Data Recovery Approach with Optimized Cauchy Coding in Distributed Storage System

Snehalata Funde*, Gandharba Swain
Department of Computer Science and Engineering
Koneru Lakshmaiah Education Foundation
Vaddeswaram-522302, Guntur, Andhra Pradesh, India

***Abstract*—In the professional world, the impact of big data is pulsating to change things. Data is currently generated by a wide range of sensors that are part of smart devices. It necessitates data storage and retrieval that is fault tolerant. Data loss can be caused by natural calamities, human error, or mechanical failure. Several security threats and data degradation attacks attempt to destroy storage disks, causing partial or complete data loss. The data encoding and data recovery mechanisms is proposed in this research. To produce a set of matrices utilizing matrix heuristics, the suggested system proposes an efficient Optimized Cauchy Coding (OCC) approach. In this paper, the Cauchy matrix is used as a generator matrix in Reed Solomon (RS) code to encode data blocks with fewer XOR operations. It reduces the encoding algorithm's time complexity. Furthermore, in the event of a disk failure, missing data from any data block is made available through the Code word. In terms of data recovery, it outperforms the Optimal Weakly Secure Minimum Storage Regenerating (OWSPM-MSR) and Product-Matrix Minimum Storage Regenerating (PM-MSR) methods. For data coding, a 1024KB file with various combinations of data blocks *l* and parity blocks m is evaluated. In the first scenario, *m* is 1 and *l* ranges from 4 to 10. The value of *l* is 4 in the second scenario, while *m* ranges from 1 to 10. The existing OWSPM-MSR approach takes an average of 0.125 seconds to encode and 0.22 seconds to decode, whereas the PM-MSR approach takes an average of 0.045 seconds to encode and 0.16 seconds to decode. The proposed OCC approach speeds up data coding by taking an average of 0.035 seconds to encode and 0.116 seconds to decode data.**

Keywords*—Optimized cauchy coding; fault tolerant; data availability; reed solomon code

I. Introduction

Nowadays, social media is the primary source of data generation that we refer to as "big data," and it generates tremendous amounts of data. It is both enormous in size and intricate in nature. As a result of the current pandemic crisis, the use of the internet and smart phones is increasing every day. As a result, numerous businesses generate a high volume of records in the Petabyte or Exabyte range. Traditional systems are limited in their ability to capture, store, and analyze such large amounts of data, according to Wang et al. [1]. Large datasets can now be stored, compiled, and evaluated because of advancements in processing and storage capabilities. Big data analytics offers novel approaches to analyzing massive datasets. Few applications are rigorously found in the domains of healthcare, finance, traffic management, education, and retail [2]. Furthermore, the increase in data necessitates attention to several difficulties such as privacy, integrity, and access control, all of which are required to protect data from various attacks such as data degradation attacks and man-in-the-middle attacks. The current research uses a Cauchy matrix generation method to build and create models that can be used to manage data recovery for dispersed datasets.

Yang et al. [3] describe a multi-cloud storage system that optimizes security and affordability. Chervyakov et al. [4] proposed a distributed data storage system for processing encrypted data and controlling calculations. Furthermore, the Redundant Residue Number System is used to discover and remedy errors. For distributed cloud storage, a block chain approach to security architecture is being investigated. Li et al. [5] used a genetic algorithm to solve file block duplication among numerous users and data centers. Bhuvaneshwari & Tharini [6] argue that scalable and dependable distributed storage systems are essential due to the growing volume of data. For massive data storage, Low Density Parity Check (LDPC) codes are being investigated. Tang & Zhang [7] demonstrate how a Vander-monde matrix is concatenated with an identity matrix to ease encoder and decoder design. Furthermore, in bit matrices of the Cauchy matrix, several finite fields are explored to decrease one. Erasure-resilient codes are Cauchy Reed-Solomon (RS) codes. Makovenko et al. [8] describe the use of heuristic perturbation to optimize coding bit-matrices for fault tolerant data storage. The ability to serve multiple clients at the same time is one of the most significant requirements of cloud storage solutions. The help rate region is a new addition to the map. Kazemi et al. [9] defines a Quality of Service (QoS) metric for coded, distributed setups. It's a collection of information access requests that the system manages all at once.

Dai & Boroomand [10] offered a combination of big data and man-made consciousness technologies to improve security by detecting attacks on internet-connected devices. Ravikumar & Kavitha [11] demonstrate how to use adaptive hybrid mutation to optimize a black widow clustering approach. To improve remote sensing retrieval mechanisms, Li & Zhang [12] propose widely open datasets with evaluation metrics. Xiao & Zhou [13] propose classifiers based on resource optimization and framework to minimize data repair and update concerns in erasure coding schemes. Gong & Sung [14] introduce zig-zag codes as erasure codes for data decoding. The dependability vs. storage trade-off is examined. Dau et al. [15] consider the Cauchy and Vander-monde

*Corresponding Author.

matrices while constructing maximum distance separable codes to tackle coding challenges. This method is used to select arbitrary Vander-monde/Cauchy matrix evaluation points. It necessitates the use of very small finite fields. The dimensions of the generator matrix are polynomial. Additional design limitations are implemented, rendering the Vander-monde/Cauchy matrix ineffective for code construction. Wu et al. [16] show how to use Cauchy RS codes to improve the efficiency of distributed storage systems. It starts by calculating the best post scaling encoder matrix based on the data migration policy. Following that, the data transfer process is optimized using the chosen post scaling encoder matrix. The Solomon code protocol for multi-party authentication is thought to improve the security of the system. When data is large, it can be difficult to construct Solomon code as matrix creation since it takes a long time. Another difficulty with several portions is space complexity. Information is securely stored in a wireless network using both local and geographic replication. It aids in the recovery after a disaster. Zhang et al. [17] discuss the architecture, as well as resource supply and replication technologies. It reduces the time it takes to deploy software and increases customer adoption. The disadvantage is that it necessitates a larger development pool.

