Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.035856 Published Online: 28 September 2023

ARTICLE

Efficient Technique for Image Cryptography Using Sudoku Keys

M. A. P. Manimekalai[1], M. Karthikeyan[1], I. Thusnavis Bella Mary[1], K. Martin Sagayam[1], Ahmed A Elngar[2], Unai Fernandez-Gamiz[3] and Hatıra Günerhan[4,*]

[1]Department of ECE, Karunya Institute of Technology and Sciences, Coimbatore, 641114, India

[2]Faculty of Computer and Artificial Intelligence, Beni-Suef University, Beni-Suef, 62511, Egypt

[3]Nuclear Engineering and Fluid Mechanics Department, University of the Basque Country, Bilbao, 48940, Spain

[4]Department of Mathematics, Faculty of Education, Kafkas University, Kars, 36100, Turkey

*Corresponding Author: Hatıra Günerhan. Email: gunerhanhatira@gmail.com

Received: 07 September 2022 Accepted: 27 February 2023

ABSTRACT

This paper proposes a cryptographic technique on images based on the Sudoku solution. Sudoku is a number puzzle, which needs applying defined protocols and filling the empty boxes with numbers. Given a small size of numbers as input, solving the sudoku puzzle yields an expanded big size of numbers, which can be used as a key for the Encryption/Decryption of images. In this way, the given small size of numbers can be stored as the prime key, which means the key is compact. A prime key clue in the sudoku puzzle always leads to only one solution, which means the key is always stable. This feature is the background for the paper, where the Sudoku puzzle output can be innovatively introduced in image cryptography. Sudoku solution is expanded to any size image using a sequence of expansion techniques that involve filling of the number matrix, Linear X-Y rotational shifting, and reverse shifting based on a standard zig-zag pattern. The crypto key for an image dictates the details of positions, where the image pixels have to be shuffled. Shuffling is made at two levels, namely pixel and sub-pixel (RGB) levels for an image, with the latter having more effective Encryption. The brought-out technique falls under the Image scrambling method with partial diffusion. Performance metrics are impressive and are given by a Histogram deviation of 0.997, a Correlation coefficient of 10^{-2} and an NPCR of 99.98%. Hence, it is evident that the image cryptography with the sudoku kept in place is more efficient against Plaintext and Differential attacks.

KEYWORDS

Sudoku; image cryptography; pixels; performance metrics

1 Introduction

In the modern world, technological advancement makes our daily lives very convenient, so global access to information is very much possible. The Multimedia domain has paved the way for online transactions and processing. On the other hand, the threat imposed on information security is big. Hence, improving information security and overcoming information leakage are critical tasks for technology users. As far as information carrying ability is concerned, the image data is superior to the text since the images have the advantage of holding large volumes of information via efficient techniques Traditional encryption algorithms such as Data Encryption standards and Rivest-Shamir-Adleman algorithms offer solutions in text cryptography. However, such solutions have partially met the criteria of data security and time objectives through respective cryptography techniques. Hence image cryptography has become more effective and popular in the research field [1]. Chaos-based Image encryption is a research field. With the rise of Metaverse, the Encryption of 3D models is more focused. One of the approaches is the 3DME-SC method, where 2D chaotic systems using a logistic map, infinite collapse (2D-LAIC), and semi-tensor product (STP) theory are used. Although chaos is complex, the 3DME-SC exhibits good performance and effectiveness [2]. In the field of watermarking, Fractional-order continuous orthogonal moments (FrCOMs) is a recent research topic, limited to planar images. One of the approaches extends the application to stereoscopic images by combination with Trinion theory [3].

Image encryption is categorized into two methods, namely Scrambling and Diffusion, based on encryption strategy. The pixel locations are interchanged or shuffled in the former technique. Hence, the correlation between neighboring pixels will be eliminated and result in efficient Encryption. The pixel values are subjected to changes in the Diffusion technique so that the property of the original image is broken and the pixel correlation is lost. By transposition, the encrypted image has less chance of data loss during decryption. By value transformation, the reconstruction of images may suffer significant variation from the actual image [4]. These factors are considered during the encryption algorithm selection process.

