely to solve storage system extension. We can initialize storage systems using shortened codes instead of standard codes. When new devices are added, we can simply zero them and let them play the roles of disks that are deleted in horizontal shortening.

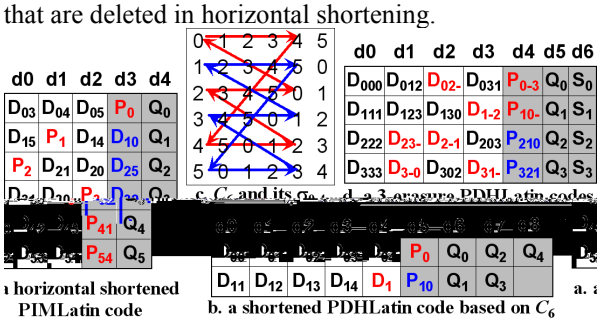


Figure 5. **Code Shortening.**

Another strategy is vertical shortening - deleting some rows of the code and reserving all Latin parity symbols. Vertical shortening makes storage efficiency worse, but performance and reliability better. Considering the high cost-capacity and low cost-reliability/performance of hard disks nowadays, it is worthwhile to do this shortening. WEAVER code [17] has practiced this idea. Its best storage efficiency is only 50%. But it has a very short parity group size and good locality, so it has good fault-mode performance especially for distributed storage applications.

Fig 5.b shows a shortened PDHLatin code. We can see that its average group size is 3.86, which is far less than the system size. Thus, the per disk load increase under fault mode is far lower than that of standard PDHLatin codes. Moreover, the fault tolerance of this code is beyond 2. For example, the triple-erasure (d0, d1, d3) can be recovered.

Vertical shortening makes horizontal codes irregular. This flaw can be alleviated by splitting check disks and repeating the code cycle rotationally like RAID5. Since non-MDS and irregularity are inevitable, parity dependent (RDP/PDHLatin) and "S" (EVENODD/PIHLatin) are unnecessary. We can bring the deleted Latin parity group back. For example, we

can add Q_5 into the code shown in Fig 5.b, and let P_{10} be independent and D_1 join Q_5 again. This strategy unifies shortened PIHLatin and PDHLatin codes, and results in optimal computational performance.

The greatest advantage of vertical shortening is that of providing a wide range of choices in performance/efficiency trade-off space with a unique construction method. Vertical shortening also spawns another useful method - constructing array codes via non-hamiltonian Latin squares (non-HLS).

4.4. Constructing array codes by non-HLS

The code shown in Fig 5.b is just constructed by deleting the last three rows of C_6 though C_6 is not a CHLS. The reason for this is that deleting the last three rows breaks all cycles in any pair of columns. In fact, we have a general code construction method via Cayley tables. The Cayley table C_n of order n can be described by $(i, j, <i+j>_n) \in C_n, 0 \le i, j \le n-1$, where $<x>_m$ denotes $x\%m$. We examine some properties of C_n first.

Property 1. $\sigma_{j,k}$ of C_n ($0 \le j < k \le n-1$) consists of $gcd(n, d)$ $2l$-long ($l=n/gcd(n,d)$) cycles. The i^{th} cycle is (i, j, $<i+j>_n$)-(i, k, $<i+k>_n$)-($<i+k-j>_n$, j, $<i+k>_n$)-($<i+k-j>_n$, k, $<i+2k-j>_n$)-($<i+2k-2j>_n$, j, $<i+2k-j>_n$)-...-($<i+(l-1)k-(l-1)j>_n$, k, $<i+lk-(l-1)j>_n$)-($<i+lk-lj>_n$, j, $<i+lk-(l-1)j>_n$).