Chen & Ma [18] present a disk recovery strategy that is both efficient and cost-effective. It decreases the amount of data that is read from the disk. Furthermore, the workload quantity on all surviving disks is the same. To recover the failed disk, only a few disk reads are needed. Recovery of all the data takes a long time. Shen et al. [19] demonstrate how to determine the optimal number of code word pictures for any XOR-based erasure code. It produces recapture rosters with the least amount of data possible. It also improves I/O performance for cloud document frameworks that use large square sizes. For the length of recuperation, it reduces the amount of data read. Its drawback is that it necessitates a significant amount of area. On database servers, it causes a heating problem. Burihabwa et al. [20] suggest using a flat XOR-code for data recovery. Recovery schedules are also generated. As a result, the execution time is minimized. Periodic system backup is not included in this system's configuration. Hadoop is a platform for storing and analyzing massive data sets. Hadoop Distributed File System (HDFS) is used to store the large amount of data. Partitioning a large record into blocks is considered a block position strategy. Following that, it was assigned to the homogenous cluster. Shah & Padole [21] offer a novel approach to updating the information blocks on explicit hubs. Total nodes are split into two categories: homogenous and heterogeneous, and high and low performing nodes. It improves node burden revision. Its drawback is that it requires a capacity approach that can work on both homogeneous and heterogeneous clusters. It requires apps that can run in a diverse and homogenous environment in a short amount of time.

Information is stored in a NoSQL database in a hub-like way. As a result, the information becomes skewed. These data sets should store information that is extremely accessible. It also has no reservations about the parcel's adaptability and durability. Pandey [22] introduces this strategy for balancing load in a circulating environment. It divides data into small parts, making it easier to move things around on their own. As erasure codes, Cauchy RS codes are gaining in use. It is preferred in Redundant Array of Independent Disks (RAID) arrays that use Solid-State Drives (SSDs). The use of Galois Field (GF) math to accomplish matrix-vector multiplication is necessary for such coding on a processor-based RAID regulator. The erasure coding performance of resistive RAM (ReRAM) can be improved. RAID controllers and SSD controllers use it as core memory. The Cauchy-Vandermonde matrix is used by Han et al. [23] as the encoding generating matrix. SSDs are used to distribute rebuilding efforts for a single failure. It boosts the speed of encoding and decoding. More memory accesses result from increased computation complexity. It has a high level of durability while requiring less storage space. To improve failure recovery performance, Xu et al. [24] introduce a PBD-based Data Layout (PDL). In the event of part failures, it clearly gives superior assistance to front-end applications. For mixed erasure codes, it provides a uniform data layout. Each lump containing different symbols from a coding strip resides in a different node and on a different disk in distributed storage. Pirahandeh & Kim [25] present a scalable, multisector scheduler that allows for three levels of adaptability to non-critical failure over time, plates, and nodes. It can now recover from multiple levels of failure. The innermost assembly of each node, which consists of total disks and segments for each node, is alarming. It raises encoder and decoder average performance. It has been determined that it is suitable for use in a distributed storage environment that is adaptive. To determine parity chunks, our approach uses fewer XOR tasks. It provides a consistent information layout for integrated Erase Code (EC).

The rest of the paper is written in the following format. The related work is discussed in Section II. The proposed Optimized Cauchy Coding (OCC) method is detailed in Section III. The evaluation findings are given in a real-time scenario in Section IV. The proposed OCC system benefits and potential scope are summarized in Section V.

II. Related Work

A. Related Work

This section discusses relevant work in the data storage domain for fault tolerant systems that addresses time, space, and complexity issues in matrix formation.

In light of the equipment execution system for decoding, Wu et al. [26] suggested a novel high-speed Cauchy algorithm to construct an encoding calculation. To achieve the same results as Cauchy, this technique requires a less sophisticated calculating process. Ca-Co, a useful Cauchy coding method for information storage in distributed systems, was proposed by Zhang et al. [27]. To produce a scheduling sequence, it uses a Cauchy matrix with XOR operations. In comparison to existing systems, the system provides an effective data storage approach. It can quickly generate a Ca-Co matrix using the XOR operation. Tang & Cai [28] offer a novel erasure decoding approach. To protect against capacity node disappointments, erasure coding is preferred in the appropriate stockpiling framework. The data is first separated into data blocks, which are then coded to create encoding blocks. The decoding method is used to recover the lost data blocks. In

light of the changing system of the translating change matrix, the decoding failures are acknowledged. Erasure coding improves interpreting effectiveness while requiring less bandwidth for modernization. The hypothetical examination, on the other hand, demonstrates the method's accuracy. The system of data restoration benefits from available erasure-coded data insertion strategies. To recover data, the operator first replaces all unproductive nodes with one new node. The Node Replacement Process (NRP) takes hours or days in practice. Due to the lack of durability, the enhanced statistics are lost once more. Information accessibility and dependability are maintained by a few methods in circulating record systems.

The Erasure Coding (EC) method is popular for increasing space efficiency. It has incalculable execution encoding, decoding, and input/yield corruption elements in general. Kim [29] proposes a buffering and joining method in which several I/O demands that occur during encoding are combined and treated as one. It offers four recovery options for distributing the disk input/yield loads generated during decoding. Erasure coding is a generally complex technology used in the allocated storage framework to protect against capacity node disk failure. It improves resilience while also reducing repetitiveness. First, the data is separated into b chunks. The encoding technique is then used to convert the b blocks into c blocks. The decoding technique recovers the $c - b$ blocks that were missed. Clients of distributed storage typically assign different redundancy configurations to the repetitiveness settings (b, c, and d) of EC. It is based on the perfect balance of execution and internal failure adaption. This work aids in the observation of a very low likelihood encoding approach for a design with available configurations that produces superior results. With low data repetition, EC codes are used in the cloud. Just a hypothetical investigation demonstrates the Ca-Co technique's accuracy.

The Optimal Weakly Secure Minimum Storage Regenerating (OWSPM-MSR) [31] approach was proposed by Bian et al. Its implementation is based on Jerasure, and the test was run on a Linux machine with an Intel Core i5 CPU and 4GB RAM. A Galois field size of 28 and a block size of 1024 are used in the implementation. They compared many factors, such as calculation time and storage overhead, with the previous system. This method use the Cauchy-network based RS coding algorithm. Over constrained fields, division and multiplication may be converted into subtraction and addition tasks, and they can be recognized by XOR tasks. Furthermore, they calculated the aforementioned during data recovery in terms of translating time. In the most pessimistic case, it takes a little longer than Product-Matrix Minimum Storage Regenerating (PM-MSR). The time between techniques became longer as the number of data blocks (k) and parity blocks (p) increased (m). Because it takes longer to invert a matrix, the time consumed by the OWSPM-MSR approach is longer than the time necessary to encode the data.

