BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en

Vol. 29 No. 2. 2024

UDC 004.43
DOI: 10.62660/bcstu/2.2024.10

A software model to generate permutation keys through a square matrix

Emil Faure*
Doctor of Technical Sciences, Professor
Cherkasy State Technological University
18006, 460 Shevchenko Blvd., Cherkasy, Ukraine
https://orcid.org/0000-0002-2046-481X

Anatoly Shcherba
PhD in Physical and Mathematical Sciences, Associate Professor
Cherkasy State Technological University
18006, 460 Shevchenko Blvd., Cherkasy, Ukraine
https://orcid.org/0000-0002-3049-3497

Artem Skutskyi
Postgraduate Student
Cherkasy State Technological University
18006, 460 Shevchenko Blvd., Cherkasy, Ukraine
https://orcid.org/0000-0002-8632-1176

Artem Lavdanskyi
PhD in Technical Sciences, Associate Professor
Cherkasy State Technological University
18006, 460 Shevchenko Blvd., Cherkasy, Ukraine
https://orcid.org/0000-0002-1596-4123

Abstract. Information security and data protection are among the key aspects, which should be intensively developing in the 21st century. A conventional approach to cryptographic algorithms offers to apply matrices to represent information. However, more recent approaches deploy other data structures, including permutations, thus necessitating accordance between differing data structures to integrate different methods into a wholistic system of processing and transmitting information. This study aims to generate permutations, which serve as a key for factorial data coding according to a known key matrix. The paper presents two algorithms for transforming a square matrix into a permutation. An example of matrix transformation following each of the proposed algorithms is given. A software model was created and described to investigate the transformation of square matrices into permutations with the Matlab software product. The authors have considered the built-in methods of statistical information processing in the Matlab program and their graphical representation by built-in functions, which are applied in the process of the software model. A matrix transformation has been performed according to the proposed algorithms. The paper investigates all possible combinations of a square matrix of order 2 with elements referring to the finite integer field modulo $p = 17$ and $p = 23$. According to each transforming algorithm, the results of a square matrix transforming into a permutation number are obtained in the lexicographic order. The statistical properties of the obtained results have been studied, and the most

Article's History: Received: 17.02.2024; Revised: 01.05.2024; Accepted: 27.05.2024.

Suggested Citation:
Faure, E., Shcherba, A., Skutskyi, A., & Lavdanskyi, A. (2024). A software model to generate permutation keys through a square matrix. *Bulletin of Cherkasy State Technological University*, 29(2), 10-23. doi: 10.62660/bcstu/2.2024.10.

*Corresponding author

efficient algorithm for transforming matrices into permutations has been determined based on the distribution uniformity criterion for the generated permutation numbers. The study demonstrates that this algorithm can potentially be deployed in information exchange systems based on factorial data coding

Keywords: factorial data coding; data security; key agreement; statistics; uniform distribution

INTRODUCTION

Software is widely used in everyday life. Every second, millions of operations are performed globally, from ordinary requests to requests that require a high level of security and reliability of information transmission. These operations include managing bank accounts, electronic digital signatures for documents, automated technological processes, controlling unmanned aerial vehicles, providing secure communication channels for the Internet of Things (IoT), etc.

Rapid development of information technologies involves information security as well. In particular, Ye. Kaptiol & I. Horbenko (2020) in their work used cloud computing with IBM quantum computers to investigate Grover's algorithm, which is a necessary method to solve cryptology problems. S. Joshi *et al*. (2023) discussed research on the ways quantum computing can break the RSA encryption algorithm in polynomial time, an operation that conventional computers are not able to perform because of processing power limitations. P. Li *et al*. (2024b) propose a scalable parallel circuit for ultrafast random bit generation at 100 terabits per second based on a single micro-ring resonator. In an investigation into Nvidia's development, A. Lavdanskyi *et al*. (2023) demonstrates how software and hardware technology of compute unified device architecture (CUDA) of graphics processors increases the performance of operations on permutations.

Based on the above, users have access to hardware with potent computing capabilities, which allows them to test the reliability of conventional encryption algorithms. However, modern technology may be used to achieve malicious goals. When determining the cryptographic stability of encryption algorithms, the breaking time is an important parameter. Passwords with a total length of 8 characters (including numbers, uppercase letters, lowercase letters, and special characters) and encrypted with MD5 hash function require approximately one hour to be computed when using an RTX4090 graphics card. The cloud computing detailed in S. Rani *et al*. (2023) is able to reduce this time by a factor of 4. Factorial data coding according to the definition offered in V. Shvydkyi *et al*. (2021) is the process of transforming information for its protection based on the factorial numbering system. Non-separable factorial data coding is the coding that uses π permutations of length M as signal carriers. The elements of the permutation are integer numbers limited by the range $[0; M - 1]$. This transformation enables implementation of integrated information protection against unauthorised access and communication channel errors. An advantage to non-separable factorial coding is that the cardinality of the key space is a function of $M!$, which, as M increases, grows faster than all polynomial and exponential functions of M.

On the other hand, a considerable number of cryptographic transformations are implemented by matrix information representations. In particular, H. Huang *et al*. (2022) analysed the imminent cryptographic systems based on tropical circular matrices. The results offer a new encryption algorithm with an open key based on the tropical circular matrices. This cryptosystem is able to resist KU and RM attacks. A.R. Naseri *et al*. (2023) applied the class of square Fibonacci matrices $(q + 1)\times(q + 1)$ in cryptographic transformations, where q – an integer number. This type of cryptographic permutations uses matrices as a key in their encryption algorithm. The approach simplified and enhanced the key exchange procedure. M. Maxrizal (2022) proposes matrices with zero determinant as a key to implement three-pass data transmission protocols. A study by F. Al-Shaarani & A. Gutub (2022) demonstrates a cryptographic key transmission algorithm through a matrix. Along with that, a matrix is employed to obtain stereograms, which are additionally encrypted and transmitted through an unsecured communication channel.

Compatible implementation of matrix data representation, for cryptographic key agreement in particular, as well as for factorial coding of data to be transmitted through secure channels, requires transforming the received matrix key into a permutation key. Consequently, a need arises to build and investigate algorithms of transforming matrices into permutations. Considering the fact that key lengths and information block lengths for matrix-based permutations and factorial protocols do not coincide in the majority of cases, reflecting matrices into permutations is not bijective, being simultaneously surjective and injective instead (Maturin *et al.*, 2023).

The aim of this research was to agree permutation keys for factorial data coding based on the informational interaction of the key matrix known to the conversers. The main goal includes several objectives:

- to develop algorithms to transform a matrix into a permutation;
- to design and to analyse bar charts reflecting permutation frequency after enumerating every square matrix of the given order for each algorithm;
- to compare the bar charts obtained under various transformation parameters.

MATERIALS AND METHODS

According to the definition offered by W. Greub (1975), a square matrix is a matrix that has a number of rows equal to the number ofcolumns: $i = j$. In this case, $A = (a_{ij})_{n\times n}$ is a square matrix of order n. The elements of a_{ij} matrix are distributed within the $0 \leqslant a_{ij} \leqslant p - 1$ range, where p – an integer number; i, j – the row and the column of the matrix, correspondently. In this research, it is proposed two algorithms to transform the A matrix into a permutation: 1) by representing A matrix in decimal and factorial notations; 2) by using auxiliary square matrices.

Permutation of the matrix A according to the first algorithm includes three steps.

1) Transforming matrix A into a decimal A_{10} value by the calculation noted as:

$$A_{10} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} \cdot p^{(i-1)n+j-1} = \\ = a_{11} \cdot p^0 + a_{12} \cdot p^1 + \ldots + a_{nn} \cdot p^{n^2-1}. \quad (1)$$

2) Limiting the decimal equivalent A_{10} of matrix A to the range $[0; M! - 1]$:

$$A_{10}^{norm} = |A_{10}|_{M!}. \quad (2)$$

3) Transforming the decimal representation of the number A_{10}^{norm} selected from the range $[0; M! - 1]$ into the factorial number A_F and permutation π of length M:

$$A_{10}^{norm} \rightarrow A_F \rightarrow \pi. \quad (3)$$

Transformation (3) is simple and shall not be studied in this paper. The second algorithm to transform a matrix into a permutation is to be implemented in four steps.

