Information Dispersal Matrices for RAID Error Correcting Codes

Sarah Edge Mann, Michael Anderson, and Marek Rychlik

September 19, 2011

Contents

1 Introduction

This paper is intended as a mathematical guide to RAID error correcting codes. Proof is given of the correctness of certain coding algorithms that have been proposed in [4] and [5]. As well, a new algorithm is introduced, and its correctness shown.

Error correcting codes are used in a number of applications, including deep space communications, telecommunications, CDs, and hard drive systems. In this paper, we will consider the particular application of arrays of hard drives, although the theory is applicable to other settings. In particular, we are motivated by the need to store data on physical devices in a way that is robust to device failure. In particular, if we have N drives of data, we will use them to create an additional M drives of data such that if any M of the $N+M$ drives fail, it is possible to recover the data stored on all failed drives.

Section 2 gives some mathematical background on Vandermonde matrices that will be useful in developing the theory of RAID systems. Section 3 discusses various algorithms for RAID error correcting codes. Section 4 considers the computational costs of preparing for and recovering from device failures.

In what follows, all arithmetic is done over the field $\mathbb{F}$. For concreteness and relevance to applications, $\mathbb{F}$ will be taken to be the finite field $GF(2^k)$ in all examples. It might be more intuitive to think of $\mathbb{F}$ as $\mathbb{R}$ at first, to understand some of the proofs.

2 The Generalized Vandermonde Matrix

Definition 2.1. *Let $\vec{\alpha}$ be a vector in $\mathbb{F}^m$, and α_i be the ith element of $\vec{\alpha}$. The $m \times n$* Vandermonde Matrix *generated by $\vec{\alpha}$ is defined to be*

$$V^{m,n}(\vec{\alpha}) = \begin{bmatrix} \alpha_1^0 & \alpha_1^1 & \alpha_1^2 & \dots & \alpha_1^{n-1} \\ \alpha_2^0 & \alpha_2^1 & \alpha_2^2 & \dots & \alpha_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_m^0 & \alpha_m^1 & \alpha_m^2 & \dots & \alpha_m^{n-1} \end{bmatrix} = \left[\vec{\alpha}^0 | \vec{\alpha}^1 | \vec{\alpha}^2 | \dots | \vec{\alpha}^{n-1}\right]. \tag{1}$$

(Note: some authors define the Vandermonde matrix as the transpose of this matrix.)

Definition 2.2. *Let $\vec{\alpha}$ be a vector in $\mathbb{F}^m$, and let $\{f_i\}_{i=0}^{n-1}$ be a set of n linearly independent degree $n-1$ polynomials over $\mathbb{F}$. Define the $m \times n$* Generalized Vandermonde Matrix[1] *by*

$$G(\{f_i\}, \vec{\alpha}) = \left[f_0(\vec{\alpha}) | f_1(\vec{\alpha}) | \dots | f_{n-1}(\vec{\alpha})\right]. \tag{2}$$

Remark 2.3. *Notice that the Vandermonde matrix is the special case of the generalized Vandermonde matrix where $f_i(x) = x^i$.*

Theorem 2.4. *The square generalized Vandermonde matrix is invertible if and only if α_i are distinct.*

Proof. Consider an $n \times n$ generalized Vandermonde matrix $G(\{f_i\}, \vec{\alpha})$. Trivially, if α_i are not distinct then $G(\{f_i\}, \vec{\alpha})$ will contain repeated rows, and will not be invertible.

Assume α_i are distinct, and, in search of contradiction, assume that $G(\{f_i\}, \vec{\alpha})$ is not invertible. Then there exists $\vec{c} \in \mathbb{F}^n$ such that $\vec{c} \neq \vec{0}$ and

$$\sum_{i=0}^{n-1} c_i f_i(\alpha_j) = 0 \tag{3}$$

for any $0 \leq j \leq n-1$. $\sum_{i=0}^{n-1} c_i f_i(x)$ is a degree $n-1$ polynomial that vanishes at n points, and is thus uniformly 0. This contradicts the assumed linear independence of $\{f_i\}$, so $G(\{f_i\}, \vec{\alpha})$ must be invertible. □

Corollary 2.5. *The square Vandermonde matrix is invertible if and only if α_i are distinct.*

Theorem 2.6. *Let $\vec{\alpha}$ be a vector in $\mathbb{F}^n$ with α_i distinct, and $\{f_i\}_{i=0}^{n-1}$ be a set of degree $n-1$ polynomials over $\mathbb{F}$. If the matrix $A = [f_0(\vec{\alpha}) | f_1(\vec{\alpha}) | \dots | f_{n-1}(\vec{\alpha})]$ is invertible, then $\{f_i\}_{i=0}^{n-1}$ are linearly independent and $G(\{f_i\}, \vec{\gamma})$ is a generalized Vandermonde matrix for any $\gamma \in \mathbb{F}^m$.*

[1] Other authors have dubbed other matrices the generalized Vandermonde matrix, see [3], for example. I am taking some liberties with this naming.

