Improving the Security of Split Data When Using Multidimensional Parity Algorithms

https://doi.org/10.3991/ijim.v16i21.36075

Tlek Akhmetgalym(✉), Gulzira Mukasheva, Karipzhanova Ardak Zhumagazievna, Aukenov Bolat Mayzhanuly, Urazbaeva Kumys Toleubekovna
Alikhan Bokeikhan University, Semey, Kazakhstan
tlek.akhmetgalym@mail.ru

Abstract—A model of distributed storage split data using algorithms, multidimensional parity, resistance to partial losses of the storage sites, is regarded as an alternative way of security that can replace the conventional multiple reservations and carrying costs of growth of the physical volume. In this model, the generated redundant data allows you to restore partial loss of the split parts. In this case, the recovery is performed at the expense of the parity files formed during the splitting procedure, in which the main action is the calculation of bitwise parity with summation modulo "two". With this additional operation between the source files, you can restore the corrupted file without distortion. A comparison with several models that use checksum codes for recovery shows that the estimated probability of losses due to storage failure in the case of multidimensional parity codes is significantly less than when using other split data storage options. Using the operation of adding the bits of undamaged files, you can completely restore corrupted files. According to the calculated data, even when theoretically non-recoverable losses are reached, it is always possible to find combinations of chain recovery. The complexity of alternative methods, including the iterative decoding method, makes a distributed split data storage system using multidimensional parity algorithms a more convenient error correction technology.

Keywords—security, distributed storage, data splitting, multidimensional parity, loss recovery, parity files

1 Introduction

The search for new methods of ensuring IT security is ongoing, but so far, replication (multiple redundancies) is widely used to save data [1] [2]. The constant increase in the volume of information leads to the fact that replication turns into an increase in hardware, energy and as a result, financial costs [3]. You can try to avoid expensive backup storage by performing operational information recovery (error correction). In particular, the authors of this article tested the possibility of storing files in distributed databases in a split state, when splitting files and restoring them is carried out using proprietary multidimensional parity algorithms that are resistant to a partial loss of

storage locations [4-6] (patents: GB2467989 «Distributed Storage», UK; GB2463078 «Distributed Storage», UK; GB2463085 «Communication System», UK; GB2463087 «Data Storage», UK; GB2492981 «Data Storage», UK; HO1175001 «Data Storage», Hong Kong; US9026844 «Distributed Data Storage and Communication», USA).

The frontend of the system is implemented on web technologies and uses a simple hypertext link protocol HTTP in conjunction with interactive technologies on AJAX/PHP (WEB 2.0). The basic content management system of the web part is implemented on CMS Word Press. The site is located on the dedicated server of PS Internet Company LLP, under the Linux Ubuntu Server operating system and the Apache 2.25 c PHP 7.0 web server. MySQL 5.0 is used as the database [7].

Due to proprietary algorithms, redundant data is generated that allows you to recover partial losses of split parts [8][9]. The split data does not carry meaningful content, and therefore it can be stored in any available places without fear of unauthorized access. Even after collecting all the parts of the split data, it is impossible to recover the information without knowing the splitting method. The splitting algorithm can use an infinite number of splitting methods, and therefore only the owner/creator of the information has access to the data [10][11]. At the same time, in the case of partial data loss, their recovery is performed at the expense of parity files formed during the splitting procedure, in which the main action is the calculation of bitwise parity with summa-tion modulo "two". Using this addition operation between the source files, you can restore a damaged (lost) file without distortion.

2 Methods

Parity algorithms operate on binary information and the summation that we use when splitting the data is reduced to the definition of parity [12]. Moreover, the summation is carried out strictly positionally ("bitwise") and modulo. It turns out that the third file in which to store the bits are summarized. The summation is performed modulo "two", namely: the mathematical operation "exclusive OR" is included. If, for example, the first file has zero and the second file has zero, then the third file will have zero. If the first one is zero, and the second one is one, then the third one will be one. If the first one is one, and the second one is zero, then the third one will also be one. If both there and here are one, then in the third file it turns out to be zero. This third file is the parity file, and the whole operation is a parity operation: two files make three. If you delete any of these files, then by performing the addition operation between the remaining two, you can always restore the third file without distortion.