1) Building auxiliary matrices $H_k = (h_{ij}(k))_{(n\times n)}$, where $(h_{ij}(k)) = |(k-1)\cdot n^2 + (i-1)\cdot n + j - 1|_M$, while $1 \leq k \leq \left[\frac{M}{n^2}\right]$.

Thus, when M value exceeds n^2, there are several auxiliary matrices H_k. For example, if $M = 8$, $n = 2$, two auxiliary matrices will be necessary: $H_1 = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}$, $H_2 = \begin{pmatrix} 4 & 5 \\ 6 & 7 \end{pmatrix}$.

If $M = 9$, $n = 2$, three auxiliary matrices are to be built: $H_1 = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}$, $H_2 = \begin{pmatrix} 4 & 5 \\ 6 & 7 \end{pmatrix}$, $H_3 = \begin{pmatrix} 8 & 0 \\ 1 & 2 \end{pmatrix}$.

2) Successive multiplication from the left of the matrix A by all auxiliary matrices H_k:

$$B_k = A \cdot H_k = (b_{ij}(k))_{(n\times n)}. \quad (4)$$

3) Finding factorial coefficient a_l, $0 \leqslant l \leqslant M - 1$, of number A_F by calculation:

$$a_l = |b_{ij}(k)|_{(l+1)}, \quad (5)$$

where values i, j, k, correspondent to the coefficient a_l, are selected so that $l = h_{ij}(k)$.

4) Converting factorial number A_F into permutation π of length M:

$$A_F \rightarrow \pi. \quad (6)$$

To automate operations converting a decimal number into a permutation, the authors of this study have developed a transformation function *int2perm(number, base_permut)* in Matlab programming environment. The function accepts *number*, a number in decimal representation to be transformed into a permutation, as input parameter. Parameter *base_permut* is a base permutation presented as a vector (row). Figure 1 features the block diagram showing the function algorithm converting a decimal number into a permutation.

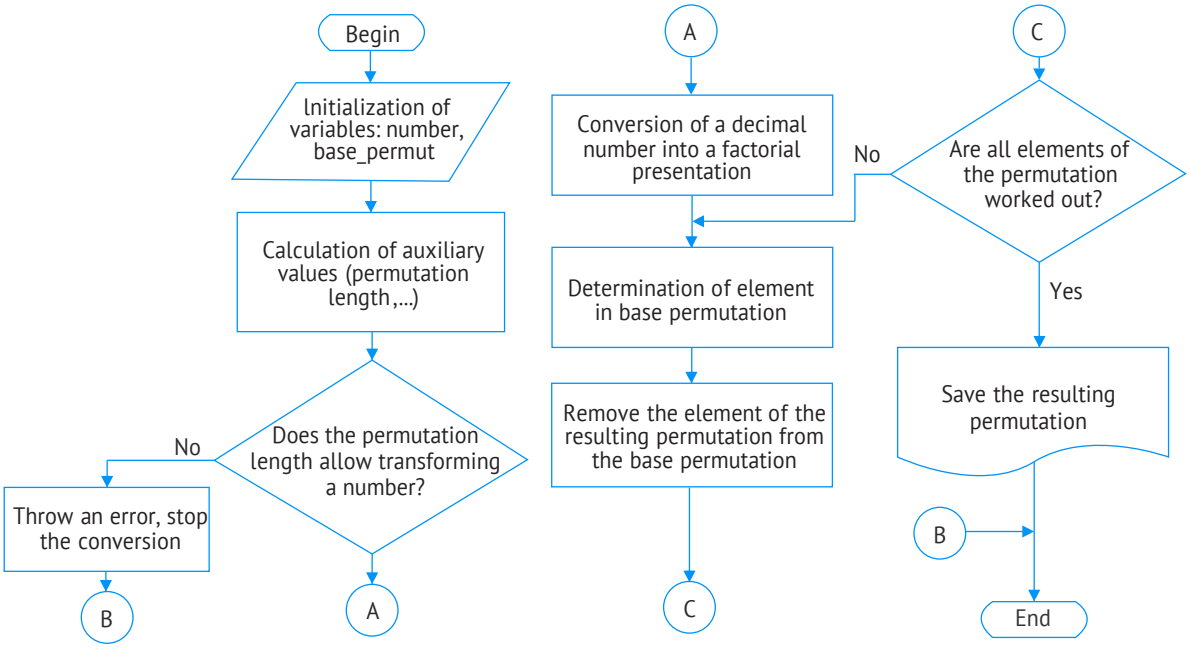

Figure 1. Algorithm for transforming a number into a permutation (*int2perm* function)

Source: developed by the authors

To conduct the research, Matlab software environment has been involved. The study applied programming principles and product capabilities developed by D.J. Higham & N.J. Higham (2005) and D.T. Valentine & B.H. Hahn (2023). Two additional methods have been developed to represent the number in the factorial and decimal notations: *int2fact()* – which is a method of transforming a number from decimal into factorial notation, and *factint()* – which is a method for reverse transformation of a number from factorial to decimal notation.

To construct the software model, the following basic variables were used:

- *A* – a square matrix;
- *n* – the order of matrix *A*;
- *M* –the permutation length;
- *pMax* denotes the maximum values for the elements of the matrix *A* (a variable that allows verifying the $a_{ij} \leqslant p - 1$ condition);
- *Adec* – the current value of the decimal number obtained from matrix *A* by the first transformation algorithm (a variable containing the result of transformation according to Equation (1) and presented in decimal notation);
- *Bdec* – the current value of the number obtained from matrix *A* according to the second transformation algorithm (a variable containing the transformed result according to Equation (5) and reduced to decimal notation);
- *AdecCollection* – the list of matrix values converted according to the first transformation algorithm (the variable is a one-dimensional array (vector), where each element is the result of stepwise transformations of the matrix *A* according to the first algorithm *Adec*);
- *BdecCollection* – the list of the matrix values converted according to the second transformation algorithm (the variable is a one-dimensional array (vector), where each element is the result of stepwise transformations of the matrix *A* according to the second algorithm *Bdec*);
- *aCollection* – the list of obtained matrix *A* values (the variable is represented as a three-dimensional array. Two dimensions define rows and columns, while the third dimension corresponds to the sequence number of each matrix generated within the range [1; p^{n^2}]).

The software model algorithm. The procedure for launching the software model includes the following steps:

1. Entering parameters: *M*, *pMax*, *n*.
2. Calculating values: number *M*!, the number of possible matrix *A* variants, number of auxiliary matrices to be processed according to the second transformation algorithm.
3. Enumerating all matrix *A* variants under the given conditions, transforming them according to the first and second algorithms.
4. Saving the obtained values.

As a result of the software model, statistical data were obtained to be further evaluated as an expansion of the **.mat* workspace.

The block diagram of the algorithm described above can be represented as shown in Figure 2.