Proof. In search of contradiction, assume that A is invertible and $\{f_i\}$ are not linearly independent. Then there exists $\vec{c} \neq \vec{0} \in \mathbb{F}^n$ such that

$$\sum_{i=0}^{n-1} c_i f_i(x) = 0, \qquad \forall x, \tag{4}$$

and, in particular,

$$\sum_{i=0}^{n-1} c_i f_i(\alpha_j) = 0, \qquad 0 \leq j \leq n-1. \tag{5}$$

Thus the columns of A are not linearly independent and A is not invertible. This is a contradiction, so $\{f_i\}$ must be linearly independent. □

Corollary 2.7. *For $m > n$, let $\vec{\alpha}$ be a vector in $\mathbb{F}^m$ with α_i distinct, and $\{f_i\}_{i=0}^{n-1}$ be a set of degree $n-1$ polynomials over $\mathbb{F}$. If*

$$A = [f_0(\vec{\alpha})|f_1(\vec{\alpha})|\ldots|f_{n-1}(\vec{\alpha})] = \begin{bmatrix} I_n \\ \star \end{bmatrix} \tag{6}$$

where I_n is the $n \times n$ identity matrix, and $\star$ is some $(m-n) \times n$ matrix, then A is a generalized Vandermonde matrix.

Proof. Notice that I_n is an invertible matrix generated by the first n elements of $\vec{\alpha}$, and thus, by Theorem 2.6, $\{f_i\}$ are linearly independent and $A = G(\{f_i\}, \vec{\gamma})$ is a generalized Vandermonde matrix. □

3 RAID and Information Dispersal Matrices

If we have N *data drives* each consisting of l *words*, and add M *check drives*, then we can express our original data as an $N \times l$ matrix D, and the encoded data as an $(N+M) \times l$ matrix C. We seek a $(N+M) \times N$ matrix F that defines our encoding scheme by

$$FD = C. \tag{7}$$

We would like F to have the following property: if any M drives fail, we want to be able to recover the original data D from the remaining encoded data $\hat{C}$. Mathematically, this requires that if any M rows are deleted from F to create $\tilde{F}$ and the same M rows are deleted from C to create $\hat{C}$, then the system

$$\tilde{F}D = \hat{C} \tag{8}$$

is solvable for D, *i.e.* $\tilde{F}$ is invertible. Any matrix with this property is called an *information dispersal matrix* or IDM [4].

Most definitions, theorems and lemmas that follow work when arithmetic is performed over an arbitrary field $\mathbb{F}$. In practice the words on each drive consist of k bits, thus arithmetic is usually preformed over $\mathrm{GF}(2^k)$, and each word is an element in this field.

Definition 3.1. *If A is an $m \times n$ matrix, $m \geq n$, such that any $n \times n$ submatrix of A is invertible, then A is an* information dispersal matrix *or* IDM.

Theorem 3.2. *An $m \times n$ generalized Vandermonde matrix $G(\{f_i\}, \vec{\alpha})$ with $m \geq n$ is an information dispersal matrix if and only if α_i are distinct.*

Proof. Any n rows of $G(\{f_i\}, \vec{\alpha})$ form a square generalized Vandermonde matrix which, by Theorem 2.4, is invertible if and only if α_i are distinct. □

Corollary 3.3. *The Vandermonde matrix $V^{N+M,N}(\vec{\alpha})$ is an IDM if and only if it is generated by $\vec{\alpha}$ with all entries distinct.*

Example 3.4. *For $\mathbb{F} = GF(2^8)$, $N = M = 4$, and $\vec{\alpha} = [0, 1, \ldots, 7]^T$,*

$$V^{8,4}(\vec{\alpha}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 5 & 15 \\ 1 & 4 & 16 & 64 \\ 1 & 5 & 17 & 85 \\ 1 & 6 & 20 & 120 \\ 1 & 7 & 21 & 107 \end{bmatrix} \tag{9}$$

is an information dispersal matrix.

Definition 3.5. *If A is an $m \times n$ information dispersal matrix such that the first n rows of A form the $n \times n$ identity matrix, then A is said to be* systematic. *If A does not have this property then it is said to be* non-systematic.[2]

When a systematic IDM is used to encode data, the encoded data C will have the original data D as its first N entries, then some check data as its last M entries. Computationally, this may be particular convenient.

Example 3.6. *The IDM in Example 3.4 is non-systematic.*

3.1 Plank's Information Dispersal Matrix

In [5] Plank proposes a particular information dispersal matrix. It is constructed in the following manner: start with $V^{N+M,N}(\vec{\alpha})$ with α_i distinct, then perform elementary column operations on this matrix such that the top N rows are transformed into the identity matrix[3] . This is accomplished by performing Gaussian elimination on the columns of $V^{N+M,N}(\vec{\alpha})$. The result will be a systematic matrix $F^{N,M}(\vec{\alpha})$.

[2]This terminology is consistent with [2], [6] and [1].

[3]In [5] , Plank used $\alpha_i = i$ and $\mathbb{F} = GF(2^k)$, however this matrix works over any field for arbitrary $\vec{\alpha}$ so long as α_i are distinct.

