XOR's, Lower Bounds and MDS Codes for Storage

James S. Plank

plank@cs.utk.edu

Technical Report CS-11-672
Department of Electrical Engineering and Computer Science
University of Tennessee

May, 2011.

This paper has been submitted for publication.
See the web link below for current publication status.

http://www.cs.utk.edu/~plank/plank/papers/CS-11-672.html

XOR's, Lower Bounds and MDS Codes for Storage

James S. Plank
Department of Electrical Engineering and Computer Science
University of Tennessee
Knoxville, TN 37919
Email: plank@cs.utk.edu

***Abstract*—MDS erasure codes are ubiquitous in storage systems that must tolerate failures. While classic Reed-Solomon codes can provide a general-purpose MDS code for any situation, systems that require high performance rely on special-purpose codes that employ the bitwise exclusive-or (XOR) operation, and may be expressed in terms of a binary generator matrix. There are known lower bounds on the density of the generator matrix; however the number of XOR operations required to encode the generator matrix, while related to the density of the matrix, has not been explored.**

This paper presents a stunning and counter-intuitive result — that the encoding of k message symbols in an MDS code may require fewer than $k-1$ XOR operations per coding symbol. The result brings into question the lower bounds on the performance of encoding and decoding MDS erasure codes.

I. Introduction

From the rise of RAID in the 1990's, storage systems have relied on MDS erasure codes to tolerate failures. The typical configuration is an extrapolation of RAID-4, as depicted in Figure 1. A system of n disks is partitioned into k disks that hold data, and m disks whose contents are calculated from the data using an erasure code. If the erasure code is a Maximum Distance Separable (MDS) code, then the system can tolerate the loss of any m disks through failures without data loss.

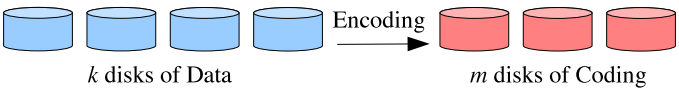


Fig. 1. An example of an n-disk RAID system that uses an MDS erasure code to encode k disks of data onto $m = n - k$ disks of coding. Because the code is MDS, the system can tolerate the failure of any m disks without losing data.

RAID-4 and RAID-5 systems arose in the early 1990's and have enjoyed considerable commercial success in providing the framework for large-scale storage systems [6]. With RAID-4 and RAID-5, there is only one coding disk (i.e. $m = 1$), whose contents are calculated as the bitwise exclusive-or (XOR) of the data disks. In the early 2000's, storage systems grew to the point where RAID-4 and RAID-5 did not provide enough protection from failures. In particular, latent sector failures that accumulated unnoticed over time exposed the storage system to data loss upon failure of a disk drive [7], [8], motivating the need for higher RAID levels. RAID-6 systems employ two coding disks (i.e. $m = 2$), and are currently sold commercially by all major disk vendors.

As storage systems have grown further, both in size and scope, larger amounts of fault-tolerance have become necessary. For example, commercial vendors such as AllMyData.com and Cleversafe sell solutions for large-scale and archival storage that employ MDS codes to tolerate four and six failures respectively [13], [16]. Cloud storage providers such as Microsoft, Amazon and Google all employ MDS erasure codes beyond RAID-6 to prevent data loss. Clearly, the trend for the next decades is for storage systems to employ larger and larger amounts of fault-tolerance.

While there are many factors that impact the performance of an erasure-coded storage system, the CPU performance of encoding and decoding can have major impact [13], [16]. Consider the example of RAID-4, where $m = 1$ and the coding disk is calculated as the parity of the data disks. The CPU performance of RAID-4 is $k - 1$ XOR operations per machine word of the coding disk. This is a lower bound on the number of XOR operations for RAID-4 encoding, and subsequent work on RAID-6 and beyond has assumed that it is a lower bound for all storage systems that employ MDS erasure codes for fault-tolerance [7], [13]. There is good reason for this assumption — theoretical work on MDS codes has proven lower bounds on the density of generator matrices for these codes [4], and intuitively, the density of the generator matrices has a direct relationship on the number of XOR operations required for encoding and decoding.