Shah et al. [32] proposed a technique which is based on a product matrix structure. PM-MSR uses the Vandermonde network to reconstruct information that has been lost due to a disc crash. Two special characteristics of the Product-Matrix structure make the codes used in this system desirable from a security standpoint. For starters, several standards in the literature, including those in, take functional repair into account. The data placed in the replacement node might vary from the data stored in the failed node in this form of repair as long as the replacement node meets the system's rebuilding and appropriate working requirements. This enables a snoop to get a greater amount of data by examining the information stored in a core over several instances of repair. PM codes, on the other hand, provide a precise fix, with information stored in the substitute node indistinguishable from that stored in the bombed node. Second, regardless of whether the fix is exact, the information downloaded during the repair of a given server may be dependent on the arrangement of n nodes cooperating in the maintenance with handling, and therefore may alter between fixes of that node. By default, the PM system ensures that the data retrieved by the substitute node is independent of the helper nodes' personalities. The Vandermonde-based has a high level of complexity in terms of duplication and division in a Galois field, resulting in a long calculation time.

The approach proposed by Wang et al. [30] involves replacing some submatrices of the polarizing change with networks on the decoder side, resulting in a less difficult evaluation of log-probability proportions. This strategy allows for a reduction in complexity for the sequential termination, list, and successive translation calculations. Later, it was suggested that an improved decoding technique for polar codes with large kernels be used. It entails replacing the instances of large kernels in the polarizing change with more straightforward frameworks in a specified way. The methodology does not require any sideways encoder enlargements and has no effect on decoder operation. It can also be used in conjunction with list and consecutive translation calculations. For efficient polar codes, many encoding computations are addressed. In comparison to previous non-recursive algorithms, an improved encoding approach reduces the number of calculations due to an iterative property of the generating matrix and a specific lower three-sided design of the lattice. When compared to current non-recursive approaches, it reduces the number of XOR processing units, which is advantageous for numerous executions over existing algorithms. Many limiting or figure units haven't been changed. This makes the framework less stable.

B. Problem Identification

In the above related work, many techniques are introduced to find a way to deal with fault tolerance with Cauchy matrix in combination with RS code. Several existing approaches require a long computation time to conduct XOR operations in encoding and decoding user data. In existing work, erasure codes like Cauchy RS use Galois Field (GF) to accomplish matrix-vector multiplication, which causes CPU overhead due to an exponential increase in read/write operations. We present an Optimized Cauchy Coding method with RS coding to overcome this problem. Proposed method makes it easier to create Cauchy matrices because it reduces the number of XOR operations in data recovery.

C. Research Contribution

The main objectives of this paper are as per the following:

- To reduce the number of read/write operations in the encoding process while storing data in a distributed environment with a certain redundancy configuration.
- To minimize XOR operations for various combinations of matrices as well as schedule fewer iterations for data recovery on the data blocks with logarithmic functions.
- To handle unnecessary resource utilization problems in a distributed environment, replica formation of the same block is replaced with a single block with fault tolerance ability to recover data.

III. Methodology

The Optimized Cauchy Coding (OCC) technique proposed is based on RS coding. The OCC paradigm for data recovery is depicted in Fig. 1. The encoding and decoding procedure here involves a number of different entities.

The entities engaged in the coding process are listed below:

1) Client
2) Data files
3) Data blocks
4) Parity blocks
5) Server nodes

In the block diagram of OCC approach at first when a client wants to write a data file, the write data request will first travel via the encoding block, where the files from the client side F1, F2, F3, and F4 can be written on server nodes.

The source file that the client wants to write on the server node will no longer be written in the original format of the source data file, but will instead be translated into a different format to secure the data. The encoding procedure encodes the file received from the client to make the system fault resistant and increases data availability in the event of a disk failure. To produce a check matrix, the OCC technique examines redundancy configurations (l, m, n) during the encoding process. The number of data blocks is represented as one, the number of parity blocks is represented as m, and the number of coding unit bits per word is represented as n in this system. The experiment considers the trade-off between matrix creation time and resource utilization. In a distributed environment, the system is emulated. XOR operations are used to produce each Ca-Co coding matrix for each chunk. Cauchy matrices are built on GF. Using the Cauchy matrix and XOR operations, GF additions and multiplications are also minimized. In terms of time, the Cauchy matrix density determines encoding performance. Algorithm 1 explains the check matrix construction process. Data blocks and parity blocks for the input files will be written on the servers S1 through S6 after the encoding procedure is completed. When clients want to read the files, the data is decoded by the decoding process. If any data blocks are missing, the data can be recovered using the decoding process, which uses the check matrix established during the encoding process to reassemble the data.