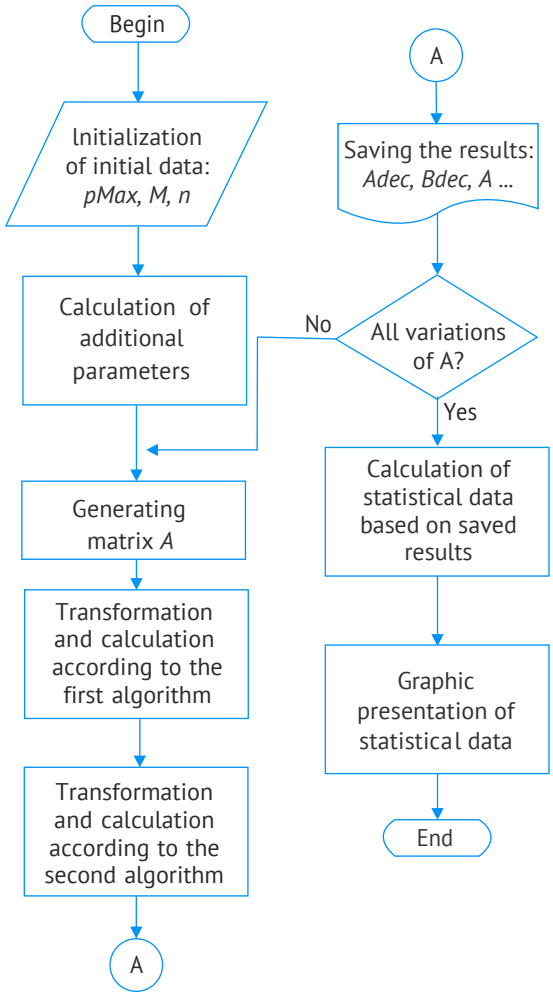

Figure 2. Software model algorithm

Source: developed by the authors

Collecting statistical data and their validation. The Matlab software product stores the results of the experiments as a workspace extension **.mat*. This format saves all the variables produced by the software. By saving the program's working space, it is possible to analyse the simulation results and access the arrays of variable values separately from the main algorithm of the model, i.e., to simulate the experiments and save the results separately for each of the experiments. To save the workspace, Matlab program employs *save(name)* function where *name* – the file name. Thus, after starting the simulation program in two experiments, the workspace was saved for each simulation. Two files containing simulation variables and their values have thus been obtained. Applying this procedure to save the workspace of the program variables, one can separate the software part of the simulation from additional calculations related to obtaining statistical parameters and to separate the experimental part from the results analysis (analytical part). In addition, this procedure optimises analysing the

results by creating a graphical interface, adds the possibility of extending the study by adding or diversifying the methods of statistical analysis applied to analyse the results. The software implementation of the model performs solely data generation, the analytical part is responsible for calculating statistical parameters, graphs, etc. Based on the defined list of variables present in the **.mat* file, an additional program has been developed that calculates the statistical parameters of the obtained results. Figure 3 demonstrates the program algorithm.

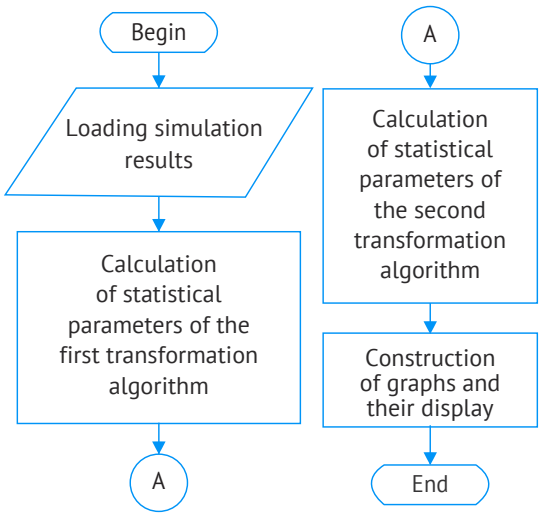

Figure 3. Calculating statistical parameters
Source: developed by the authors

While implementing the algorithm, the built-in functions of Matlab were deployed: *min* – the minimum value in a numerical sample; *max* – the maximum value in a numerical sample; *range* – the range of values in a numerical sample; *num* – sample size; *mean* – sample mean; *median* – the median of the sample values; *standart deviation* – standard deviation.

Matlab function *datastats()* can retrieve the listed values. The *var()* function will calculate the variance. The *bar(x,y,*)* method was to display bar graphs for absolute frequency, where x – the vector of values along the *Ox* axis; y – vector of values along the *Oy* axis; * denotes additional settings (line thickness, colour, etc.). The *plot(x,y,*)* method was employed to display graphs of relative frequency.

RESULTS

Simulation of the described algorithms and methods for matrix transformation into permutation was performed by conducting two experiments with the following input parameters:

1) $p = 17, M = 8, n = 2$;
2) $p = 23, M = 8, n = 2$.

A fragment of setting up the software implementation by entering the initial data for the first experiment is shown in Figure 4.

```
1    pMax = 17; %maximum p
2    pMin = 0; %minimum p
3    M = 8; % permutation lenght
```

Figure 4. A code fragment to enter primary parameters
Source: developed by the authors

The set cardinality of matrix A equals to $17^4 = 83521$ for the first experiment and to $23^4 = 279841$ for the second experiment. The data arrays resulting from experimental data are represented by sets of permutation numbers A_{10}^{norm} for the first algorithm and the obtained combinations of the matrix A. The array of results saved for the second transformation algorithm contains permutation numbers obtained by transforming the A_F values from the factorial into decimal representation. Figures 5 and 6 contain fragments of the results for the first and second experiments.

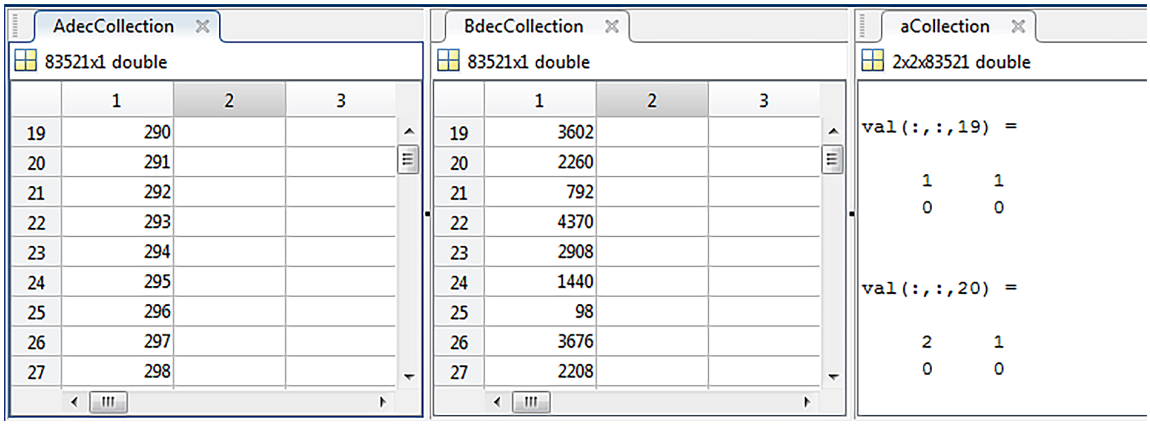

Figure 5. Generated and saved transformation results according to the first and second algorithms for $p = 17, M = 8, n = 2$
Source: developed by the authors

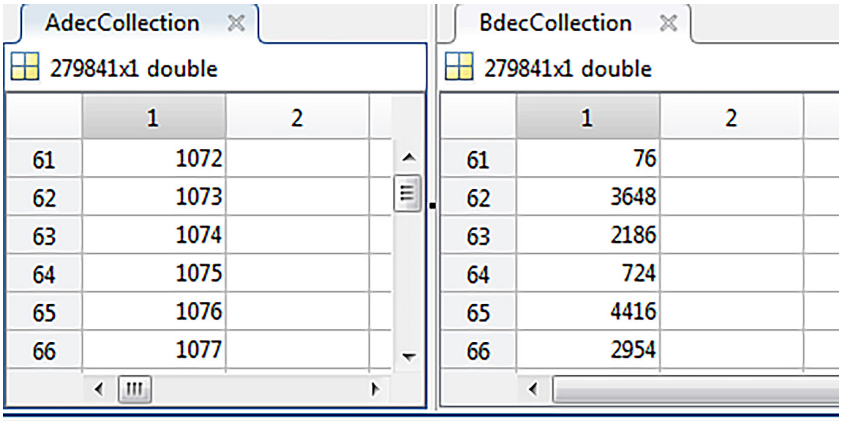

Figure 6. Generated and saved transformation results according to the first and second algorithms for $p = 23$, $M = 8$, $n = 2$

Source: developed by the authors

Variable *AdecCollection* contains the transformation results according to the first algorithm (A_{10}^{norm} value). The variable *BdecCollection* contains the transformation results according to the second algorithm (decimal representation of the received permutation number A_F). The variable *aCollection* contains all the obtained *A* matrix values.

