EURASIAN JOURNAL OF MATHEMATICAL AND COMPUTER APPLICATIONS
ISSN 2306–6172
Volume 12, Issue 3 (2024) 35 – 50

MAZE BASED IMAGE ENCRYPTION METHOD CONSTRUCTED BY RANDOM NUMBER GENERATION

Gasimov V. A., Mammadzada N. F., Mammadov J. I., Mustafayeva E. A.

Abstract Due to the large size of image files and the high correlation between the numerical values of the pixels, various improved algorithms have been used for image encryption in recent years. In most cases improvement is obtained by taking into account the characteristics of chaotic and random processes in encryption algorithms. In this study, an algorithm is proposed that encrypts images based on a maze created by keys consisting of random numbers generated with Feigembaum's quadratic function. For the generation of the maze, deep first search (DFS) algorithm has been used. During the generation with this algorithm, numbers representing the walking paths and as well as the length of the used stack are included in the encryption process. Algorithm security verification was carried out according to key space and differential analysis, statistical analysis such as entropy estimation, and analysis of correlation between the histogram and neighboring pixels of the image.

Key words: randomness, data shuffling, chaos, image encryption, maze matrices.

AMS Mathematics Subject Classification: 94A60, 11T71, 68P25.

DOI: 10.32523/2306-6172-2024-12-3-35-50

1 Introduction

In modern age, the development of information technologies and the frequent use of image-type information have stipulated the increase of security in their transmission. The large volume of image-type information, and the high correlation within this information, necessitated to create new algorithms for their encryption, which are different from the classical methods. On the other hand, classical encryption systems (DES, AES, etc.) encrypt images at the level of bits, in other words, they do not provide encryption of image by pixels. This, in turn, increases the duration of the encryption process. Exactly for these reasons, the need arises for specially created algorithms for images that treat images as a two-dimensional matrix and minimize inter-pixel correlation as a result of encryption. In this regard, in recent years, studies have been carried out in the direction of developing algorithms for image encryption [1-3]. Majority of the recent use the features of deterministic chaotic systems to solve the cryptographic protection of images problem [4-5]. In general, deterministic chaotic systems are dynamic systems that depend exponentially on the initial conditions, and in such systems, a slight change in the initial state causes a significant change in the trajectory of the system in the phase plane, meaning that the change in the initial conditions increases exponentially with time [6]. Researchers paying attention to these features since the mid-90s of the last century, have already came to the conclusion that chaotic systems

can greatly contribute to the development of reliable cryptographic encryption algorithms [7]. This is explained by the non-linear transformation of information both in crytography and in chaotic systems. On the one hand, such a transformation is completely deterministic, because it is carried out by software or hardware on a computer, and on the other hand, it is practically unpredictable by an outside observer, which is a basic requirement in cryptosystems.

In cases where the images are based on chaotic processes, the randomness of the elements included in the ciphertext increases, and this ultimately significantly increases the efficiency of ensuring security. An image encryption where a bit-level permutation strategy is applied can be shown as an example of this [8]. In some methods based on chaotic processes, the prolem of image encryption using the structure of real deoxyribonucleic acid (DNA) molecules has been considered [9-12]. According to the image encryption algorithm suggested in [9], the encryption process is carried out by coordinates of a chaotic set of points obtained by a chaos game image of a DNA symbol sequence, the sequence of DNA symbols, and shuffling and displacement of the image pixels based on the coding order operations. And, in another research work, an image encryption algorithm obtained by using a sequence of DNA pseudosymbols obtained on the basis of real DNA symbols and a chaotic transformation function is given [10]. Using labyrinths as chaos in image encryption is also one of the interesting directions. The main feature here is that it is possible to create a single-use random keys according to the complex structure of the paths going through the labyrinth (or maze, which is a general name for labyrinths), and this, in its turn, means increasing the cryptodurability of the encryption process. In [13], the process of image encryption based on maze and pixel values is depicted. Here, it is meant that the maze is created based on the starting value defined by the user using the DFS algorithm. The encryption process is carried out by summing (XOR operation) each pixel value modulo 2 with the matrix elements obtained from the maze. In this work, we study the encryption method of images using a maze has been. In the proposed method, together with the sequence of paths obtained during the maze generation, both the color values and locations of the image pixels changing are implemented by using the length of the stack memory formed as a result of each path, which allows for a more reliable encryption process.

2 DFS method to generate maze

A maze means a structure with one or more inputs and outputs, branching in many directions, and having confusing paths that include choices and dead ends. A labyrinth is a special case of a maze. Finding the path from the starting point to the end point is a complicated process, therefore at each point it is necessary to check all possible paths in four directions. There are many algorithms for generating mazes. Examples of these can include Depth-First-Search (DFS), Randomized Prim's algorithm, Eller's algorithm, Hunt-and-Kill Algorithm, etc. The mazes generated by the DFS algorithm have a complex structure and from a cryptographic point of view are considered advisable to use. The general structure of the maze generated by this algorithm is presented in Figure 1.