This idea, based on the properties of parity, is the basis for solving problems of distributed information storage with data splitting using multidimensional parity algorithms [13]. So far, the parity property has only been used to detect file corruption and determine what information is lost, nothing more (RAID). In such systems, the information is divided into identical blocks, then the parity is calculated, which is written on a separate file containing the parity bits. When, for example, the parity bit is lost when transmitting data, the parity bit will not match when adding data. However, this result in RAID is only needed to delete the file if it is found to be corrupted. In case

of data loss during transmission, a second request is made. The lost file is then restored from the backup locations (replication method).

Meanwhile, in the method of multidimensional parity that we tested; the lost information is restored with a calculated probability. As mentioned above, if you divide a file (data carrier) into two identical parts, calculate the bitwise parity, and write the result to a third file, you can always recover the loss (or damage, if you know which file is damaged) of any of the three files.

Consider, for example, the case of three files, when one of them is lost. Bitwise, we add the remaining two files modulo. The resulting parity file is exactly equal to the lost file. This is the case of one-dimensional parity.

If we now divide the file into four identical parts, arrange them in the form of a square, and calculate the bitwise parity in rows and columns, we get 4+4+1=9 files. This is the case of two-dimensional parity.

In the case of two-dimensional parity, it is obvious that we can lose any three files out of nine, which is equivalent to losing files at one of the coordinates, but then restore all the original information. To show this, consider for simplicity the case of writing one bit at a time (the information is stored on 9 disks).

The result of the summation is shown in Table 1.

Table 1. The result of the parity summation for 9 disks

1st disk	1	2nd disk	0	**7th disk, pari the 1st line**	**1**
3rd disk	0	4th disk	1	**8th disk, pari the 2nd line**	**1**
5th disk, parity, an column	**1**	**6th disk, parity o 2nd column**	**1**	**9th disk, par parity**	**0**

In such a system with two-dimensional parity, 3 disks out of 9 can fail at the same time, and the data can always be restored since bits can be restored at two coordinates. And this is equivalent to a 3-fold copy during replication. Moreover, it is possible to lose even 4 disks out of 9, and it will still be possible to recover the information with a 92% probability. You can lose even up to 5 disks and recover with a 67% probability [14].

In a system with two-dimensional parity, of course, in reality, more disks and a large amount of data are needed, because in this version there should be more parity data. But, in a two-dimensional parity, the redundancy is 2.25 (data is written to 4 disks and additionally to 5 disks: parity 9/4=2.25).

If we compare it with replication, the two-dimensional parity redundancy is almost the same as with double replication with 2.0 redundancy, when two copies are written. And in a system with two-dimensional parity, you can lose from 3 to 5 disks, i.e., reliability-as with 3-5-fold replication.

3 Related work

Today, the standard way to ensure security is the method of multiple replications. Besides, to ensure the security of information, methods based on the Reed-Solomon algorithm [15], fountain technologies [16], as well as using checksums in the RAID5/6 system, in codes with local parity LRC [17] were tested.

Within the framework of the model of correcting codes, the theory of channels with erasures were developed, which was proposed by V. Peterson [18]. The model of channels with erasures was subsequently developed in the work of M. Labi [19].

3.1 Method of replication

The most widely used correction algorithms have found simple and fast algorithms for parity. In particular, they are used with different efficiency in common RAID systems of fault-tolerant disk arrays for servers [20]. But, on the other hand, high-speed and simple parity algorithms in conventional applications do not allow you to recover multiple data losses. For example, a common variant of RAID5, when a single disk fails, goes into the rebuild state, in which intensive reading begins from all the disks in the array to restore the data of the failed disk. If the disk sizes are on the order of several gigabytes, this does not cause problems, then with an increase in the volume of disks to terabytes, the process can take a significant time, during which any failure of another disk leads to a complete loss of all data.

Problems with the reliability of standard RAID arrays force the use of more complex RAID configurations with a combination of different schemes, such as RAID6, to apply additional mirroring [21]. Commercial proprietary RAID variants are known, such as HP EVA with its vRAID technology, RAID m + n using Reed-Solomon erasure codes. But all this leads to an even greater increase in the cost of the technology.Therefore, RAID arrays often allocate a single disk specifically for writing parity, which can be used to restore data from a broken disk.

To explain the meaning of parity recovery, consider, for example, the case of a RAID of 5 disks. For simplicity, let's write four bytes to four disks: 1, 2, 3, and 4. The result of the parity summation is written to the 5th disk (Table 2).