$d=k-j$, and $gcd(n, d)$ denotes the greater common divisor of n and d. According to the definitions of C_n and σ, it isn't hard to conclude this property. Fig 5.c shows $\sigma_{0,4}$ of C_6. It has 2 6-long cycles because $gcd(4, 6)=2$. Generally, $\sigma_{j,k}$ contains at most $n/\mathrm{pr}(n)$ cycles ($\mathrm{pr}(n)$-long each, when $d=n/\mathrm{pr}(n)$) and at least $\mathrm{pr}(n)$ cycles ($n/\mathrm{pr}(n)$ -long each, when $d=\mathrm{pr}(n)$), where $\mathrm{pr}(n)$ is the smallest prime divisor of n. This conclusion is coincident with the fact that C_n is a CHLS when n is a prime number.

Property 2. Given C_n, for any $0 \le i \le n-1$, and any $0 \le r < s \le n-1$, rows i, $<i+1>_n$, ..., $<i+n/\mathrm{pr}(n)-1>_n$ intersect all cycle of $\sigma_{r,s}$. And there exist $0 \le r < s \le n-1$ and $0 \le k \le gcd(n,s-r)-1$, rows i, $<i+1>_n$, ..., $<i+n/\mathrm{pr}(n)-2>_n$ don't intersect the k^{th} cycle of $\sigma_{r,s}$.

This property can be concluded by examining cycle patterns of all $\sigma_{r,s}$. It deduces property 3 directly.

Property 3. Deleting any k ($n/\mathrm{pr}(n) \le k \le n-1$) consecutive rows (with wrap-around) from C_n, we get a Latin rectangle in which for any $0 \le r < s \le n-1$, $\sigma_{r,s}$ contains no cycles.

These properties can be converted into 1F version according to LS-1F transformation described in Section 3. Therefore, deleting at least $p/\mathrm{pr}(p)$ rows from C_n, we can construct a 2-erasure PDHLatin code. The construction algorithm and the fault tolerance theorem are very similar to algorithm 1 and theorem 2 respectively, so we omit them.

This method greatly improves structure variety. For example, there are 37 main classes containing CHLS of order 9, but 19,270,853,541 main classes containing LS of order 9 [15]! This method also improves practical parameter flexibility. Even if the size of n isn't a P1F number, we still can construct PDHLatin/PIMLatin/PIHLatin codes via C_n.

This method can also generalize 2-erasure HoVer code [18]. HoVer is a class of multi-erasure mixed codes. It provides a wide range of choices in performance/efficiency trade-off space. We can show that HoVer codes are constructed by deleting two sets of consecutive rows from Cayley tables. The details are omitted for the paper length limit.

5. Performance analysis

Gibson et al present 5 metrics for erasure codes [5]: reliability, check disk overhead, update penalty, group size and extensibility. We compare the performance of PDHLatin/PIMLatin and other array codes in these aspects. In this section, n denotes the number of data disks, p denotes the order of CHLS, and h denotes the number of data rows. So standard and shortened codes can be denoted consistently by notation $\mathrm{PDHLatin}_{p,n,h}$. Thus $p=n$ ($p=n+1$) and $h=p-1$ mean standard EVENODD/PIHLatin (RDP/PDHLatin).

The check disk overhead of $\mathrm{PDHLatin}_{p,n,h}$/ $\mathrm{PIMLatin}_{p,n,h}$ is $\frac{h+p-1}{nh+h+p-1}$. That is to say, standard codes are MDS codes, and achieve optimal check disk overhead. If $h=p-1$ (merely horizontally shortened), the MDS property remains. If n (h) is fixed, the check disk overhead increases as h (n) decreases. When $n=p-1$, the worst value is $\frac{p}{2p-1}$. (Latin parity symbols degenerate into copies of data symbols). The global worst value is 2/3 (the code is just 3-way mirroring). Analysis of EVENODD and RDP is similar.

As mentioned above, all of the codes can achieve optimal update penalty.

