Enhancing Data Recovery in RAID6: A Comparative Analysis of Row-Diagonal Parity Codes

Wenxu Zhang*

School of Automation and Software Engineering, Shanxi University, 030006, Taiyuan, China

Abstract. RAID6 systems, known for their robust data protection and redundancy capabilities, encounter challenges in data recovery efficiency and computational complexity. This study investigates the efficacy of Row-Diagonal Parity (RDP) codes within RAID6 frameworks, emphasizing their ability to recover from dual disk failures. The exploration includes a detailed examination of the encoding and decoding processes of RDP codes, employing concrete examples to demonstrate these techniques. Comparative analysis highlights the computational advantages of RDP codes over traditional methods such as Reed-Solomon and EVENODD Codes. Findings reveal that RDP codes not only reduce computational complexity but also enhance data recovery speed significantly. Given these attributes, RDP codes offer a promising solution for large-scale data environments demanding high reliability and swift recovery. This approach markedly improves RAID6's functionality by optimizing recovery processes, thus supporting high-volume storage systems with stringent data integrity requirements. The potential for future enhancements in RAID6 data recovery through further research into hardware acceleration and artificial intelligence is also acknowledged, aiming to refine recovery times and efficiency in large-scale storage applications.

1 Introduction

In the rapidly evolving landscape of information technology, the demand for robust data storage and recovery solutions has become paramount across various sectors. Redundant Array of Independent Disks (RAID), particularly RAID6, stands as a cornerstone technology for ensuring data integrity and availability through sophisticated redundancy and error recovery mechanisms. RAID6, recognized for its double parity feature, offers the ability to recover from the failure of two concurrent disks, making it indispensable for high-dependency environments such as enterprise storage and data centers [1]. Despite its advantages, RAID6 faces challenges in recovery performance and computational overhead, sparking significant research interest aimed at enhancing its data recovery processes [2].

Research to date has primarily focused on refining RAID6's data recovery strategies, exploring coding techniques such as Reed-Solomon (RS Code), Horizontal Diagonal Parity

* Corresponding author: zhangwenxu@alu.sxu.edu.cn

(HDP), and EVENODD Code. Each method enhances data reliability to some extent but often at the cost of increased recovery time and computational complexity. Recently, Row-Diagonal Parity codes have emerged as a promising alternative, noted for their reduced computational demands and expedited recovery speeds, addressing some of the persistent limitations of existing approaches [3].

This study delves into the application of RDP codes within RAID6 frameworks, examining their encoding and decoding methodologies through detailed examples. The research aims to assess the computational efficiency of RDP in comparison with traditional methods such as Reed-Solomon and EVENODD Codes, highlighting its potential to streamline data recovery in RAID6 systems. This analysis is intended to foster improvements in RAID6's operational efficacy, particularly in large-scale data environments where rapid recovery and high reliability are paramount.

2 Relevant theories

2.1 Reed-solomon code

Reed-Solomon Code is a coding method for RAID 6. In RAID 6, the use of Reed-Solomon code allows for strong data redundancy and error correction. It allocates data and parity information among multiple disks such that even if two disks fail simultaneously, the lost data can be retrieved by the data and parity information on the remaining disks.

Its specific implementation process is as follows:

Specify the generating matrix of the Reed-Solomon code as

$$G = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \cdots & \cdots & \cdots & \cdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_n^{k-1} \end{bmatrix} \quad (1)$$

Here $\alpha_1, \alpha_2, \cdots, \alpha_n,$ are distinct elements of $GF(p^t)$, where p is a prime and t is a positive integer [3].

$mG = c$, where m represents the length-k input and c represents the length-n codeword.