Table 2. The result of the parity summation in the RAID system

1 disk	1	1	1	0	1	0	1	0
2 disk	0	0	1	0	1	1	1	0
3 disk	0	1	0	1	1	1	0	1
4 disk	1	1	1	0	0	1	0	1
5 disk – parity	**0**	**1**	**1**	**1**	**1**	**1**	**0**	**0**

The sum of the parity calculation is made by the "exclusive OR" of the bits between the different drives. It is also called "addition modulo 2". At the same time 0^0=0, 0^1=1, 1^0=1, 1^1=0. The method is called parity summation because the result must always be an even number of bits along with the calculated one.

The table shows that if an even number of units is added, the calculated parity bit must be zero: the parity does not change. For example, in the first column of bits: 1 disk = 1; 2 disk = 0; 3 disk = 0; 4 disk = 1. Two units, 0 is written to the 5th disk. If the number of bits is odd, then the parity bit will be 1, thus turning the total number of ones into an even number. For example, in the second column: 1 disk = 1; 2 disk = 0; 3 disk = 1; 4 disk = 1. Three units, 1 is written to the 5th disk. Then we get 4 units, i.e. we "achieve" an even number of 4. If any of the five disks now break, you can always restore the missing data by writing "1" to the new disk to even if the number of units on the remaining 4 disks is odd, and writing " 0 " if the number of units on the remaining 4 disks is even.

But still, in modern RAID arrays, they prefer to refuse to allocate a disk for writing parity [20] Why? In RAID systems that use a parity disk, you can recover data if only one disk in the array fails. The disk is simply changed to a new one, and data recovery begins. This process is called rebuild, and data is read simultaneously from all the remaining disks to calculate the parity.

If another disk fails during the rebuild, then all the data can be lost completely, since it is no longer possible to restore anything [22]. The longer the rebuild process takes, the higher the risk of double failure. At current volumes, when they reach terabytes or more, this can last for several days. There may be a situation with the presence of the so-called "rotten" bit (bad bit), which is one of the factory parameters of hard drives. In this case, which manifests itself precisely with large amounts of storage, the risk of data loss in RAID becomes unacceptably high.

Therefore, in modern multi-disk Big Data storage systems, RAID arrays are abandoned, and the only de facto method of data protection is multiple copying (replication). Moreover, replication occurs not only between nodes but also within nodes. Even inside the DATA center, you have to do replication from one server to another [22].

3.2 Correction codes with a checksum

The most common and simplest method of recovering losses is to use checksum codes [12]. Let there be a data set D = {d0, d1, … dn}, divided into blocks of the same size. If we know the amount of CD = d0 + d1 +… di …+ dn, then any loss of the block di you can restore it by recalculating the amount of the remaining data C'D, the difference of which with the sum file will be the missing block di = CD – C'D. When adding, the summation operation is used modulo, since for control, you only need to know the differences in the data blocks, i.e. in general, the amount of data required for controlling the sums does not exceed the block size. In binary logic, the calculation of checksums is reduced to summation modulo 2, which is implemented by a single operation that excludes OR (XOR) and turns into parity control (we use the term "parity control" since we describe technologies related to the processing of binary data).

When applied to multi-disk systems, this technology was first implemented in RAID systems [22]. In this technology (in particular, in RAID 3/4/5/6), data is divided into blocks, parity is calculated, and data blocks with a parity block are scattered across different disks in the array. Such an array can recover single disk failures. When storing

large amounts of data, RAID cannot provide an acceptable level of reliability. Recovery after a failure (recovery of data from a failed disk and writing it to a healthy new disk) with increased volumes begin to take an unacceptably long time since it is necessary to read data from all disks for calculations [13]. In the process of recovery (Rebuild), the storage system is not protected from repeated failures, and in the event of a failure of another disk, all data is lost irretrievably.

On the other hand, RAID arrays are very expensive devices, because the main area of their application, for which everything is optimized – is high-speed devices for online storage and data exchange [23]. The device's RAID controller can simultaneously read and write across all disks, increasing the overall speed of data exchange. For high-capacity storage, specialized systems are being developed that use relatively slow but high-capacity disks, combined with distributed storage technology that is resistant to disk failures. Such storage systems are also called RAID, although they do not use "proprietary" parity technologies. There are various designations of multi-disk storage technologies, both open and proprietary, for example, RAID 7.3 or RAID m+n, which have nothing to do with traditional RAID systems and refer us to the original meaning of the abbreviation RAID-Redundant Array of Inexpensive Disks – - a redundant array of inexpensive disks [24].