In this paper, we demonstrate that $k-1$ XOR operations per coding word is *not* a lower bound for encoding with an MDS code. We find this result both counter-intuitive and stunning; yet we demonstrate how it arises in codes with (n,k) values as small as (6,3). We explore the phenomenon with other small codes, and we challenge the Information Theory community to explore its theoretical underpinnings.

II. Problem Statement

We are exploring systematic (n,k) MDS erasure codes whose sole encoding and decoding operation is the bitwise exclusive-or (XOR). Our code takes a vector D of k message symbols, $d_0, \ldots, d_{k-1}$, each of which is w bits in length. It outputs a vector C of $m = n-k$ coding symbols $c_0, \ldots, c_{m-1}$, each of which is also w bits. The vector $(C|D)$ composes the

codeword of size n. If any m of the n symbols in the codeword are erased, then the original message may be restored from the surviving symbols.

When an erasure code is implemented in a disk-based storage system, the bits that compose each symbol are each stored on a separate sector of the disk. The only operation allowed is the XOR operation, which means that when bits are XOR'd in the erasure code, whole sectors are XOR'd on disk. This is an efficient operation. As such, we refer to the individual bits of each word, d_i or c_j as $d_{i,0}, \ldots, d_{i,w-1}$ or as $c_{j,0}, \ldots, c_{j,w-1}$. Morover, we can equivalently view D and C as bit vectors of size kw and mw respectively. In fault-tolerant storage systems, disks typically hold "data" rather than "messages." However, to keep consistent with standard coding theory nomenclature, we will call D the "message."

A systematic erasure code is defined by a $kw \times nw$ generator matrix $G = (I|P)$. Each element of G is a bit, and the vector-matrix product DG over $GF(2)$ equals the codeword. Thus $DP = C$. When erasures occur, we decode by forming a square *decoding* matrix from G that corresponds to k of the surviving words in the codeword, inverting it and multiplying it by the survivors. A generator matrix G defines an MDS code if every decoding matrix that can be made in this way is invertible.

The acts of encoding and decoding, therefore, may be viewed as calculating vector-matrix products over $GF(2)$. We provide a concrete example in Figure 2. This is the (6,4) RDP code for RAID-6 [7]. Each disk holds a 4-bit symbol, one bit on each of four sectors.

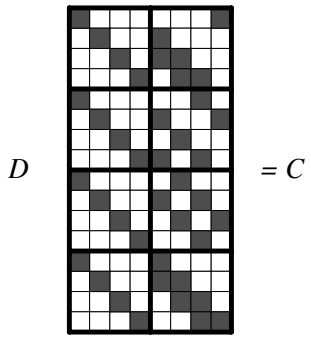


Fig. 2. The (6,4) RDP RAID-6 erasure code, defined as $DP = C$ over 4-bit words. Thus, D is a vector of four 4-bit words, or equivalently a bit vector of size 16. Similarly, C is a vector of two 4-bit words, or a bit vector of size eight.

The straightforward way to calculate this vector-matrix product is to calculate each bit of C independently. For example, $c_{0,1}$ and $c_{1,0}$ may be calculated as:

$$\begin{aligned} c_{0,1} &= d_{0,1} + d_{1,1} + d_{2,1} + d_{3,1} \\ c_{1,0} &= d_{0,1} + d_{0,2} + d_{1,1} + d_{1,3} + d_{2,1} + d_{3,0} + d_{3,1} \end{aligned}$$

Since arithmetic is over $GF(2)$, addition is equal to XOR.

When the bits of C are calculated independently, the number of XOR operations is determined by the density of P. In particular, if a column has o ones, then its corresponding coding bit may be calculated with $o-1$ XOR's. The example in Figure 2 requires 33 XORs, or 4.125 XORs per coding bit, when encoded in this manner.

However, RDP has been designed so that its generator has a particular stucture. In particular, each sum $c_{1,i}$ for $0 \leq i < w-1$ contains $c_{0,i-1}$ as a subexpression. In the example above:

$$c_{1,0} = c_{0,1} + d_{0,2} + d_{1,3} + d_{3,0}.$$

Therefore, if the bits of c_0 are calculated before the bits of c_1, then they may be used to calculate c_1 with far fewer XOR operations. In the example of Figure 2, this leads to 24 total XOR's, or exactly three XOR's per coding bit.

