A Method of Increasing the Integrity of Stored Information at the RAID 6 Level Using the Reed-Solomon Code and Combination of Syndrome and Probabilistic Decoding

Alina Lanina
Faculty of Computer Science and Technology
Saint Petersburg Electrotechnical University "LETI", 197022
St. Petersburg, Russia
lanina-20019@yandex.ru

Alla Levina
Faculty of Computer Science and Technology
Saint Petersburg Electrotechnical University "LETI", 197022
St. Petersburg, Russia
alla_levina@mail.ru

Abstract—**Currently, the volumes of information that need to be stored are growing. To store such data, specialized storage systems are used — they are capable of accommodating ever-increasing amounts of data. During storage and transmission, errors may occur, which can lead to data corruption or loss. The technology of adding redundant disks to a RAID disk array is widely known — it is used to subsequently recover lost information. This article proposes a method that combines the Reed-Solomon code (used at the RAID 6 level) with a decoding technique that is a combination of syndrome-based and probabilistic decoding methods.**

Keywords: coding theory, syndrome decoding, soft decoding, storage systems, redundant array of independent disks

I. INTRODUCTION

In modern conditions, large volumes of data are processed and stored. The information to be stored may be housed in data centers (DCs — data processing centers). According to statistics presented in the article Uptime Institute Data Center Outage Report 2025: Trends, Causes, and Recommendations, 2025 [1], which is based on data from the Annual Outage Analysis 2025 report by the Uptime Institute [2], the issue of outages is highly relevant. Specifically, 60 % of surveyed organizations reported experiencing outages caused by failures at their own facilities or with third-party service providers. Among the most frequently cited causes were network/connectivity issues, power supply problems, IT systems/software issues.

The financial damage caused to organizations by the outage and the wait for restoration of the data processing center's functionality was also assessed. In 2020, among the surveyed organizations: 60 % reported that losses amounted to less than $100 thousand; 28 % — between $100 thousand and $1 million; 12 % — more than $1 million. By 2025, the share of organizations with losses between $100 thousand and $1 million had risen to 53 %.

As follows from the statistical data presented above, preserving and enhancing the integrity of stored information is an urgent task. There are various methods and technologies capable of recovering data after equipment failures. One of them is RAID technology, which allows restoring information on failed disks through the use of error-correcting coding.

This article will present a method for combining Reed-Solomon codes and a technique for enhancing the integrity of information stored in storage systems (SS), proposed in [3] for the RAID 6 level.

II. RAID TECHNOLOGY

A. Redundant Arrays of Inexpensive Dicsks

Redundant disk arrays, known as RAID (Redundant Arrays of Inexpensive Disks), were introduced by researchers Peterson, Gibson, and Katz from the University of California, Berkeley, in 1987 [4].

The RAID technology distinguishes various levels. The key differences between them lie in the methods of data formation and placement, as well as in the algorithms for distributing data across disks. The RAID levels will be discussed in greater detail below.

B. RAID 5 u RAID 6 levels

RAID 5 level uses parity checksums and data striping. In RAID 5, parity checksums are distributed across the entire array, which improves write speed thanks to parallel operations. The system requires at least three disks to operate, and the storage space allocated for parity checksums equals the capacity of one disk (Figure 1).

Manuscript received April 20, 2026; revised May 26, 2026; accepted June 4, 2026. Published June 15, 2026.
Issue category: Regular
Paper category: Regular
DOI: 10.64552/wipiec.v12i1.131

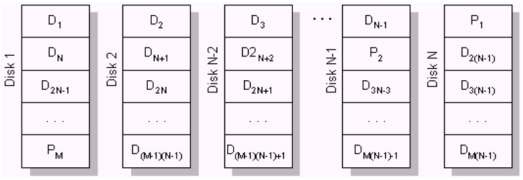

Figure 1 - RAID 5

- In RAID 6, information is divided at the block level in the same way as in RAID 5; however, this architecture incorporates a second, additional mechanism to improve fault tolerance. Reed-Solomon encoding is commonly used for this purpose. The resulting structure can withstand dual disk failures (Figure 2).

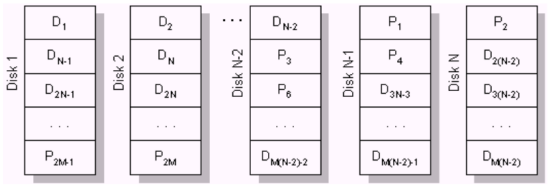

Figure 2 - RAID 6

A pressing issue for every RAID level is errors that occur when reading from or writing to a specific disk sector. Sectors that generate errors upon access are called bad blocks. The causes of their appearance may include external physical impact on the storage device, natural wear and tear of the disk surface, power surges or unstable power supply to the device, malfunctions in the software or hardware components during data transmission.