3.3 Application of Reed-Solomon codes

The high redundancy of replication, the high cost and low reliability of RAID systems make it necessary to develop storage technologies using more complex methods. Various systems are known, usually referred to as RAID n+m with the Reed-Solomon noise-tolerant Data encoding (RS) algorithm. The most well–known is RAID 7.3 from RAIDIX of St. Petersburg, a storage system from IBM based on the developments of Cleversafe, purchased by IBM.

The RS algorithm allows you to set the required fault tolerance in the form of the number of blocks whose loss can be restored. The PC can detect damaged or lost blocks and calculate the location of blocks that need to be restored. In theory, the algorithm can detect t errors and recover information using redundant data. The total number of blocks will then be n+2t, where n is the number of data blocks.

The PC method most economically increases the total amount of stored data; the redundancy is almost close to the theoretical limit [25][26]. However, the algorithm is resource-intensive, because it uses complex calculations, and intensive information exchange with the storage system, it can greatly increase the latency of the system. Also, the algorithm does not scale well, any increase in storage locations requires a complete recalculation of the data of the entire array since the PC has a rigid algebraic structure. In the modern multi-drive storage with a high failure rate of a high risk of data loss at the slightest accidental exceeding of the threshold of failure, because fault tolerance is RS threshold character – the slightest excess leads to complete failure of the entire repository. Another feature of the algorithm, when applied to distributed storage, that causes big problems is that the PC requires data from all the disks in the array for calculations, and spikes in traffic in the storage network will greatly affect the availability of data.

There are various methods of noise-tolerant coding, called multidimensional error correction codes, used in noise-tolerant coding [27]. We have developed a simple, non-resource-intensive algorithm with reliability comparable to that of a PC, but free from its disadvantages. The redundancy in this case, although higher than that of the PC, is much less than when using replications [14][28].

4 Results

4.1 One-dimensional parity

We divide the data into two blocks of equal size and calculate the parity:

	(3.1)

which denotes the operation XOR, implementing bitwise addition modulo 2?

Note that, based on the properties of the XOR operation, the resulting data blocks have the property of restoring any missing part. We show that, for example,

Perform the operation with both sides of equality. Given the involution of the operation, we reduce, and we get the correct equality (3.1).

	(3.2)

Similarly, we prove the equality using the operation:

	(3.3)

From here we have a triple of relations, from which we can recover the loss of any data block from the triple:

	(3.4)

The resulting data triple can be represented as a one-dimensional vector at the x coordinate, the data which is related by parity relations:

	(3.5)

where calculated modulo 3. This means that in the parity formula (3.5) 0–1=3 and 3+1=0.

4.2 Two-dimensional parity

Let us split the original data into four blocks of the same size. Let us write the blocks in a two-dimensional 2x2 matrix. Convert the matrix to 3x3 by adding an empty column

to the right and an empty row to the bottom of the original matrix. We will calculate the checksums for the rows and columns and write the results in empty spaces.

	(3.6)

The resulting data matrix now allows you to recover any data loss in two ways-both by columns and by rows. The resulting matrix demonstrates the two-dimensional parity of the Multidimensional parity-check code [24].

Let us complete the matrix by calculating the parity values of the parity values. We get two equalities – and, where, recall, denotes the operation XOR, implementing bitwise addition modulo 2. We prove that. If you expand the parity values and take into account, the commutativity of the XOR operation:

	(3.7)

we get.

The resulting 3x3 two-dimensional matrix will be a matrix with all the data in the columns and rows connected by taking an XOR operation between adjacent data pairs. In this case, each data block can be obtained in two ways – by XOR operation, both by rows and columns:

	(3.8)

where the index values are calculated modulo 3, i.e., they are the residue ring. These relations can be interpreted as two-dimensional parity in the x, y coordinates corresponding to one-dimensional vectors of rows and columns.