Reed-Solomon codes are founded on algebraic coding theory and possess efficient error correction performance as well as flexibility to adapt to various storage requirements and failure circumstances [4]. However, it also has certain shortcomings. Since the encoding and decoding process of Reed-Solomon codes involves complex mathematical operations and a large amount of computational resources, the computational complexity is high. This means that in practical applications, it needs to consume more computational resources and time, which may have some impact on the performance of the system. Its application in RAID 6 improves the reliability and data security of the storage system, which is especially suitable for scenarios that require high data integrity, such as enterprise-level storage and data centers.

2.2 EVENODD code

EVENODD is a coding method specifically designed for RAID6, whose main goal is to protect data from dual disk failures.

In terms of coding structure, EVENODD code adopts the design of two parity blocks [5]. One parity block is used for horizontal parity, which checks the rows of the data block to ensure the accuracy of each row. The calculation process is to perform XOR operation on the data blocks of the identical row to get the row parity block of the respective row.

The other parity block is used for diagonal parity, which provides additional redundancy information by checking specific combinations of data blocks. It is computed by first

calculating the special sum $S = \sum_{t=1}^{m-1} \alpha_{m-1-t,t}$ and then by this formula $\alpha_{l,\ m+1} = S + \sum_{t=0,t\neq l+1}^{m-1} \alpha_{<l-t>m,t}$, where m is the number of data disks and l is the serial number of the data block starting from 0 [6].

This structure enables the EVENODD code to use both parity blocks effectively for data recovery when both disks fail simultaneously.

Compared to Reed-Solomon codes, EVENODD codes have reduced computational complexity. This reduces the computational resources and time needed for encoding and decoding, thereby enhancing the overall system performance. Simultaneously, the EVENODD code still maintains high efficiency in correcting two simultaneous disk failures. However, compared to RDP, the EVENODD code is slightly less efficient. Nevertheless, as a simpler method than the Reed-Solomon code, it still has a wide range of applications in scenarios where there is a limitation of computational resources or where the speed of data recovery is not extremely demanding.

2.3 Horizontal-diagonal parity

HDP is another effective coding scheme used in the data storage field to correct dual disk failures. It provides a strong guarantee for data security through its unique parity method.

In the specific parity method, HDP on the one hand calculates the horizontal parity of the data block. This indicates that it computes parity for data blocks along the same row, ensuring the integrity of each individual line of data. On the other hand, it also calculates diagonal parity for a specific set of data blocks. This diagonal parity can span different rows and columns, providing more comprehensive redundancy information. The specific calculation formula is as follows:

Horizontal-Diagonal Parity Element:

The calculation of the horizontal-diagonal parity element $C_{i,i}(0 \leq i \leq p-2)$ is based on $C_{i,i} = \sum_{j=0}^{p-2} C_{i,j}(j \neq i)$, with p denoting the number of disks involved. As an example, within a 6-disk array $(p = 7)$ implementing HDP code, the value of $C_{1,1}$ can be determined by $C_{1,1} \oplus C_{1,2} \oplus C_{1,3} \oplus C_{1,4} \oplus C_{1,5}$ [7].

Anti-Diagonal Parity Element:

The calculation of the anti-diagonal parity element $C_{i,p-2-i}(0 \leq i \leq p-2)$ is done by computing $C_{i,p-2-i} = \sum_{j=0}^{p-2} C_{<2i+j+2>_p,j}$. For instance, in a 6-disk array $(p = 7)$ HDP code, to calculate $C_{0,5}(i = 0)$, first, for $j = 0$, $2i + j + 2 = 2$, and $< 2 >_p = 2$, so the first data element is $C_{2,0}$. The other data elements $C_{3,1}$, $C_{4,2}$, and $C_{5,3}$ can be calculated in the same way. Then, $C_{0,5} = C_{2,0} \oplus C_{3,1} \oplus C_{4,2} \oplus C_{5,3}$.

By combining horizontal and diagonal parity, HDP can leverage this redundancy to accurately recover data when both disks fail.