A rectangular table of nxm size (consisting of n rows and m columns) is initially

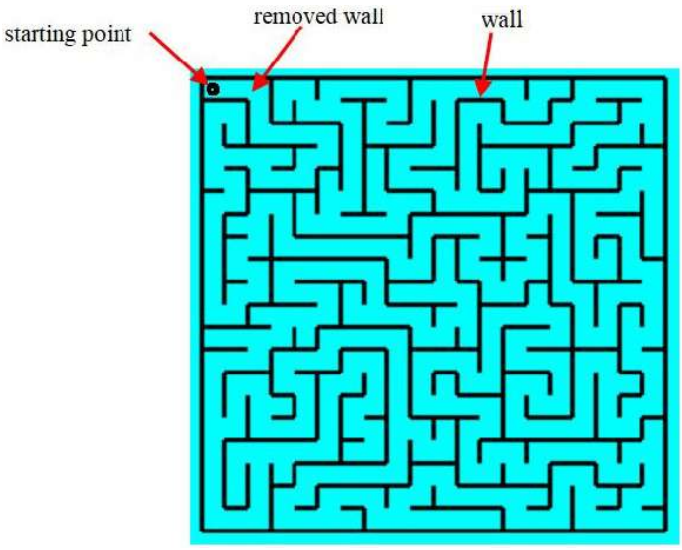


Figure 1: Maze structure.

created to generate the maze with the DFS algorithm (Fig. 2a). The position where lines and columns intersect is called a cell. The algorithm for generating the maze on this table is as follows:

1. Any cell is randomly selected within the table and the maze is generated starting from this cell.

2. From that cell, it is checked whether there are 4 directions (up, down, right, left) and the available directions are added to the temporary list. If an adjacent cell has been visited, that cell is not added to the list.

3. One direction is randomly selected from the list of available directions.

4. In relation to the current cell, a transition is made to the adjacent cell located in the selected direction and the list is emptied.

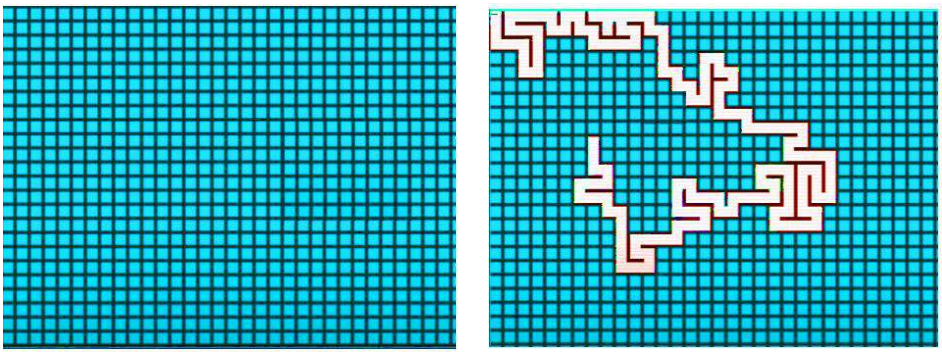
5. During the transition, the wall between the two cells is eliminated (Figure 2b) and the numbers of the cells to be eliminated are added to the stack memory.

6. If there is no direction from the current cell to walk, then the current cell is removed from the stack.

7. If there are cells in the stack, then you return to the last cell (before the current deleted cell) and pass to step 2.

8. If the stack is empty (it means, there are no cells left in the stack), then the maze generation process has been completed and the algorithm completes its work.

9. The end.



a) b)

Figure 2: a) Maze walls before being removed b) Removing walls during cells walking.

As mentioned above, the direction of transition from each current cell to the next cell is selected randomly. Here, to ensure random selection of starting point and directions during maze construction pseudo-random number generation (PRNG) algorithms are used.

3 Maze and maze matrices generation algorithm for encryption

The suggested image encryption method is realized through a maze created corresponding to the size of the image (to the number of pixels in the horizontal and vertical directions). A maze is generated according to the size of the image that is going to be encrypted (Fig. 1). For this purpose, to use the DFS algorithm is suggested. As a pseudo-random number generator Feigenbaum's logical map is used:

$$x_{n+1} = rx_n(1 - x_n). \tag{1}$$

Here, the coefficient r changes between 3.6 and 4. It should be noted that the coefficient r, the starting value x_0, as well as the coordinates of the starting point chosen for the construction of the maze are the quantities used during the generation of the secret key for the encryption and decryption processes. The secret key is agreed between the parties before each session through the exchange protocol described in [14]. Of course, standard key exchange protocols can also be used for this purpose. Maze generation algorithm is given below.

1. A table consisting of cells of the size apprporiate to the size of the image that is going to be encrypted is created. A two-dimensional matrix $M = \{m_{ij}\}_{nl}$ is created according to the size of the maze and zero is appropriated to all its elements: $m_{i,j} = o, i = \overline{1,n}, j = \overline{1,l}$.

2. $S = \{s_{ij}\}_{nl}$ matrix is created to store the lengths of the information in the stack memory according to each cell of the maze.

3. In order to generate the maze, the starting coordinate (cell) is randomly selected in the table (Feigembaum PRNG), for example, the cell at the intersection of line i and column j is selected. This cell is called the current cell.