Given this backdrop, the question we address is the following:

> *What is the lower bound on the number of XOR operations required to encode and/or decode a systematic (n, k) MDS erasure code?*

When $m = 1$, it is trivial to show that RAID-4, which requires $k-1$ XORs to encode and decode, is the lower bound. For $m = 2$, RDP and its supersets [18] also require $k-1$ XOR's per coding or decoded bit, and this is claimed to be optimal without proof [7]. Blaum gives constructions that achieve this bound for higher values of m [1], and these are also claimed to be optimal. Given that the P portion of any systematic MDS generator matrix must have at least kmw non-zero entries [4], it is reasonable to assume that $k-1$ XOR's per coding bit is the lower bound for encoding, and that $k-1$ XOR's per decoded bit is the lower bound for decoding.

This paper demonstrates that for certain (n,k) combinations, the lower bound is *lower* than $k-1$ XOR's. We briefly discuss the concept of scheduling bit matrix/vector products, and show results of an exploration of performing scheduling on Cauchy Reed-Solomon generator matrices. In many cases, the performance of encoding is less than $k-1$ XOR's per coding bit.

While our work is in some sense happenstance, in that we made this discovery through experimentation, we believe that the result is important, and bears exploration by the theoretical community.

III. Scheduling Matrix-Vector Products in $GF(2)$

Given a $r \times c$ bit matrix P, we define a schedule S to be an ordered set $\{s_0, s_1, \ldots\}$ of r-bit column vectors. We refer to the bits of s_i as $s_{i,0}, \ldots, s_{i,r-1}$. To be a valid schedule, S must have the following properties:

- For $0 \leq i < r$, s_i is defined such that $s_{i,i} = 1$ and for all $j \neq i$, $s_{i,j} = 0$.
- For $i \geq r$, there must exist $x < y < i$ such that for all j, $s_{i,j} = s_{x,j} + s_{y,j}$.
- For each column P_j of P, $P_j \in S$.

Intuitively, the first r elements of S represent each bit of the message, and each subsequent element represents a sum that

is generated by one XOR operation. Since each column of P is an element of S, S gives a schedule of XOR operations that can generate P. The total number of XOR operations is $|S|-r$.

We may represent S as a matrix of column-vectors as in Figure 3. This schedule is the optimal way of performing encoding with the (6,4) RDP code from 2. The starred and colored columns depict elements of S that are also columns of the generator matrix. Since the schedule contains 40 elements, it requires $40 - 16 = 24$ XOR's, which is optimal for RDP.

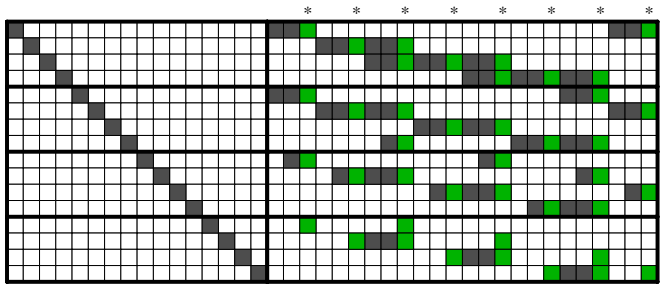


Fig. 3. An optimal scheduling of the (6,4) RDP code of Figure 2. The starred columns are columns of RDP's generator matrix that are in the schedule.

It is an open problem to generate optimal schedules for arbitrary matrices. Huang *et al* conjecture that it is NP-Hard [10]. However, there have been three separate research projects that develop heuristics for generating good schedules. Hafner *et al* propose an algorithm called "Code-Specific Hybrid Reconstruction" (CSHR) which, as in the RDP example, attempts to use previous product bits as starting points for calculating subsequent product bits to reduce XOR's [9]. Their technique generates optimal schedules for RDP and is a necessary optimization for decoding Blaum-Roth and Liberation codes [4], [13].

Huang *et al* propose a technique that uses Edmonds' general matching algorithm to identify common subexpressions within the matrix for XOR reduction [10]. Like CSHR, their technique generates optimal schedules for RDP, and additionally it generates optimal schedules for EVENODD coding [2].