In the first experiment with input parameters $p = 17$, $M = 8$, $n = 2$, the following statistical results have been obtained for the transformation executed according to the first algorithm:

- the number of generated matrix *A* combinations, attribute *num* in the *datastat()* function: $N = 83521$;
- the maximum value of the number obtained in the result of transforming the matrix *A*: $Nmax = M! - 1 = 40319$;
- the minimum value of the number obtained in the result of transforming the matrix *A*: $Nmin = 0$;
- the mean value in the generated sample: $mean = 19514$;
- the median values in the generated sample: $median = 19439$;
- the maximum number of permutation repetitions: $rep_{max}^{A1} = 3$;
- the minimum number of permutation repetitions: $rep_{min}^{A1} = 2$;
- A_{10}^{norm} values from the range [0; 2880] repeat rep_{max}^{A1} times;
- A_{10}^{norm} values from the range [2881; 40320] repeat rep_{min}^{A1} times;
- variance: $D_{A1} = 142498178.841$;
- the maximum value of the relative frequency for a permutation to occur is: $W_{max}^{A1} = 3.5919 \cdot 10^{-5}$;
- the minimum value of the relative frequency for a permutation to occur is: $W_{min}^{A1} = 2.3946 \cdot 10^{-5}$.

Distribution of absolute frequencies N_j, $0 \leqslant j \leqslant M! - 1$ of permutations with the number $x_j = j$ in the lexicographic order following the first transformation algorithm is shown in Figure 7 as a bar graph.

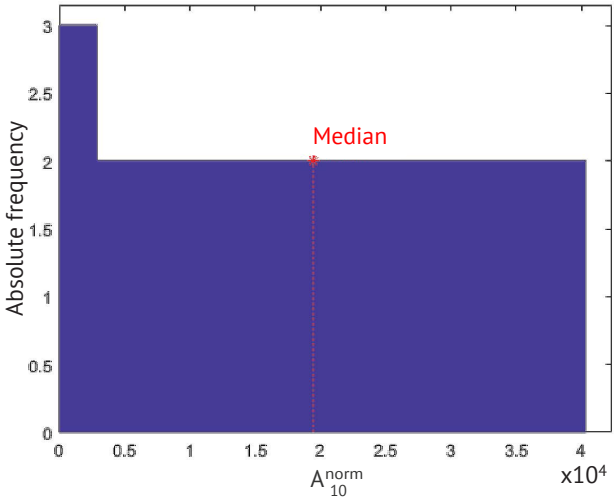

Figure 7. Distribution of absolute frequencies for a permutation to occur according to the first transformation algorithm based on the results of the first experiment

Source: developed by the authors

Figure 8 shows the graph of the relative frequency of occurrence of permutations according to the first transformation algorithm is shown in. *Ox* axis contains the generated values of the permutation sequence numbers when matrix *A* is transformed by formula A_{10}^{norm}, and *Oy* axis is the relative frequency of the generated number.

In the experiment with input parameters $p = 17$, $M = 8$, $n = 2$, the following statistical results were obtained for the transformation according to the second algorithm:

- the number of generated combinations of the matrix *A*, attribute *num* in the *datastat()* function: $N = 83521$;
- the maximum value of the number resulting from transforming the matrix *A*: $Nmax = 40198$;
- the minimum value of the number resulting from transforming the matrix *A*: $Nmin = 0$;

- the mean value in the generated sample: *mean* = 19995;
- the median values in the generated sample: *median* = 20012;
- the maximum repetition number of the generated numbers (sequence number of the permutation), $rep^{B1}_{max} = 52$;
- the minimum repetition number of the generated numbers (sequence number of the permutation), $rep^{B1}_{min} = 0$;
- the number of ordinal number values of the permutations that have the maximum number of repetitions rep^{B1}_{max} is 408;
- the number of ordinal number values of the permutations that have the minimum number of repetitions rep^{B1}_{min} is 38159;
- variance: $D_{B1} = 136109307.2089$;
- the maximum value of the relative frequency for a permutation to occur: $W^{B1}_{max} = 62.2597 \cdot 10^{-5}$;
- the minimum value of the relative frequency for a permutation to occur: $W^{B1}_{min} = 35.9191 \cdot 10^{-5}$.

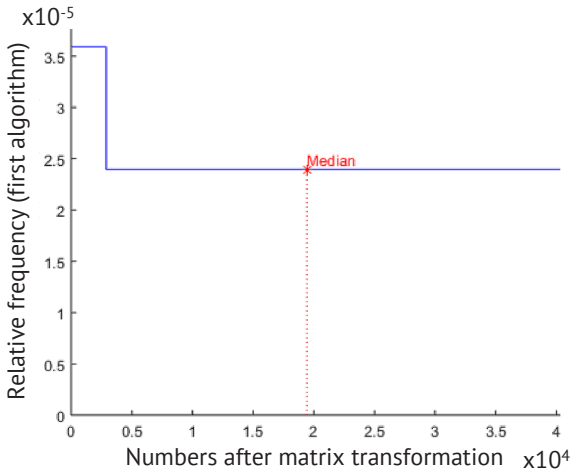

Figure 8. Relative frequency for a permutation to occur according to the first transformation algorithm based on the results of the first experiment
Source: developed by the authors

Figure 9 shows a bar graph for distribution of the relative frequency N_j, $0 \leqslant j \leqslant M! - 1$ of permutations with number $x_j = j$ in the lexicographic order according to the first transformation algorithm. The graph describing the relative frequency of permutation occurrence according to the first transformation algorithm is shown in Figure 10. *Ox* axis contains the generated values of the permutation sequence numbers when the matrix *A* is transformed by formula A^{norm}_{10}, and *Oy* axis is the relative frequency of the generated number.

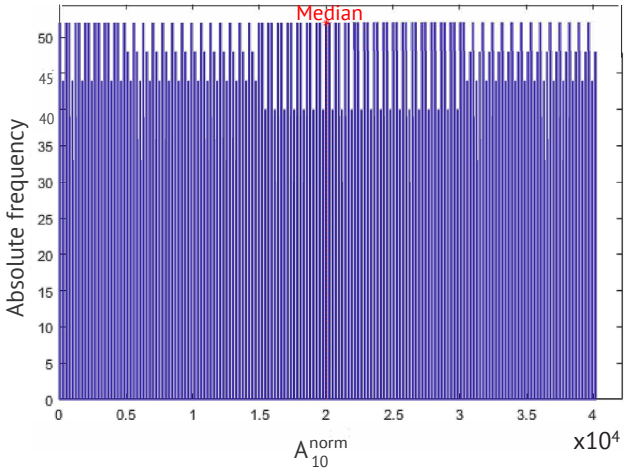

Figure 9. Distribution of the absolute frequencies for a permutation to occur according to the second transformation algorithm, based on the results of the first experiment
Source: developed by the authors

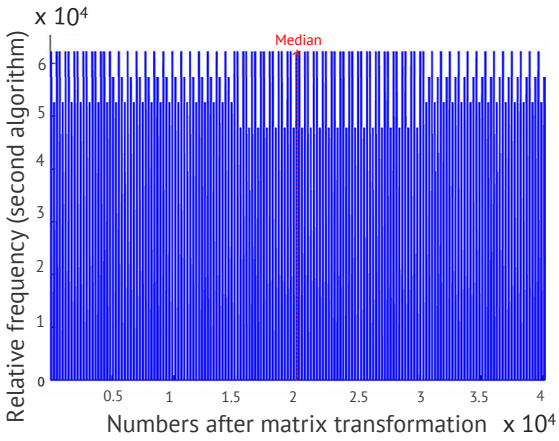

Figure 10. Relative frequency for a permutation to occur according to the second transformation algorithm, based on the results of the first experiment
Source: developed by the authors

In the experiment with input parameters $p = 23$, $M = 8$, $n = 2$, the following statistical results were obtained for the transformation executed according to the first algorithm:

- the number of generated combinations of the matrix *A*, attribute *num* in the *datastat()* function: *num* = 279841;
- the maximum value of the number resulting from transforming the matrix *A*: *max* = *M*! – 1 = 40319;
- the minimum value of the number resulting from transforming the matrix *A*: *min* = 0;
- the mean value in the generated sample: *mean* = 19996.957;

• the median values in the generated sample: $median = 19988$;

• the maximum number of permutation repetitions: $rep^{A1}_{max} = 7$;

• the minimum number of permutation repetitions: $rep^{A1}_{min} = 6$;

• A^{norm}_{10} values from the range [0; 37920] repeat rep^{A1}_{max} times;

• A^{norm}_{10} values from the range [37921; 40320] repeat rep^{A1}_{min} times;

• variance: $D_{A1} = 133524637.447$;

• the maximum value of the relative frequency for a permutation to occur is: $W^{A1}_{max} = 2.5014 \cdot 10^{-5}$;

• the minimum value of the relative frequency for a permutation to occur is: $W^{A1}_{min} = 2.1441 \cdot 10^{-5}$.