HDP has some conceptual similarity to the EVENODD code, but it optimizes certain aspects of the parity calculation. Such optimizations may include more efficient computation methods, more reasonable parity block layouts, or better fault recovery strategies. Through such optimizations, HDP is capable of achieving a moderate improvement in both the efficiency and reliability of data recovery.

2.4 Row-diagonal parity

RDP demonstrates excellent performance in preventing dual disk failures, making it one of the ideal choices for large-scale storage systems. In terms of checksums, RDP cleverly combines row-based and diagonal-based parity. By checking the rows of a data block, the integrity of each row is ensured. At the same time, the introduction of diagonal checksums

further enhances the redundancy of the data and can span different rows and columns, providing more comprehensive error detection and correction capabilities. The integration of row and diagonal parity allows RDP to swiftly and accurately reconstruct lost data when both disks fail.

One of the advantages of RDP is its low computational complexity. Compared to some other encoding methods, RDP requires fewer computational resources and less time to perform data recovery and error correction operations. This makes it perform well in large-scale storage systems that can efficiently handle large amounts of data and multiple disks. The low-computational-complexity advantage of RDP is especially evident in data storage scenarios that require fast response and high reliability, such as cloud computing data centers and large enterprise storage networks. The following section continues with an in-depth description of RDP.

3 The encoding and decoding processes of RDP

3.1 Encoding of RDP

3.1.1 Calculation of parity1

RDP's row parity (Parity 1) is calculated in the same way as EVENODD by performing an XOR operation on blocks of data in the same row.

3.1.2 Calculation of parity2

The diagonal parity (parity 2) of RDP is calculated differently from the diagonal parity of EVENODD in that it does not use a special sum S and includes the row parity (parity 1) in the operation for XOR. The specific calculation process can be carried out according to the following formula, with m indicating the total data disks and i representing the data block's serial number, beginning at 0 [8].

$$\alpha_{i,\ m+1} = \sum_{j=0,j\neq i+1}^{m} \alpha_{<i-j>(m+1),j} \tag{2}$$

3.2 A specific example of RDP encoding

Table 1. Example of RDP encoding.

Disk/Block	0	1	2	3	Parity1	Parity2
0	1	0	1	0	0	0
1	0	1	0	1	0	0
2	1	1	0	0	0	1
3	1	0	1	0	0	0

As shown in the Table 1. Parity1 is calculated as follows:

$$\begin{aligned} \alpha_{0,\ 4} &= \alpha_{0,\ 0}\oplus\alpha_{0,\ 1}\oplus\alpha_{0,\ 2}\oplus\alpha_{0,\ 3} = 1\oplus0\oplus1\oplus0 = 0 \\ \alpha_{1,\ 4} &= \alpha_{1,\ 0}\oplus\alpha_{1,\ 1}\oplus\alpha_{1,\ 2}\oplus\alpha_{1,\ 3} = 0\oplus1\oplus0\oplus1 = 0 \\ \alpha_{2,\ 4} &= \alpha_{2,\ 0}\oplus\alpha_{2,\ 1}\oplus\alpha_{2,\ 2}\oplus\alpha_{2,\ 3} = 1\oplus1\oplus0\oplus0 = 0 \\ \alpha_{3,\ 4} &= \alpha_{3,\ 0}\oplus\alpha_{3,\ 1}\oplus\alpha_{3,\ 2}\oplus\alpha_{3,\ 3} = 1\oplus0\oplus1\oplus0 = 0 \end{aligned} \tag{3}$$