III. Error-Correction Codes

A. Linear Block Codes

Let there be a message consisting of *n* symbols, of the form $m = (m_1, m_2, \dots, m_n)$. The encoded message is a word $c = (c_1, c_2, \dots, c_n)$; such words are called codewords and together they form a code.

One of the encoding methods is linear coding [5]. In linear coding, the first part of the codeword coincides with the symbols of the original message (Formula 1):

$$x_1 = u_1, x_2 = u_2, \dots, x_k = u_k \quad (1)$$

The following are the $n-k$ check (parity) symbols: $x_{k+1}, \dots, x_n$. Codewords must satisfy Equation 2:

$$H \cdot \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} = Hx^T = 0 \quad (2)$$

where H is a parity-check matrix of size $\big((n-k) \times n\big)$, of the following form (Formula 3):

$$H = [A|I_{n-k}] \quad (3)$$

By the matrix H, we mean a fixed matrix consisting of 0 and 1, and I_{n-k} is the identity matrix of size $\big((n-k) \times (n-k)\big)$ — as given in Formula 4:

$$I_{n-k} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix} \quad (4)$$

B. Reed-Solomon Codes

These belong to linear block codes [6] and are also cyclic. They are specified as $RS(n,k)$ s −bit symbols, where: k is the number of information symbols of s bits each; n is the number of symbols in a codeword, obtained as the information symbols plus the added parity symbols. A Reed-Solomon decoder can correct up to *t* symbols containing errors in a codeword, where $2t = n - k$.

The decoder can correct any 16 erroneous symbols in a codeword — but no more than that. With a symbol size of s bits, the maximum codeword length is $n = 2^s - 1$. Thus, the maximum code length with 8-bit symbols is 255 bytes.

Reed-Solomon codes operate using Galois field arithmetic (GF), also known as finite field arithmetic.

To form a Reed-Solomon codeword, an external (generator) polynomial is used. Any valid codeword must be divisible by all factors of the generator polynomial. In general form, the generator polynomial is presented in Formula 5:

$$g(x) = (x - a^i)(x - a^{i+1}) \dots (x - a^{i+2t}) \quad (5)$$

The codeword is formed using the operation $c(x) = g(x)i(x)$ where: $g(x)$ is the generator polynomial; $i(x)$ is the information block (message polynomial); $c(x)$ is the codeword (an element of the field).

Encoder architecture

To construct a codeword, $2t$ parity symbols are required.

There are two possible encoding variants: systematic and nonsystematic. In nonsystematic encoding, the information vector $i(x)$ is multiplied by the generator polynomial vector $g(x)$ as shown in Formula 6:

$$c(x) = i(x)g(x) \quad (6)$$

In systematic encoding, the $2t$ parity symbols are found as the remainder of dividing the information vector $i(x)$ by the generator polynomial $g(x)$ as shown in Formula 7:

$$R(x) = \frac{i(x)}{g(x)} mod\ g(x) \quad (7)$$

Then, the information vector is shifted by a certain number of positions toward the higher-order bits (multiplication by x^{n-k}), and the computed remainder is appended to the right as shown in Formula 8:

$$c(x) = i(x) \cdot x^{n-k} + R(x) \quad (8)$$

Each of the 6 registers holds a symbol (8 bits). Arithmetic operators perform addition or multiplication on the symbol as an element of a finite field.

IV. Method of Increasing Integrity

Data integrity is enhanced by encoding the stored information using a Hamming code. Subsequently, decoding is performed using a combination of syndrome decoding and soft decoding methods [3, 7, 8]. The decoding device decides on the decoding algorithm for each input word: syndrome decoding is applied if the word contains up to *t* errors; soft decoding is used if there are more than *t* errors. This decision-making process provides the following benefits: an improvement in the number of correctable errors compared to using syndrome decoding alone; a gain in decoding time and resource consumption

compared to using soft decoding alone, since probabilistic symbols do not need to be processed for all words.

