# OPTIMAL CODING SCHEME FOR CLOUD STORAGE SYSTEM

Mr. Yuvraj P. Rasal[1]

[1]*PG Student, Department of Computer Engineering, SND College of Engineering and Research Centre, Yeola, Nashik, Maharashtra, India*

## ABSTRACT

*Cloud storage generally provides different redundancy configuration to users in order to maintain the desired balance between performance and fault tolerance. Data availability is critical in distributed storage systems, especially when node failures are prevalent in real life. Our study finds that with very low probability, one coding scheme chosen by rules of thumb, for a given redundancy configuration performs best. In this paper, we propose CaCo, an efficient Cauchy coding approach for data storage in the cloud. First, CaCo uses Cauchy matrix heuristics to produce a matrix set. Second, for each matrix in this set, CaCo uses XOR schedule heuristics to generate a series of schedules. Finally, CaCo selects the shortest one from all the produced schedules. Furthermore, CaCo outperforms and auditing technique to detect any fraud done on data based on hashing and signature techniques. We implement CaCo in the cloud environment and tested that it provides good performance in encoding time.*

**Keyword :** - *Cloud storage, Cauchy matrix, Fault Tolerance, Reed-Solomon Codes, XOR Scheduling, hashing technique, Auditing.*

## 1. INTRODUCTION

One of the biggest challenges in designing cloud storage systems is providing the reliability and availability that users expect. Once their data is stored, users expect it to be persistent forever, and perpetually available. Unfortunately, in practice there are a number of problems that, if not dealt with, can cause data loss in storage systems. So, the failure protection offered by the standard RAID levels has been no longer sufficient in many cases, and storage designers are considering how to tolerate larger numbers of failures [1] Technology shifts and market forces are changing the composition and design of storage systems. Topics for this diverse issue include the emergence of nonvolatile storage technologies, virtualization technologies that reduce the distinction between storage and computing platforms, advances in tape densities, the growing use of commodity and distributed storage, and the increasing importance of error and disaster recovery, autonomic storage management, pet scale file and archival storage, and long-term data preservation.[2] Cloud services inevitably fail: machines lose power, networks become disconnected, pesky software bugs cause sporadic crashes, and so on. Unfortunately, failure recovery itself is often faulty; e.g. recovery can accidentally recursively replicate small failures to other machines until the entire cloud service fails in a catastrophic outage, amplifying a small cold into a contagious deadly plague. Cauchy Reed-Solomon (CRS) codes improve Reed-Solomon codes by using neat projection to convert Galois Field multiplications into XOR operations [3]. Currently, CRS codes represent the best performing general purpose erasure codes for storage systems. In addition, CRS coding operates on entire strips across multiple storage devices instead of operating on single words. In particular, strips are partitioned into w packets, and these packets may be large. Figure 1 illustrates a typical architecture for a cloud storage system with data coding. The redundancy configuration of the system is k = 4 and m = 2. With CRS codes, k data blocks are encoded into m coding blocks. In such a way, the system can tolerate any m disk failures without data loss. Note that those k data blocks and m coding blocks should be stored on different data nodes. Otherwise, the failure of one node may lead to multiple faults in the same group of n = k +m blocks.