4.3 Three-dimensional parity

According to the described algorithm, you can get a three-dimensional matrix by splitting the data into 8 blocks of the same size and generating parity data already on three coordinates:

	(3.9)

with data blocks that are already linked by three relationships:

	(3.10)

If the two-dimensional matrix described above consists of one-dimensional (3.5), then the three-dimensional matrix, respectively, consists of sets of three planes of two-dimensional matrices (3.9) arranged sequentially under each other. The third matrix-the plane with the coordinate z=2-is formed from the calculated data of the parity of the upper two planes. In general, a data block that has a value of any index equal to 2 is a parity block.

Thus, we get a 2x2x2 cube with the original data, the block indexes of which have the values 0 or 1, and three boundary planes on which the parity data is located, one of the index values of which is 2.

4.4 N-dimensional parity

The described model of data splitting into blocks and the method of generating parity blocks can be extended to any *n*-dimensional parity. In this case, the original data will be split into 2^n blocks, after processing by the parity generation algorithm, we will get 3^n blocks, where n is the parity dimension. The resulting blocks will be linked by parity relations:

	(3.11)

Basic properties of multidimensional parity:

1. As the parity dimension increases, so does the number of ways to recover lost data blocks.

In *n*-dimensional space, the main property of a one-dimensional parity vector is that restoring the loss of one of the three blocks with the help of the remaining two turns into cross-parity control in each of the *n* directions.

2. With increasing dimensionality, the options for not only cross-parity control but also additional options for chain recovery are growing simultaneously.

With the growth of the dimension, additional recovery options appear, when the "missing" data for any of the coordinates (in the case of a conditionally fatal loss for this file) can, in turn, be restored using other files, with subsequent recovery, at the next step, of the required file.

4.5 Example of chained recovery

Suppose that 5 blocks of data are lost in a two-dimensional matrix. For a file this is a conditionally fatal case since it is impossible to restore it from any of the coordinates:

	(3.12)
But there are two options for chained recovery: *or* -in four steps	(3.13)

The presence of chained recovery options means that multidimensional parity codes do not have a threshold for the probability of data recovery. That is, even when theoretically non-recoverable losses are reached, there is always a chance to find combinations of chain recovery.

In a two-dimensional parity matrix of 9 files, each file is linked by parity relations to two neighboring files and can be restored using them. This means that:

a) losses from 1 to 3 of any file storage locations are always guaranteed to be recovered.
b) losses from 4 to 5 places have chain recovery options, so there is a non-zero probability of recovery.
c) of course, the fatal scenario of file loss is the loss of 6 files out of 9.

4.6 Combinations of failures, storage locations, resulting in a loss of data

Denote the fatal combination of DL^{kn} losses, when the original file cannot be restored when k storage locations fail out of n. For example, for a two-dimensional matrix, if you lose 4 storage locations out of 9, the list of fatal combinations will look like this:

	(3.14)

The total number of possible combinations of k places from n is equal to the combinations of n by k [27]:

	(3.15)

The loss of four blocks has no chain recovery options if the coordinates of the failed storage locations are located at the vertices of a two-dimensional matrix in such a way that paired losses are formed over the intersecting coordinates of the files.

For each combination of 2 out of 3, there will be 2 combinations of 2 out of 3. Total combinations: $DL^{4,9} = 9$.

The loss of 5 blocks has no chain recovery options if the coordinates of the storage locations are arranged in such a way that any four of them form the combination described above.

Given that there are five free positions, we get that there is a total of combinations: $DL^{5,9} = DL^{4,9} \cdot 5 = 45$.

We calculate the combinatorial probability of losses in the event of storage failure.

Suppose that 4 disks out of 9 randomly fail. What is the probability that we will get a combination of $DL^{4,9}$, leading to data loss?

$Q^{4,9}$probability of data loss when four storage locations fail:	(3.18)

Now let us assume that 5 disks randomly fail. What is the probability that we will get a combination of $DL^{5,9}$ leading to data loss?

The probability of $Q^{4,9}$ data loss when five storage locations fail:	(3.19)

The probability of $Q^{6,9}$, $Q^{7,9}$, $Q^{8,9}$, $Q^{9,9}$ data loss with a loss of 6 storage locations and higher is 1.0. That is, such storage losses cannot be restored, and they are fatal.

