Note: Correction to the 1997 Tutorial on Reed-Solomon Coding

James S. Plank   Ying Ding

University of Tennessee

Knoxville, TN 37996

[plank,ying]@cs.utk.edu

Technical Report UT-CS-03-504

Department of Computer Science

University of Tennessee

April 24, 2003

1 Introduction

In 1997, SPE published a tutorial by Plank [19] on implementing Reed-Solomon codes for erasure correction in redundant data storage systems. The motivation of this tutorial was to present these codes, which are typically described mathematically by coding theorists, in a way accessible to the programmers who need to implement them. The tutorial as published presented an information dispersal matrix A, which does not have the properties claimed – that the deletion of any m rows results in an invertable $n \times n$ matrix. The purpose of this note is to present a correct information dispersal matrix that has the desired properties, and to put the work in current context.

2 The Continued Need For Erasure Correcting Codes

As disk array technology continued to blossom in the 1990's [4, 5], a need arose to tolerate a disk's failure without waiting for the disk to be repaired. Straight replication performs this fault-tolerance, but at a high storage overhead. RAID Level 5 encoding, termed "N+1 Parity" [5], reduces the storage overhead for fault-tolerance, and allows a *parity disk* to store redundancy for n data disks in such a way that the failure of any single disk may be tolerated. However, as the number of disks in a disk array grows, so does the the need to tolerate multiple simultaneous failures. Reed-Solomon coding has the properties necessary to add arbitrary levels of fault-tolerance to disk array systems. One may add m coding disks to n data disks so that the failure of *any* m disks may be tolerated, and although none of the levels of RAID employs Reed-Solomon codes, the original work on disk arrays make note of the codes' desirable properties [5].

As wide-area network computing has become more popular, the uses of erasure-correcting codes have broadened. Rizzo employs them to avoid retransmission in point-to-point [20] and multicast [21] communication protocols. This work has resulted in standardization efforts for such codes in multicast scenarios from the IETF [15, 16]. Additional uses of Reed-Solomon codes have been in cryptography [8], distributed data structures [10], energy-efficient wireless communication [7] and distributed checkpointing [18].

The advent of wide-area and peer-to-peer storage systems has further motivated the need for erasure-correcting codes. For example, OceanStore employs Reed-Solomon coding for RAID-like fault-tolerance in a wide-area file system [9]. More interestingly, several content dispersal systems have noted that erasure coding can be used for caching rather than for fault-tolerance [2, 3, 22]. Specifically, suppose that n blocks of a file need to be stored in a wide-area storage substrate, so that clients in all parts of the network may access it. With replication, clients must find the closest copies of each of the n blocks in order to retrieve the file. However, with erasure-correcting codes, m extra coding blocks may be distributed with the n blocks of the file so that each client need only retrieve the n *closest* blocks in order to reconstruct the file. As files grow in size, the power of this application will be immense; hence the need to correct the error of the 1997 Reed-Solomon coding tutorial.

There are other erasure coding techniques in addition to the one which this tutorial addresses. Examples are Tornado codes [13, 14], Cauchy Reed-Solomon codes [1] and other parity-based schemes [6]. Of these, Tornado codes are worth special mention, as they form the backbone of the Digital Fountain content dispersal system [2]. Tornado codes have a randomized structure so that with the addition of m extra parity blocks, a file may be reconstructed from any $n + \epsilon$ blocks. The randomized structure ensures that ϵ should be small. In a performance evaluation conducted by Luby [12], Tornado codes display significantly better encoding and decoding performance than the codes in this paper (termed "Vandermonde-based Reed-Solomon codes") for large data sizes and large values of m. For small values of m, a true comparison has yet to be performed. Currently, there is no implementation guide for Tornado codes akin to [19]. As the knowledge of their performance advantages become more widespread, perhaps this will change.

3 A Correct Information Dispersal Matrix B

The desired properties for the information dispersal matrix for Reed-Solomon coding is that:

- It is an $(n + m) \times n$ matrix.
- The $n \times n$ matrix in the first n rows are the identity matrix.
- Any submatrix formed by deleting m rows of the matrix is invertible.