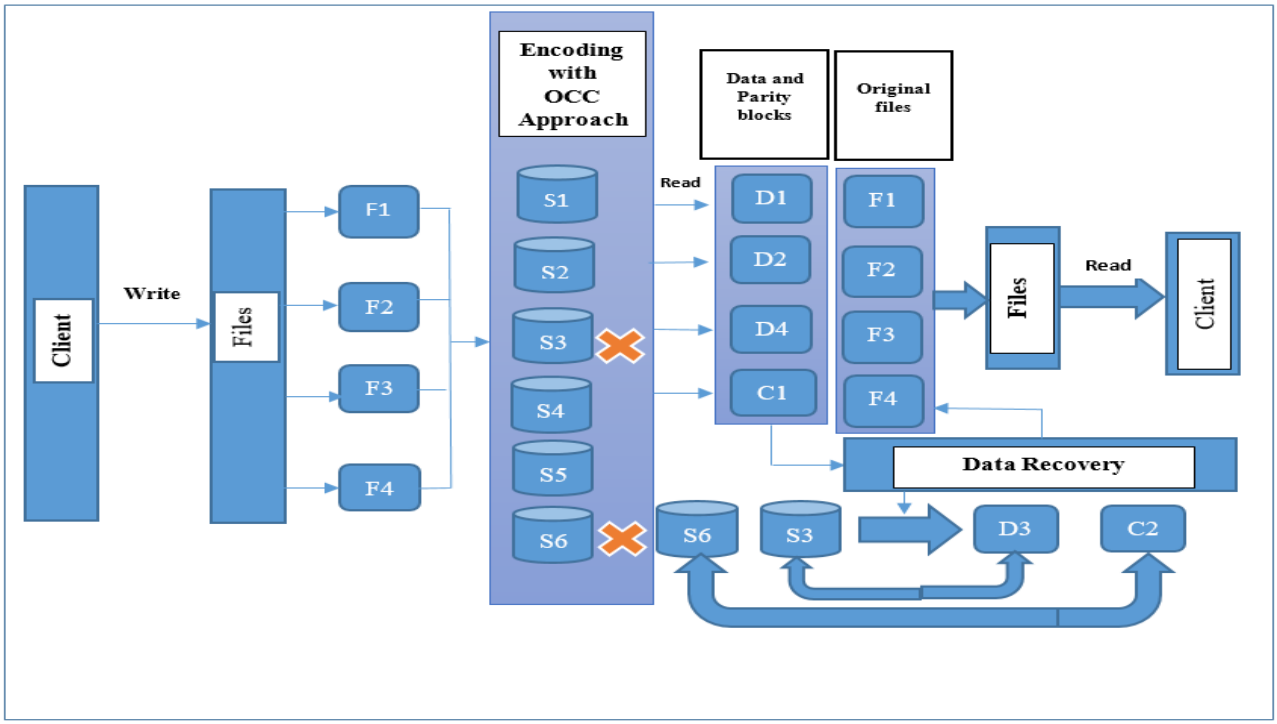

Fig. 1. OCC Data Recovery Approach.

The suggested method lets users control their access in a safe and private way. Algorithm 1 shows how to make an optimized Cauchy matrix and encode data blocks in a data write operation in a step-by-step way.

The proposed OCC data recovery technique converts the Cauchy encoding entities into Boolean form and stores them as a binary matrix. Further binary XOR operations are performed for the elements obtained from Galois Field $GF(2^n)$. It helps in reducing the time complexity of XOR calculations. Here in the encoding algorithm, three parameters l, m and n are considered to generate the Cauchy matrix, where l is the number of data blocks, m is the number of parity blocks and n is the coding unit bits per word.

Algorithm 1: OCC encoding algorithm
Input: $\{l, m, n\}$ where $l = data\ blocks$, $m = parity\ blocks$ and $n = coding\ unit\ bits\ per\ word$
Output: Code word check matrix

The following are the steps for the encoding strategy.

Step 1: We need to generate elements of $GF(2^n)$ with the help of specific polynomial then generate its binary form matrix. So first we need to generate polynomial to generate a $GF(2^n)$.

For different value of n different polynomials are there as below. If we have prime polynomial of degree n over $GF(2)$ then $GF(2^n)$ can be constructed. For $n = 3$ generate $GF(2^3)$ from $GF(2)$ using prime polynomial $p(x) = x^3 + x + 1$.

To calculate elements of $GF(2^3)$: As $p(x) = x^3 + x + 1$, polynomial elements over $GF(2^3)$ is as below:

$GF(8) = GF(2^3)$

z^0 1

z^1 z

z^2 z^2

z^3 $z + 1$

z^4 $z^2 + z$

z^5 $z^2 + z + 1$

z^6 $z^2 + 1$

$z^7 \equiv z^0$ 1

This shows set of elements generated on $GF(2^n)$ using polynomial.

As per binary matrix rules $BM(d)$ is generated using d elements of polynomial and coefficient vector $V(d)$.

Step 2: Generate a Cauchy coding matrix.

In the binary Cauchy coding matrix, the count of ONES decides how many operations of XOR will be required. A lower number of ONES yields better performance. In this matrix generation, the Cauchy Good (CG) method is used for enhancing the performance because the generated matrix will have a lower number of ONES as compared to other heuristic methods.

With the help of CG, constructing the Cauchy matrix, which is defined as MG. Furthermore, we divide each element of MG, as in column j by $MG_{0,j}: MG\ is\ modified\ and\ row\ 0\ elements\ are\ all\ ONES$. For other rows like row i, present number of ONES are calculated which are considered as N. Further by dividing row i, elements by $MG_{i,j}$ and record the number of ONES count, which is represented as $N_j (j \in [0, l-1])$.

Select a minimum of $\{N_0, N_1, \dots \dots, N_{l-1},\}$ which generates CG position one. Repeat step 2 for the remaining matrix elements of CG.

Step 3: Expand the Cauchy coding matrix of size $l \times m$ to the binary matrix of size $ln \times mn$. Consider the extended matrix as a matrix E.

Step 4: Create an array.

Divide m data blocks into n parts similar to $d_1, d_2, \dots d_m$ into $d_{1,1}, d_{1,2}, \dots d_{m,n}$.

$C = E \times D$

$$C = \begin{bmatrix} E_{1,1} & E_{1,2} & \dots E_{1,mn} \\ E_{2,1} & E_{2,2} & \dots E_{2,mn} \\ \vdots & \vdots & \vdots \\ E_{ln,1} & E_{ln,2} & \dots E_{ln,mn} \end{bmatrix} \times \begin{bmatrix} d_{1,1} \\ d_{1,2} \\ \vdots \\ d_{m,n} \end{bmatrix} = \begin{bmatrix} C_{1,1} \\ C_{1,2} \\ \vdots \\ C_{l,n} \end{bmatrix} \quad (1)$$

$C_{1,1} = (E_{1,1} \cdot d_{1,1}) \oplus (E_{1,2} \cdot d_{1,2}) \oplus (E_{1,mn} \cdot d_{m,n})$

The result array is as below:

$$R1 = \begin{bmatrix} d_{1,1} & d_{2,1} \cdots & d_{m,1} C_{1,1} & C_{2,1} \dots & C_{m,1} \\ d_{1,2} & d_{2,2} \cdots & \vdots \quad \vdots & \vdots \quad \vdots & \vdots \\ \vdots & \vdots & & & \\ d_{1,n} & d_{2,n} \dots & d_{m,n} C_{1,n} & C_{2,n} \dots & C_{l,n} \end{bmatrix} \quad (2)$$

Further XOR operations are calculated then sorted. Further result is replaced with new data as $R2$.

Algorithm 2 shows how to write data to a cloud storage service using the OCC approach.

Algorithm 2: Data recovery algorithm
Input: Code word check matrix
Output: Plain text

Step 1: The client sends a request to the server for data extraction.

Step 2: The decoding matrix generation phase.

As R1 has been partitioned into the blocks of (m +l) ×n and matrix BM(d) which has (m +l) n× (m +l) w in size is formed.

$$BM(d) = \frac{Identity\ matrix | Zero\ matrix}{Parity\ matrix\ of\ R1} \quad (3)$$

Step 3: Evaluate the missing data blocks.

i: When a data node fails, the information for the whole segment is lost. So while re-establishing information, a relating list L of missing components is made for the missing node.

ii: The component d in the missing component list L is iterated through. Assuming it has a place in the data component; continue to step iii.