Computational complexity is important though [5] didn't discuss them. We examine standard codes first. Because encoding a parity group with g data symbols does g-1 XOR operations, per data word encoding XORs of 2-erasure EVENODD is $2-\frac{1}{p-2}$, those of RDP/PDHLatin/PIMLatin are all $2-\frac{2}{p-1}$ which is the optimum. Analysis of decoding cost is similar. Per

word updating cost of 2-erasure EVENODD (PIHLatin), RDP (PDHLatin) is $4-\frac{4}{p-1}$ and $4-\frac{2}{p-1}+\frac{1}{(p-1)^2}$. That of PIMLatin is 3 - the optimal value. Note that as mentioned above, these codes all adopt "column implementation." Thus the updating unit is a column.

We now examine horizontally shortened codes. We only compared RDP and PDHLatin because the analysis of EVENODD and PIMLatin is similar. As mentioned above, deleting the first data disk shrinks every parity group exactly by 1. Deleting any other data disk shrinks every horizontal parity group by 1 and p-2 Latin parity groups by 1 and all Latin parity groups are missed in turn. Thus, the per word encoding cost of a horizontally shortened RDP code $RDP_{p,n,p-1}$ is $2-\frac{1}{p-1}-\frac{1}{n}-\frac{1}{n(p-1)}$. That of a ($n$+1)-disk standard PDHLatin $_{n+1,n,n}$ is $2-\frac{2}{n}$. The difference is $\frac{1}{n}-\frac{1}{p-1}-\frac{1}{n(p-1)}$ that increases as p-n increases. The performance gap is huge if n is far less than p. This will happen in practice. Note that RDP requires "column implementation." So, p-1 must be a power of 2 to avoid disk space waste. Moreover p must be a prime number. Qualified numbers are very few, 17 followed by 257! 17 is too small, 257 is reasonable. Therefore, we have to construct a small RDP code from the 258-disk standard RDP code. By contrast, we can construct a small standard PDHLatin code directly. The performance gap approaches 10% when n is small. For heavily-loaded systems, that means a heavy extra computational load. Analysis of the decoding cost is similar. The per word updating cost of shortened RDP codes is $4-\frac{2}{p-1}$, and that of standard PDHLatin codes is $4-\frac{2}{n}+\frac{1}{n^2}$. We can see that the advantage of PDHLatin (PIHLatin) in parameter flexibility over RDP (EVENODD) benefits performance greatly.

As mentioned above, vertically shortened PIHLatin (EVENODD) codes and PDHLatin (RDP) codes can be unified into one kind of 2-erasure parity independent codes in which each data symbol participates in exactly two parity groups. The per word encoding cost of this kind of code is $2-\frac{1}{p}+\frac{1}{h}$. As h decreases, the performance improves. The per word updating cost of the unified codes is obviously 3 which is the optimum. The general case is similar to merely horizontal shortening, so we omit it.

As mentioned above, vertical shortening provides a wide range of choices in performance/efficiency trade-off space. When h is close to p-1, the codes have good storage efficiency. Deeply shortened codes have a small group size that benefits degraded- and reconstruction-mode performance especially for distributed storage applications because it decreases the interaction between failed disks and surviving disks (network communication between storage nodes). Generally, the average group size of the unified code is $\frac{2h(n+1)}{p+h}$. Obviously, as h decreases, the average group size decreases. As mentioned above, vertical shortening also improves reliability.

As such, PDHLatin/PIMLatin's performance is comparable with or higher than codes of the type, such as EVENODD/RDP. Moreover they are far superior to other codes in parameter flexibility and structure variety, which benefit performance.

6. Conclusion

In this paper, we have presented a new class of 2-erasure horizontal codes with dependent parity symbols. We call it PDHLatin because it is based on column-hamiltonian Latin squares. We have proven that the PDHLatin codes are MDS codes. We also presented a new class of 2-erasure parity independent mixed codes based on CHLS - PIMLatin. There is a bijection between 2-erasure PDHLatin and 2-erasure PIMLatin. We have shown that their performance is comparable with RDP, and slightly better than EVENODD. And PIMLatin has outstanding updating performance. The new codes have perfect parameter flexibility and structure variety. We showed that these advantages improve performance. We also discussed code shortening. Horizontal shortening can improve parameter flexibility and can be used reversely to solve system extension. Vertical shortening can improve degraded- and reconstruction-mode performance especially for distributed storage systems. It also can improve parameter flexibility, structure variety and reliability. Borrowing ideas from vertical shortening, we present a 2-erasure array code construction method using non-hamiltonian Latin squares.