Figure 3 shows the block diagram of the proposed data integrity increasing method:

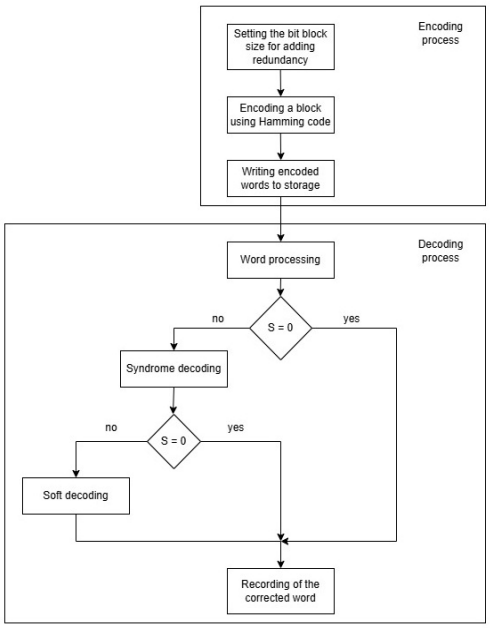

Figure 3 – Block diagram of the data integrity increasing method

The process of applying checksum methods for integrity control of information stored using RAID 5 technology was also simulated. To simulate checksum [9] and proposed method a simulation model was created. Information words were recorded in files simulating disks, the information in which is distributed using RAID technology.

Characteristics of the computer on which the process was modeled: processor: Intel(R) Core(TM) i5-82500; RAM: 8.00 GB; x64 processor; Windows 10 operating system; Python programming language 3.10.4; Visual Studio Code Development Environment; Software Package Management System (pip) written in Python;

The information storage process was modeled using three data stripes and one parity stripe providing parity for all three data stripes. For the CRC32 case, checksum values computed using the CRC32 algorithm were stored separately (one checksum per data block). In the proposed method, words in both the data and parity stripes were additionally encoded using a Hamming (12,8) code with the addition of redundant symbols, resulting in stored codewords instead of raw data. Each codeword was processed individually, with the number of injected errors (ranging from 0 to 4) and their positions randomly determined. When errors were introduced into a single stripe, recovery was performed via the standard RAID array mechanism; however, if errors occurred in more than one stripe, the RAID mechanism alone proved insufficient, necessitating decoding via a combination of syndrome decoding and soft decoding. In contrast, for the CRC32 case, recovery was not possible when errors affected more than one stripe.The simulation results are presented in Table 1.

TABLE 1. SIMULATION RESULTS FOR CHECKSUM METHODS AND THE PROPOSED METHOD

	crc2 code	Proposed method
	Errors in 1 strip	
Detection of a strip with errors	+	+
Number of errors introduced	2758	
Number of errors corrected	2758/2758	2758/2758
Corrected errors, %	100%	100%
	Errors in 2 strips	
Detection of a strip with errors	+	+
Number of errors introduced	5444	
Number of errors corrected	-	1682/5444
Corrected errors, %	-	30,89%
	Errors in 3 strips	
Detection of a strip with errors	+	+
Number of errors introduced	8094	
Number of errors corrected	-	2524/8094
Corrected errors, %	-	31,18%

The simulation results showed that the proposed method enables full data recovery in case of errors in a single stripe (using redundancy information stored on the additional stripe), with similar results achieved when using checksum methods. However, if errors are introduced into more stripes than the number of redundant stripes, checksum methods fail to recover the data. In contrast, the proposed method can recover: 1682 out of 5444 errors (for 2 erroneous stripes); 2524 out of 8094 errors (for 3 erroneous stripes).

V. EVENODD SCHEME

The EVENODD algorithm [10] provides an efficient encoding procedure based on exclusive-OR (XOR) operations and independent relationships. It also presents a simple decoding procedure for two erasures as well as for a single error.

Encoding procedure (equation 9):

$$S = \bigoplus_{t=1}^{m-1} a_{m-1-t,t} \tag{9}$$

Then, for each l, where $0 \le l \le m-2$, the redundant symbols are obtained as follows (equations 10, 11):

$$a_{l,m} = \bigoplus_{t=0}^{m-1} a_{l,t} \tag{10}$$

$$a_{l,m+1} = S \oplus \left(\bigoplus_{t=0}^{m-1} a_{<l-t>m,t} \right) \tag{11}$$

Equations (10) and (11) define the encoding. We have two types of redundancy: horizontal redundancy and diagonal redundancy. Disk m is simply a parity disk containing the XOR result of disks $0,1,\dots,m-1$. Disk $(m+1)$ contains the

diagonal redundancy as defined by equation (11). We see that there are two possible diagonal redundancies: the parity can be even or odd. This even or odd parity is determined by the bit S in equation (9), which defines the parity of the diagonal $(m-2,1),(m-3,2),\dots,(0,m-1)$. If this diagonal has an EVEN number of 1, then we have even parity in the remaining diagonals. Otherwise, we have ODD parity. It is for this reason that we call this scheme the EVENODD (EVEN–ODD) scheme.