The reliability of multi-disk systems decreases with the increase in the number of disks in the array. Using a statistical model of the probability of disk failures, it can be stated that the frequency of disk failure λ in an array of n disks increases almost proportionally by n times [14]. The probability of a Q_{dev} disk failure during time t will be:	(3.20)

For an array of n disks, the probability of a Q^n_{arr} disk failure will be:

	(3.21)

The parameter λ is responsible for the factory characteristic of the disk quality MTBF (Mean time between failures) – the conditional number of hours of disk operation before the first failure.

In this case, we are interested in the probability of simultaneous failure disks.

There will be such failures in an array of n disks, but only a fatal scenario will lead to the failure of the storage system $DL^{k,\,n.}$ This means that we have a joint probability of failure of k disks out of n with the probability of $Q^{k,\,an}$ occurrence of a fatal scenario. Considering the probability of failure of k disks as both random and independent events, we get the probability of failure of the storage system.

	(3.22)

4.7 Probability of failure of a two-dimensional parity storage system

Here is an approximate calculation of the probability of failure of a storage system with two-dimensional parity, using typical hard drives with MTBF of ≈ 800 thousand hours (Mean Time Between failures or time between failures). The probability of a disk failure within a year with MTBF will be:

	(3.23)

Where λ = 1/MTBF, and t = 1 year (8760 hours). Disks with this MTBF have a 1.09% chance of failure within a year.

The probability of failure of any disk within a year in an array of 9 disks, respectively, will already be 9.38%:

	(3.24)

The probability of storage failure when 4 disks fail, i.e., getting a fatal scenario will be 0.00055273%:

	(3.25)

The probability of a storage failure with a failure of 5 disks will be 0,0002593%:

	(3.26)

5 Discussion

The redundancy of a storage system that uses multidimensional parity technology to increase the reliability of storage in multi-disk arrays is determined by the ratio of the original files to the total number of files, along with the redundancy [14]. As mentioned in the discussion of the properties of multidimensional parity, it is equal to the ratio R = $(3/2)^n$, where n is the parity dimension.

For storage systems with two-dimensional parity, R = 2.25, which is almost equivalent to double replication or mirroring. The probability of failure of a storage system with mirroring to two disks will be equal to the probability of failure of two disks at the same time. Given the formula (3.21) for the probability of disk failure in the array and considering the failure events to be independent and joint, we get the probability of failure 2.166%:

	(3.28)

In storage systems with two-dimensional parity technology, with comparable redundancy, we get failure probabilities from 0.00055273% to 0.0002593%.

Thus, the estimated probability of losses in the event of storage failure when using multidimensional parity codes is significantly less than when using other storage options[29]. In principle, multidimensional parity codes do not have a threshold value for the probability of data recovery. That is, even when theoretically non-recoverable losses are reached, there is always a chance to find combinations of chain recovery. Of course, the reliability of multi-disk systems decreases with the increase in the number of disks in the array. But the redundancy of storage systems that use multidimensional parity technology to increase the reliability of storage in multi-disk arrays is theoretically equivalent to multiple replications [30].

6 Conclusion

The method of storing and transferring files without using encryption, but only by splitting files according to multidimensional parity algorithms, ensures confidentiality and guarantees the security of the content from unauthorized access. The fundamental difference between this technology and existing security methods is the ability to implement an internally consistent and up-to-date model of stored and processed data with a high degree of protection against external intrusion . First, the system does not allow you to use the information in case of unauthorized access. Split data itself does not carry meaningful information. The second aspect is related to the fact that the recovery procedure can be performed using only a part of the split data, that is, the storage locations are regenerated . At the same time, unlike the backup method (multiple copies), the system is designed so that the more storage locations are used, the more information is split. Fragments of information are stored in different places. Anyone who wants to get information will have to collect fragments from all storage locations. But even if he collects what is split, he must know how to connect the parts.

The method of splitting/restoring is written in a meta-file that provides access to the information. There is a key in the encryption and a metafile in our method. This is the file that describes the splitting method. Only with its help can you collect what is split.

The ultimate goal of the technology involves the operation of a database in which private users, companies and organizations can store their files, being sure of their security.