If it has a place in the redundant component, skip it.

iii: The r is the row set whose element d is one in the parity check matrix H. Assume that there is no component in r, set row d to zero. In case there is a component in r, then, at that point, we select the row with the little load of hamming subsequent to eliminate the missing component from r as r_1. The row corresponding to the component in r and the row d performs XOR with r_1 and r_1 is set to 0.

iv: After handling each component of the list L, the values in the column corresponding to the redundant components in the list can be set to zero.

Step 4: Data recovery phase

Now, line r_d is an analogous row to the component d of the missing component list with the name L which is the relating matrix BM(d). C_d is a set of columns with 1 in r_d. The information block relating to the component in C_d is the data block expected for restoring the component d.

As per the above component list with the name L and the binary matrix BM(d), all missing records can be recovered as before. Lost data can be recovered with XOR operations with available dataset.

IV. Result

This section examines performance in terms of the amount of time it takes to encode and decode data in a distributed system. The requirement to run a test-engine to evaluate system performance is shown in Table I.

At the output, it encodes these data blocks into m parity blocks. The storage system is resistant to numerous disk failures in this case. The OCC code works with binary data. Each word consists of n bits such that $2^n \geq l + m$. For data encoding, redundancy configuration(l, m, n) varies by keeping different combinations of data blocks and parity blocks.

TABLE I. System Requirement

Particulars	Specifications
CPU with RAM	Pentium 2.5 GHz with 4 GB RAM
Language	Java
Operating system	Linux

The work of the proposed system is compared to that of the present system in terms of encoding and decoding times, utilizing various redundancy configurations. Encoding time refers to the time it takes to convert the contents of a file into a specified format that allows data to be securely sent and stored in a specific data node. The time it takes to convert an encoded data sequence back to its native format is known as decoding time. Table II demonstrates the time necessary to code files ranging in size from 1MB to 50MB using the proposed OCC coding technique in seconds.

TABLE II. Encoding and Decoding Time for Various Size Input File for Proposed OCC (L=10, M=20)

File size	Encoding Time	Decoding Time
1 MB	0.12	0.145
5 MB	0.26	0.31
10 MB	0.54	1.3

In the event of a disk failure, Table III outlines the various criteria, such as data availability, recoverability, efficiency, and storage overhead. It displays great data availability, high recoverability, and high efficiency for the inputs $l = 4$ and $m = 1$. For the same arrangement, the storage overhead is minimal. The amount of storage space required grows as the number of parity blocks grow.

TABLE III. Performance Analysis of Proposed OCC for Various Input Pair Values (L, M)

Encoding	Availability	Recoverability	Efficiency	Storage Overhead
l=4, m=1	High	High	High	Low
l=5, m=1	High	High	Medium	Low
l=6, m=1	High	High	High	Low
l=7, m=1	High	High	High	Medium
l=4, m=2	High	High	High	Low
l=4, m=3	High	Medium	High	Low
l=4, m=4	High	High	Medium	Low
l=4, m=5	High	Medium	Low	High

V. Discussion

Table IV demonstrates the encoding time required for various ways to encode the data when different numbers of data blocks ($l = 4$ to $l = 10$) and parity block ($m = 1$) are examined. For comparison of the proposed OCC with the existing approaches of Optimal Weakly Secure Minimum Storage Regenerating (OWSPM-MSR) [31] and Product-Matrix Minimum Storage Regenerating (PM-MSR) [32], a total of seven different combinations are evaluated.

TABLE IV. Encoding Time for Various Number of Data Block (L)

Techniques	Encoding Time(s)						
	l=4, m=1	*l*=5, m=1	*l*=6, m=1	*l*=7, m=1	*l*=8, m=1	*l*=9, m=1	*l*=10, m=1
OWSPM-MSR	0.0065	0.0065	0.0082	0.0119	0.0125	0.0119	0.0135
PM-MSR	0.0065	0.0077	0.039	0.053	0.098	0.27	0.38
Proposed OCC	0.006	0.0063	0.008	0.0084	0.009	0.0096	0.01

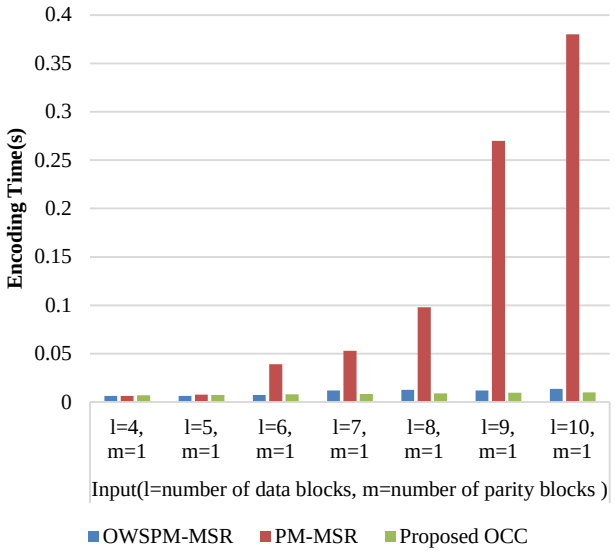

Fig. 2. Comparison of Encoding Time with Various Number of Data Blocks.

Fig. 2 depicts an encoding time plot, with the X axis representing input and the Y axis representing encoding time for a single input. The inputs are the number of data blocks (l = 4 to 10) and the number of parity blocks (m = 1). The OCC encoding procedure outperforms the previous OWSPM-MSR and PM-MSR algorithms for various combinations of one and m. Fig. 3 depicts an encoding time plot, with the X axis representing input and the Y axis representing encoding time for a single input. The number of data blocks (l = 4) and the number of parity blocks (m = 1 to 10) are used as inputs. The OCC encoding procedure outperforms the previous OWSPM-MSR and PM-MSR algorithms for various combinations of l and m.

Table V shows the encoding time required for various approaches to encode the data when m = 1 to m = 10 parity blocks and data blocks l = 4 are examined. For comparison of the proposed OCC with the existing approaches OWSPM-MSR and PM-MSR, a total of seven different combinations are explored.

Table VI demonstrates the decoding time necessary for various ways to decode the data when different numbers of data blocks (l = 4 to l = 10) and parity block (m = 1) are examined. For a comparison of the new OCC with the existing OWSPM-MSR and PM-MSR approaches, seven different combinations are tried.