The purpose of this work is to attempt cryptography on images of any size using a number matrix defined by a well-known sudoku protocol. The primary hypothesis considered for this work is that Sudoku can encrypt and decrypt images without compromising the encryption robustness. The number matrix in a solved sudoku puzzle forms the base for the key construction. The Property of this matrix is that numbers do not get duplicated within the defined pattern. This scheme is a simple solution for generating the shuffled number matrix that defines the pixel position for encryption. Rotational shifting of the cells and reverse shifting techniques across a full matrix add ruggedness to the crypto key.

The challenges imposed on the crypto key for such encryption are compact keys, easily processable, the robustness of encrypted products over plaintext and differential attacks on data, etc. The key must also be capable of encrypting images of any size. With these factors as mandatory constraints, Sudoku is used in cryptography in this work. Sudoku must be interpreted as a controlled non-repetitive random number sequence, which can be predicted from the intermediate numbers. When the complexity of the number puzzle increases, the volume of the intermediate numbers decreases. Encryption has been tried using various entities like DNA, fingerprint patterns, etc. Sudoku can also be visualized as a suitable crypto key for image encryption.

2 Literature Survey

A Literature survey is conducted to explore the image encryption techniques and sudoku concepts. Image encryption techniques have drastically improved to maintain image confidentiality from data security threats. Image encryption techniques are compared to assess the encryption efficiency, and choose the best suitable cryptography technique. The trade-off between the Encryption complexity i and the computation time is the critical factor for choosing the suitable encryption technique. Fast processing technique, the encryption complexity may not satisfy the required encryption ruggedness. Pixel permutation assists in guarding the multimedia data. Many algorithms emerge to improve speed and data protection [4].

An algorithm by the name of Rijndael was applied, along with image shuffling. The image is grouped under small sections of 4×4 pixels, and their location is shuffled, followed by restructuring so that a wholly shuffled image is obtained. Data pre-shuffling before implementing the Rijndael algorithm ensures that the encrypted image becomes alien compared with the original image. When the shuffling pattern is inversely applied along with the Rijndael decryption, the actual image is obtained. These techniques must perfectly suit the Cryptography requirements in a private environment [5,6].

An affine transform is used to cross-locate the image pixels, followed by exclusive OR-ing. The result is an efficient, simple encrypted image. The 64-bit key is used in this technique. The first 32 bits (4×8-bit key) are used for the location transformation via the affine transformation, and the second 32 bits (4×8-bit key) are used for exclusive OR-ing on every 2×2 clusters of pixels. Thus the output image becomes completely decorrelated from the actual image [7].

Another method involves Scrambling, where the original image is broken based on bit-planes. Resultant images are shuffled using a suitable encryption technique. Shuffled images are used to reconstruct the image based on bit-planes, and hence the encrypted image is obtained. This technique exhibits good encryption efficiency when compared with conventional scrambling techniques, due to the introduction of bit-plane-based image decomposition [8].

A contemporary technique is proposed, where the RGB sub-pixel values are shuffled for the given image without disturbing the sub-pixel values. For an image of size $m * n$, this shuffling retains the image sub-pixels as such but with each pixel's components distributed to random locations resulting in a completely different output image. The adjacent pixel correlation between pixels, as well as sub-pixels, gets broken. Any two sub-pixel shifts are not alike, and the shuffling pattern is also different between every sub-pixel matrix as a whole [9].

The Sudoku gains attention in the signal processing domain as it is analogous to the error-correcting codes in terms of a discrete characteristic problem. The techniques used to resolve such error-correcting codes are applied to sudoku with suitable transformation. This analogy indicates that solutions for sudoku are alternatively applied to error-correcting codes and provide a different approach to solution estimation. The similar nature is possible due to the perspective approach and the possible representations of a problem. Such new perspectives are compared with Sudoku for innovative solutions [10]. Recent research has been done [11–18].

3 Sudoku-Based Cryptography

Sudoku is a puzzle in which different symbols (usually digits 1 through maximum) are arranged in an array such that the arrangement agrees with given clues and meets the puzzle constraints [19]. Sudoku is a discrete characteristic problem that must be seen beyond a simple number puzzle. Sudoku is short for "Su-ji wa dokushin ni kagiru" that is translated as "the numbers must be single". 6,670,903,752,021,072,936,960 different sudoku puzzles and equivalent solutions can be made for a 9×9 Sudoku [20]. If the Sudoku solution is used for Image encryption, there are plenty of provisions to choose one of these as a key for encryption.