Theorem 3.7. *The matrix $F^{N,M}(\vec{\alpha})$ proposed by Plank is a systematic information dispersal matrix.*

Proof. We obtain $F^{N,M}(\vec{\alpha})$ by transforming $V^{N+M,N}(\vec{\alpha}) = \left[\vec{\alpha}^0|\vec{\alpha}^1|\vec{\alpha}^2|\ldots|\vec{\alpha}^{N-1}\right]$ by performing elementary column operations. Thus each column of $F^{N,M}(\vec{\alpha})$ is a linear combination of the columns of $V^{N+M,N}(\vec{\alpha})$, *i.e.* a polynomial of degree $N-1$ over $\mathbb{F}$. So, $F^{N,M}(\vec{\alpha}) = [f_0(\vec{\alpha})|f_1(\vec{\alpha})|\ldots|f_{N-1}(\vec{\alpha})]$, where f_i are polynomials of degree $N-1$. Moreover, the top N rows of $F^{N,M}(\alpha)$ form the identity matrix. Thus, by Corollary 2.7, $F^{N,M}(\alpha)$ is a generalized Vandermonde matrix which is an information dispersal matrix by Theorem 3.2. By construction, $F^{N,M}(\vec{\alpha})$ is systematic. □

Algorithm 3.8. *What follows is pseudo-code for constructing Plank's IDM $F^{N,M}(\vec{\alpha})$. A version of this algorithm appears in [5]. All arithmetic should be performed over $GF(2^k)$.*

1. *Construct $V^{N+M,N}(\alpha)$ for $\alpha = [0, 1, \ldots, M+N]^T$:*

```
for c = 0 to N - 1
    for r = 0 to N  + M - 1
        V(r, c) =  r^c
    end
end
```

2. *Perform Gaussian elimination on the columns $V^{N+M,N}(\vec{\alpha})$ until the first N rows of the matrix are the identity.*

```
F = V   (start with the Vandemonde matrix constructed above)

for c = 0 to N-1
    % rescale the column so the entry on the diagonal is 1
    if F(c, c) !=1
        for r = i to N+M-1
            F(r, c) = F(r, c)/F(c, c)
        end
    end

    % zero out the entry in row c of each column except for column c
    for i = 0 to N - 1, i != c
        for r = c to N+M-1
            V(r, i) = V(r, i) - V(r, c)*V(c, i)
        end
    end
end
```

Example 3.9. *For* $\mathbb{F} = GF(2^8)$*,* $N = M = 4$*, and* $\vec{\alpha} = [0, 1, \ldots, 7]$*,*

$$F^{4,4}(\vec{\alpha}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 27 & 28 & 18 & 20 \\ 28 & 27 & 20 & 18 \\ 18 & 20 & 27 & 28 \\ 20 & 18 & 28 & 27 \end{bmatrix} \tag{10}$$

is the information dispersal matrix obtained by performing Gaussian elimination on $V^{8,4}(\vec{\alpha})$ *shown in (9).*

3.2 Lagrange Polynomials and IDMs

As noted in the proof of Theorem 3.7, the columns of $F^{N,M}(\vec{\alpha})$ are generate by degree $N-1$ polynomials $\{f_i\}_{i=0}^{N-1}$. Using the constraint $f_i(\alpha_j) = \delta_{ij}$, we can write an explicit formula for these functions using Lagrange polynomials.

Theorem 3.10. *The information dispersal matrix* $F^{N,M}(\vec{\alpha})$ *discussed in Section 3.1 has the form*

$$F^{N,M}(\vec{\alpha}) = [f_0(\vec{\alpha})|f_1(\vec{\alpha})|\ldots|f_{N-1}(\vec{\alpha})] \tag{11}$$

where

$$f_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{N-1} \frac{x - \alpha_j}{\alpha_i - \alpha_j}. \tag{12}$$

Proof. From the proof of Theorem 3.7, $F^{N,M}(\vec{\alpha}) = [f_0(\vec{\alpha})|f_1(\vec{\alpha})|\ldots|f_{N-1}(\vec{\alpha})]$, where $\{f_i\}_{i=0}^{N-1}$ are degree $N-1$ polynomials. The constraint $f_i(\alpha_j) = \delta_{ij}$ for $0 \leq i, j < N$ defines these polynomials uniquely. Using Lagrange polynomials yields the form of $f_i(x)$ given. □

Algorithm 3.11. *What follows is pseudo-code for an algorithm to construct* $F^{N,M}(\vec{\alpha})$ *via Lagrange polynomials. This produces the same matrix as Algorithm 3.8.*

1. *Construct the functions* $f_i(x)$ *per equation 12.*
2. *Construct the matrix:*

```
for i = 0 to N + M - 1
    for j = 0 to N -1
        F(i,j) = f_j(alpha_i)
    end
end
```

3.3 The Parity Row

A RAID-5 system has exactly one check drive which is often called the *parity* drive. If the data is broken into k-bit words on each disk, then the word-wise sum of all data drives it taken modulo 2^k and stored on the check disk. If $k = 1$ and words are single bits, then the check drive contains the parity of the sum of the data drives. The RAID-5 system corresponds to an IDM that is an identity matrix with one extra row consisting of all ones appended to the end.