In this example m = 4 and i is 0 to 3, then the formula for Parity2 is:

$$\alpha_{i,\ 5} = \sum_{j=0,j\neq i+1}^{4} \alpha_{<i-j>5,j} \tag{4}$$

Then the exact calculation process is:

$$\alpha_{0,\ 5} = \alpha_{0,\ 0}\oplus\alpha_{3,\ 2}\oplus\alpha_{2,\ 3}\oplus\alpha_{1,\ 4} = 1\oplus1\oplus0\oplus0 = 0$$

$$\alpha_{1,\ 5} = \alpha_{1,\ 0} \oplus \alpha_{0,\ 1} \oplus \alpha_{3,\ 3} \oplus \alpha_{2,\ 4} = 0 \oplus 0 \oplus 0 \oplus 0 = 0$$
$$\alpha_{2,\ 5} = \alpha_{2,\ 0} \oplus \alpha_{1,\ 1} \oplus \alpha_{0,\ 2} \oplus \alpha_{3,\ 4} = 1 \oplus 1 \oplus 1 \oplus 0 = 1 \quad (5)$$
$$\alpha_{3,\ 5} = \alpha_{3,\ 0} \oplus \alpha_{2,\ 1} \oplus \alpha_{1,\ 2} \oplus \alpha_{0,\ 3} = 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

3.3 Decoding of RDP

The decoding process of RDP is performed to recover disk data after a disk failure. Therefore, different disk failure scenarios will result in different recovery processes, which means different decoding processes. The following will explain this by considering various cases.

3.3.1 Single disk failure

Single data disk failure： In the case of a single disk failure, the first possibility is the failure of an individual data disk. In this situation, the row parity disk (Parity1) can be utilized to recover the data from the failed disk [9]. The lost data blocks can be restored by performing an XOR operation between the remaining data blocks in the same row and Parity1.

Single parity disk failure: Another possibility is the failure of one of the parity disks, either row parity or diagonal parity, which can be restored by recalculating the corresponding parity formula.

3.3.2 Double disk failure

In distributed storage systems, handling disk failures efficiently is crucial for maintaining data integrity and availability. This study delineates the recovery mechanisms for various failure scenarios in a system employing erasure coding with two parity disks.

The first scenario involves the failure of both parity disks. This situation is the simplest to address; recovery is achieved by directly recalculating the contents of the failed parity disks from the remaining data disks.

The second scenario concerns the simultaneous failure of a data disk and the diagonal parity disk. Recovery in this case is also straightforward: the failed data disk is initially restored using the row parity, followed by a recalculation of the diagonal parity to restore the diagonal parity disk.

The third scenario features the failure of one data disk alongside a row parity disk. Recovery here is more complex and involves locating a set of diagonals with only one failed block. This block can be swiftly restored using the diagonal parity, enabling the reconstruction of another faulty block in the same row using row parity.

The fourth and most complex scenario involves the failure of two data disks. Similar to the previous case, the recovery process begins by identifying a diagonal containing only one failed block, which is restored using diagonal parity. Subsequently, row parity is utilized to calculate and restore another faulty block in the same row. In each of these scenarios, the recovery techniques exploit the redundancy inherent in the coding scheme to restore lost data effectively. Although the methods vary in complexity based on the nature and number of disks that fail, they consistently provide a systematic approach for data recovery. This analysis highlights the robustness of erasure coding in ensuring data resilience in distributed storage environments and suggests further research into hybrid coding schemes and machine learning optimizations to enhance recovery efficacy and efficiency.

3.4 A specific example of decoding in the case of a double disk failure in RDP

Table 2. Example of RDP decoding.

Disk/Block	0	1	2	3	Parity1	Parity2
0	1	⑥	1	⑤	0	0
1	0	②	0	③	0	0
2	1	④	0	①	0	1
3	1	⑧	1	⑦	0	0

As shown in the table 2. The following is a demonstration of the RDP decoding process using the example of two data disks that have failed, in this case disk 1 and disk 3.