Fig. 4 depicts a plot for decoding time, with the X axis representing input and the Y axis representing decoding time for a certain input. The inputs are the number of data blocks (l = 4 to 10) and the number of parity blocks (m = 1). OCC outperforms the previous OWSPM-MSR and PM-MSR algorithms for various combinations of l and m decoding.

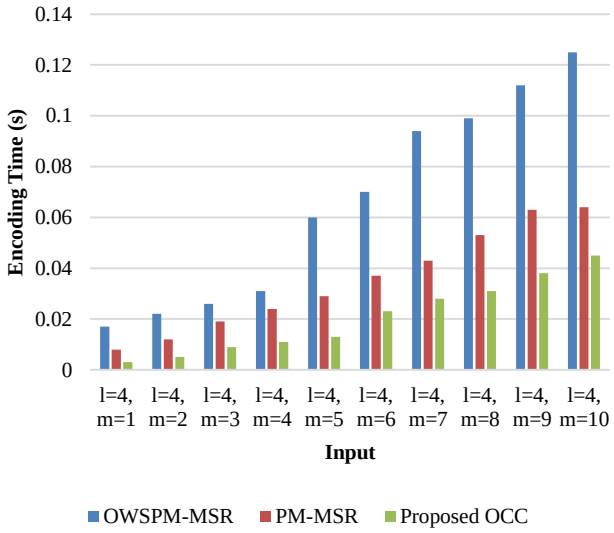

Fig. 3. Comparison of Encoding Time for Various Number of Parity Blocks.

TABLE V. ENCODING TIME (IN SECONDS) FOR VARIOUS NUMBER OF PARITY BLOCKS

Techniques	Encoding Time(s)									
	l=4, m=1	*l*=4, m=2	*l*=4, m=3	*l*=4, m=4	*l*=4, m=5	*l*=4, m=6	*l*=4, m=7	*l*=4, m=8	*l*=4, m=9	*l*=4, m=10
OWSPM-MSR	0.017	0.022	0.026	0.031	0.06	0.07	0.094	0.099	0.112	0.125
PM-MSR	0.008	0.012	0.019	0.024	0.029	0.037	0.043	0.053	0.063	0.064
Proposed OCC	0.003	0.005	0.009	0.011	0.013	0.023	0.028	0.031	0.038	0.045

TABLE VI. DECODING TIME (IN SECONDS) FOR VARIOUS NUMBER OF DATA BLOCKS

Techniques	Decoding Time(s)						
	l=4, m=1	*l*=5, m=1	*l*=6, m=1	*l*=7, m=1	*l*=8, m=1	*l*=9, m=1	*l*=10, m=1
OWSPM-MSR	0.010	0.023	0.045	0.075	0.12	0.19	0.26
PM-MSR	0.0092	0.015	0.035	0.050	0.070	0.085	0.13
Proposed OCC	0.0077	0.0083	0.0089	0.0097	0.010	0.016	0.018
PM-MSR	0.0092	0.015	0.035	0.050	0.070	0.085	0.13

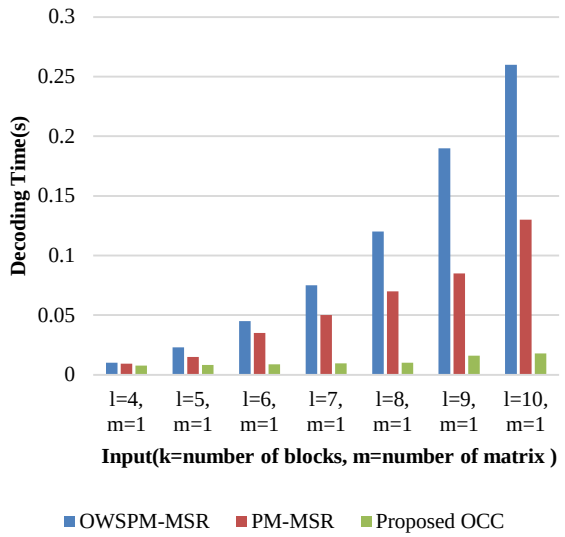

Fig. 4. Comparison of Decoding Time for Various Number of Data Blocks.

Table VII compares the average time required for encoding and decoding 1024KB of data using different approaches. In comparison to the existing two methodologies, the proposed system takes less time.

Table VIII demonstrates the decoding time required for various ways to decode the data when $m = 1$ to $m = 10$ parity blocks are used in combination with data block $l = 4$. For a comparison of the new OCC with the existing OWSPM-MSR and PM-MSR approaches, seven different combinations are tried.

TABLE VII. AVERAGE ENCODING AND DECODING TIME OF DIFFERENT METHODS (FILE SIZE 1024KB, L=4, M=10)

Techniques	Encoding Time(s)	Decoding Time(s)
OWSPM-MSR	0.125	0.22
PM-MSR	0.045	0.16
Proposed OCC	0.035	0.116

Fig. 5 depicts a plot for decoding time, with the X axis representing input and the Y axis representing encoding time for a certain input. As inputs, the data block count ($l = 4$) and the m parity block count ($m = 1$ to 10) are used. OCC outperforms the previous OWSPM-MSR and PM-MSR algorithms for various combinations of l and m decoding.

Fig. 6 and Fig. 7 provide a comparison graph for average data encoding and decoding times for various techniques, showing that the proposed OCC outperforms existing techniques with a 0.035 second encoding time and 0.116 second decoding time for a 1024KB data file.

The OCC data recovery strategy outperforms the OWSPM-MSR and PM-MSR data recovery strategies. Because PM-MSR employs the Vandermonde matrix in the encoding process, the encoding time is longer than the OCC technique, which uses the Cauchy matrix in conjunction with the RS code. In the encoding process, the OCC technique employs a Cauchy Good matrix with fewer ONES, which decreases the time required for the XOR operation. Ultimately it optimized overall processing time for encoding and decoding. As a result, the OCC technique outperforms the OWSPM-MSR and PM-MSR strategy.