When arithmetic is performed over $\mathrm{GF}(2^k)$, summing elements is computational cheaper than multiplying elements. Constructing a parity drive requires $N-1$ additions, whereas constructing an arbitrary check drive requires N multiplications and $N-1$ additions, so parity drives are much less expensive to construct. It is also cheaper to maintain a parity drive when data on a data drive is updated, and to recover data from a data drive in the event of a drive failure. Hence, we would like to work with systematic IDMs that include a parity row. In this section, we will consider when $F^{N,M}(\vec{\alpha})$ is guaranteed to have a parity row.

We will restrict our attention to the case $M = 1$. Notice that if for some N, $\vec{\alpha}$, $F^{N,1}(\vec{\alpha})$ has a parity row as its only non-identity row, then the matrix $F^{N,M}([\vec{\alpha}, \vec{\beta}])$ will also have a parity row as its first non-identity row for any $\vec{\beta}$ of length $M-1$. Since we can only have one parity row, studying $M = 1$ yields all necessary information for the general case.

Example 3.12. *Let $\mathbb{F} = GF(2^8)$, and $M = 1$. Then for $N = 3$ and $\vec{\alpha} = [0, 1, 2, 3]$,*

$$F^{4,4}(\vec{\alpha}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{13}$$

Notice this matrix contains the desired parity row.

Example 3.12 shows one case where such a parity row is found. It is a stroke of luck that any such matrix is to be found! Looking further, we see that for $\mathbb{F} = GF(2^8)$, $M = 1$, $N = 3, 7, 15, 31, 63, 127$ and $\vec{\alpha} = [0, 1, 2, \ldots, N-1]$, $F^{N,M}([\vec{\alpha}])$ always contains a parity row. Notice for that these matrices $N = 2^n - 1$ for some n. From this observation (and some more thought) we arrive at Theorem 3.13.

Theorem 3.13. *Let $\mathbb{F} = GF(2^k)$, and $\mathcal{A}$ be a coset of an additive subgroup of $\mathbb{F}$. Let $N = |\mathcal{A}| - 1$, and $\vec{\alpha}$ be a vector containing all elements of $\mathcal{A}$ exactly once in any order. Then the last row of $F^{N,1}([\vec{\alpha}])$ is a parity row.*

Proof. Recall from Theorem 3.10 that

$$F^{N,1}(\vec{\alpha}) = [f_0(\vec{\alpha})|f_1(\vec{\alpha})|\ldots|f_{N-1}(\vec{\alpha})]$$

where

$$f_i(x) = \prod_{\substack{0 \le j \le N-1 \\ j \ne i}} \frac{x - \alpha_j}{\alpha_i - \alpha_j}.$$

Let

$$\begin{aligned} \mathcal{N}_i &= \{\alpha_N - \alpha_j : 0 \le j \le N-1, j \ne i\} \quad \text{and} \\ \mathcal{D}_i &= \{\alpha_i - \alpha_j : 0 \le j \le N-1, j \ne i\}. \end{aligned}$$

Then $|\mathcal{N}_i| = |\mathcal{D}_i| = N - 1$, and

$$f_i(\alpha_N) = \frac{\prod_{n \in \mathcal{N}_i} n}{\prod_{d \in \mathcal{D}_i} d}.$$

Since $\mathcal{A}$ is the coset of an additive subgroup of $\mathbb{F}$, there exists $\mathcal{B}$ and an additive subgroup of $\mathbb{F}$ and $g \in \mathbb{F}$ such that for any $\alpha_i \in \mathcal{A}$ there exists $b_i \in \mathcal{B}$ such that $\alpha_i = g + b_i$. Thus for any j, $\alpha_N - \alpha_j = g + b_N - (g + b_j) = b_N - b_j \in \mathcal{B}$, and $\mathcal{N}_i \subset \mathcal{B}$ for any i. Similarly, $\mathcal{D}_i \subset \mathcal{B}$ for any i. $|\mathcal{B}| = |\mathcal{A}| = N + 1$, so there are exactly two elements that are in $\mathcal{B}$ and not in $\mathcal{N}_i$ and two elements in $\mathcal{B}$ that are not in $\mathcal{D}_i$. Notice that $0 \notin \mathcal{N}_i$ since $N \ne j$ and $0 \notin D_i$ since $i \ne j$. As well, $\alpha_N - \alpha_i \notin \mathcal{N}_i$ since $j \ne i$, $\alpha_i - \alpha_N = \alpha_N - \alpha_i \notin \mathcal{D}_i$ since $j \ne N$ and $a = -a$ for any $a \in GF(2^k)$, and $\alpha_N - \alpha_i \in \mathcal{B}$. Thus we have identified two elements that are in $\mathcal{B}$, namely 0 and $\alpha_N - \alpha_i$, that are not in either $\mathcal{N}_i$ or $\mathcal{D}_i$. Thus for any i, $\mathcal{N}_i = \mathcal{D}_i$, $\prod_{n \in \mathcal{N}_i} n = \prod_{d \in \mathcal{D}_i} d$, and $f_i(\alpha_N) = 1$. Therefore the Nth (last) row of $F^{N,1}(\vec{\alpha})$ is a parity row. □