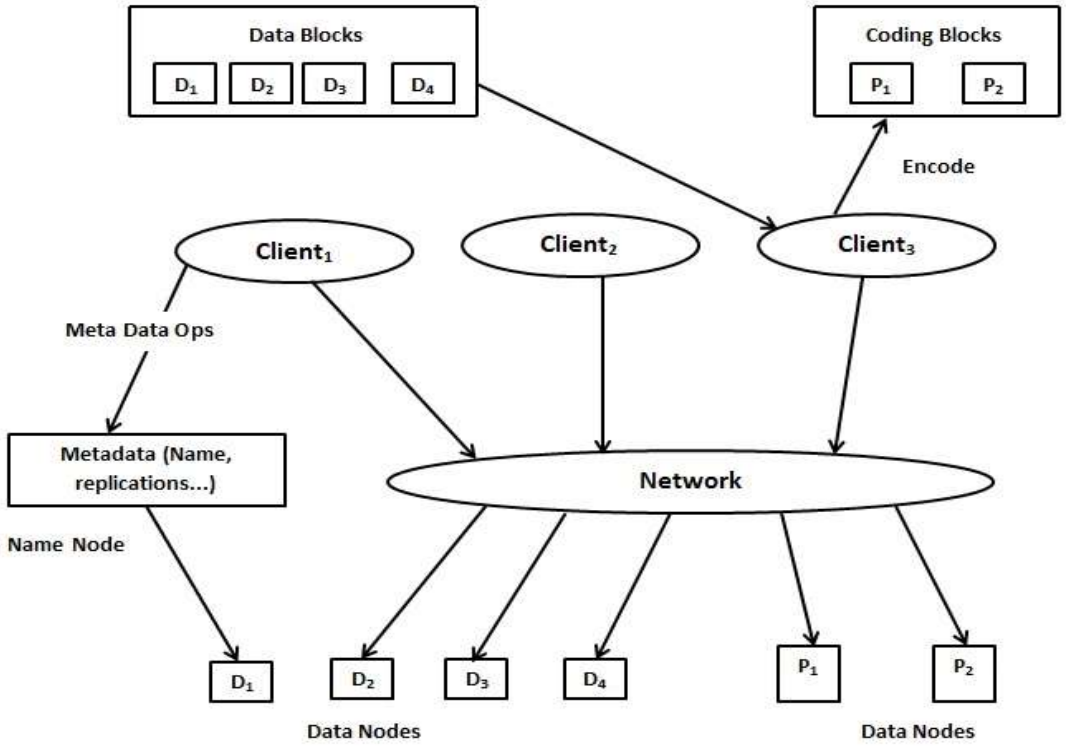



**Fig-1.1:** A Distributed Architecture for a Cloud Storage System with Data Coding.

## 2. LITERATURE REVIEW

In this section, we first give a general overview of Cauchy Reed- Solomon (CRS) Coding. Then, we provide a description in brief of the research and related work on generating Cauchy matrices and encoding with schedules. From these descriptions, we can make clearer the motivation of our work.

### 2.1 Cauchy Reed-Solomon Coding Reed-Solomon (RS)

Cauchy Reed-Solomon Coding Reed-Solomon (RS) codes [4] are based on a finite field, often called Galois field. When encoding data using RS codes, to implement a Galois filed arithmetic operation (addition or multiplication) requires many computations, so the performance is often unsatisfactory. CRS [3] codes modify RS codes and give two improvements. First, CRS codes use a Cauchy matrix instead of a Vandermonde matrix [5]. Second, CRS codes convert Galois field multiplications into XOR operations. The key to CRS codes is construction of Cauchy matrices, and we can achieve that in the following way. Given a redundancy configuration (k; m;w) where k+m ≤ 2w, let X = {x1; : : : ; xm}, Y = {y1; : : : ; yk}, and X ∩ Y = _, so that each xi and yj is a distinct element of GF(2w). Then we calculate the Cauchy matrix in element (i; j) using 1=(xi + yj) (the addition and division are defined over Galois field) [3]. Since the elements of GF(2w) are the integers from zero to 2w − 1, each element e can be represented by a w-bit column vector, V (e), using the primitive polynomial over Galois Field. Furthermore, each element e of GF(2w) can be converted to a (w×w) binary matrix, M(e), whose i-th(i = 1; : : : ;w) column is equal to the column vector V (e2i−1) [6]. Thus according to the value of w, we can transform the Cauchy matrix into a (mw × kw) binary matrix, denoted as A. We divide every data block X and erasure codes block B into w trips. In this way, when there exists "1" in every row of A, we can do XOR operations on the corresponding data in X, to obtain the elements of B. As Figure 2 shows [7], the erasure codes require 11 XOR operations.