Two-erasure decoding algorithm

Consider an array of symbols a_{ij} with dimensions $(m-1)\times(m+2)$, such that the last two columns are redundant according to equations (9), (10) and (11). If a failure occurs in one column (disk), say in column (disk) i, where $i \neq m+1$, then it can be reconstructed using the exclusive-OR (XOR) operation across the remaining columns (disks).

If column $(m+1)$ is faulty, then the symbols can be restored using equations (9) and (11).

Next, suppose that columns (disks) i and j have failed, where $0 \le i < j \le m+1$. We have four cases:

- $i = m$ **and** $j = m+1$, i.e., both redundant disks have failed. We can reconstruct disk m using equation (10), and disk $(m+1)$ using equations (9) and (11). In other words, the reconstruction is equivalent to encoding.
- $i < m$ **and** $j = m$, namely, one data disk and one redundant disk have failed. We can reconstruct disk i as follows (equation 12):

$$S = a_{<i-1>_m,m+1} \oplus \left(\bigoplus_{l=0}^{m-1} a_{<i-l-1>_m,l} \right) \qquad (12)$$

where we suppose that $a_{m-1,1} = 0$ for $0 \le l \le m+1$. Then (equation 13):

$$a_{k,i} = S \oplus a_{<i-1>_m,m+1} \oplus \left(\bigoplus_{l=0, l\neq i}^{m-1} a_{<k+i-l>_m,l} \right), \quad 0 \le k \le m-2 \qquad (13)$$

and $a_{k,m}, 0 \le k \le m-2$ it is obtained using equation (10) after reconstructing disk i.

- $i < m$ **and** $j = m+1$, namely, one redundant disk and one data disk have failed. We can reconstruct disk i using equation (10), and disk $(m+1)$ using equations (9) and (11), once disk i has been reconstructed.
- $i < m$ **and** $j < m$. This is the main case. Both failed disks contain data, and we cannot reconstruct them using the individual relations as in the previous three cases. This case is analyzed in detail in [10].

VI. COMBINATION OF REED-SOLOMON AND METHOD OF INCREASING THE INTEGRITY AND COMPARISON WITH EVENODD

A. Combination of Reed-Solomon and method of increasing the integrity

We will describe the encoding and recovery processes for a combination of Reed-Solomon code and an integrity enhancement method.

Encoding process:

1. Computation of checksums using the XOR operation.
2. Computation of Reed-Solomon checksums.
3. Determination of the block size (in bits) for adding redundancy.
4. Encoding of data blocks using a Hamming code.
5. Writing the encoded information words to the data drives.
6. Writing the encoded checksums to the parity drives.

Recovery:

If one drive fails:

1. If it is a parity drive, recovery is performed using the data drives — similar to recalculating parity.
2. If it is a data drive, recovery is done using XOR parity.

If two drives fail:

1. If both are parity drives, recovery is performed using the data drives — similar to recalculating both parity values.
2. If one is a parity drive and one is a data drive: recover the data drive using available parity; recover the second parity drive by recalculating it from all data drives.
3. If both are data drives, recovery is performed using the two available parity drives.

If more than two drives fail, recovery of the codewords encoded with a Hamming code is performed using a combination of syndrome decoding and probabilistic (soft) decoding methods.

B. EVENODD Scheme

We will describe the encoding and decoding processes for the EVENODD method.

Encoding:

1. Computation of checksums using the XOR operation.
2. Computation of checksums using diagonal parity (diagonal redundancy).
3. Writing information symbols to the data drives.
4. Writing parity symbols to the parity drives.

Recovery:

If one drive fails:

1. If it is a parity drive, recovery is performed using the data drives — similar to recalculating parity.
2. If it is a data drive, recovery is done using XOR parity.

If two drives fail:

1. If both are parity drives, recovery is performed using the data drives — similar to recalculating both parity values.
2. If one is a parity drive and one is a data drive: recover the data drive using available parity; recover the second parity by recalculating it from all data drives.
3. If both are data drives, recovery is performed using the two available parity drives (for more details, see [10]).

Next, Figure 4 shows a block diagram of the combination of the Reed-Solomon code and the integrity increasing method for the RAID 6 architecture.