4. The value 1 is given to the element of the matrix M corresponding to the position of the current cell.

5. The number (address) of the current cell is added to the stack. The value of the current length of the stack is given to the relevant element of the matrix $S : s_{ij} = len(Stack)$.

6. If there exist neighboring cells (up, down, left and right) that are reachable from the current cell (i.e., the corresponding element of the matrix M has the value 0), those cells are detected, their addresses are stored in the direction list, otherwise, go to step 12.

7. The value of the current cell is given to the variable $c : c = m_{ij}$

8. With the help of the Feigenbaum method (logical map), by generating a random number, one of the possible directions of movement from the current cell to neighboring cells is selected.

9. The switch is made to cell located in the selected direction, that cell is accepted as the current cell, and the direction list is emptied. For example, if the switch is made from cell (i, j) to cell $(i, j + 1)$, then the following update are conducted: $i = i$ and $j = j + 1$.

10. 1 is added to the value (c) of the element of the matrix M corresponding to the previous cell and is given to the element corresponding to the position of the current cell of the matrix: $m_{ij} = c + 1$.

11. Go back to step 5.

12. If it is not possible to move in any direction from the current cell, the address of that cell is eliminated from that stack. If the stack is not empty, then go back to the last cell in the list (before the current cell that has been deleted) and go to step 6, otherwise go to step 13.

13. The end.

The execution of the algorithm results in the generation of a maze appropriate to the size of the image that is going to be encrypted and a matrix appropriate to the maze. In the end, the elements of the matrix get values from 0 to $n \times m - 1$. The values of the elements of the matrix contain the trajectory of movement in the maze.

As an example, in Figure 3, a maze generated starting from cell $(0, 0)$ for encoding a 20×20 pixel image is shown.

The matrix formed for movement paths (routes) in such a maze, the values of its elements are shown in figure 4.

As can be seen from Figure 4, the elements of the generated matrix take unique values in the interval from 0 to $20 \times 20 - 1$ $(n \times m - 1)$. In other words, two separate

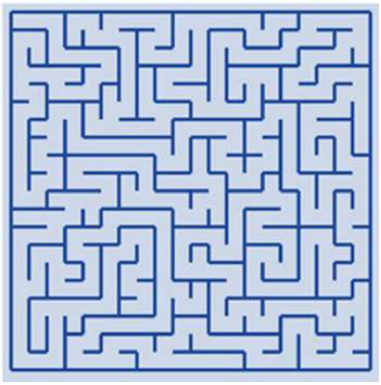


Figure 3: Generated 20×20 maze with DFS algorithm and Feygembaum PRNG.

0	18	17	16	21	24	25	26	27	28	29	355	319	318	317	316	356	357	358	359
1	19	20	15	22	23	399	35	34	31	30	354	320	313	314	315	382	383	384	360
2	3	4	13	12	38	37	36	33	32	325	322	321	312	311	310	364	363	362	361
156	157	5	14	11	39	40	41	331	330	324	323	304	305	385	309	365	366	286	285
155	154	6	9	10	167	166	42	332	329	326	353	303	306	307	308	367	288	287	284
152	153	7	8	164	163	165	43	333	328	327	301	302	371	370	369	368	289	282	283
151	158	159	160	161	162	168	44	334	339	338	300	373	372	377	378	381	290	281	390
150	147	146	175	174	173	169	45	335	336	337	299	374	375	376	379	380	291	280	279
149	148	145	129	128	171	170	46	47	352	340	298	297	296	295	294	293	292	277	278
142	143	144	130	127	172	50	49	48	351	341	342	232	233	234	386	387	275	276	391
134	133	132	131	126	125	51	350	349	348	344	343	231	236	235	389	388	274	273	392
135	120	121	122	123	124	52	55	56	347	345	346	230	229	237	240	241	242	272	393
136	119	118	180	181	176	53	54	57	58	225	226	227	228	238	239	244	243	271	394
137	138	117	179	178	177	64	63	60	59	224	218	217	210	209	208	245	246	270	269
140	139	116	182	183	66	65	62	61	223	222	219	216	211	212	207	248	247	395	268
141	114	115	101	100	67	68	69	70	71	221	220	215	214	213	206	249	265	266	267
112	113	103	102	99	90	85	86	87	72	73	74	202	203	204	205	250	264	263	262
111	184	104	97	98	91	84	89	88	77	76	75	201	199	198	197	251	252	396	261
110	106	105	96	93	92	83	82	81	78	398	190	191	200	195	196	254	253	259	260
109	107	108	95	94	185	186	187	80	79	188	189	192	193	194	256	255	257	258	397

Figure 4: Path route of the maze described in figure 3.

elements in the matrix cannot have the same value, and each number in this interval participates only once in this matrix. This, in turn, allows to use the matrix generated from the paths through the maze as a displacement table. Thus, to mix the pixels of the image, it is possible to use the elements of the maze matrix, which have values in the interval $[0, n \times m - 1]$ standing against each pixel of the image that is going to be encrypted. At the same time, it should be noted that during the generation of the maze, the length value of the stack memory that is used to organize the visits of the cells in the DFS algorithm changes randomly for the current cells, and these values can be used for the purpose of generating random numbers, and also in the encryption process. in Figure 5, the image of the matrix which is formed from the current lengths of the stack memory during the generation of the 20×20 maze is given.