We denote the correct information matrix B. B is derived from an $(n+m) \times n$ Vandermonde matrix using a sequence of elementary matrix transformations:

1. Any column C_i may be swapped with column C_j.

2. Any column C_i may be replaced by $C_i * c$, where $c \neq 0$.

3. Any column C_i may be replaced by adding a multiple of another column to it: $C_i = C_i + c * C_j$, where $j \neq i$ and $c \neq 0$. Since arithmetic is over a Galois field, the addition operation is bitwise exclusive-or.

The i, j-th element of a Vandermonde matrix is defined to be i^j:

$$\begin{bmatrix} 0^0(=1) & 0^1(=0) & 0^2(=0) & \ldots & 0^{n-1}(=0) \\ 1^0 & 1^1 & 1^2 & \ldots & 1^{n-1} \\ 2^0 & 2^1 & 2^2 & \ldots & 2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ (n+m-1)^0 & (n+m-1)^1 & (n+m-1)^2 & \ldots & (n+m-1)^{n-1} \end{bmatrix}$$

By definition, this matrix has the property that any submatrix formed by deleting m rows of this matrix is invertible [17]. Moreover, any matrix derived from this matrix by a sequence of elementary matrix transformations maintains this property (since elementary matrix operations do not change the rank of a matrix [11]). Therefore, constructing the matrix B is a simple matter of performing elementary transformations on the Vandermonde matrix until the first n rows are the identity matrix.

The algorithm for doing constructing B is as follows:

- Suppose the first $i-1$ rows of the matrix are identity rows, and $i < n$. At each step, we will turn row i into an identity row, without altering the other identity rows. If the i-th element of row i is equal to zero, find a column j such that $j > i$ and the j-th element of row i is non-zero, and swap columns i and j. Such a column is guaranteed to exist; otherwise the first n rows of the matrix would not compose an invertible matrix. Moreover, since $j > i$, swapping columns i and j will not alter the first $i-1$ rows of the matrix.

- Let $f_{i,i}$ be the value of the i-th element of row i. Let $f_{i,i}^{-1}$ be the multiplicative inverse $f_{i,i}$. In other words, $f_{i,i} * f_{i,i}^{-1} = 1$. Since $f_{i,i} \neq 0$, $f_{i,i}^{-1}$ is guaranteed to exist. if $f_{i,i} \neq 1$, replace column C_i with $f^{-1}i, i * C_i$.

- Now $f_{i,i}$ = 1. For all columns $j \neq i$ and $f_{i,j} \neq 0$, replace column C_j with $C_j - f_{i,j} C_i$, where $f_{i,j}$ is the j-th element in row i. At the end of this step, rows 0 through i are identity rows, and the matrix still has the property that the deletion of any m rows yields an invertible matrix.

- Repeat this process until the first n rows are identity rows, and the construction of B is complete.

Example

As an example, we construct B for $n = 3, m = 3$, over GF(2^4). As detailed in [19], in GF(2^4), addition is performed by exclusive-or, and multiplication/division may be performed using logarithm tables, reproduced in Table 1.

The 6×3 Vandermonde matrix over $GF(2^4)$ is:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
`gflog[i]`	—	0	1	4	2	8	5	10	3	14	9	7	6	13	11	12
`gfilog[i]`	1	2	4	8	3	6	12	11	5	10	7	14	15	13	9	—

Table 1: Logarithm tables for $GF(2^4)$

$$\begin{bmatrix} 0^0 & 0^1 & 0^2 \\ 1^0 & 1^1 & 1^2 \\ 2^0 & 2^1 & 2^2 \\ 3^0 & 3^1 & 3^2 \\ 4^0 & 4^1 & 4^2 \\ 5^0 & 5^1 & 5^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 5 \\ 1 & 4 & 3 \\ 1 & 5 & 2 \end{bmatrix}$$