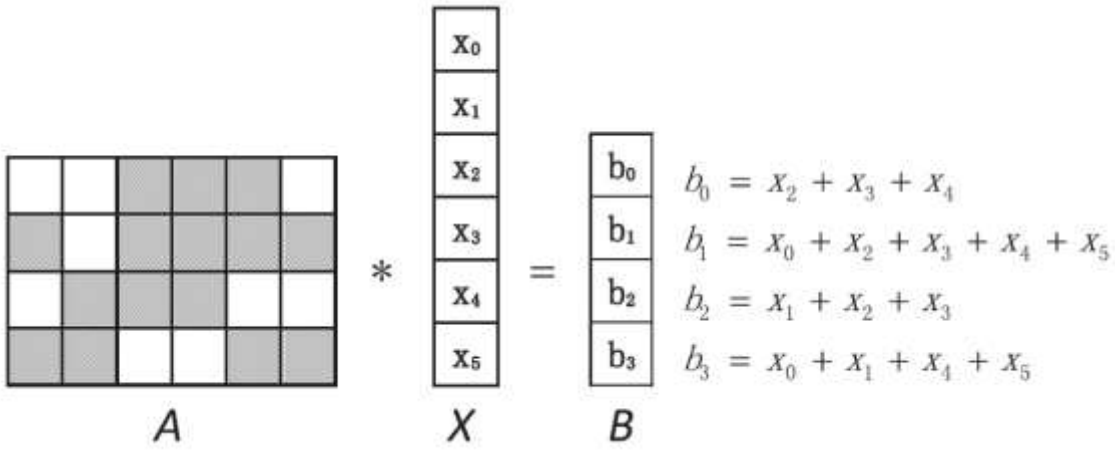



**Fig-2.2:** Erasure Coding with an Optimal Schedule

**2.2 Observations and Motivation of Our Work**

Cloud systems always use different redundancy configurations (i.e., (k; m; w)), depending on the desired balance between performance and fault tolerance. Through the preceding discussions and a number of experiments and analyses, we get some observations as follows.

• For different combinations of matrix and schedule, there is a large gap in the number of XOR operations.
• No one combination performs the best for all redundancy configurations.
• With the current state of the art, from the (2w k+m) (k+mk ) Cauchy matrices, there is no method discovered to determine which one can produce the best schedule.
• Giving a Cauchy matrix, different schedules generated by various heuristics lead to a great disparity on coding performance.
• For a given redundancy configuration, it is with very low probability that one coding scheme chosen by rules of thumb performs the best. In view of the problems above, it is necessary to discover an efficient coding approach for a cloud storage system. And this approach is desired to be able to identify the optimal coding scheme in the current state of the art, for an arbitrary given redundancy configuration.

## 3. PROBLEM STATEMENT

Given a redundancy configuration (k, m, w) our goal is to find a Cauchy matrix, whose schedule is desired to be the shortest. In this paper, we propose CaCo, a coding approach that incorporates all existing matrix and schedule heuristics, and therefore is able to discover an optimal solution for data coding in a cloud storage system, within the capability of the current state of the art.

## 4. PROPOSED SYSTEM

To provide user security for file transfer we requires proposed system. As in many roll based access system if the user have the access of the file then user can access the file any time but if the user found unauthorized then there is main challenge is revoking the access of that user .KIDS provide that facility of revoking the access of the user also signature concept for the particular file.

The figure shows the Architecture of Proposed System. The system consists of four basic modules which are listed and explain below in detail.