First, it is easy to see that there are two sets of diagonals with only one missing data block, block 1($\alpha_{2,\ 3}$) and block 2($\alpha_{1,\ 1}$) by diagonal parity.

$$\because \alpha_{0,\ 5} = \alpha_{0,\ 0} \oplus \alpha_{3,\ 2} \oplus \alpha_{2,\ 3} \oplus \alpha_{1,\ 4}$$
$$\alpha_{2,\ 5} = \alpha_{2,\ 0} \oplus \alpha_{1,\ 1} \oplus \alpha_{0,\ 2} \oplus \alpha_{3,\ 4}$$
$$\therefore \alpha_{2,\ 3} = \alpha_{0,\ 5} \oplus \alpha_{0,\ 0} \oplus \alpha_{3,\ 2} \oplus \alpha_{1,\ 4} = 0 \oplus 1 \oplus 1 \oplus 0 = 0 \quad (6)$$
$$\alpha_{1,\ 1} = \alpha_{2,\ 5} \oplus \alpha_{2,\ 0} \oplus \alpha_{0,\ 2} \oplus \alpha_{3,\ 4} = 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

Then block 3($\alpha_{1,\ 3}$) and block 4($\alpha_{2,\ 1}$) by row parity of the corresponding rows.

$$\because \alpha_{1,\ 4} = \alpha_{1,\ 0} \oplus \alpha_{1,\ 1} \oplus \alpha_{1,\ 2} \oplus \alpha_{1,\ 3}$$
$$\alpha_{2,\ 4} = \alpha_{2,\ 0} \oplus \alpha_{2,\ 1} \oplus \alpha_{2,\ 2} \oplus \alpha_{2,\ 3} \quad (7)$$
$$\therefore \alpha_{1,\ 3} = \alpha_{1,\ 4} \oplus \alpha_{1,\ 0} \oplus \alpha_{1,\ 1} \oplus \alpha_{1,\ 2} = 0 \oplus 0 \oplus 1 \oplus 0 = 1$$
$$\alpha_{2,\ 1} = \alpha_{2,\ 4} \oplus \alpha_{2,\ 0} \oplus \alpha_{2,\ 2} \oplus \alpha_{2,\ 3} = 0 \oplus 1 \oplus 0 \oplus 0 = 1$$

Then, there is only one unknown data block on the diagonal line where block 4 is located, block 5($\alpha_{0,\ 3}$) through diagonal parity.

$$\because \alpha_{3,\ 5} = \alpha_{3,\ 0} \oplus \alpha_{2,\ 1} \oplus \alpha_{1,\ 2} \oplus \alpha_{0,\ 3}$$
$$\therefore \alpha_{0,\ 3} = \alpha_{3,\ 5} \oplus \alpha_{3,\ 0} \oplus \alpha_{2,\ 1} \oplus \alpha_{1,\ 2} = 0 \oplus 1 \oplus 1 \oplus 0 = 0 \quad (8)$$

After getting block 5, block 6($\alpha_{0,\ 1}$) in the same row through row parity.

$$\because \alpha_{0,\ 4} = \alpha_{0,\ 0} \oplus \alpha_{0,\ 1} \oplus \alpha_{0,\ 2} \oplus \alpha_{0,\ 3}$$
$$\therefore \alpha_{0,\ 1} = \alpha_{0,\ 4} \oplus \alpha_{0,\ 0} \oplus \alpha_{0,\ 2} \oplus \alpha_{0,\ 3} = 0 \oplus 1 \oplus 1 \oplus 0 = 0 \quad (9)$$

After getting block 6, only block 7() on the diagonal where block 6 is located is unknown, which can be obtained through diagonal parity,

$$\because \alpha_{1,\ 5} = \alpha_{1,\ 0} \oplus \alpha_{0,\ 1} \oplus \alpha_{3,\ 3} \oplus \alpha_{2,\ 4}$$
$$\alpha_{3,\ 3} = \therefore \alpha_{1,\ 5} \oplus \alpha_{1,\ 0} \oplus \alpha_{0,\ 1} \oplus \alpha_{2,\ 4} = 0 \oplus 0 \oplus 0 \oplus 0 = 0 \quad (10)$$

Then block 8($\alpha_{3,\ 1}$) on the same line as block 7 can be obtained through row parity.