4 Image encryption algorithm through Maze

The maze and the corresponding matrices are generated by the algorithms that are described in the previous sections. But in this section, the image encryption algorithm that uses these obtained matrices is viewed. So, the following matrices are used in the $m \times n$ sized image encryption algorithm:

- $M = \{m_{ij}\}_{nl}$ - the matrix that show the movement directions in the maze and

0	17	16	15	16	19	20	21	22	23	24	207	202	201	200	199	200	201	202	203
1	18	19	14	17	18	33	30	29	26	25	206	203	196	197	198	209	210	211	204
2	3	4	13	12	33	32	31	28	27	208	205	204	195	194	193	208	207	206	205
139	140	5	14	11	34	35	36	213	212	207	206	187	188	195	192	209	210	169	168
138	137	6	9	10	145	144	37	214	211	208	209	186	189	190	191	210	171	170	167
135	136	7	8	143	142	143	38	215	210	209	184	185	214	213	212	211	172	165	166
134	137	138	139	140	141	144	39	216	221	220	183	216	215	220	221	224	173	164	167
133	130	129	150	149	148	145	40	217	218	219	182	217	218	219	222	223	174	163	162
132	131	128	119	118	147	146	41	42	229	220	181	180	179	178	177	176	175	160	161
125	126	127	120	117	148	45	44	43	228	221	222	117	118	119	178	179	158	159	162
124	123	122	121	116	115	46	229	228	227	224	223	116	121	120	181	180	157	156	163
125	110	111	112	113	114	47	50	51	226	225	226	115	114	121	124	125	126	155	164
126	109	108	119	120	115	48	49	52	53	110	111	112	113	122	123	128	127	154	165
127	128	107	118	117	116	59	58	55	54	109	104	103	96	95	94	129	130	153	152
130	129	106	119	120	61	60	57	56	109	108	105	102	97	98	93	132	131	154	151
131	104	105	92	91	62	63	64	65	66	107	106	101	100	99	92	133	148	149	150
102	103	94	93	90	81	80	81	82	67	68	69	88	89	90	91	134	147	146	145
101	104	95	88	89	82	79	84	83	72	71	70	87	86	85	84	135	136	147	144
100	97	96	87	84	83	78	77	76	73	78	77	78	87	82	83	138	137	142	143
99	98	99	86	85	86	87	88	75	74	75	76	79	80	81	140	139	140	141	144

Figure 5: Stack lengths of maze described in figure 3.

that contains the values corresponding to each cell of the maze;

- $M^{'} = \left\{m_i^{'}\right\}_{n\times l}$ - $M = \{m_{ij}\}_{nl}$ a one-dimensional matrix obtained by writing the matrix elements successively (line by line);
- $S = \{s_{ij}\}_{nl}$ - a matrix that contains the stack length values according to each cell of the maze;
- $S^{'} = \left\{s_i^{'}\right\}_{n\times l}$ - $S = \{s_{ij}\}_{nl}$ a one-dimensional matrix obtained by writing the elements of the matrix successively (line by line);
- $M_R = \left\{m_i^R\right\}_{n\times l}$, $M_G = \left\{m_i^G\right\}_{n\times l}$, $M_B = \left\{m_i^B\right\}_{n\times l}$ - one-dimensional matrices that contain the values of the RGB color channels (Red, Green, Blue) of the encrypted (initial) image pixels (the values are read line by line and entered into those matrices);
- $M_{RE} = \left\{m_i^{RE}\right\}_{n\times l}$, $M_{GE} = \left\{m_i^{GE}\right\}_{n\times l}$, $M_{BE} = \left\{m_i^{BE}\right\}_{n\times l}$ - one-dimensional matrices that contain the values of the RGB color channels (Red, Green, Blue) of the image pixels during the encryption process;
- $M_{RE2} = \left\{m_i^{RE2}\right\}_{n\times l}$, $M_{GE2} = \left\{m_i^{GE2}\right\}_{n\times l}$, $M_{BE2} = \left\{m_i^{BE2}\right\}_{n\times l}$ - two-dimensional matrices that contain the values of the RGB color channels (Red, Green, Blue) of the encrypted image pixels;

The algorithm for encryption of images through maze is as follows:

1. The elements of the matrix $M^{'}$ are used to encrypt the initial image. So, for the encryption purpose, the pixels of the initial image are mixed and written together with the pixels of the encrypted image. In other words, the pixel corresponding to the value of the relevant element of the $M^{'}$ matrix of the initial image is written to the pixel of the encrypted image. For this purpose, the value of the matrix $M^{'}$ appropriate to the current pixel i of the initial image is taken. The values of the pixel corresponding to $m_i^{'}$ element of the initial image are given to the elements number i of the M_{RE}, M_{GE}, M_{BE} matrices of the encrypted image:

$$m_i^{RE} = m_i^R \left[m_i^{'}\right]$$

$$m_i^{GE} = m_i^G \left[m_i^{'} \right]$$

$$m_i^{BE} = m_i^B \left[m_i^{'} \right]$$

.