Row 0 is already an identity row. To convert row 1, we note that $f_{1,0} = f_{1,1} = f_{1,2} = 1$, so we need to replace C_0 with $(C_0 - C_1)$ and C_2 with $(C_2 - C_1)$. The resulting matrix is:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 2 & 6 \\ 2 & 3 & 6 \\ 5 & 4 & 7 \\ 4 & 5 & 7 \end{bmatrix}$$

All that is left is to convert row 2. First, since $f_{2,2} \neq 1$, we need to replace C_2 with $6^{-1}C_2$ = $7C_2$:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 2 & 1 \\ 2 & 3 & 1 \\ 5 & 4 & 6 \\ 4 & 5 & 6 \end{bmatrix}$$

Then we replace C_0 with $(C_0 - 3C_2)$ and C_1 with $(C_1 - 2C_2)$ to yield our desired B:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 15 & 8 & 6 \\ 14 & 9 & 6 \end{bmatrix}$$

References

[1] J. Blomer, M. Kalfane, M. Karpinski, R. Karp, M. Luby, and D. Zuckerman. An XOR-based erasure-resilient coding scheme. Technical Report TR-95-048, International Computer Science Institute, August 1995.

[2] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *ACM SIGCOMM '98*, pages 56–67, Vancouver, August 1998.

[3] J. W. Byers, M. Luby, and M. Mitzenmacher. Accessing multiple mirror sites in parallel: Using tornado codes to speed up downloads. In *IEEE INFOCOM*, pages 275–283, New York, NY, March 1999.

[4] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, June 1994.

[5] G. A. Gibson. *Redundant Disk Arrays: Reliable, Parallel Secondary Storage*. The MIT Press, Cambridge, Massachusetts, 1992.

[6] G. A. Gibson, L. Hellerstein, R. M. Karp, R. H. Katz, and D. A. Patterson. Failure correction techniques for large disk arrays. In *Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 123–132, Boston, MA, April 1989.

[7] P. J. M. Havinga. Energy efficiency of error correction on wireless systems, 1999.

[8] C. S. Jutla. Encryption modes with almost free message integrity. *Lecture Notes in Computer Science*, 2045, 2001.

[9] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS*. ACM, November 2000.

[10] W. Litwin and T. Schwarz. Lh*rs: a high-availability scalable distributed data structure using Reed Solomon codes. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 237–248. ACM Press, 2000.

[11] F. Lowenthal. *Linear Algebra with Linear Differential Equations*. John Wiley & Sons, Inc, New York, 1975.

[12] M. Luby. Benchmark comparisons of erasure codes. `http://www.icsi.berkeley.edu/~luby/erasure.html`, 2002.

[13] M. Luby, M. Mitzenmacher, and A. Shokrollahi. Analysis of random processes via and-or tree evaluation. In *9th Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1998.

[14] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann. Practical loss-resilient codes. In *29th Annual ACM Symposium on Theory of Computing,*, pages 150–159, 1997.

[15] M. Luby, L. Vicisano, J. Gemmell, L. Rizo, M. Handley, and J. Crowcroft. Forward error correction (FEC) building block. IETF RFC 3452 (`http://www.ietf.org/rfc/rfc3452.txt`), December 2002.

[16] M. Luby, L. Vicisano, J. Gemmell, L. Rizo, M. Handley, and J. Crowcroft. The use of forward error correction(FEC) in reliable multicast. IETF RFC 3453 (`http://www.ietf.org/rfc/rfc3453.txt`), December 2002.

[17] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes, Part I*. North-Holland Publishing Company, Amsterdam, New York, Oxford, 1977.

[18] J. S. Plank. Improving the performance of coordinated checkpointers on networks of workstations using RAID techniques. In *15th Symposium on Reliable Distributed Systems*, pages 76–85, October 1996.

[19] 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.

[20] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM SIGCOMM Computer Communication Review*, 27(2):24–36, 1997.

[21] L. Rizzo and L. Vicisano. RMDP: an FEC-based reliable multicast protocol for wireless environments. *Mobile Computer and Communication Review*, 2(2), April 1998.

[22] A. I. T. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Symposium on Operating Systems Principles*, pages 188–201, 2001.