$$\because \alpha_{3,\ 4} = \alpha_{3,\ 0} \oplus \alpha_{3,\ 1} \oplus \alpha_{3,\ 2} \oplus \alpha_{3,\ 3}$$
$$\therefore \alpha_{3,\ 1} = \alpha_{3,\ 4} \oplus \alpha_{3,\ 0} \oplus \alpha_{3,\ 2} \oplus \alpha_{3,\ 3} = 0 \oplus 1 \oplus 1 \oplus 0 = 0 \quad (11)$$

Finally, the two disks that failed were fully restored.

4 Complexity comparison

4.1 Encoding complexity comparison

The total count of XOR operations required to create the parity disk defines the computational cost of RDP. In an array including $p - 1$ rows and $p + 1$ disks, there exist $p - 1$ data blocks for each row. In order to implement row parity, $(p-1)(p-2) = p^2 - 3p + 2$ XOR operations are required, while building diagonal parity requires $(p-1)(p-2) = p^2 - 3p + 2$ XOR operations. Therefore, the overall quantity of XOR operations needed to construct an RDP is $2p^2 - 6p + 4$.

The paper also demonstrates through a theorem that, in an array of n data disks, the minimum number of XOR operations per block needed to ensure protection against two failures is $(2 - 2/n)$ [10]. Specifically, RDP requires $2p^2 - 6p + 4$ XORs to protect $(p-1)^2$ data blocks. By setting $n = p - 1$, the number of XOR operations required to protect n^2 data blocks becomes $(2n^2 - 2n)$, which achieves the optimal ratio.

In order to compare RDP and EVENODD Code, consider an array consisting of n data disks, where each disk contains $n - 1$ data blocks. $(n-1)(n-1)$ XOR operations are required for EVENODD Code to compute row parity; $(n-2)n$ XOR operations are required to calculate the parity across n diagonals. In addition, EVENODD Code requires an additional $n - 1$ XORs to complete the diagonal parity calculation, the parity of a given diagonal must be added to the parity of the other $n - 1$ diagonals. Therefore, EVENODD Code requires a sum of $2n^2 - 3n$ XOR operations to generate the parity in a block array of size $n(n-1)$, and an average of $(2n^2 - 3n)/(n^2 - n) = 2 - 1/(n-1)$ XOR operations per block of data. This does not correspond to the optimal rate.

4.2 Decoding complexity comparison

The complexity of decoding plays a key role in failure recovery, and RDP demonstrates optimal computational cost in this process. Since different disk failure scenarios lead to different decoding processes, they are explained in the following cases.

The first scenario involves the recovery process after a single disk failure, for which RDP requires $(p-1)(p-2) = p^2 - 3p + 2$ XOR operations. When setting $n = p - 1$, the total of XOR operations needed to recover n data blocks is $n^2 - n$, meaning each parity block requires $n - 1$ XOR operations. It is important to note that, as has already been demonstrated, the count of XOR operations required by RDP to create a double-protection parity array is minimal and that all parity sets are of equal size, the computational cost of recovering any lost disk is the same in all cases and is optimal.

The second case involves a two-disk failure, which includes a diagonal parity disk. In this case, the recovery cost is equivalent to the cost of constructing the parity disk. Specifically, the recovery process consists of first reconstructing the lost data disk or recomputing the row parity disk using the row parity data, and subsequently reconstructing the diagonal parity disk. Since the cost of recovering any data disk from the row parity is equal to the computational cost of constructing the row parity, and since the cost of constructing the RDP is optimal, the cost of recovery in this case is likewise optimal.

The last case is the recovery process for a double disk failure of two data disks or a disk containing row parity, where RDP needs to recover $2(p-1)$ blocks of data. The size of each parity is $p - 1$, so the cost of recovering each data block is still $p - 2$ XOR operations. This is the same as the computed cost of $2p^2 - 6p + 4 = 2n^2 - 2n$ XOR operations for the construction and other recovery scenarios, which is also optimal.