Figure 11 contains a bar graph for distribution of the absolute frequencies N_j, $0 \leqslant j \leqslant M! - 1$ of permutations with number $x_j = j$ in the lexicographic order according to the first transformation algorithm. The graph illustrating the relative frequency for permutations to occur according to the first transformation algorithm is demonstrated in Figure 12. *Ox* axis contains the generated values of the permutation sequence numbers when the matrix A is transformed by formula A^{norm}_{10}, and *Oy* axis is the relative frequency of the generated number. In the experiment with input parameters $p = 23$, $M = 8$, $n = 2$, the following statistical results were obtained for the transformation according to the second algorithm:

• the number of generated combinations of the matrix A, attribute *num* in the *datastat()* function: $num = 279841$;

• the maximum value of the number resulting from transforming the matrix A: $max = 40198$;

• the minimum value of the number resulting from transforming the matrix A: $min = 0$;

• the mean value in the generated sample: $mean = 20072.2193$;

• the median values in the generated sample: $median = 20162$;

• the maximum repetition number of the generated numbers (sequence number of the permutation), $rep^{B1}_{max} = 230$;

• the minimum repetition number of the generated numbers (sequence number of the permutation), $rep^{B1}_{min} = 0$;

• the number of ordinal number values of the permutations that have the maximum number of repetitions rep^{B1}_{max} is 64;

• the number of ordinal number values of the permutations that have the minimum number of repetitions rep^{B1}_{min} is 37679;

• variance: $D_{B1} = 1335421877.799$;

• the maximum value of the relative frequency for a permutation to occur: $W^{B1}_{max} = 82.1895 \cdot 10^{-5}$;

• the minimum value of the relative frequency for a permutation to occur: $W^{B1}_{min} = 28.5877 \cdot 10^{-5}$.

Figure 13 contains the bar graph showing the distribution of the absolute frequency N_j, $0 \leqslant j \leqslant M! - 1$ of permutations with number $x_j = j$ in the lexicographic order following the first transformation algorithm.

The graph illustrating the relative frequency for permutations to occur according to the second transformation algorithm is shown in Figure 14. *Ox* axis contains the generated values of the permutation sequence numbers when the matrix A is transformed by formula A^{norm}_{10}, and *Oy* axis is the relative frequency of the generated number.

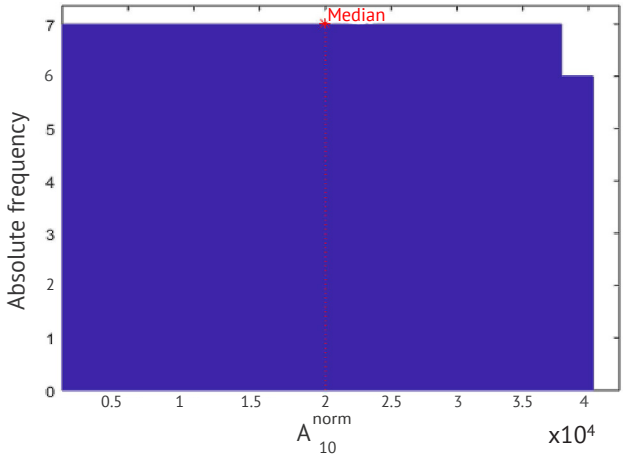

Figure 11. Absolute frequency for a permutation to occur according to the first transformation algorithm, based on the results of the second experiment

Source: developed by the authors

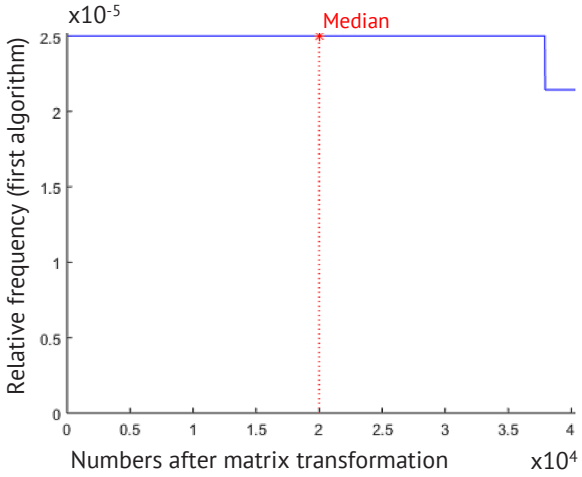

Figure 12. Relative frequency for a permutation to occur according to the first transformation algorithm, based on the results of the second experiment

Source: developed by the authors

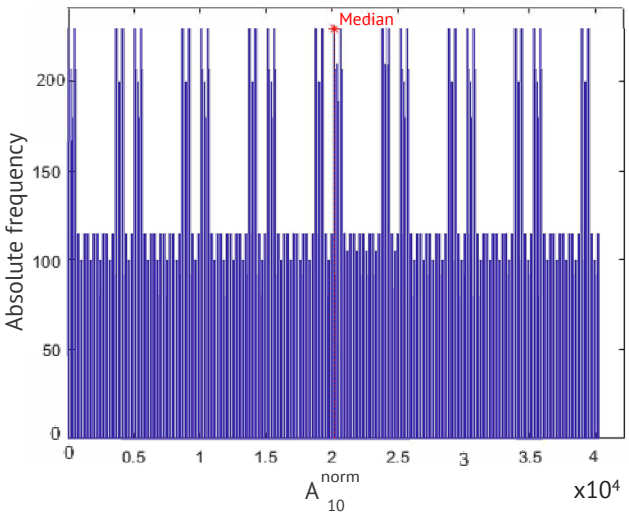

Figure 13. Absolute frequency of permutation occurrence according to the second transformation algorithm, based the results of the second experiment
Source: developed by the authors

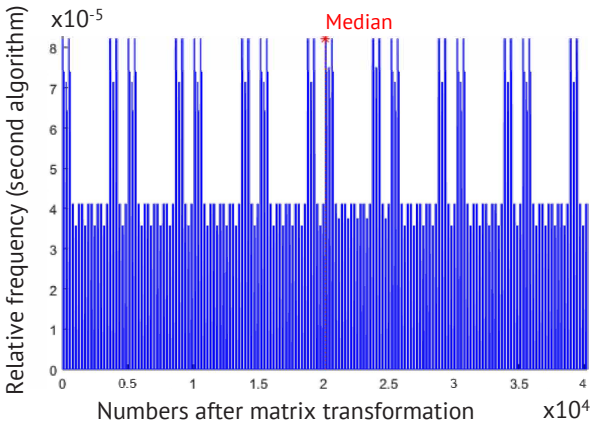

Figure 14. Relative frequency of permutation occurrence according to the second transformation (results of the second experiment)
Source: developed by the authors

The statistical results for the first and second experiments are shown in Table 1.