TABLE VIII. DECODING TIME FOR VARIOUS NUMBER OF PARITY BLOCKS

Techniques	Decoding Time(s)									
	l=4, m=1	*l*=4, m=2	*l*=4, m=3	*l*=4, m=4	*l*=4, m=5	*l*=4, m=6	*l*=4, m=7	*l*=4, m=8	*l*=4, m=9	*l*=4, m=10
OWSPM-MSR	0.03	0.045	0.065	0.080	0.11	0.13	0.15	0.16	0.19	0.22
PM-MSR	0.02	0.035	0.050	0.070	0.08	0.085	0.095	0.010	0.12	0.16
Proposed OCC	0.0086	0.0092	0.0099	0.0103	0.0135	0.0156	0.0187	0.0256	0.0285	0.0321

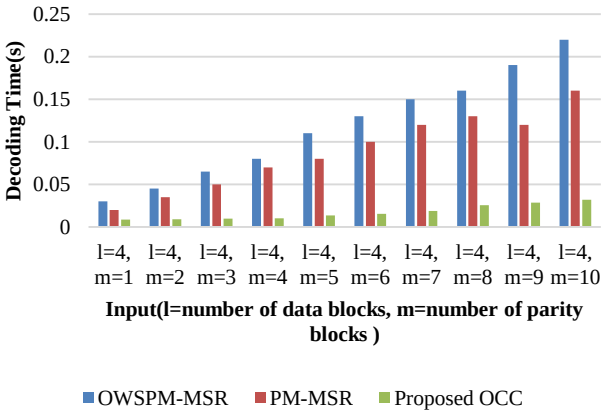

Fig. 5. Comparison of Decoding Time for Various Number of Parity Blocks.

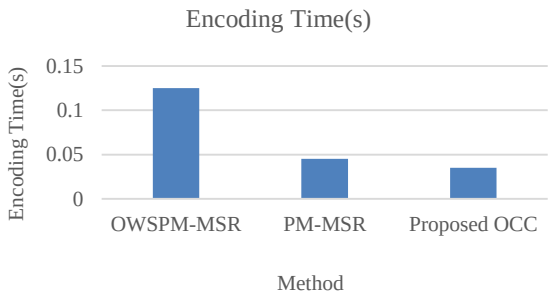

Fig. 6. Comparison of Average Encoding Time of Different Methods.

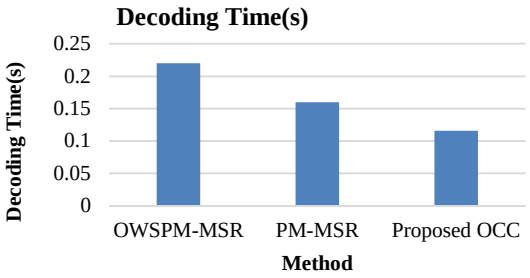

Fig. 7. Comparison of Average Encoding and Decoding Time of Different Methods.

VI. Conclusion and Future Work

This paper proposed an optimized Ca-Co approach to utilize matrix as well as schedule heuristics to attain an optimal coding structure. In terms of data recovery, it outperforms the OWSPM-MSR and PM-MSR approaches. For data coding, a 1024KB file with various combinations of data blocks l and parity blocks m is evaluated. For coding, two data block and parity block combinations are evaluated. In the first scenario, m equals 1 and l ranges from 4 to 10. The value of l is 4 in the second scenario, while m ranges from 1 to 10. The existing OWSPM-MSR approach takes an average of 0.125 seconds to encode and 0.22 seconds to decode, whereas the PM-MSR approach takes an average of 0.045 seconds to encode and 0.16 seconds to decode. The proposed OCC enhances data coding performance by taking an average of 0.035 seconds to encode and 0.116 seconds to decode data. In future, we intend to use the system in a hybrid cloud environment to balance energy savings with resource virtualization using punctured Cauchy RS code, which can further optimize network resources like power and bandwidth.

References

[1] J. Wang, Y. Yang, T. Wang, R. S. Sherratt, J. Zhang, "Big Data Service Architecture: A Survey," *Journal of Internet Technology*, vol. 21, no. 2, pp. 393-405, Mar. 2020.

[2] N. A. Ghani, S. Hamid, I. A. T. Hashem, E. Ahmed, "Social Media Big Data Analytics: A Survey", *Computers in Human Behavior*, Vol. 101, pp. 417-428, 2019, ISSN 0747-5632, https://doi.org/10.1016/j.chb.2018.08.039.

[3] J. Yang, H. Zhu, T. Liu, "Secure and Economical Multi-Cloud Storage Policy with NSGA-II-C", *Applied Soft Computing*, Vol. 83, 2019, 105649, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2019.105649.

[4] N. Chervyakov, M. Babenko, A. Tchernykh, N. Kucherov, V. Miranda-Lopez, J. M. Cortes-Mendoza, "AR-RRNS: Configurable Reliable Distributed Data Storage Systems for Internet of Things to Ensure Security", *Future Generation Computer Systems*, Vol. 92, 2019, Pages 1080-1092, ISSN 0167-739X, https://doi.org/10.1016/j.future.2017.09.061.

[5] J. Li, J. Wu, L. Chen, "Block-secure: Blockchain Based Scheme for Secure P2P Cloud Storage", *Information Sciences*, Vol. 465, pp. 219-231, 2018, ISSN 0020-0255.

[6] P. V. Bhuvaneshwari & C. Tharini, "Review on LDPC Codes for Big Data Storage", *Wireless Personal Communications*, Vol. 117, Issue 2, pp. 1601-1625, 2021.

[7] Y. J. Tang and X. Zhang, "Fast En/Decoding of Reed-Solomon Codes for Failure Recovery", *IEEE Transactions on Computers*, vol. 71, no. 3, pp. 724-735, 1 March 2022, https://doi: 10.1109/TC.2021.3060701.

[8] M. Makovenko, M. Cheng and C. Tian, "Revisiting the Optimization of Cauchy Reed-Solomon Coding Matrix for Fault-Tolerant Data Storage," *IEEE Transactions on Computers*, https://doi: 10.1109/TC.2021.3110131.

[9] F. Kazemi, S. Kurz, Soljanin and A. Sprintson, "Efficient Storage Schemes for Desired Service Rate Regions", *2020 IEEE Information Theory Workshop (ITW)*, pp. 1-21, 2020.

[10] D. Dai and S. Boroomand, "A Review of Artificial Intelligence to Enhance the Security of Big Data Systems: State-of-Art, Methodologies, Applications, and Challenges", *Archives of Computational Methods in Engineering*, pp. 1291–1309, 2022.

[11] S. Ravikumar and D. Kavitha, "A New Adaptive Hybrid Mutation Black Widow Clustering Based Data Partitioning for Big Data", Analysis *Wireless Pers Commun*, Vol. 120, pp. 1313–1339, 2021, https://doi.org/10.1007/s11277-021-08516-x.