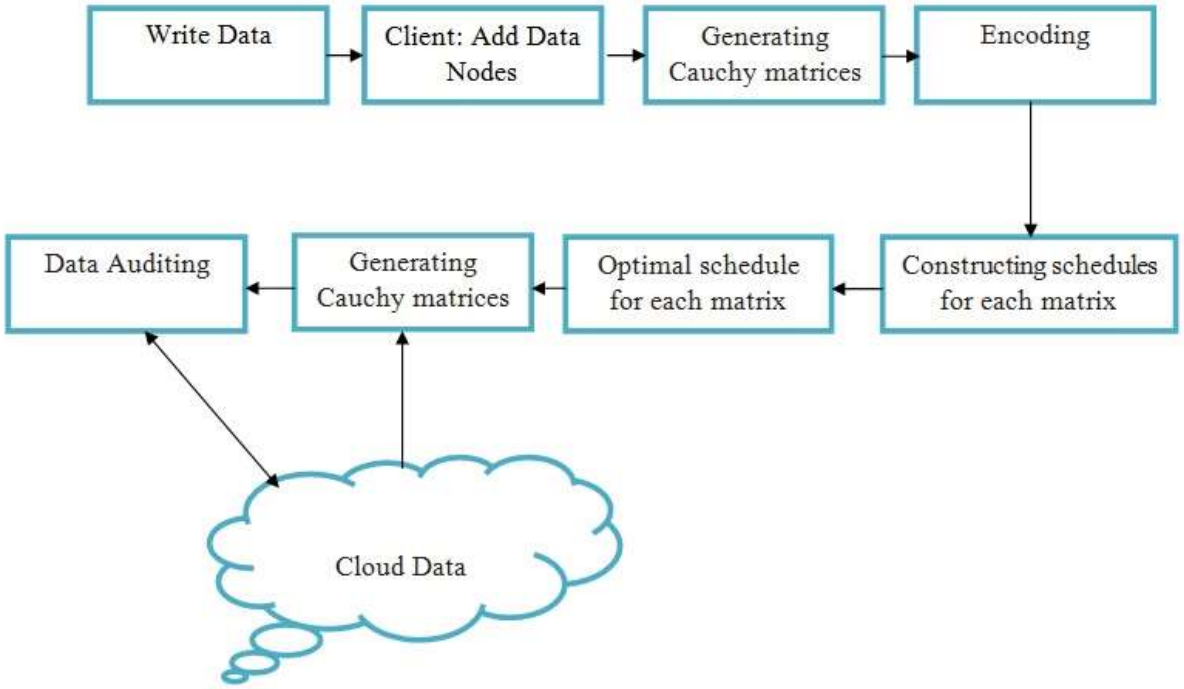



**Fig -4.1**: Architecture of Proposed System

**4.1 Generating Cauchy matrices.**

Selecting the best one from Cauchy matrices using the enumeration method is a combinatorial problem. Given a redundancy configuration (10; 6; 8), the magnitude of the matrices to be constructed can be up to 1029, and it is unrealistic to enumerate them. We cannot even determine which one of the matrices will produce better schedules. In the CaCo approach, we choose only a certain number of them for scheduling.

**4.2 Constructing schedules for each matrix.**

For each matrix mi($0 < i < p$) in the set Sm, we pass the parameters including k, m, w and pointer of the matrix to the function do schedule (int k, int m, int w, int * matrix) to perform q heuristics in the function, such as Uber-CSHR, X-Sets, and so on. In this manner, we get a set of schedules, denoted as $S_{si=\{}s_{0i}, s_{1i}, \ldots, sq\text{-}1i\}$. If there appears a good heuristic for scheduling at a later date, we can add it to the function do schedule.

**4.3 Selecting the locally optimal schedule for each matrix.**

For each matrix mi($0 < i < p$) in the set Sm, we select the shortest schedule from the set Ss;i, denoted as si, so that we get a set of matrices and their shortest schedules, denoted as S = {(m0, s0); (m1; s1); . . . ; (mp_1; sp_1)}. For mi in the set Sm, we can encode data in an order of XORs given by si. In this way, the times of XOR operations no longer have direct relationship with the density of the matrix. Therefore, scheduling excludes the influence of the lower limit of the number of ones in the matrix, so the performance improves significantly.

**4.4 Selecting the globally optimal solution**

From the collection of combinations of Cauchy matrix and schedule, namely {(m0; s0); (m1; s1); . . . ; (mp_1; sp_1)}, we choose the combinations with the shortest schedule. On this basis, for better performance, we tend to select the one containing the fewest ones in the matrix to be (mbest; sbest). Once selected, subset can be used for encoding data.

**4.5 Data Auditing**

Security monitoring on the cloud is important, because computers sharing data are most readily available to an attacker. Without mechanisms in place to detect attacks as they occur, an system may not realize its security. Therefore it is vitally important that computers residing in the cloud are carefully monitored for a wide range of audit events. The auditing in a system consists of three steps. The first step is the attack has attempted on any node in system, secondly the attack is detected by the system by hashing algorithm after detection of attack the

notifications are send to data owner. Due to this security is improved.

## 5. ALGORITHM USED FOR IMPLEMENTATION

**5.1 Algorithm 1: Write Operation with CaCo**

1 The Client sends a write request to the NameNode.
2 The NameNode allocates some DataNodes to the Client.
3 Write the data blocks into DataNodes.
4 Make a copy of data and put it into DataQueue.
5 Encode data with the schedule selected by CaCo.
6 Write the coding blocks into DataNodes.
7 Data encoding finishes.
8 Remove the copies of data from DataQueue.