2. In the first step, by changing the places of the pixels of the initial image, the values of the pixels of the encrypted image (M_{RE}, M_{GE}, M_{BE}) and the corresponding values of the stack $S^{'} = \left\{ s_i^{'} \right\}_{n \times l}$ are collocted by XOR operation and the obtained values are given to the matrices M_R, M_G, M_B :

$$m_i^R = m_i^{RE} \oplus s_i^{'}$$

$$m_i^G = m_i^{GE} \oplus s_i^{'}$$

$$m_i^B = m_i^{BE} \oplus s_i^{'}$$

3. 1-st and 2-nd steps are repeated 5 times.

4. Two-dimensional matrices M_{RE2}, M_{GE2}, M_{BE2} are formed by grouping the elements of matrices M_R, M_G, M_B without changing their values, with the number of blocks appropriate to the width of the initial image.

5. A new image is formed by taking the values of the RGB color channels from the appropriate elements of the M_{RE2}, M_{GE2}, M_{BE2} matrices. In this way, an encrypted image is obtained.

6. The end.

It should be noted that during the generation of the maze, the values being unique in the interval $[0, m \times n - 1]$ (m and n are the width and length of the image) of the sequence of paths allow you to successfully mix the pixels of the image. In addition to this, changing the values of the image pixels with the help of a matrix that consists of stack memory lengths allows to provide more reliable encryption. At the same time, it was detected that a more effective (reliable) result is obtained in terms of security when the process of changing the places of the pixels and considering the stack values is repeated at least 5 times in the encryption process.

The decrypting process of encrypted images is performed in a similar manner, using the same key. Here, as in the encryption process, the corresponding generated maze is used. The only difference is that, contrary to the sequence in the encryption process, in the decrypting process, taking the values of the color channels, grouping the elements of the matrix, XOR operations on the numerical values of the pixels, and displacement operations are performed in the opposite direction.

5 Software of the suggested method

The software of the suggested method was realized in the Visual Studio environment using the c# programming language. Here, during the encryption of the image, first of all, the "Select image" button is pressed from the "Form1" program window (Fig. 6) and the image that is required to be encrypted is selected from the relevant folder. The selected image is shown in the "Plain image" area. In the next step, the encryption key is added in the "Encryption key" area and the "Encryption" button is pressed. And with that a multi-stage encryption process starts by mixing the pixels and changing their values. The result of the process, that is, the encrypted image, is removed to the "Encryption image" area of the window and is also stored in the folder intended by the program.

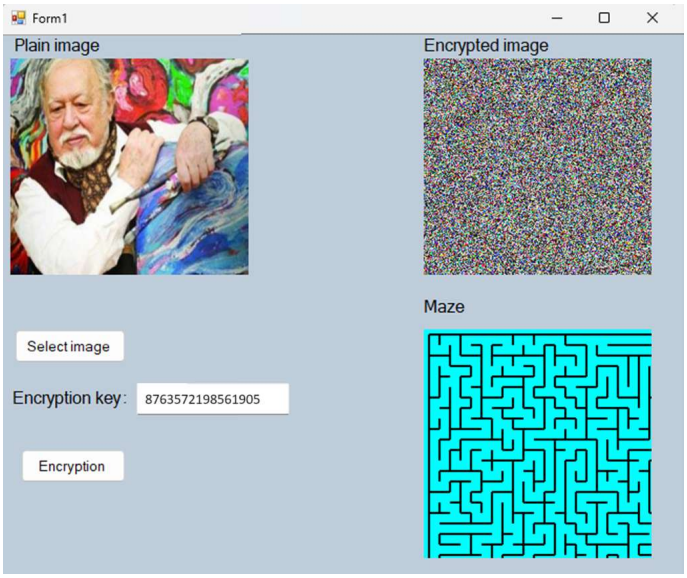


Figure 6: Encryption process.

It should be noted that the visual image of the maze, which is used in the process to control the progress of the encryption process and is established on the basis of PRNG, is shown in the "Maze" area of the program window. Maze generation starts by pressing the "Encryption" button, and the encryption key components are used as initial parameters. Since the suggested method is a symmetric encryption method, the program window of the decryption process (fig. 7) has a similar structure. During the decryption, the encrypted image file is selected from the relevant folder with the "Select encryption image" button in the "Form2" window and removed to the "Encryption image" area, and after entering the decryption key (it is the same as the encryption key) in the "Decryption key" area, the "Decryption" button is pressed. After the process is completed, the decrypted image along with being extracted to the "Decryption image" area also is written to the relevant folder of the computer memory. Here, the "Maze" area is intended for the visual image of the generated maze to control the process.