7 References

[1] Keser, H., & Semerci, A. (2019). Technology trends, Education 4.0 and beyond. Contemporary Educational Research Journal, 9(3), 39–49. https://doi.org/10.18844/cerj.v9i3.4269

[2] Bagel, S., Kok, A., Zubeida, A., Suleimenova, Z., Riskulbekova, A., & Uaidullakyzy, E. (2019). Teaching Primary School Pupils Through Audio-Visual Means. International Journal of Emerging Technologies in Learning (iJET), 14(22), 122-140. https://doi.org/10.3991/ijet.v14i22.11760

[3] Elsayed, M., & Salama, R. (2020). Educational games for miss-concentration students (ADHD students). International Journal of Innovative Research in Education, 7(1). https://doi.org/10.18844/ijire.v7i1.4762

[4] Syrgabekov I., Sagauli E., Kurmanbaev E. Protection of databases by the method of distributed storage // Reports of the National Academy of Sciences of the Republic of Kazakhstan. – No. 5. - 2014. - pp. 141-153.

[5] Zadauly E., Kurmanbayev E., Syrgabekov I. Innovative security system based on distributed information storage with data splitting / / Patriot Engineering. – №2 (7). – 2015. – Pp. 111-119.

[6] Karipzhanova A. Zh., Gudov A.M. Organization of distributed databases of information systems by the method of data splitting / / Materials of the XIV (XLV) International Scientific Conference of Students and Young Scientists "Education, Science, Innovation: the contribution of young researchers", Kemerovo, Russia, April 25, 2019 – Kemerovo, 2019.

[7] Kurmanbaev E.A., Syrgabekov I. N., Zadauly E. Karipzhanova A.Zh., Urazbaeva K.T. Information Security System based on the Distributed Storage with Splitting of Data // International Journal of Applied Engineering Research. – 2017. – Vol. 12. – № 8. – pp. 1703-1711.

[8] Ãœlker, E. D. (2020). The effect of applying 4-stages on learning analysis and design of algorithms. Cypriot Journal of Educational Sciences, 15(5), 1238–1248. https://doi.org/10.18844/cjes.v15i5.4621

[9] Taygan, U., & Ozsoy, A. (2020). Performance analysis and GPU parallelisation of ECO object tracking algorithm. New Trends and Issues Proceedings on Advances in Pure and Applied Sciences, (12), 109–118. https://doi.org/10.18844/gjpaas.v0i12.4991

[10] Tekkanat, E., & Topaloglu, M. (2018). Developing java design patterns modeller with object-oriented programming. Global Journal of Computer Sciences: Theory and Research, 8(3), 132–135. https://doi.org/10.18844/gjcs.v8i3.4024

[11] Bhuyan, M. H., & Tamir, A. (2020). Evaluating COs of computer programming course for OBE-based BSc in EEE program. International Journal of Learning and Teaching, 12(2), 86–99. https://doi.org/10.18844/ijlt.v12i2.4576

[12] Kurmanbaev E.A., Syrgabekov I. N., Zadauly E. Karipzhanova A.Zh., Urazbaeva K.T. Information Security System based on the Distributed Storage with Splitting of Data //