Our recent work has extrapolated upon these projects to yield two new techniques, called "Uber-CSHR" and "X-Sets," which are the most effective at reducing XOR's for arbitrary matrices [14]. We have posted open-source implementations of all three research projects [12], and it is these implementations that we use to generate the results of this paper.

IV. When $k \leq 2$ or $m \leq 2$

It is trivial to show that $k - 1$ XOR's is a lower bound when $k = 1$ or $m = 1$. When $k = 1$, replication provides an MDS code with zero XOR's. When $m = 1$, it is a necessity for each input symbol to be touched once, and this can only occur when k symbols are XOR'd as in RAID-4. This yields $k - 1$ XOR's.

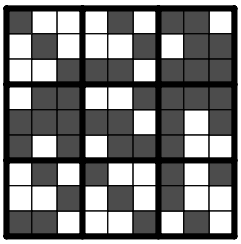
For a code to be MDS, each $w \times w$ panel of P must be an invertible matrix [4]. Moreover, each column of P must be unique; otherwise, a decoding matrix can be created from P that has two identical columns and is therefore not invertible.

When $k = 2$, each column of P must have at least two non-zero bits; otherwise, its $w \times w$ panels are not invertible. Moreover, each column must be unique. Therefore any schedule that generates P must have at least mw columns in addition to the first kw identity columns. This means that the schedule requires at least mw XOR's. There are mw coding bits that need to be generated, so the lower bound is one XOR per coding bit. This equals $k - 1$.

When $m = 2$, we conjecture without proof that $k-1$ XOR's per coding bit is also the lower bound. This is achieved by RDP, which claims its optimality [7].

1	2 (α)	3 (α^3)
6 (α^4)	4 (α^2)	7 (α^5)
2 (α)	1	5 (α^6)

(a): In $GF(2^3)$



(b): In $GF(2)$

Fig. 4. The P portion of a systematic generator matrix for $k = m = w = 3$. This is generated with a Cauchy Reed-Solomon code with $X = \{0, 2, 3\}$ (or $\{0, \alpha, \alpha^3\}$) and $Y = \{1, 5, 6\}$ (or $\{1, \alpha^6, \alpha^4\}$).

V. A (6,3) Code Whose Lower Bound is less than 2

In Figure 4, we present the P part of a generator matrix for $k = m = w = 3$. This is an MDS matrix generated using the Cauchy Reed-Solomon coding technique [5] with $X = \{0, 2, 3\}$ (or $\{0, \alpha, \alpha^3\}$) and $Y = \{1, 5, 6\}$ (or $\{1, \alpha^6, \alpha^4\}$). The 3×3 matrix in $GF(2^3)$ is shown in Figure 4(a). It is calculated with $P_{i,j}$ equal to $\frac{1}{x_i + y_j}$ in $GF(2^3)$. In Figure 4(b), we convert the matrix in $GF(2^3)$ to a 9×9 bit matrix as specified in [5].

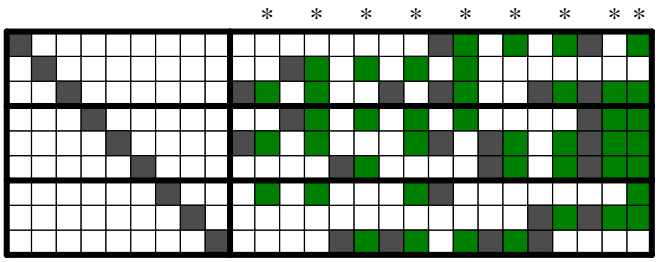


Fig. 5. A schedule that encodes using the generator matrix of Figure 4(b) with fewer than $k - 1$ XOR's per encoded word.

We ran the "X-Sets" scheduling program from [12], using the "Uber-XSet" selection parameter with a threshold of zero, and $L = 2$. which yields the schedule depicted in Figure 5. This schedule has 26 columns, which means that it requires 17 XOR operations, or 1.89 XOR's per encoded word. This is less than $k - 1$ XOR's, meaning that the lower bound for encoding MDS codes in general is *not* $k - 1$ XOR's.

We attained this result by enumerating all potential X and Y sets for Cauchy Reed-Solomon coding, and then applying the

X-Sets heuristic to the results. Of the $\binom{8}{3}\binom{5}{3} = 560$ potential sets that we tested, 18 of them could be scheduled with 17 XOR's. We cannot claim that these are the only ones that can be scheduled with 17 XOR's, because our scheduler is based on heuristics. Regardless, these are the smallest values of k, m and w for which the number of XOR's required is less than $k-1$ per bit.