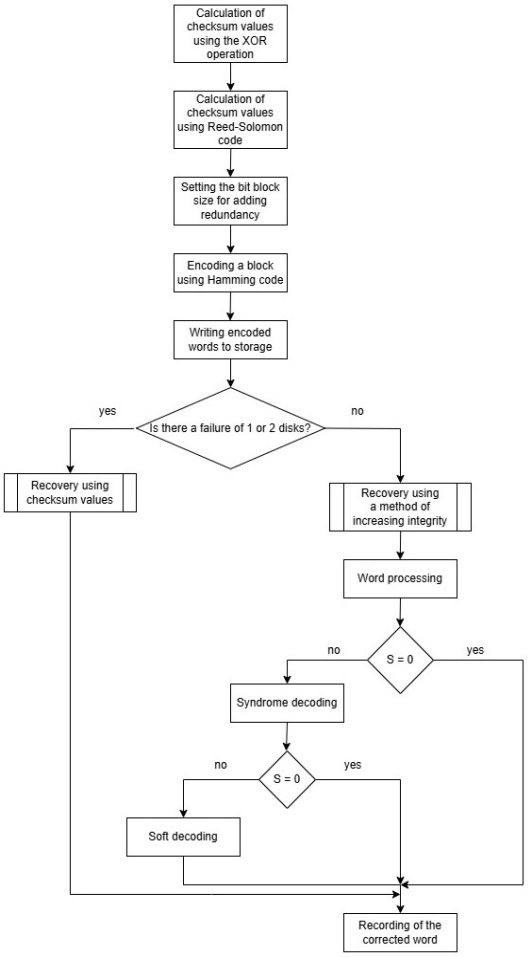

Figure 4 – Block diagram of the Reed-Solomon code and integrity increasing method combination for RAID 6

Next, the table provides a comparison of the capabilities of the proposed combination of the Reed-Solomon code with the integrity enhancement method and the EVENODD method. The analysis focused on scenarios where drives did not fail completely, but errors occurred and needed to be detected. Cases with errors in 1, 2, and 3 stripes were considered. Comparison results are presented in Table 2.

TABLE 2. RESULTS OF THE COMPARISON BETWEEN THE REED-SOLOMON CODE AND INTEGRITY INCREASING METHOD COMBINATION AND THE EVENODD ALGORITHM FOR ERRORS IN MULTIPLE STRIPES

	Combination of the Reed-Solomon algorithm and method of increasing integrity	EVENODD
	Errors in 1 strip	
Detection of a disk with errors	+	+
Error correction	+	+
	Errors in 2 strips	
Detection of a disk with errors	+	-
Error correction	+	-
	Errors in 3 strips	
Detection of a disk with errors	+	-
Error correction	+	-

As the comparison shows, EVENODD — described as an algorithm that corrects a single error — is outperformed by the proposed combination in cases where errors occur in 2 or more stripes.

ACKNOWLEDGMENT

This research was funded by the Ministry of Science and Higher Education of the Russian Science Foundation (Project “Goszadanie” No.075-00003-24-02, FSEE-2024-0003.

REFERENCES

[1] Uptime Institute Data Center Outage Report 2025: Trends, Causes, and Recommendations, 2025. [Online]. Available: https://telecombloger.ru/600923?ysclid=mnxplfv2zv991216637

[2] Uptime Institute, Annual Outage Analysis 2025, 2025. [Online]. Available: https://uptimeinstitute.com/resources/research-and-reports/annual-outage-analysis-2025

[3] Lanina, Alina A.; Levina, Alla B. A method of increasing the integrity of information stored in data storage systems by combining syndrome and probabilistic decoding. IT Security (Russia), [S.l.], v. 32, no. 3, p. 44–56, 2025. ISSN 2074-7136. URL: https://bit.spels.ru/index.php/bit/article/view/1814. DOI: http://dx.doi.org/10.26583/bit.2025.3.04.

[4] David A Peterson, Garth Gibson, Randy H Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID) – 1988

[5] MacWilliams, F. J., & Sloane, N. J. A. (1977). The theory of error-correcting codes (North-Holland Mathematical Library). Amsterdam: North-Holland Publishing Co. ISBN 0-444-85193-3

[6] B. D. Kudryashov, Osnovy teorii kodirovaniya: uchebnoe posobie. Saint Petersburg: BKhV-Peterburg, 2016.

[7] Plotnikov A. I., Levina A. B., Lanina A. A., Zikratov I. A. (2024). Comparison of Methods Syndrome and SoftDecoding for Hamming Code. Vestnik komp'yuternyh i informatsionnyh tekhnologiy, 21(11), 46 – 53. [in Russian lan-guage]. DOI: 10.14489/vkit.2024.11. pp.046-053

[8] V. V. Kvashennikov, Method for soft decoding of error-correcting code, Russian Federation Patent 2738724, Dec. 16, 2020.

[9] B. Kroth and S. Yang, CheckSumming RAID, 2013. [Online]. Available: https://gradebuddy.com/doc/233998/checksumming-raid/

[10] 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.