Table 1. Experimental statistical results

Parameter	Experiment 1		Experiment 2	
Input conditions for the experiment	$p = 17, n = 2, M = 8$		$p = 23, n = 2, M = 8$	
Matrix transformation algorithm	I	II	I	II
The volume of the generated sample,	83521	83521	279841	279841
N_{max}	40319	40198	40319	40198
N_{min}	0	0	0	0
Variance, D	142498178.84	136109307.21	133524637.45	13542187.80
Relative frequency, W_{max}	$3.5919 \cdot 10^{-5}$	$62.2597 \cdot 10^{-5}$	$2.50142 \cdot 10^{-5}$	$82.1895 \cdot 10^{-5}$
Relative frequency, W_{min}	$2.3946 \cdot 10^{-5}$	$35.9191 \cdot 10^{-5}$	$2.1440 \cdot 10^{-5}$	$28.5877 \cdot 10^{-5}$
Number of repetitions rep_{max}	3	52	7	230
Number of repetitions rep_{min}	2	0	6	0
The sample median	19439	20012	19988	20162
The sample mean	19514	19995	19996.96	20072.22

Source: developed by the authors

Transformation of the matrix A by the first algorithm generated the maximum repetitions of the permutation numbers in the lexicographic order $rep^{A1}_{max} = 3$ in the first experiment and $rep^{A2}_{max} = 7$ in the second experiment. The minimum number of the generated permutation repetitions is: $rep^{A1}_{min} = 2$ in the first experiment and $rep^{A2}_{min} = 6$ in the second experiment.

The result of transformations with auxiliary matrices (the second algorithm) is the maximum number of repeated permutation numbers $rep^{B1}_{max} = 52$ in the first experiment and $rep^{B2}_{max} = 230$ in the second experiment. The minimum number of repeated permutation numbers is $rep^{B1}_{min} = 0$ in the first experiment and $rep^{B2}_{min} = 0$ in the second experiment.

The variance value of the obtained permutation numbers in the first experiment and matrix processing according to the first algorithm is $D_{A1} = 142498178.841$. The variance value in the first experiment and the processing algorithm with auxiliary matrices is $D_{B1} = 136109307.2089$. The obtained relations $D_{A1} > D_{B1}$ indicate that random variable values in the first algorithm have larger deviations from the distribution centre, which is a prerequisite for more uniform distribution over a set of possible values. The bar graph featuring the absolute distribution of the matrix transformation according to the second algorithm (Fig. 13) shows zeros, which means that the corresponding permutation number has not been generated in the sample.

To test the null hypothesis H_0 that the received samples of permutation numbers in the lexicographic order correspond to a uniform distribution over the segment $[0; M! - 1]$, the χ^2 Pearson test was applied.

H. Chernoff & E.L. Lehmann (1954) argue that the resulting sample must be grouped into classes to correctly apply the χ^2 criterion. H.B. Mann & A. Wald (1942) found that the optimal $r = r(N, \alpha)$ number of grouping classes (data grouping intervals) based on the sample size N and the significance level α for the χ^2 criterion is defined by the equation:

$$r(N,\alpha) = 2 \cdot 2^{1.2} \cdot \left(\frac{N-1}{Norm_{0;1;\alpha}}\right)^{0.4}, \quad (7)$$

where $Norm_{0;1;\alpha}$ – the upper limit of the standard normal distribution $Norm_{0;1}$.

Later, it was established that without significant loss of the criterion's length, the $r(N, \alpha)$ value can be halved, and for large N select the following (Turchyn, 2014):

$$r(N,\alpha) = 2 \cdot 2^{0.2} \cdot \left(\frac{N-1}{Norm_{0;1;\alpha}}\right)^{0.4}. \quad (8)$$

For $\alpha = 0.05$, $Norm_{0;1;0.05} = 1.645$. The following is an analysis of the results of experiment 1 for the first matrix transformation algorithm. For N=83521 and α=0.05, the value $r(83521,0.05) \approx 175.18$. Therefore, the number of grouping intervals is r=175. The length of the grouping interval is $H = \frac{40319}{175} \approx 230.3943$. The value of the interval width is taken as $H = 230.395$. Then $x_i^* = H \cdot i$, $0 \leqslant i \leqslant r$. v_i denotes absolute frequencies of grouping intervals: $v_i = \sum_{x_{i-1}^* < x_j \leq x_i^*} N_j$, $0 \leqslant i \leqslant r$, and $v_1 = \sum_{x_0^* \leq x_j \leq x_1^*} N_j$, N_j is absolute frequency of x_j value in the i-th grouping interval. If the degree of freedom m of the χ^2_m distribution acquires large values, then by virtue of the central limit theorem, this distribution approaches the normal one with parameters $a = m$ and $\sigma^2 = 2m$:

$$\chi^2_m(x) \approx Norm_{m;2m}(x) \text{ for } m > 30. \quad (9)$$

Since $Norm_{m;2m}(x) = Norm_{0;1}\left(\frac{x-m}{\sqrt{2m}}\right)$, the equation for the upper α-limit of the distribution is $Norm_{m;2m}(x)$: $1 - \alpha = Norm_{m;2m}(x_\alpha) = Norm_{0;1}\left(\frac{x_\alpha - m}{\sqrt{2m}}\right) = Norm_{0;1}(t_\alpha)$. Then $\frac{x_\alpha - m}{\sqrt{2m}} = t_\alpha$ or:

$$x_\alpha = m + t_\alpha\sqrt{2m}. \quad (10)$$

The Pearson test statistics for a uniform distribution has degrees of freedom $m = r - 1$. If $\chi^2_{\alpha;m}$ is the upper α-limit of χ^2 distribution with m degrees of freedom, the H_0 hypotheses regarding the continuous uniform distribution of the sample is accepted at the level of significance α if:

$$\chi^2_{observ.} = \sum_{i=1}^{r} \frac{(v_i - Np_i)^2}{Np_i} < \chi^2_{\alpha;m}, \quad (11)$$

where p_i – theoretical relative occurrence frequency of the investigated random variable in the i-th grouping interval.

Worth noting that

1) $p_1 = P(0 \leq X \leq x_1^*) = P(x_{i-1}^* < X \leq x_{i1}^*) = p_i = \frac{1}{r}$, $2 \leqslant i \leqslant r$. For $N = 83521$ and $\alpha = 0.05$, the value $p_i = \frac{1}{175}$, $1 \leqslant i \leqslant 175$;

2) for each i, $1 \leqslant i \leqslant 175$ value, $Np_i = 83521 \cdot \frac{1}{175} \gg 10$ has been performed;

3) by virtue of relations (9), (10) and (11), $\chi^2_{observ.} = \sum_{i=1}^{r} \frac{\left(v_i - \frac{N}{r}\right)^2}{\frac{N}{r}} = \frac{r}{N}\sum_{i=1}^{r}\left(v_i - \frac{N}{r}\right)^2 < \chi^2_{\alpha;r-1} \approx r - 1 + t_\alpha\sqrt{2}\sqrt{r-1}$ holds. Thus, the verification rule is:

$$\sum_{i=1}^{r}\left(v_i - \frac{N}{r}\right)^2 < \frac{N}{r}\left(r - 1 + t_\alpha\sqrt{2}\sqrt{r-1}\right). \quad (12)$$

For $N = 83521$, $\alpha = 0.05$, $t_\alpha = 1.645$, $r = 175$ hypothesis H_0 about uniform distribution of the permutation numbers in the segment [0; 40319] is conditionally accepted:

$$\sum_{i=1}^{175}(v_i - 477.26)^2 < 97689.54. \quad (13)$$

Substituting the absolute frequencies v_i obtained from experiment 1 for the first matrix transformation algorithm into (13) results in the value $\sum_{i=1}^{175}(v_i - 477.26)^2 = 603773.91$, which significantly exceeds the permissible value 97689.54. Table 2 demonstrates the results of applying the Pearson test as Equation (12) for the results of experiments from Table 1.

Table 2. Results of testing the hypothesis about the uniform distribution of permutations

Parameter	Experiment 1		Experiment 2	
Matrix transformation algorithm	I	II	I	II
Number of intervals r	175		284	
Interval width H	230.395		141.969	
Value $\sum_{i=1}^{r}\left(v_i - \frac{N}{r}\right)^2$	603773.91	1973127.91	317923.08	350321441
Critical value $\frac{N}{r}\left(r - 1 + t_\alpha\sqrt{2}\sqrt{r-1}\right)$	97689.54		317418.32	

Source: developed by the authors

As has been proved here, none of the obtained permutation distributions corresponds to the uniform distribution according to χ^2 Pearson's test. However, with increasing $\frac{p^{n^2}}{M!}$ ratio, such discrepancy disappears for the first matrix transformation algorithm. On the contrary, the second matrix transformation algorithm requires additional research to improve its statistical properties.