VI. Enumeration of Small Codes when $w \in \{3, 4\}$

To explore this phenomenon for small matrices, we considered all values of $n = k+m \leq 8$ and $w \in \{3, 4\}$. For each set of parameters, we enumerated all X and Y sets for Cauchy Reed-Solomon coding and then applied the X-Sets heuristic to the results to derive the schedules with the fewest XOR's. We tabulate the results in Tables I and II. In each table, we plot the total number of XOR's in the schedule, and compare it against $(k-1)mw$, which is the number of XOR's that corresponds to $k-1$ XOR's per word. We plot the average XOR's per word, plus the number of matrices tested and found.

k	m	$(k-1)mw$	XOR's	Diff	Avg	Tested	Found
3	2	12	13	1	2.17	560	22
4	2	18	19	1	3.17	420	25
5	2	24	25	1	4.17	168	11
6	2	30	32	2	5.33	28	8
3	3	18	17	-1	1.89	560	14
4	3	27	25	-2	2.78	280	4
5	3	36	33	-3	3.67	56	2
3	4	24	22	-2	1.83	280	18
4	4	36	32	-4	2.67	70	7
3	5	30	26	-4	1.73	56	16

TABLE I
THE BEST SCHEDULES FOR CAUCHY REED-SOLOMON CODING MATRICES FOR $n = k + m \leq 8$ WHEN $w = 3$.

We omit values when $k \leq 2$ and $m = 1$, since those have a provable lower bound of $k-1$ XOR's per word. It is significant that for each test where $m > 2$, the number of XOR's in the best schedule is less than $k-1$ XOR's per word, the previously assumed lower bound. Additionally, the data suggests that as k and m increase, the distance from $k-1$ also increases. When $m = 2$, we conjecture that $(k-1)$ XOR's per word is a lower bound, and the data does not refute the conjecture.

When $k = 4$, $m = 3$ and $w = 4$, the schedules discovered encode with fewer XOR's than generalized RDP for the same set of parameters [1]. This is true for $k = 4$, $m = 4$ and $w = 4$ as well. This is notable, since generalized RDP is the best known code for these parameters. We show the P portion of this latter generator matrix, plus a schedule that encodes it with 44 XOR's in Figure 6.

VII. Of Matrix Density and XOR's

When one encodes without scheduling, the density of the generator matrix dictates the number of XOR's. For this reason, numerous works have sought to design codes with low density [4], [13], [15]. Moreover, some lower bounds have been derived on the density of MDS generator matrices.

k	m	$(k-1)mw$	XOR's	Diff	Avg	Tested	Found
3	2	16	16	0	2.00	43,680	16
4	2	24	24	0	3.00	120,120	34
5	2	32	33	1	4.12	240,240	8
6	2	40	42	2	5.25	360,360	49
3	3	24	23	-1	1.92	53,936	4
4	3	36	35	-1	2.92	400,400	6
5	3	48	47	-1	3.92	720,720	8
3	4	32	31	-1	1.94	400,400	88
4	4	48	44	-4	2.75	900,900	2
3	5	40	36	-4	1.80	720,720	4

TABLE II
THE BEST SCHEDULES FOR CAUCHY REED-SOLOMON CODING MATRICES FOR $n = k + m \leq 8$ WHEN $w = 4$.

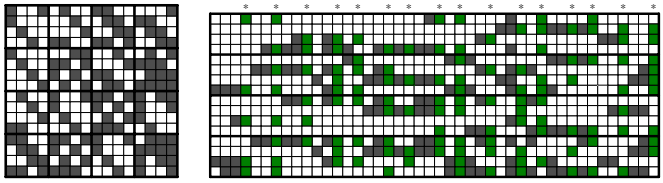


Fig. 6. The P portion for a generator matrix for $k = m = w = 4$, and a schedule that encodes it in 44 XOR's

The most specific is that the P portion of any MDS RAID-6 generator matrix must have at least $kmw + (k-1)(m-1)$ non-zero entries [4]. The generator matrix for any MDS code where $m > 2$ must contain multiple RAID-6 codes as submatrices; therefore the lower bound for RAID-6 applies to all codes where $m \geq 2$.