Remark 3.14. *Since the order of a subgroup must divide the order of the group, Theorem 3.13 only guarantees the existence of systematic IDMs with a parity row when $\mathbb{F} = GF(2^k)$ and $N = 2^n - 1$ for $n \le k$.*

Lemma 3.15. *Let $\mathbb{F} = GF(2^k)$, $n \le k$, $N = 2^n - 1$, $\alpha_i = i$ for $0 \le i \le N$. α_i may be arbitrary for $N < i < N + M$. Then $F^{N,M}(\vec{\alpha})$ is a systematic IDM with Nth row a parity row.*

Proof. This follows directly from Theorem 3.13 noting that $\mathcal{A} = \{0, 1, 2, \ldots N\}$ is an additive subgroup of $GF(2^k)$. □

3.4 A New IDM

In the previous section we have shown how to construct systematic IDMs with parity rows. Unfortunately, this method only works for $N = 2^n - 1$. In this section, we will learn how to construct such an IDM for arbitrary N. This new class of IDMs will be constructed from the IDMs proposed by Plank by removing certain rows and columns from the matrix $F^{N,M}(\vec{\alpha})$.

Theorem 3.16. *Fix $n > N$ and $\vec{\alpha} \in \mathbb{F}^{n+M}$. Construct a new $(N+M) \times N$ matrix $H_n^{N,M}(\vec{\alpha})$ by discarding any $n-N$ columns of $F^{n,M}(\vec{\alpha})$ and discarding the corresponding rows as well (*i.e. *if column k is discarded then discard row k as well). Then $H_n^{N,M}(\vec{\alpha})$ is a systematic IDM.*

Proof. Using the notation of Theorem 3.7, let $F^{n,M}(\vec{\alpha}) = [f_0(\vec{\alpha})|f_1(\vec{\alpha})|\dots|f_{n-1}(\vec{\alpha})]$, where $\{f_i\}$ is a set of linearly independent degree $n-1$ polynomials. In search of contradiction, assume that $H_n^{N,M}(\vec{\alpha})$ is not an IDM. Then there exists an $N \times N$ sub matrix of $H_n^{N,M}(\vec{\alpha})$ that is not invertible. Let $\mathcal{I} \subset \{0, \dots, n-1\}$ with $|\mathcal{I}| = N$ be the set of the indices of the columns of $F^{n,M}(\vec{\alpha})$ that are *kept* when constructing $H_n^{N,M}(\vec{\alpha})$, and $\mathcal{J} \subset \mathcal{I} \cup \{n, \dots n+M-1\}$, $|\mathcal{J}| = N$, be the indices of the rows from $F^{n,M}(\vec{\alpha})$ that are used to form the non-invertible $N \times N$ sub matrix of $H_n^{N,M}(\vec{\alpha})$. Then there exists $\vec{c} \neq \vec{0} \in \mathbb{F}^N$ such that

$$\sum_{i \in \mathcal{I}} c_i f_i(\alpha_j) = 0, \qquad \forall j \in \mathcal{J}. \tag{14}$$

Let $F(x) = \sum_{i \in \mathcal{I}} c_i f_i(x)$. F is a degree $n-1$ polynomial that vanishes at the N points α_j, $j \in \mathcal{J}$. As well, for $a \in \{0, \dots, n\} \backslash \mathcal{I}$ (the identity rows discarded from $F^{n,M}(\vec{\alpha})$ when constructing $H_n^{N,M}(\vec{\alpha})$), $f_i(\alpha_a) = 0$ for all $i \in \mathcal{I}$ and thus $F(\alpha_a) = 0$. Therefore, F vanishes at all N points in $\mathcal{J}$ and all $n-N$ points in $\{0, \dots, n\} \backslash \mathcal{I}$, n points in total. Since F is a degree $n-1$ polynomial, this implies that F is uniformly zero. This contradicts the linear independence of the polynomials $\{f_i\}$, hence $H_n^{N,M}(\vec{\alpha})$ is an IDM and by construction it is systematic.

□

Algorithm 3.17. *What follows is pseudo code for generating an IDM for any $N \leq 127$ and $M \leq 129$.*

1. *Construct $F = F^{127,129}([0, 1, \dots, 255]^T])$ by Algorithm 3.8 or 3.11.*
2. *Construct $H = H_{127}^{N,M}([0, 1, \dots, 255]^T)$, an IDM for N data disks and M check disks, by setting the first N rows of H equal to the $N \times N$ identity matrix, and the last M rows of H equal to rows 127 through $126 + M$ and columns 0 through $N-1$ of F.*