**5.2 Algorithm 2: Auditing Algorithm**

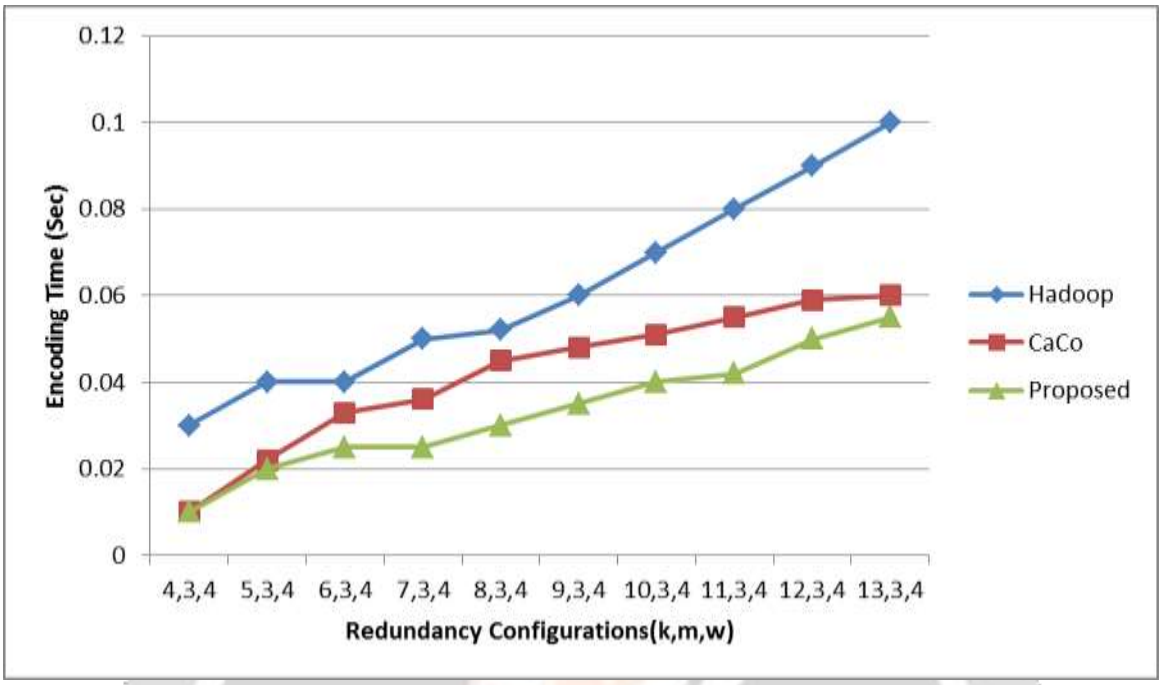
1. Start
2. Read user data owner id (udoid)
3. If (doid ≠ udoid)
4. Stop
5. Else Read file name from TPA xml
6. Retrieve No. of blokes for Auditing
7. Select the block number that user want to verify.
8. Get the auxiliary information for block from TPA xml
9. Based on Auxiliary information generate new root for Auditing
10. If (new root ≠ root) file modified
11. Else File not modified
12. Stop.

## 6. RESULT AND DISCUSSION

Table I shows an expected result of proposed system. We compare Hadoop, CaCo and Proposed result in the encoding time when recovering from failures of two data disks and one coding disk. First, the coding times of Hadoop, CaCo and Proposed result take on an upward trend as k increases along the x-axis.

**Table 6.1:** Execution Time Of A Redundancy Configurations (K,M,W).

| | **Hadoop** | **CaCo** | **Proposed** |
|---|---|---|---|
| 4,3,4 | 0.03 | 0.01 | 0.01 |
| 5,3,4 | 0.04 | 0.022 | 0.02 |
| 6,3,4 | 0.04 | 0.033 | 0.025 |
| 7,3,4 | 0.05 | 0.036 | 0.025 |
| 8,3,4 | 0.052 | 0.045 | 0.03 |
| 9,3,4 | 0.06 | 0.048 | 0.035 |
| 10,3,4 | 0.07 | 0.051 | 0.04 |
| 11,3,4 | 0.08 | 0.055 | 0.042 |
| 12,3,4 | 0.09 | 0.059 | 0.05 |
| 13,3,4 | 0.1 | 0.06 | 0.055 |




## 7. CONCLUSION

We propose CaCo, a new approach that incorporates all existing matrix and schedule heuristics, and thus is able to identify an optimal coding scheme within the capability of the current state of the art for a given redundancy configuration. The selection process of CaCo has an acceptable complexity and can be accelerated by parallel computing. It should also be noticed that the selection process is once for all. We had also provide security to data which is in form of auditing technique. We had observe that for different combinations of matrices and schedules, there is a large gap in the number of XOR operations, and no single combination performs best for all redundancy configurations. We had also tried to reduce the encoding and decoding complexity.