In summary, RDP demonstrates optimality over the EVENODD Code in terms of computational and I/O efficiencies, both in the construction process during normal operation and in the recovery process after a failure.

5 Conclusion

This paper has successfully demonstrated the advantages of employing Row-Diagonal Parity codes within RAID6 systems, significantly enhancing data recovery speed and reducing computational complexity. The comparative analysis with established coding techniques like Reed-Solomon and EVENODD has underscored RDP's superior efficiency and performance in handling dual disk failures. By integrating RDP codes, RAID6 systems can achieve

heightened fault tolerance and operational reliability, crucial for managing the extensive data storage demands of modern enterprises and cloud-based architectures. Looking forward, the potential for further advancements in RAID6 data recovery is substantial. Future research could explore the integration of hardware acceleration technologies, such as GPUs, to further diminish recovery times and enhance throughput. Additionally, the application of artificial intelligence and machine learning could revolutionize fault tolerance mechanisms in RAID6, potentially leading to more adaptive and resilient storage solutions. Continual refinements in RDP algorithms and their implementation in broader system environments will likely yield more robust methodologies for data protection, shaping the future of storage technology in high-stake applications.

References

1. R. Wang, J. Zhu, S. Wang, T. Wang, J. Huang, X. Zhu, "Multi-modal emotion recognition using tensor decomposition fusion and self-supervised multi-tasking," International Journal of Multimedia Information Retrieval, **13(4)**, 39 (2024)
2. X. Zhu, C. Guo, H. Feng, Y. Huang, Y. Feng, X. Wang, R. Wang, "A Review of Key Technologies for Emotion Analysis Using Multimodal Information," Cognitive Computation, pp. 1-27 (2024)
3. H. Hou, Y.S. Han, K.W. Shum, H. Li, A unified form of EVENODD and RDP codes and their efficient decoding. IEEE Trans. Commun. **66,** 5053–5066 (2018)
4. J. Yang, W. Jia, Z. Gao, Z. Guo, Y. Zhou, Z. Pan, Cuckoo-Store Engine: A Reed–Solomon code-based ledger storage optimization scheme for blockchain-enabled IoT. Electronics **12,** 15 (2023)
5. Y. Huang, R. Wang, M. Ju, X. Zhu, Y. Xie, "Reconstructing global daily XCO2 at 1°× 1° spatial resolution from 2016 to 2019 with multisource satellite observation data," Journal of Applied Remote Sensing, **18(2),** 028502-028502 (2024)
6. S. Tang, S. Cai, X. Ma, A new chase-type soft-decision decoding algorithm for Reed-Solomon codes. Alexandria Eng. J. **61,** 13067–13077 (2022)
7. Y. Zhang, H. Xu, X. Zhu, Z. Zhao, J. Zuo, Detection and Quantization Technique of Optical Distributed Acoustic Coupling Based on φ-OTDR. J. Shanghai Jiaotong Univ. (Sci.) **25**, 208-213 (2020)
8. Y. Zhang, X. Wang, J. Wen, X. Zhu, "WiFi-based non-contact human presence detection technology," Scientific Reports, **14(1),** 3605 (2024)
9. X. Wei, X. Zhu, X. Wang, Z. Zhao, J. Zuo, "Fuzzy Fault Tree Analysis Method and Its Application in Fault Diagnosis of Denitration System in Thermal Power Plant," in Proceedings of the 2020 8th International Conference on Information Technology: IoT and Smart City, pp. 227-232 (2020)
10. H. Xu, X. Zhu, Z. Zhao, X. Wei, X. Wang, J. Zuo, "Research of Pipeline Leak Detection Technology and Application Prospect of Petrochemical Wharf," in 2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), vol. **9**, pp. 263-271, IEEE (2020)