Remark 3.18. *Algorithm3.17 has two advantages over previous IDMs discussed:*

1. *The first non-identity row (row 127) of $F^{127,129}([0, 1, \dots, 255]^T])$ is a parity row, and so in any IDM derived from it, the first check drive will be a classical parity checker. That is, H is a systematic IDM with parity row for any N and M.*
2. *F may be computed once, and IDMs for* any *$N \leq$ and $M \leq 129$ derived from it with virtually no additional computational cost. In particular, one could store F in hardware, then the user could set N and M later but still reap the speed advantages of using a hard-coded IDM.*

Example 3.19. *For $\mathbb{F} = GF(2^8)$, $n = 4$, $N = 3$ and $M = 4$, and $\vec{\alpha} = [0, 1, \ldots, 7]$,*

$$H_4^{3,4}(\vec{\alpha}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 27 & 28 & 18 \\ 28 & 27 & 20 \\ 18 & 20 & 27 \\ 20 & 18 & 28 \end{bmatrix} \tag{15}$$

is the information dispersal matrix obtained by removing column and row 3 from $F^{4,4}(\vec{\alpha})$ shown in (10).

Example 3.20. *For $\mathbb{F} = GF(2^8)$, $n = 127$, $N = 3$ and $M = 4$, and $\vec{\alpha} = [0, 1, \ldots, 255]$,*

$$H_{127}^{3,4}(\vec{\alpha}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 191 & 158 & 109 \\ 168 & 137 & 145 \\ 101 & 175 & 183 \end{bmatrix}. \tag{16}$$

This is the $N = 3$, $M = 4$ IDM obtained using Algorithm 3.17. Notice that row 3 consists of all ones.

3.5 Error Detection and Correction

In the preceding sections in designing our IDMs, we have restricted our attention to the situation where some drives are known to have failed, and we wish to recover the information on those drives. However, it is sometimes the case that there are errors in the stored data that are not known to exist before reading the data. This is the case when data is miswritten or misread to and from the drive, or when the data has been somehow corrupted on the drive, for example. In this section we will consider under which conditions these errors can be detected and corrected, and how to accomplish both of these tasks.

Whereas previously we have considered a matrix D of the original data and C of the encoded data, in this section we will consider only a single column from each of these matrices. Let d be a vector of one word from each data disk, and c be the encoding of d under $c = Fd$.

Definition 3.21. *Let $c \in \mathbb{F}^n$, and F be an $n \times m$ IDM. c is a* codeword *under F if there exists $d \in \mathbb{F}^m$ such that $c = Fd$.*

Definition 3.22. *Let $\vec{a}$ and $\vec{b}$ be vectors in $\mathbb{F}^n$. Then the* Hamming distance *between $\vec{a}$ and $\vec{b}$ is*

$$D(\vec{a}, \vec{b}) = n - \sum_{i=1}^{n} \delta_{a_i, b_i}.$$

$D(\vec{a}, \vec{b})$ is the count of the indices where $\vec{a}$ and $\vec{b}$ differ. The Hamming distance was introduced by Hamming in [2] where he noted that D is a metric on $\mathbb{F}^n$.

Remark 3.23. *Let c be a codeword that has been stored in our RAID system. Upon reading, c' is returned. $D(c, c')$ gives the number of errors in the read data as compared to the codeword that was intended.*

Theorem 3.24. *Let F be an $(N + M) \times N$ IDM and c_1 and c_2 be two codewords under F. Then $D(c_1, c_2) \geq M + 1$.*

Proof. In search of contradiction assume there exists distinct codewords c_1 and c_2 with $D(c_1, c_2) \leq M$. Then c_1 and c_2 agree at at least N indices. There exists $d_1, d_2 \in \mathbb{F}^N$ such that $c_1 = Fd_1$ and $c_2 = Fd_2$. Construct $\hat{F}$ from N rows of F corresponding to indices where c_1 and c_2 agree. Then $\hat{F}d_1 = \hat{F}d_2$. Since F is an IDM, $\hat{F}$ is invertible, so we have $d_1 = d_2$ and thus $c_1 = c_2$. This contradicts c_1 and c_2 being distinct, and so no such c_1 and c_2 may exist. □

Lemma 3.25. *Let F be an $(N + M) \times N$ IDM, and c' be some read word. Then if there are no more than $\frac{M}{2}$ errors in c' then it may be uniquely decoded.*

Proof. This follows from Theorem 3.24. □

4 The Computational Cost of RAID

In the preceding sections, we have discussed mathematical methods for creating RAID systems. Here we evaluate the computational cost of creating check drives and recovering from drive failures. As in Section 3, assume that there are N data drives and M check drives, each with l bytes of data. We represent our original data as D, the encoded data as C and the IDM as F. F may be either Plank's IDM from Section 3.1 or the new IDM proposed in Section 3.4, but is assumed to be systematic. We will assume that $\mathbb{F}$ is a finite field.

4.1 Generation of Check Drives

Some computation is necessary to construct the encoded data C from the original data D using (7). Clearly, no computations are required to calculate the first N rows of C since these rows are exact copies of D. To calculate each byte in the M check drives in C, one must dot a row of F with a column of D. Each of these vectors has length N,

so this requires N multiplications and $N-1$ additions. Therefore NMl multiplications and $(N-1)Ml$ additions are required to create all M check drives. This is essentially M multiplications and additions per original byte of data.