The study of t-erasure PDHLatin codes for $t > 2$ is an important work for the future. We have found some instances. Fig 5.d shows a 3-erasure PDHLatin code based on C_5 and a cyclic-shift transformation of C_5. But the existence of multi-erasure PDHLatin codes needs a lot more work. Another important research

direction is to study performance optimizing via the good structure variety of the Latin codes. Obviously, different Latin squares may lead to quite a different performance. Code shortening, especially vertical shortening (constructing codes by non-HLS), is worth further study too. In this paper, we only did simple performance analysis and compared the new codes with few other codes. Detailed performance analysis and reliability analysis (especially reliability of vertically shortened codes), and performance comparison with other codes (such as Liberation code [19]) are valuable works. We also plan to apply this methodology to other kinds of codes, such as X-code and Liberation code. Finally, implementation and performance testing are 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] J. S. Plank, "Erasure Codes for Storage Applications", Tutorial of the 4th Usenix Conference on File and Storage Technologies, San Francisco, CA, Dec, 2005.

[3] J. S. Plank, "A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems", Software - Practice & Experience, Vol. 27, No.9, Sep, 1997, pp.995-1012.

[4] 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.

[5] 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, Vol. 12, No. 2/3, Aug,1994, pp.182-208.

[6] M. Blaum, J. Brady, J. Bruck, J. Menon, "EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures", IEEE Trans. on Computers, Vol. 44, No. 2, pp. Feb, 1995, 192-202.

[7] M. Blaum, J. Bruck, and A. Vardy, "MDS array codes with independent parity symbols", IEEE Trans. on Information Theory, Vol. 42, No. 2, Mar, 1996, pp. 529-542.

[8] L. Xu and J. Bruck, "X-Code: MDS Array Codes with Optimal Encoding", IEEE Trans. on Information Theory, Vol. 45, No. 1, Jan, 1999, pp.272-276.

[9] Cheng Huang, Lihao 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.

[10] Wang Gang, Dong Sha-sha, Liu Xiao-guang, Lin Sheng, Liu Jing, "Construct double-erasure-correcting Data Layout Using P1F", ACTA ELECTRONICA SINICA, Vol. 34, No. 12A, 2006, pp.2447-2450.

[11] L. Xu, V. Bohossian, J. Bruck, and D.G. Wagner, "Low-Density MDS Codes and Factors of Complete Graphs", IEEE Trans. on Information Theory, Vol. 45, No. 6, Sep, 1999, pp.1817-1826.

[12] Gang Wang, Sheng Lin, Xiaoguang Liu, Guangjun Xie, Jing Liu, "Combinatorial Constructions of Multi-Erasure-Correcting Codes with Independent Parity Symbols for Storage Systems", IEEE PRDC 2007, Melbourne, Victoria, Australia, Dec, 2007, pp. 61-68.

[13] 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, Oct, 2003, pp.1379-1386.

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

[15] Charles. J. Colbourn, Jeffrey H. Dinitz, et al, "Handbook of Combinatorial Designs (Second Edition)" , CRC Press, 2007.

[16] V. Bohossian, J. Bruck, Shortening Array Codes and the Perfect 1-Factorization Conjecture, 2006 IEEE International Symposium on Information Theory, pp. 2799-2803, Seattle, WA, USA.

[17] 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.

[18] James Lee Hafner, "HoVer Erasure Codes For Disk Arrays," International Conference on Dependable Systems and Networks (DSN'06), Philadelphia, PA, USA, Jun, 2006, pp. 217-226.

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