DISCUSSION

If the computing power of the computer system is sufficient, additional hardware is inexpedient. However, storing and transmitting the matrix for key generation becomes an issue. In this case, it is possible to select a complex approach to the organization of a secure system. Transmitting a matrix through an open, unprotected communication channel between two conversers is possible by taking the following steps:

- a square matrix is generated in accordance with the requirements of the cryptosystem;
- the transformation of the matrix into a permutation is performed through successive steps described in "materials and methods";
- an exchange protocol using permutations is implemented.

Factorial coding ensures implementing protective functions against unauthorised access, as well as detecting and correcting errors. However, the issue of matching the common key permutations persists. To solve this issue, this study proposes an approach based on matrices. The advantages of matrix calculations when solving cryptography problems include:

- complexity of analysis: matrix operations (especially matrix inversions) complicate analysing and breaking cryptographic systems, increasing their security;
- flexibility and efficiency: matrix methods allow efficient encoding and decoding of large data arrays;
- increased resistance to quantum attacks: some matrix cryptographic schemes (for example, lattice-based) are resistant to quantum computer attacks, which enhances their prospects for future use.

Matrix operations, such as multiplication or inversion of large matrices, are computationally complex. This may lead to a significant increase in processing time and resource consumption, especially in cases with large matrix sizes. Note that transforming a matrix according to the first algorithm through representing matrix *A* in decimal and factorial notations requires significant computer system resources. In particular, a way to achieve a uniform distribution is to increase the $\frac{p^{n^2}}{M!}$ ratio, which, as a result, is able to increase the amount of time required for calculations. However, this flaw can be insignificant if part of the calculations is distributed and transferred to the hardware component of the information exchange system. One of the examples of realising this approach is offered in a study by M. Issad *et al.* (2020). The authors proposed to transfer encrypting and decrypting server's algorithms to a programmable gate array (FPGA). Another work by S. Sheikhpour *et al.* (2021) proposes two schemes based on a combination of hardware and time reservation: a novel hardware reservation for the cyclic AES function and time reservation for the AES key extension module hardware.

Another limitation of matrices may be related to the high memory volume requirements by information processing tools, since processing matrices requires significant amounts of memory, especially large dimensional matrices or a significant quantity of matrix operations. This may be problematic for resource-constrained devices, such as embedded devices and especially mobile devices. Thus, applications of the proposed approaches in low-resource cryptography should be investigated in more detail. Key management procedures can also be complicated. In case of matrix cryptosystems, matrices are the keys. Managing these keys (generation, storage, transmission) is more complex compared to traditional encryption algorithms, such as Advanced Encryption Standard (AES) (National Institute for Standards and Technology, 2001), where keys are scalar values. In addition, encryption and decryption algorithm offered by Z.Y. Karatas *et al.* (2019) involves finding an inverse matrix, which is a complicated computational task. Cryptosystems using lattice-based encryption have a similar property (Li *et al.,* 2024a).

Regarding the issue of cryptographic stability, matrix cryptosystems can be stable. For example, a study by H. Huang *et al.* (2022) emphasises that, based on the results of the research, an algorithm based on ring tropical matrices can be considered as a new post-quantum cryptosystem. However, the specific properties of matrices used for decryption can increase efficiency of certain cryptographic analysis methods, for example, attacks based on the matrix structure. An important stage in investigating cryptographic stability of matrix cryptosystems is hence analysing vulnerabilities to attacks performed through specialised methods, for example, lattice-based analysis or other algebraic methods.

In addition to the above, matrix cryptosystems may be vulnerable to linear dependencies between matrix elements. This can degrade the system stability and increase its chances of being hacked. To combat linear dependencies between elements, various approaches are used, which can be attributed to: nonlinear transformations (in AES and DES cryptosystems, s-blocks are deployed that perform bit transformations), which as a result reduce linear dependencies between input and output elements; another approach to reduce linear dependencies is to pad with redundant data or use longer keys to increase the key space; use of algebraic transformations using matrices with high entropy; adding randomness during key generation. An example is a study by M. Gafsi *et al.* (2020).

Note that matrix cryptosystems work with data blocks of fixed size, which can impede processing messages that are not a multiple of the block size and require additional mechanisms to overcome the barrier (such as adding placeholders). The existing methods of adding placeholders include PKCS No. 5/7, ANSI X.923, ISO/IEC 9797-1, bit padding, filling the message content to the required block size with bytes of zeros. Cryptosystems applying these techniques are AES, which often use PKCS #7, or DES, which may use PKCS #5 or ANSI X.923. However, implementing the mentioned

mechanism in the cryptosystem reduces its stability because of possible attacks targeted at this aspect, for instance, a boomerang type attack proposed back in 1998 (Bleichenbacher Attack Explained, 2019) and its improved algorithm (Kelesidis, 2022). In the second algorithm for converting a matrix into a permutation proposed in this paper, a mechanism for solving the backfill problem is introduced for the value of the permutation length M, which is greater than the number of elements of the matrix. Nonetheless, considering the possible identified risks, it is necessary to provide for a flexible encryption mechanism that would not depend on the length of the incoming message in the future.

Matrix fields are a powerful tool in cryptography, offering efficient and robust methods for protecting information. Involving matrices in cryptographic algorithms assists in constructing complex and reliable encryption systems that satisfy modern security requirements under the condition of choosing the appropriate algorithms, matrix sizes and protection methods. Nonetheless, high computational complexity can be attributed to the potential disadvantages of using matrices in cryptographic information exchange systems; high requirements for the volume of memory used; complexity of key management; vulnerability to certain types of attacks; message length and block size.

CONCLUSIONS

This study attempted to integrate methods of matching matrix key elements with factorial data coding, which relies on permutations to transfer information. For this purpose, a software model for transforming the square matrix A into the permutation π was developed and implemented. Two matrix transformation algorithms have been considered and tested: by representing the matrix A in decimal and factorial notations and by using auxiliary square matrices. The first transformation algorithm involves step-by-step transformation of matrix elements into coefficients of positional (decimal and factorial) notations. The second transformation algorithm involves successive multiplication from the left of the matrix A by all auxiliary matrices followed by generating factorial coefficients from the elements of the product matrices.

Analysis of existing cryptosystems built on matrices has revealed both advantages and disadvantages in applying matrix transformations to generate key elements of factorial coding. The study has also determined new vectors of developing and improving matrix transformations in systems with factorial data coding as a tool for key agreement. A potential weak point to implement an attack on a cryptographic system is encryption, which takes place in blocks of a fixed size. Despite the above-mentioned shortcomings, combinations of matrix cryptographic systems and factorial data coding can be effective and safe, provided they are used correctly, and possible risks are taken into account. Selecting appropriate algorithms, matrix sizes, and protection methods, as well as algorithms for converting matrices into permutations, can help minimise these flaws and ensure reliable information security.

Two experiments have been conducted for different dimensions of Z_p finite field of matrix elements: $p = 17$ and $p = 23$. Histograms of permutation distributions at the output of the simulated model have been constructed. The results of the study indicate that the transformation algorithm that involves representing the matrix A in decimal and factorial notations for large $\frac{p^{n^2}}{M!}$ has proved that this method can achieve a uniform distribution of possible permutations and can be used as a means of generating keys in systems with factorial coding. The second transformation algorithm with auxiliary matrices does not ensure a uniform distribution of the generated permutations and hence requires further research.

ACKNOWLEDGEMENTS

This research was funded by the Ministry of Education and Science of Ukraine under grant 0123U100270.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Al-Shaarani, F., & Gutub, A. (2022). Securing matrix counting-based secret-sharing involving crypto steganography. *Journal of King Saud University - Computer and Information Sciences*, 34(9), 6909-6924. doi: 10.1016/j.jksuci.2021.09.009.