4.2 Recovery of Lost Data

We will now consider the computational cost of recovering lost data. To recover from data lost, it will be helpful to rearrange the matrices F, D, and C, so that we focus on recovering failed drives and use the remaining data and the structure of F most efficiently. This will require some new notation which is summarized in Table 1 below.

Notice that since F is systematic, it has the form

$$F = \begin{bmatrix} I_N \\ \star \end{bmatrix},$$

where I_N is $N \times N$ identity matrix, and $\star$ some matrix. We have the relation $FD = C$, and notice due to the structure of F,

$$C = \begin{bmatrix} D \\ \star \end{bmatrix},$$

where $\star$ represents some matrix of check data. I will refer to the drives in C that are copies of the data D as the data drives, and the drives that correspond to $\star$ as the check drives. Assume m of the data drives of C fail, and thus $k = N - m$ drives survive. Then so long as no more than $M - m$ of the check drives fail, we can recover the data from the failed drives on C. We need only N drives total to recover this data, so we may discard extra check drives if fewer than M total drives have failed.

Construct $\hat{C}$ as N drives of C that have not failed, including all working data drives. Mathematically, we can represent this by matrix multiplication as

$$\hat{C} = QC$$

where Q is the $M+N$ identity matrix with the M rows corresponding to failed (or ignored) drives removed.

It will also be advantageous to rearrange D so that good drives appear first and failed drives appear last. Mathematically, this may be accomplished by

$$\hat{D} = PD = \begin{bmatrix} X \\ Y \end{bmatrix}$$

where P is the appropriate permutation matrix, X is the surviving data, and Y is the missing data. If we then set $\hat{F} = QFP^T$ (noting that $P^T = P^{-1}$) we have the reduced system

$$\hat{F}\hat{D} = \hat{C}.$$

matrix	dimensions	meaning	relations
F	$(N+M) \times N$	IDM	$C = FD = \begin{bmatrix} D \\ \star \end{bmatrix}$
D	$N \times l$	Original data	
C	$(N+M) \times l$	Encoded data (original + check)	
Q	$N \times (N+M)$	Drive selection matrix	
P	$N \times N$	Permutation matrix	
$\hat{C}$	$N \times l$	Surviving encoded data	$\hat{C} = QC$, $\hat{C} = \hat{F}\hat{D}$
$\hat{D}$	$N \times l$	Original data, permuted	$\hat{D} = PD$
$\hat{F}$	$N \times N$	Transformed IDM	$\hat{F} = QFP^T$
X	$k \times l$	Surviving original data	$\hat{D} = \begin{bmatrix} X \\ Y \end{bmatrix}$, $\hat{C} = \begin{bmatrix} X \\ W \end{bmatrix}$
Y	$m \times l$	Lost original data	
W	$m \times l$	Surviving check data	
I_k	$k \times k$	Identity matrix	$\hat{F} = \begin{bmatrix} I_k & 0 \\ A & B \end{bmatrix}$
0	$k \times m$	Matrix of zeros	
A	$m \times k$		
B	$m \times m$		

Table 1: List of matrices used in the analysis of the computational complexity of recovering lost data, their dimensions, meaning, and relations between them. l is the number of bytes on each data drive, m is the number of failed data drives (not including failed check drives), and $k = N - m$ is the number of surviving data drives.

Furthermore, $\hat{F}$ has the additional block structure

$$\hat{F} = \begin{bmatrix} I_k & 0 \\ A & B \end{bmatrix},$$

so we have the system of equations

$$\begin{matrix} \hat{F} & & \hat{D} & & \hat{C} \end{matrix}$$
$$\begin{bmatrix} I_k & 0 \\ A & B \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} X \\ W \end{bmatrix}.$$

This can be rewritten as

$$\begin{aligned} X &= X \\ AX + BY &= W. \end{aligned}$$

Y is the unknown lost data, and all other variables are known, so we simply solve this system for Y:

$$Y = B^{-1}(W - AX). \tag{17}$$

It is the computation of Y via this formula in which we are interested. I will assume that the operation cost associated with constructing B, W, A, and X is negligible, as through clever referencing of F, and C they need not be explicitly constructed. There are three steps in computing Y with operations counts as follows:

1. Compute AX. This requires kml multiplies and $(k-1)ml$ additions.
2. Compute $W - AX$. This requires ml subtractions.
3. Solve $BY = W - AX$. There are a few algorithms available to solve this system. Let's assume this is done using the LU factorization of B.
   (a) Compute the LU factorization of B without pivoting[4]. This requires $\frac{1}{2}m(m-1)$ divides, $\frac{1}{3}(m^3 - m)$ multiplies, and $\frac{1}{3}(m^3 - m)$ subtractions.
   (b) Solve $LZ = W - AX$ for Z using forward substitution. This requires $(m-1)l$ subtractions, $\frac{1}{2}(m^2 - 3m + 1)l$ additions, and $\frac{1}{2}(m^2 - 3m + 1)l$ multiplications.
   (c) Solve $UY = Z$ for Y using backward substitution. This requires $(m-1)l$ subtractions, ml divisions, $\frac{1}{2}(m^2 - 3m + 1)l$ additions, and $\frac{1}{2}(m^2 - 3m + 1)l$ multiplications.