Unfortunately, when scheduling is applied, the density of the matrix does not have a direct effect on the number of XOR's. For example, the P portion of RDP's generator matrix has $k(3w-1)$ non-zero entries, yet it encodes with exactly $k-1$ XOR's per word. This is superior to, for example, the Blaum-Roth code, which attains the lower bound on density, yet encodes with $(k-1) + \frac{k-1}{2w}$ XOR's per word.

To gain a feeling for the density of the matrices that encode with fewer than $(k-1)$ XOR's per word, in Table III we plot the density of the sparsest and densist matrices that generate the best schedules for each value of k and m greater than two in our tests. We plot the densities normalized by kmw, so that the densities are independent of matrix size. Additionally, for each value of k, m and w we plot the lower bound for matrix density $(kmw + (k-1)(m-1))$ [4], and for $w = 4$, we plot the density of the generalized RDP matrix $(kw + k(m-1)(2w-1))$ [1].

In all cases, the density of the matrix generating the best schedule is significantly higher than the lower bound. In all cases but one, it is also higher than the generalized RDP matrix.

VIII. Incompleteness

Even in the very small parameter space that we have explored, we acknowledge that our work is quite incomplete. First, we have only explored MDS matrices that result from Cauchy Reed-Solomon constructions [5]. Even within the

		$w=3$			$w=4$			
k	m	Min	Max	LB	Min	Max	LB	RDP
3	3	1.59	1.59	1.15	1.72	1.72	1.11	1.44
4	3	1.61	1.64	1.17	1.65	1.75	1.12	1.44
5	3	1.60	1.82	1.18	1.67	1.87	1.13	1.44
3	4	1.61	1.75	1.17	1.44	2.08	1.12	1.50
4	4	1.62	1.67	1.19	1.88	1.88	1.14	1.50
3	5	1.60	1.82	1.18	2.03	2.03	1.13	1.53

TABLE III
THE NORMALIZED MINIMUM AND MAXIMUM DENSITIES OF THE MATRICES THAT GENERATE THE BEST SCHEDULES IN TABLES I AND II.

realm of MDS generator matrices for $GF(2^w)$, there are many more constructions. For example, the **jerasure** open-source erasure coding library implements a heuristic that massages the rows and columns of the generator matrix in an attempt to minimize matrix density [13]. None of these constructions have been considered. Additionally, there are numerous MDS generator matrices that do not correspond to Reed-Solomon codes. Examples are the generator matrices that correspond to EVENODD [2], RDP [7], Liberation [13] and Blaum-Roth [4] codes for RAID-6, and STAR [11], generalized EVENODD [3] and generalized RDP [1] for higher values of m. Clearly the space of MDS codes to explore is huge; however for smaller values of k, m and w, there may be insight to be gained by considering all possible MDS codes.

Similarly, we have only addressed scheduling experimentally, using heuristics that attempt to optimize the number of XOR operations for a given generator matrix [9], [10], [14]. The XOR numbers presented in Tables I and II may well not be the minima for each value of k, m and w. As with MDS matrix generation, when k, m and w are sufficiently small, determining optimality is computationally feasible [17] and may be illustrative.

IX. CONCLUSION

We have unearthed a counter-intuitive result concerning MDS erasure codes for storage — that the lower bound on XOR operations to encode an (n, k) MDS erasure code is in some circumstances less than $(k-1)$ XOR operations per encoded word. In Tables I and II, we have identified 12 sets of $[k, m, w]$ parameters where we can encode with fewer than $k-1$ XOR operations per word. This is smaller than any known erasure code.

The nature of this work is more experimental than theoretical, but we believe the results bear theoretical scrutiny. In particular, it raises the following open questions:

- What is the lower bound on XOR's for an MDS code?
- Can we design codes that achieve it?
- What is the impact on decoding?

It is our hope that the community of Information Theory researchers will be spurred to attack these questions. We will be happy to provide any of the results, codes and/or schedules produced in this paper to those who contact us.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grants CNS-0615221 and CSR-1016636, and a research award from Google. The author would like to thank Jim Hafner, Mario Blaum, Cheng Huang, and Jin Li for valuable discussions on this topic.