Sudoku is bound by the rule that Every row, column and Block of cells has every value only once. These characteristics ensure that an N-element sudoku solution can generate an $N * N$ number sequence. A proper sudoku is a single solution matrix, which means that for a given sudoku clue, there exists one and only one solution. For image encryption, this is a mandatory requirement to avoid ambiguity in deriving the crypto key. The complexity of Sudoku is determined by many patterned terminologies like Single candidate, X-wing, forcing chain, swordfish, etc. [21,22]. These terminologies refer to the clue pattern. These concepts need to be taken into consideration for key compacting. Key compacting is not discussed in this paper; instead, the focus is more on the Encryption technique.

Two block diagrams correspond to key generation and encryption schemes (Refer to Figs. 1 and 2). Key generation from a sudoku solution matrix is the first block flow, while the encryption concept is the second block flow. Sudoku solution ($n \times n$ size) has to be converted to a proper crypto key ($M \times N$ size) which involves Expansion, Sequencing and Shuffling. The result of the process is a crypto key to the required image size.

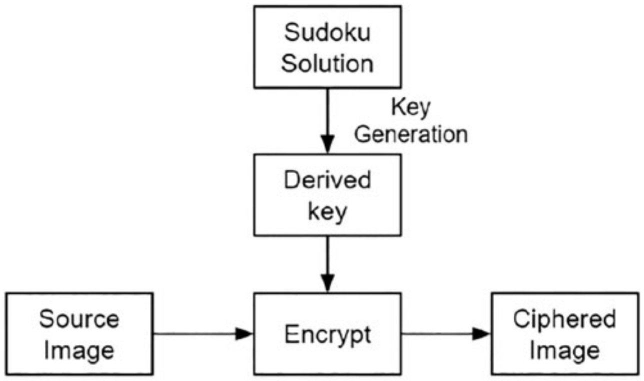


Figure 1: Block diagram-image ciphering

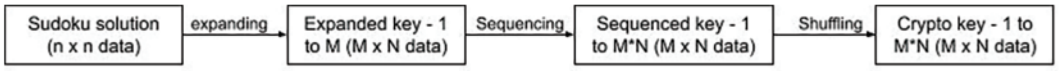

Sudoku solution (n x n data)
expanding
Expanded key - 1 to M (M x N data)
Sequencing
Sequenced key - 1 to M*N (M x N data)
Shuffling
Crypto key - 1 to M*N (M x N data)

Figure 2: Block diagram-crypto key generation

Based on the cipher key, Encryption is performed by pixel position transformation on the RGB layer of subpixels. Thus, the robustness is increased, and a better-ciphered image is obtained. encrypted output is assessed by histogram analysis, correlation coefficient analysis and differential measure analysis [23]. Implementation is done in Python software using Google Colaboratory with runtime file access for I/O operations [24].

4 Sudoku Key Validation

The Input key needs to be validated to yield the correct crypto key. The method involves validation of the number matrix to ensure all are non-zero, relevant, and non-repeating. The Length and Count mismatches in all Rows and Columns are checked. Following are the steps to be followed in input key validation.

- The order of the matrix $= n$. The row count and column count of the read text are cross-checked with the order value of the matrix to validate the size of the sudoku solution key.
- Each element of the read matrix is checked for $0 < \text{Key}(i, j) < n + 1$. When the values of elements exceed this limit, then the key is considered invalid.
- The matrix key is checked for the sudoku characteristics All elements in each row and column are different. All rows have different elements in every column.