International Journal of Applied Engineering Research. – 2017. – Vol. 12. – № 8. – pp. 1703-1711.
[13] Kaseb, M. R., Khafagy, M. H., Ali, I. A., & Saad, E. M. (2018, March). Redundant independent files (RIF): a technique for reducing storage and resources in big data replication. In World Conference on Information Systems and Technologies (pp. 182-193). Springer, Cham. https://doi.org/10.1007/978-3-319-77703-0_18
[14] Karipzhanova A., Sagindykov K., Dimitrov K. Justification of the method and algorithm of multidimensional parity control in distributed databases of information systems // Proc. X National Conference with International Participation «Electronica 2019», May 16-17, 2019, Sofia, Bulgaria. https://doi.org/10.1109/ELECTRONICA.2019.8825600
[15] Plank J.S. Optimizing Cauchy Reed-Solomon Codes for Fault-Tolerant Storage Applications. – Tennessee, 2005.
[16] Shokrollahi A. Transactions on Information Theory // Raptor Codes. – 2006. – Vol. 52. – P. 2551-2567. https://doi.org/10.1109/TIT.2006.874390
[17] Lee J.H. WEAVER Codes: Highly Fault-Tolerant Erasure Codes for Storage Systems, in FAST-2005: 4th Usenix Conference on File and Storage Technologies, 2005.
[18] Peterson W.W., Weldon E.J. Error-Correcting Codes / 2nd edition. – Cambridge, Massachusetts: MIT Press, 1972.
[19] Luby M. LT Codes // Proc. of the 43rd Annual IEEE Symp. on Foundations of Computer Science (FOCS), 2002.
[20] Farman, A., & Hasnain, M. (2019). Raid Storage Technology: A Survey. i-Manager's Journal on Computer Science, 7(4), 24. https://doi.org/10.26634/jcom.7.4.17269
[21] Huang, W. (2017). Coding for security and reliability in distributed systems (Doctoral dissertation, California Institute of Technology).
[22] Rahman, P. A. (2017, February). Analysis of the mean time to data loss of nested disk arrays RAID-01 on basis of a specialized mathematical model. In IOP Conference Series: Materials Science and Engineering (Vol. 177, No. 1, p. 012088). IOP Publishing. https://doi.org/10.1088/1757-899X/177/1/012088
[23] Chan, H. H., Li, Y., Lee, P. P., & Xu, Y. (2018). Elastic Parity Logging for SSD RAID Arrays: Design, Analysis, and Implementation. IEEE Transactions on Parallel and Distributed Systems, 29(10), 2241-2253. https://doi.org/10.1109/TPDS.2018.2818171
[24] Patterson D.A., Gibson G., Katz R.H. A Case for Redundant Arrays of Inexpensive Disks (RAID) // Proceed. of the 1988 ACM SIGMOD conf. on Management of Data. – Chicago IL, 1988. – P. 109-116. https://doi.org/10.1145/971701.50214
[25] Karagozlu, D. (2020). Determination of cyber security ensuring behaviours of pre-service teachers. Cypriot Journal of Educational Sciences, 15(6), 1698–1706. https://doi.org/10.18844/cjes.v15i6.5327
[26] Benmammar, S. (2020). Teaching English for specific purposes to computer science students with reading difficulties. Global Journal of Foreign Language Teaching, 10(3), 159–166. https://doi.org/10.18844/gjflt.v10i3.5072
[27] Wong J., Shea M., Tan F. Multidimensional Codes. – The Wiley Encyclopedia of Telecommunications, 2016.
[28] Belaidi, R., Bendib, B., Ghribi, D., Bouzidi, B., & Larafi, M. M. (2019). A comparative study on conventional and modern maximum power point tracking algorithms applied to photovoltaic systems. World Journal of Environmental Research, 9(2), 29–35. https://doi.org/10.18844/wjer.v9i2.4625
[29] Karipzhanova A. Zh., Sagyndykov K. M. Ways to improve the reliability of information stored in databases. Vestnik KazGUIU. – 2018. – №3(39). – Pp. 265-270.

[30] Le, Q. M. (2019). Shingled Magnetic Recording Disks for Mass Storage Systems (Doctoral dissertation, Santa Clara University).

8 Authors

Tlek Akhmetgalym is a 3rd year doctoral student, majoring in computer science (e-mail: tlek.akhmetgalym@mail.ru. ORCID - https://orcid.org/0000-0002-4361-9810. Address: Abay street 107, 071405, Semey city, Kazakhstan).

Gulzira Mukasheva is a 3rd year doctoral student, majoring in computer science (e-mail: gulzira_7777@mail.ru. ORCID - https://orcid.org/0000-0001-9766-1371. Address: Abay street 107, 071405, Semey city, Kazakhstan).

Karipzhanova Ardak Zhumagazievna PhD, is Dean of the Faculty of Information Technology and Economics, Senior Lecturer of the Department of Information and Technical Sciences (e-mail: kamilakz2001@mail.ru and ORCID - https://orcid.org/0000-0002-0113-6132. Address: Abay street 107, 071405, Semey city, Kazakhstan).

Aukenov Bolat Mayzhanuly is a PhD Candidate of Physical and Mathematical Sciences, Associate Professor in mathematics, Department of Information and Technical Sciences (e-mail: abm58@mail.ru and ORCID - https://orcid.org/0000-0002-1350-5660. Address: Abay street 107, 071405, Semey city, Kazakhstan).

Urazbaeva Kumys Toleubekovna is a Candidate of Physical and Mathematical Sciences, Associate Professor (e-mail: urazbaeva57@mail.ru and ORCID - https://orcid.org/0000-0002-8296-8394. Address: Abay street 107, 071405, Semey city, Kazakhstan).

Article submitted 2022-09-08. Resubmitted 2022-10-11. Final acceptance 2022-10-11. Final version published as submitted by the authors.