[4]The standard algorithm for LU factorization is done with partial pivoting in which in step i, the row with the largest entry in column i is chosen as the pivot. This is done to improve the numerical stability of Gaussian elimination. For this application, all arithmetic is done over a finite field and thus numerical stability is not a concern. Thus partial pivoting is not necessary. It may still be necessary to exchange rows during Gaussian elimination when a diagonal entry is zero. However, the cost of this step is negligible and is not included in this analysis. See [7] for a full discussion of the LU factorization.

The overall operation count is (taking into account only the arithmetical operations in the finite field and not considering operations such as copying and comparisons):

Additions (including subtractions): $ml(m+k-1) - l + \frac{1}{3}(m^3 - m))$;

Multiplications: $ml(m+k-3) + l + \frac{1}{3}(m^3 - m)$;

Divides: $ml + \frac{1}{2}m(m-1)$.

RAID systems are generally measured by their throughput, that is, how many bytes can be accepted or delivered within a fixed time interval. With this in mind, the most useful perspective on the cost of computation is with regard to each data byte that is accepted or delivered, *i.e.* the number of original data bytes in the system which is the number of data bytes in D, Nl. We can think of the operation counts listed above as a cost per original byte plus some overhead associated with computing the LU factorization of B. Table 2 summarizes these costs, also using the relationship $N = m + k$ to further simplify expressions.

	count	count per byte of original data	overhead
+, -	$mNl - ml - l + \frac{1}{3}(m^3 - m))$	$m - \frac{m+1}{N}$	$\frac{1}{3}(m^3 - m))$
$\cdot$	$mNl - 3ml + l + \frac{1}{3}(m^3 - m)$	$m - \frac{3m-1}{N}$	$\frac{1}{3}(m^3 - m))$
$\div$	$ml + \frac{1}{2}m(m-1)$	$\frac{m}{N}$	$\frac{1}{2}m(m-1)$

Table 2: Summary of the count of operations required to recover lost data, both as an overall count and broken down into the cost per byte recovered plus some fixed overhead cost. N is the total number of data drives (not including check drives), m is the number of data drives that failed, l is the number of bytes per drive, and Nl is the number of bytes of original data. The operations addition and subtraction have been listed together as they are the same operation over $GF(2^k)$.

In summary, if m data drives are lost, then it costs about m additions and multiplies and < 1 divide per byte of original data requested to read data plus an additional m^3 additions and multiplies and m^2 divides of overhead costs associated with computing the LU factorization. When $m \ll l$ as is typically the case, the cost of m additions and multiplies per original data byte is the dominating cost of data delivery. As discussed in Section 4.1, it costs about M additions and multiplies per original data byte to generate (or regenerate) the check drives. Thus the cost of preparation for the failure of M drives (generating M check drives) is essentially the same as the cost of recovering from the failure of M drives: M additions and M multiplications.

5 Conclusions

In the current paper we examined in detail the technique of constructing *systematic information dispersal matrices* (IDM), originally proposed by Plank in [5]. We also developed a new technique based on dropping certain columns and rows of an existing IDM, thus allowing construction of smaller IDM's from a single large IDM.

We analyzed the application of these IDM's to constructing RAID system. We provided the mathematical equations which allow building check drives and recovering data in case of failure of several drives. Subsequently, we gave the count of arithmetic operations required to perform each of the two tasks. In particular, we showed that each task requires M additions/subtractions, M multiplies, and M/N divides in the underlying finite field, per piece of the original data, where M is the number of check drives in the RAID. It should be noted, these numbers are independent of N or decrease with N, the number of data drives. Although there are startup costs depending on N, these costs turn out to be negligible for realistic drive sizes.

6 About the Authors

Sarah's Pic — Sarah Edge Mann..... a short bio should go here.

Mike's Pic — Mike Anderson.....a short bio should go here.

Marek Rychlik.....a short bio should go here

References

[1] W.A. Burkhard and J Menon. Disk array storage system reliability. *Fault-Tolerant Computing, 1993. FTCS-23. Digest of Papers., The Twenty-Third International Symposium on*, pages 432–441, 1993.

[2] R. W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 26(2):147–160, 1950.

[3] Dan Kalman. The generalized Vandermonde matrix. *Mathematics Magazine*, 57(1):15–21, January 1984.

[4] J. S. Plank. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems. *Software – Practice & Experience*, 27(9):995–1012, September 1997.

[5] J. S. Plank and Y. Ding. Note: Correction to the 1997 tutorial on Reed-Solomon coding. Technical Report CS-03-504, University of Tennessee, April 2003.

[6] Jason K. Resch and James S. Plank. AONT-R: blending security and performance in dispersed storage systems. In *Proceedings of the 9th USENIX conference on ...*, February 2011.

[7] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, June 1997.