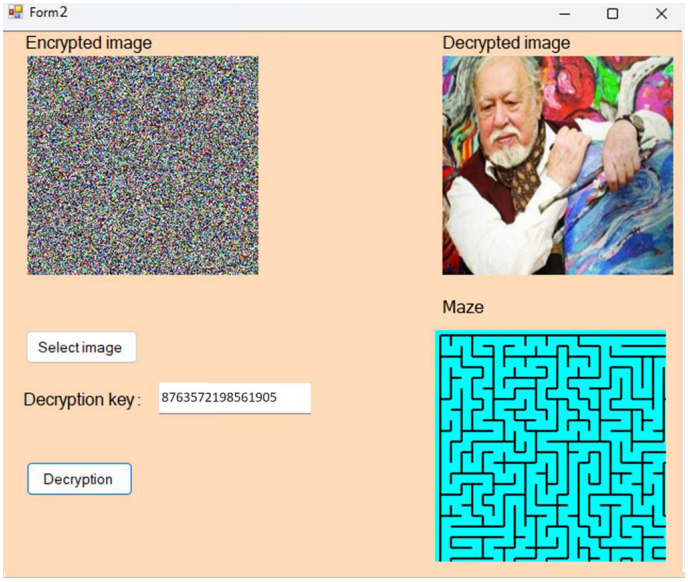


Figure 7: Decryption process.

6 Security analysis

There are many approaches to check the effectivity and reliability of image encryption methods: key space analysis, key sensitivity defining for differential analysis, histogram analysis for statistical analysis, correlation analysis between neighboring pixels of an image, entropy calculation, etc.

6.1 Key space analysis

As in any encryption method, it is important for image encryption that the key length is sustainable against a brute-force attack. In the suggested algorithm, 128 bits key length is selected, and this allows to provide the necessary stability. In order to further increase the sustainability of the algorithm, the key length can be chosen even larger.

6.2 Key sensitivity analysis

It has always been one of the important questions how a small (even 1 bit) change on the secret key that is used during encryption process affects the ciphertext. For an encryption method to be crypto-durable, a small change on the key must lead to a large change in the ciphertext. To check the key sensitivity of the suggested method, a test was performed. Thus, each image is first encrypted with the key K1, and then the same image is encrypted with a different key K2 has been obtained by making a change by 1 bit in the key K1. In this way, substantially different encrypted images were obtained as a result of encryption of the same image using slightly (1 bit) different keys (Figure 8.)

It should be noted that after encrypting the initial image with the key K1, if we decrypt it with the key K2, which differs from it by 1 bit, the image becomes completely

(a) (b) (c)

Figure 8: a-original image, b-image encrypted with the key (K1 = 8763572198561905), c- image encrypted with the key (K2 = 8763572198561805).

different from the initial image (figure 9).

(a) (b) (c)

Figure 9: a-image encrypted with the key (K1=8763572198561905), b-image decrypted with valid key (K1=8763572198561905), c-image decrypted with invalid key (K1=8763572198561805).

6.3 Statistical analysis

In order to analyze the algorithm's resistance to cryptanalysis, as a statistical method, histogram analysis and entropy measurement, correlation evaluation methods between horizontal and vertical adjacent pixels are used.

6.3.1 Histogram analysis

Histogram analysis of the image is a method of expressing the distribution of pixels over each color channel. While the distribution of colors over the histogram in the initial image meets a certain regularity, the distribution over each color channel should be uniform in order to be resistant to histogram analysis in the encrypted image. During the analysis conducted, the histogram analysis of the initial image is shown in figure 10, and the histogram analysis of the encrypted images that uses our suggested method is shown in figure 11.

(a) Red (b) Green (c) Blue

Figure 10: Histogram of the initial image colors.

(a) Red (b) Green (c) Blue

Figure 11: Histogram of the encrypted image colors.

6.3.2 Information entropy analysis

One of the important problems in the image encryption is the randomness of the information in the encrypted image. It is also considered a factor that represent the average amount of information in the ciphertext. The S - entropy value for a ciphertext is calculated using the following formula:

$$H(s) = -\sum_{i=0}^{255} P(i) log_2(P(i))$$

Here $P(i)$ is the probability to find each i value ($0 \leqslant i \leqslant 255$) on 3 separate color channels. Entropy on 10 initial images was calculated by the above given formula and it has been found that this value varies between 4.2 - 7.5 After the encryption process, these values increased and became in the interval of 7.9998 - 7.9999, and this indicates that the security of the encryption algorithm is enough.