[12] Y. Li, J. Ma, Y. Zhang, "Image Retrieval from Remote Sensing Big Data: A Survey", *Information Fusion*, Vol. 67, pp. 94-115, 2021, ISSN 1566-2535, https://doi.org/10.1016/j.inffus.2020.10.008.

[13] Y. Xiao, S. Zhou and L. Zhong, "Erasure Coding-Oriented Data Update for Cloud Storage: A Survey," in *IEEE Access*, vol. 8, pp. 227982-227998, 2020, doi: 10.1109/ACCESS.2020.3033024.

[14] X. Gong, C. Wan Sung, Zigzag, "Decodable Codes: Linear-Time Erasure Codes with Applications to Data Storage", *Journal of Computer and System Sciences*, Vol. 89, pp. 190-208, 2017, ISSN 0022-0000, https://doi.org/10.1016/j.jcss.2017.05.005.

[15] S. H. Dau, W. Song, A. Sprintson and C. Yuen, "Constructions of MDS Codes via Random Vandermonde and Cauchy Matrices over Small Fields," *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2015, pp. 949-955, https://doi: 10.1109/ALLERTON.2015.7447110.

[16] S. Wu, Y. Xu, Y. Li and Z. Yang, "I/O-Efficient Scaling Schemes for Distributed Storage Systems with CRS Codes," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 27, No. 9, pp. 2639-2652, 1 Sept. 2016, https://doi: 10.1109/TPDS.2015.2505722.

[17] Y. Zhang, J. Yang, A. Memaripour and S. Swanson, "Mojim: A Reliable and Highly-Available Non-Volatile Memory System", *SIGARCH Comput. Archit. News*, Vol. 43, No. 1, pp. 3–18, 2015, https://doi.org/10.1145/2786763.269437.

[18] X. Chen and X. Ma, "Optimized Recovery Algorithms for RDP $(p, 3) $ Code," in *IEEE Communications Letters*, vol. 22, no. 12, pp. 2443-2446, Dec. 2018, https://doi: 10.1109/LCOMM.2018.2875468.

[19] Z. Shen, J. Shu and P. P. C. Lee, "Reconsidering Single Failure Recovery in Clustered File Systems," *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2016, pp. 323-334, doi: 10.1109/DSN.2016.37.

[20] D. Burihabwa, P. Felber, H. Mercier, and V. Schiavoni, "A Performance Evaluation of Erasure Coding Libraries for Cloud-Based Data Stores: (Practical Experience Report)", *In Distributed Applications and Interoperable Systems: 16th IFIP WG 6.1 International Conference, DAIS 2016*, Proceedings. Springer-Verlag, Berlin, Heidelberg, pp. 160–173, 2016, https://doi.org/10.1007/978-3-319-39577-7_13.

[21] A. Shah and M. Padole, "Load Balancing through Block Rearrangement Policy for Hadoop Heterogeneous Cluster," *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2018, pp. 230-236, doi: 10.1109/ICACCI.2018.8554404.

[22] Sudhakar and S. K. Pandey, "An Approach to Improve Load Balancing in Distributed Storage Systems for NoSQL Databases: MongoDB", *In: Pattnaik, P., Rautaray, S., Das, H., Nayak, J. (eds) Progress in Computing, Analytics and Networking. Advances in Intelligent Systems and Computing*, Vol. 710, pp. 529-538, Springer, Singapore, https://doi.org/10.1007/978-981-10-7871-2_51.

[23] L. Han, S. Tan, B. Xiao, C. Ma and Z. Shao, "Optimizing Cauchy Reed-Solomon Coding via ReRAM Crossbars in SSD-based RAID Systems," *2019 IEEE Non-Volatile Memory Systems and Applications Symposium (NVMSA)*, 2019, pp. 1-6, https://doi: 10.1109/NVMSA.2019.8863519.

[24] L. Xu, M. Lyu, Z. Li, C. Li and Y. Xu, "A Data Layout and Fast Failure Recovery Scheme for Distributed Storage Systems with Mixed Erasure Codes" in *IEEE Transactions on Computers*, no. 01, pp. 1-1, 2021, https://doi: 10.1109/TC.2021.3105882.

[25] M. Pirahandeh and D. Kim, "MS Scheduler: New, Scalable, and High-Performance Sparse AVX-2 Parity Encoding and Decoding Technique

for Erasure-Coded Cloud Storage Systems", *Future Generation Computer Systems*, Vol. 126, 2022, pp. 123-135, ISSN 0167-739X, https://doi.org/10.1016/j.future.2021.08.002.

[26] R. Wu, L. Wang, and Y. Wu, "A High-Speed Cauchy CODEC Algorithm for Distributed Storage System", *In Proceedings of the 2020 International Conference on Internet Computing for Science and Engineering (ICICSE '20). Association for Computing Machinery*, New York, NY, USA, pp. 20–24, 2020, https://doi.org/10.1145/3424311.3424313.

[27] G. Zhang, G. Wu, S. Wang, J. Shu, W. Zheng and K. Li, "CaCo: An Efficient Cauchy Coding Approach for Cloud Storage Systems," in *IEEE Transactions on Computers*, vol. 65, no. 2, pp. 435-447, 1 Feb. 2016, doi: 10.1109/TC.2015.2428701.

[28] D. Tang and H. Cai, "A Novel Decoding Method for the Erasure Codes", *Security and Communication Networks*, vol. 2021, Article ID 8755697, pp. 1-12, 2021, https://doi.org/10.1155/2021/8755697.

[29] J-J Kim, "Erasure-Coding-Based Storage and Recovery for Distributed Exascale Storage Systems", *Applied Sciences*, Vol. 11, No. 8:3298, https://doi.org/10.3390/app11083298.

[30] X. Wang, Z. Zhang, L. Shan, J. Li, Y. Wang, H. Cao, Z. Li, "An Optimized Encoding Algorithm for Systematic Polar Codes*", J Wireless Com Network 2019*, 193 (2019), pp. 1-12, https://doi.org/10.1186/s13638-019-1491-4.

[31] J. Bian, S. Luo, Z. Li and Y. Yang, "Optimal Weakly Secure Minimum Storage Regenerating Codes Scheme," in *IEEE Access*, vol. 7, pp. 151120-151130, 2019, https://doi: 10.1109/ACCESS.2019.2947248.

[32] N. B. Shah, K. V. Rashmi and P. V. Kumar, "Information-Theoretically Secure Regenerating Codes for Distributed Storage," *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, 2011, pp. 1-5, https://doi: 10.1109/GLOCOM.2011.6133754.