## 8. ACKNOWLEDGEMENT

With all respect and gratefulness, I would like to thanks all people who have helped me directly or indirectly for the paper presentation. I am grateful to my guide, Prof. I. R. Shaikh, for his guidance and support. I wish to express my sincere thanks to the, PG Coordinator Prof. V. N. Dhakane and HOD Prof I. R. Shaikh for their support. Lastly I would like to thank staff member of Department of Computer, SND COE & RC, Yeola, Nashik, India for making all the requirements possible and simply available whenever required.

## 6. REFERENCES

[1] L. N. Bairavasundaram, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, G. R. Goodson, and B. Schroeder, "An analysis of data corruption in the storage stack," Trans. Storage, vol. 4, pp. 8:1–8:28, Nov. 2008.
[2] J. L. Hafner, V. Deenadhayalan, W. Belluomini, and K. Rao, "Undetected disk errors in raid arrays," IBM journal of research and development, vol. 52, pp. 413–425, July 2008.
[3] J. Bloemer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman,"An xor-based erasure-resilient coding scheme," 1995.
[4] J. S. Plank, "A tutorial on reed-solomon coding for fault-tolerance in raid-like systems," Software Practice Experience , vol. 27, pp. 995–1012, Sept.1997.
[5] J. S. Plank and Y. Ding, "Note: Correction to the 1997 tutorial on reed& ndash;solomon coding," Software Practice Experiance, vol. 35, pp. 189–194,Feb. 2005.
[6] X. Li, Q. Zheng, H. Qian, D. Zheng, and J. Li, "Toward optimizing cauchy matrix for cauchy reed-solomon code," IEEE Communication Letters., vol. 13,pp. 603–605, Aug. 2009.
[7] Guangyan Zhang, Guiyong Wu, Shupeng Wang, Jiwu Shu, Weimin Zheng, and Keqin Li "CaCo: An Efficient Cauchy Coding Approach for Cloud Storage Systems" IEEE Transactions On Computers, Vol. 62, No. 11, November 2015

[8] J. S. Plank, "XOR's, lower bounds and MDS codes for storage," Technical Representative. CS-11-672, University of Tennessee, May 2011.
[9] J. S. Plank, "A tutorial on Reed-Solomon coding for faulttolerance in raid-like systems," Softw. Pract. Experience, vol. 27, pp. 995–1012, Sept. 1997.
[10] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," J. Soc. Ind. Appl. Math., vol. 8, no. 2, pp. 300–304, 1960. [11] J. Bl€omer, M. Kalfane, M. Karpinski, R. Karp, M. Luby, and D. Zuckerman, "An xor-based erasure-resilient coding scheme," International Computer Science Institute, Berkeley, California,USA, 1995.
[12] J. S. Plank, J. Luo, C. D. Schuman, L. Xu, and Z. Wilcox-O'Hearn, "A performance evaluation and examination of open-source erasure coding libraries for storage," in Proc. 7th Conf. File Storage Technol., Berkeley, CA, USA, 2009, pp. 253–265.
[13] M. Blaum, J. Brady, J. Bruck, and J. Menon, "Evenodd: An efficient scheme for tolerating double disk failures in raid architectures," IEEE Trans. Comput., vol. 44, no. 2, pp. 192–202, Feb. 1995.
[14] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-diagonal parity for double disk failure correction," in Proc. 3rd USENIX Conf. File Storage Technol., Berkeley, CA, USA, 2004, pp. 1–
[15] L. Xu and J. Bruck, "X-code: MDS array codes with optimal encoding," IEEE Trans. Inform. Theory, vol. 45, no. 1, pp. 272–276, Jan. 1999.
[16] C. Jin, H. Jiang, D. Feng, and L. Tian, "P-code: A new raid-6 code with optimal properties," in Proc. 23rd Int. Conf. Supercomput., New York, NY, USA, 2009, pp. 360–369.