6.3.3 Correlation coefficient analysis

In normal images, there is a high correlation among the horizontal, vertical and diagonal adjacents of pixels. In order to check the durability of the suggested algorithm, the correlation of the adjacent pixels in the encrypted image has been calculated using the following formula:

$$p_c(X,Y) = \frac{cov(X,Y)}{\sqrt{D(x)D(y)}}$$

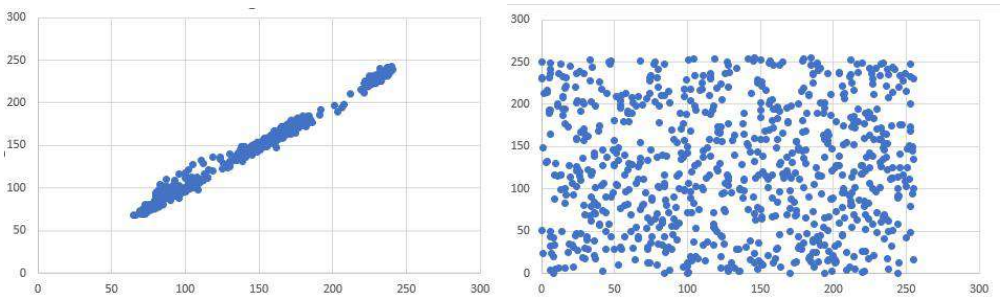
The elements in this formula are calculated as follows:

$$cov(X,Y) = \frac{1}{n}\sum_{i=0}^{n} [x_i - E(x)]\,[y_i - E(y)]$$

$$D(x) = \frac{1}{n}\sum_{i=0}^{n}[x_i - E(x)]^2$$

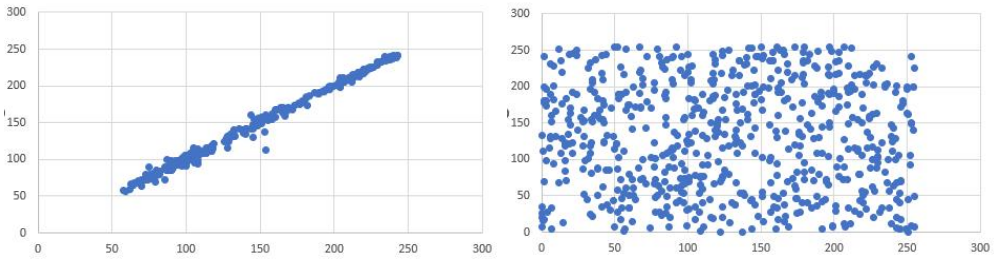
$$E(x) = \frac{1}{n}\sum_{i=0}^{n} x_i$$

Here, C-is the color channel, $D(X)$, $D(Y)$ is the variation, $cov(X, Y)$ is the covariance of two random variables. Using the suggested method, tests has been carried out on 10 images encrypted, and it has been found that the correlation coefficient varies between [-0.002,0.00093]. This result showed that there is a very weak correlation between the pixels of the encrypted image. A graphical image of the correlation between pixels is shown in figure 12-14.



(a) (b)

Figure 12: Graphic of correlation dependency of horizontal adjacent pixels of initial (a) and encrypted images (b).

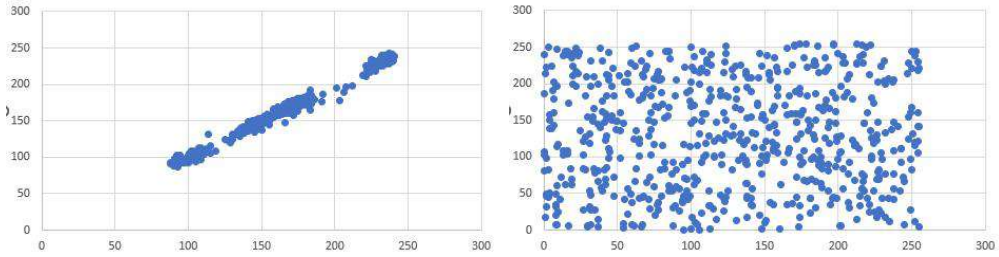


(a) (b)

Figure 13: Graphic of correlation dependency of vertical adjacent pixels of initial (a) and encrypted images (b).

In figure 12, the correlation between horizontally adjacent pixels is shown. It can be seen from the figure that while there is a strong correlation between adjacent pixels in the initial image, the correlation dependence becomes much weaker in the encrypted image. In the same figure, it is possible to see similar results in the image between vertical (figure 13) and diagonal (figure 14) pixels.

Analysis of correlation coefficients and dependencies of encrypted images have shown that the suggested algorithm is resistant to statistical cryptoanalysis.



(a) (b)

Figure 14: Graphic of correlation dependency of diagonal adjacent pixels of initial (a) and encrypted images (b).

6.3.4 Analysis with NIST tests

The statistical performance of the proposed algorithm was also verified through NIST tests. For this purpose, 7 tests were used and experiments were conducted on ciphertexts of different sizes. The results of all executed tests were positive. The table below shows the results of checking 5number of 100x100 ciphertexts. It is known that the result of the NIST test is measured by the P-value value, and the results of the tests are considered successful if this quantity is greater than 0.01 for the samples used. The average value of the P-value according to the results of the performed checks is given in the 2nd column of the table. As can be seen from the table, successful results were obtained for all tests.