REFERENCES

[1] BLAUM, M. A family of MDS array codes with minimal number of encoding operations. In *IEEE International Symposium on Information Theory* (Seattle, September 2006).

[2] BLAUM, M., BRADY, J., BRUCK, J., AND MENON, J. EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures. *IEEE Transactions on Computing 44*, 2 (February 1995), 192–202.

[3] BLAUM, M., BRUCK, J., AND VARDY, A. MDS array codes with independent parity symbols. *IEEE Transactions on Information Theory 42*, 2 (February 1996), 529– 542.

[4] BLAUM, M., AND ROTH, R. M. On lowest density MDS codes. *IEEE Transactions on Information Theory 45*, 1 (January 1999), 46–59.

[5] BLOMER, J., KALFANE, M., KARPINSKI, M., KARP, R., LUBY, M., AND ZUCKERMAN, D. An XOR-based erasure-resilient coding scheme. Tech. Rep. TR-95-048, International Computer Science Institute, August 1995.

[6] CHEN, P. M., LEE, E. K., GIBSON, G. A., KATZ, R. H., AND PATTERSON, D. A. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys 26*, 2 (June 1994), 145–185.

[7] CORBETT, P., ENGLISH, B., GOEL, A., GRCANAC, T., KLEIMAN, S., LEONG, J., AND SANKAR, S. Row diagonal parity for double disk failure correction. In *3rd Usenix Conference on File and Storage Technologies* (San Francisco, CA, March 2004).

[8] ELERATH, J. G., AND PECHT, M. Enhanced reliability modeling of RAID storage systems. In *DSN 2007: International Conference on Dependable Systems and Networks* (Edinburgh, UK, 2007), IEEE.

[9] HAFNER, J. L., DEENADHAYALAN, V., RAO, K. K., AND TOMLIN, A. Matrix methods for lost data reconstruction in erasure codes. In *FAST-2005: 4th Usenix Conference on File and Storage Technologies* (San Francisco, December 2005), pp. 183–196.

[10] HUANG, C., LI, J., AND CHEN, M. On optimizing XOR-based codes for fault-tolerant storage applications. In *ITW'07, Information Theory Workshop* (Tahoe City, CA, September 2007), IEEE, pp. 218–223.

[11] HUANG, C., AND XU, L. Decoding STAR code for tolerating simultaneous disk failure and silent errors. In *DSN-2010: The International Conference on Dependable Systems and Networks* (Chicago, June 2010), IEEE.

[12] PLANK, J. S. Uber-CSHR and X-Sets: C++ programs for optimizing matrix-based erasure codes for fault-tolerant storage systems. Tech. Rep. CS-10-665, University of Tennessee, December 2010.

[13] PLANK, J. S., LUO, J., SCHUMAN, C. D., XU, L., AND WILCOX-O'HEARN, Z. A performance evaluation and examination of open-source erasure coding libraries for storage. In *FAST-2009: 7th Usenix Conference on File and Storage Technologies* (February 2009), pp. 253–265.

[14] PLANK, J. S., SCHUMAN, C. D., AND ROBISON, B. D. Heuristics for optimizing matrix-based erasure codes for fault-tolerant storage systems. Tech. Rep. CS-10-664, University of Tennessee, December 2010.

[15] PLANK, J. S., AND XU, L. Optimizing Cauchy Reed-Solomon codes for fault-tolerant network storage applications. In *NCA-06: 5th IEEE International Symposium on Network Computing Applications* (Cambridge, MA, July 2006).

[16] RESCH, J. K., AND PLANK, J. S. AONT-RS: blending security and performance in dispersed storage systems. In *FAST-2011: 9th Usenix Conference on File and Storage Technologies* (February 2011).

[17] SCHUMAN, C. D. An exploration of optimization algorithms and heuristics for the creation of encoding and decoding schedules in erasure coding. *Pursuit - The Journal of Undergraduate Research at the University of Tennessee 2* (2010).

[18] WANG, G., LIU, X., LIN, S., XIE, G., AND LIU, J. Generalizing RDP codes using the combintorial method. In *NCA-08: 7th IEEE International Symposium on Network Computing Applications* (Cambridge, MA, 2008).