[2] Bleichenbacher Attack Explained. (2019). Retrieved from https://medium.com/@c0D3M/bleichenbacher-attack-explained-bc630f88ff25.

[3] Chernoff, H., & Lehmann, E.L. (1954). The use of maximum likelihood estimates in x^2 tests for goodness of fit. *The Annals of Mathematical Statistics*, 25(3), 579-586. doi: 10.1214/aoms/1177728726.

[4] Gafsi, M., Abbassi, N., Hajjaji, M.A., Malek, J., & Mtibaa, A. (2020). Improved chaos-based cryptosystem for medical image encryption and decryption. *Scientific Programming*, 2020, article number 6612390. doi: 10.1155/2020/6612390.

[5] Greub, W. (1975). *Linear algebra*. New York: Springer. doi: 10.1007/978-1-4684-9446-4.

[6] Higham, D.J., & Higham, N.J. (2005). *MATLAB guide*. Philadelphia: Society for Industrial and Applied Mathematics.

[7] Huang, H., Li, C., & Deng, L. (2022). Public-key cryptography based on tropical circular matrices. *Applied Sciences*, 12(15), article number 7401. doi: 10.3390/app12157401.

[8] Issad, M., Anane, N., Bellemou, A.M., & Boudraa, B. (2020). Secure hybrid crypto-system AES/RSA on FPGA for data communication. *Malaysian Journal of Computing and Applied Mathematics*, 3(1), 1-10. doi: 10.37231/myjcam.2020.3.1.38.
[9] Joshi, S., Bairwa, A.K., Pljonkin, A.P., Garg, P., & Agrawal, K. (2023). From pre-quantum to post-quantum RSA. In *Proceedings of the 6th international conference on networking, intelligent systems & security* (pp. 1-8). doi: 10.1145/3607720.3607721.
[10] Kaptiol, Ye., & Horbenko, I. (2020). Analysis of the possibilities and peculiarities of programming cryptology problems on a quantum computer. *Radiotekhnika*, 3(202), 37-48. doi: 10.30837/rt.2020.3.202.03.
[11] Karatas, Z.Y., Luy, E., & Gonen, B. (2019). A public key cryptosystem based on matrices. *International Journal of Computer Applications*, 182(42), 47-50. doi: 10.5120/ijca2019918432.
[12] Kelesidis, E.-A. (2022). An optimization of Bleichenbacher's oracle padding attack. In P.Y. Ryan & C. Toma (Eds.), *Innovative security solutions for information technology and communications* (pp. 145-155). Cham: Springer. doi: 10.1007/978-3-031-17510-7_10.
[13] Lavdanskyi, A., Faure, E., Skutskyi, A., & Bazilo, C. (2023). Accelerating operations on permutations using graphics processing units. In E. Faure, O. Danchenko, M. Bondarenko, Y. Tryus, C. Bazilo & G. Zaspa (Eds.), *Information technology for education, science, and technics* (pp. 3-12). Cham: Springer. doi: 10.1007/978-3-031-35467-0_1.
[14] Li, J., Yan, M., Peng, J., Huang, H., & El-Latif, A.A.A. (2024a). A lattice-based efficient certificateless public key encryption for big data security in clouds. *Future Generation Computer Systems*, 158, 255-266. doi: 10.1016/j.future.2024.04.039.
[15] Li, P., *et al.* (2024b). Scalable parallel ultrafast optical random bit generation based on a single chaotic microcomb. *Light: Science & Applications*, 13(1), article number 66. doi: 10.1038/s41377-024-01411-7.
[16] Mann, H.B., & Wald, A. (1942). On the choice of the number of class intervals in the application of the Chi square test. *The Annals of Mathematical Statistics*, 13(3), 306-317. doi: 10.1214/aoms/1177731569.
[17] Maturin, Yu., Komarnytska, L., & Hordiienko, I. (2023). *Discrete math*. Drohobych: Drohobych Ivan Franko State Pedagogical University.
[18] Maxrizal, M. (2022). Public key cryptosystem based on singular matrix. *Trends in Sciences*, 19(3), article number 2147. doi: 10.48048/tis.2022.2147.
[19] Naseri, A.R., Abbasi, A., & Atani, R.E. (2023). A new public key cryptography using M_q matrix. *Journal of Mathematical Modeling*, 11(4), 681-693. doi: 10.22124/jmm.2023.23982.2142.
[20] National Institute for Standards and Technology. (2001). *Specification for the Advanced Encryption Standard (AES)*. Retrieved from https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf.
[21] Rani, S., Bhambri, P., Kataria, A., Khang, A., & Sivaraman, A.K. (2023). *Big data, cloud computing and IoT: Tools and applications*. Boca Raton: Chapman and Hall/CRC. doi: 10.1201/9781003298335.
[22] Sheikhpour, S., Mahani, A., & Bagheri, N. (2021). Reliable advanced encryption standard hardware implementation: 32- bit and 64-bit data-paths. *Microprocessors and Microsystems*, 81, article number 103740. doi: 10.1016/j.micpro.2020.103740.
[23] Shvydkyi, V., Shcherba, A., Kharin, O., Lavdanskyi, A., & Faure, E. (2021). *Basics theory of inseparable factorial data coding*. Kharkiv: Novyi Kurs.
[24] Turchyn, V. (2014). *Probability theory and mathematical statistics*. Dnipro: IMA-Press.
[25] Valentine, D.T., & Hahn, B.H. (2023). *Essential MATLAB for engineers and scientists*. London: Academic Press.

Програмна модель формування ключів-перестановок через квадратну матрицю

Еміль Фауре

Доктор технічних наук, професор
Черкаський державний технологічний університет
18006, бульв. Шевченка, 460, м. Черкаси, Україна
https://orcid.org/0000-0002-2046-481X

Анатолій Щерба

Кандидат фізико-математичних наук, доцент
Черкаський державний технологічний університет
18006, бульв. Шевченка, 460, м. Черкаси, Україна
https://orcid.org/0000-0002-3049-3497

Артем Скуцький

Аспірант
Черкаський державний технологічний університет
18006, бульв. Шевченка, 460, м. Черкаси, Україна
https://orcid.org/0000-0002-8632-1176

Артем Лавданський

Кандидат технічних наук, доцент
Черкаський державний технологічний університет
18006, бульв. Шевченка, 460, м. Черкаси, Україна
https://orcid.org/0000-0002-1596-4123

Анотація. Одним із важливих аспектів, які необхідно розвивати у 21 столітті, є безпека та захист інформації. Для криптографічних алгоритмів розповсюдженим підходом є використання матричного представлення інформації. Разом з тим, нові підходи можуть використовувати інші структури даних, в тому числі перестановки. Це викликає необхідність узгодження структур даних для інтеграції різних методів у одній системі обробки та передавання інформації. Метою роботи є формування перестановок, що є ключем для факторіального кодування даних, за відомою ключовою матрицею. У роботі представлено два алгоритми перетворення квадратної матриці в перестановку. Наведено приклад перетворення матриці за кожним з запропонованих алгоритмів. Створено та описано програмну модель для дослідження перетворення квадратних матриць у перестановки з використанням програмного продукту Matlab. Розглянуто вбудовані методи обробки статистичної інформації в програмі Matlab та графічного їх відображення за допомогою вбудованих функцій, які використані в процесі роботи програмної моделі. Виконано перетворення матриці за запропонованими алгоритмами. Досліджено всі можливі комбінації квадратної матриці порядку 2 з елементами, що належать скінченному полю цілих чисел за модулем $p = 17$ та $p = 23$. За кожним алгоритмом перетворення отримано результати перетворення квадратної матриці у номер перестановки в лексикографічному порядку їх слідування. Досліджено статистичні властивості отриманих результатів, визначено найбільш ефективний алгоритм перетворення матриць у перестановки за критерієм рівномірності розподілу отриманих номерів перестановок. Показано, що цей алгоритм має перспективу використання в системах обміну інформації з використанням факторіального кодування даних

Ключові слова: факторіальне кодування даних; захист інформації; узгодження ключів; статистика; рівномірний розподіл