Table 1: Results of NIST tests

Test	Result value
Frequency (Monobit) Test	P-value= 0.799
Frequency Test within a Block	P-value= 0.569
Test for the Longest Run of Ones in a Block	P-value= 0.151
Cumulative Sums Test	P-value Forward= 0.900 P-value Reverse= 0.674
Runs Test	P-value= 0.998
Approximate Entropy Test	P-value=0.303

6.4 Time consumption analysis

The time evaluation of the proposed algorithm was performed on a medium-sized image using an Intel®Core^{TM}i7-1260P(16 CPUs) 2.1GHz computer. As a result of the analysis, the total time spent on the image encryption process was 0.1-0.2 seconds, including the time spent on creating the relevant maze. It should be noted that the speed of the encryption process can be increased many times directly by using computers with higher parameters and performing some calculations in advance.

7 Conclusion

A method of image encryption that uses parameters representing the complex structure of the maze generated on the basis of random numbers has been suggested and software has been created for the realization of this method. During the generation of the maze, each value appears exactly once among the sequence of paths walked, which makes it possible to mix the pixels of the images successfully. At the same time, for changing the color values of the image pixels, a matrix was used, the elements of which are made up of the lengths of the stack memory, and this made it possible to further increase the resistance of the ciphertext to cryptanalysis. Pixel mixing and changing its values are repeated many times to bring the encryption algorithm to the required level of resistance to cryptanalysis. According to the results of the experiments carried out, at least 5 times repetitions are needed to ensure that the satisfactory results are obtained. Cryptoanalytical analyzes have been carried out with the help of many methods that are considered important for evaluating the reliability of the suggested algorithms for image encryption, and their cryptoresistance and the effectiveness of its results were approved.

References

[1] Ozturk I., Sogukpinar I., *Analysis and comparison of image encryption algorithms*, World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering, Vol.1 Issue 3 (2007), 814-817 pages.

[2] Potdar V., Chang E., *Disguising text cryptography using image cryptography*, International Network Conference in Plymouth, UK, 6 - 9 July, 2004.

[3] Mao SY., Chen G., Lian S., *A novel fast image encryption scheme based on 3D chaotic Baker maps*, International Journal Bifurcation and Chaos, Vol. 14 Issue 10 (2004), 3613-3624 pages.

[4] Boltenkov V.A., Nicolsky E.S., *Analysis of algorithms for chaotic images encrypting*, Digital technologies, Vol.7 Issue (2010), 61-66 pages. (in Russian)

[5] Mammadov J.I., Namazov F.H., *Development of image encryption algorithm using chaotic sequences and fractals*, ASOIU, AHTS News, Vol.2 Issue (2018), 85-94 pages.

[6] Crownover R.M. , *Fractals and chaos in dynamic systems*, Moscow, 2000. (in Russian)

[7] Ptitsyn N., *Application of the theory of deterministic chaos in cryptography*, Moscow: BMSTU, 2002. (in Russian)

[8] Hu G., Xiao D., Zhang Y. and Xiang T., *An efficient chaotic image cipher with dynamic lookup table driven bit-level permutation strategy*, Nonlinear Dynamics, Vol. 87 Issue 2 (2017), 1359-1375 pages.

[9] Gasimov V.A., Mammadov J.I., *DNA-based image encryption algorithm*, IOP Conf. Series: Materials Science and Engineering, Vol. 734, IOP Publishing (2020), 1-11 pages.

[10] Gasimov V., Mammadov J., *Image encryption algorithm using DNA pseudo-symbols and chaotic map*, 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications, IEEE Turkey Section, (2021), 1-5 pages.

[11] Gasimov V.A., Mammadov J.I., Hasanova A.A., *Symmetric DNA encryption algorithm based on relative index*, Modern Movement of Science: abstracts of the 12th International Scientific and Practical Internet Conference, (2021), 53-55 pages.

[12] Akkasaligar P.T., Biradar S., *Medical Image Compression and Encryption using Chaos based DNA Cryptography*, 2020 IEEE Bangalore Humanitarian Technology Conference, (2020), 1-5 pages.

[13] Guvenoglu E., *Maze Based Image Encryption Algorithm*, International Research Journal of Engineering and Technology (IRJET), Vol.2 Issue 8 (2015), 1578-1585 pages.

[14] Gasimov V.A., Mammadzada N.F., Mammadov J.I., *New Key Exchange Protocol Based on Matrix Algebras*, 5th International Conference on Problems of Cybernetics and Informatics (PCI), (2023), 1-3 pages. https://doi.org/10.1109/PCI60110.2023.10326004, https://ieeexplore.ieee.org/document/10326004

Gasimov V.,
Azerbaijan Technical University,
H. Javid avenue 25, Baku, Azerbaijan AZ 1073,
Email: vaqif.qasimov@aztu.edu.az,

Mammadzada N.,
Azerbaijan Technical University,
H. Javid avenue 25, Baku, Azerbaijan AZ 1073,
Email: mammadzada.nargizw@gmail.com,

Mammadov J.,
Azerbaijan Technical University,
H. Javid avenue 25, Baku, Azerbaijan AZ 1073,
Email: cabir.memmedov@aztu.edu.az,

Mustafayeva E.,
Azerbaijan Technical University,
H. Javid avenue 25, Baku, Azerbaijan AZ 1073,
Email: esmira.mustafayeva@aztu.edu.az,

Received 31.03.2024, received 21.05.2024, Accepted 25.05.2024