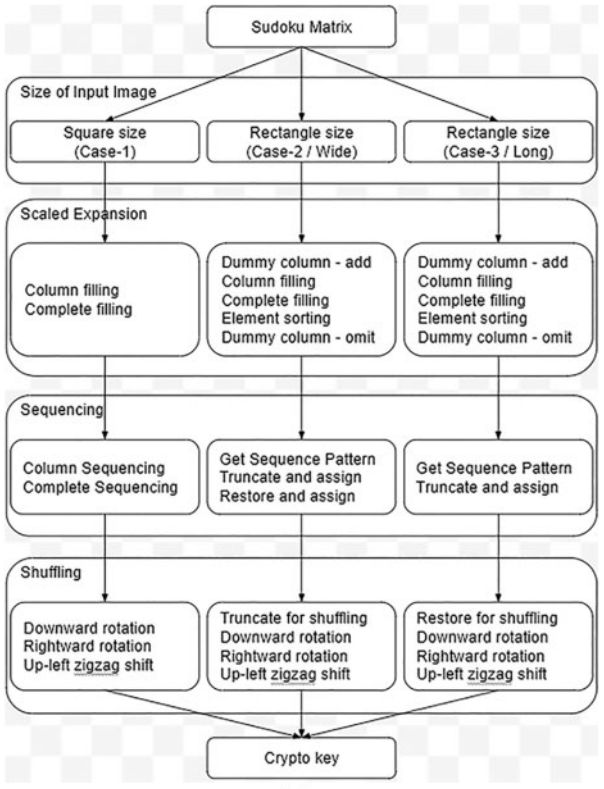
The input number matrix is first validated for a proper sudoku solution. Enhancements come into the picture for three criteria. The first criterion is that the column count in the final image must be a multiple of the column count in the sudoku solution. Once this criterion is not met, the matrix column count is revised to the roof multiple of the sudoku column count. After calculations, such columns are isolated and removed.

The Second criterion is that the length of the sequencing pattern must match the row count of the image matrix while sequencing. In case this criterion is not met, errors occur in the sequencing. When there is a mismatch due to a more extended sequencing pattern, proportional numbers of the sequence are not allocated to the number matrix, resulting in a discontinuous number sequence. When there is a mismatch due to a shorter sequencing pattern, proportional matrix numbers do not participate in the sequence, resulting in a number sequence with multiple duplicates. Mismatch is eliminated by expanding or truncating techniques.

Third criterion is that the length of the rotational shift pattern must match the column count of the image matrix during the downward shifts. The length of the rotational shift pattern must match the row count of the image matrix during the rightward shifts. Mismatches result in unshifted rows or columns. Hence few sections of the number matrix remain unshuffled, yielding a poor crypto key. The generation of the effective crypto-key lies in how these mismatches are correctly handled, meeting the criteria. The process criterion highlights the significance of the Image size in the crypto key generation. Hence, three cases are separately brought out:

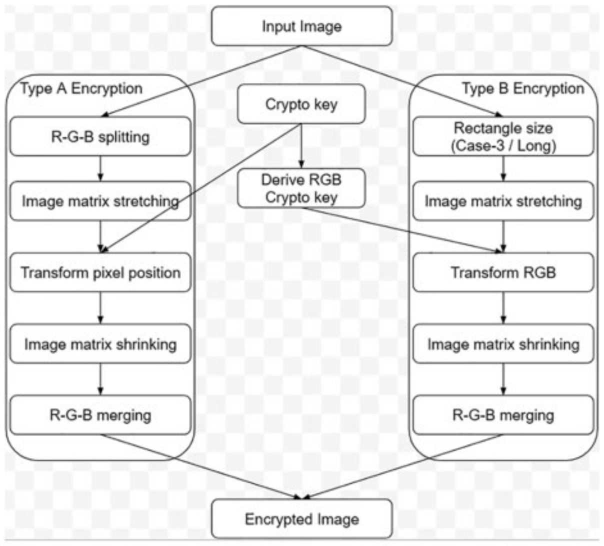
- Case-1 refers to the square size image, where the row count equals the column count in the Image matrix. The number of rows or columns are direct multiplication factors of the input key rows or columns.
- Case-2 refers to the wide rectangle size image, where the row count exceeds the column count in the Image matrix.
- Case-3 refers to the long rectangle size image, where the column count exceeds the row count in the image matrix.

In Cases 2 & 3, the row or column count may/may not be direct multiples of the Input key rows or columns. Crypto key generation involves three steps in sequence, namely Expansion, Sequencing and Shuffling. The study of this method can be replicated by parallel execution of activities in the same sequence described in the respective sections. The flowchart of this encryption algorithm explains all the processes involved in this algorithm (Refer to Figs. 3a and 3b).



(a)

Figure 3a: Flowchart-crypto key generation



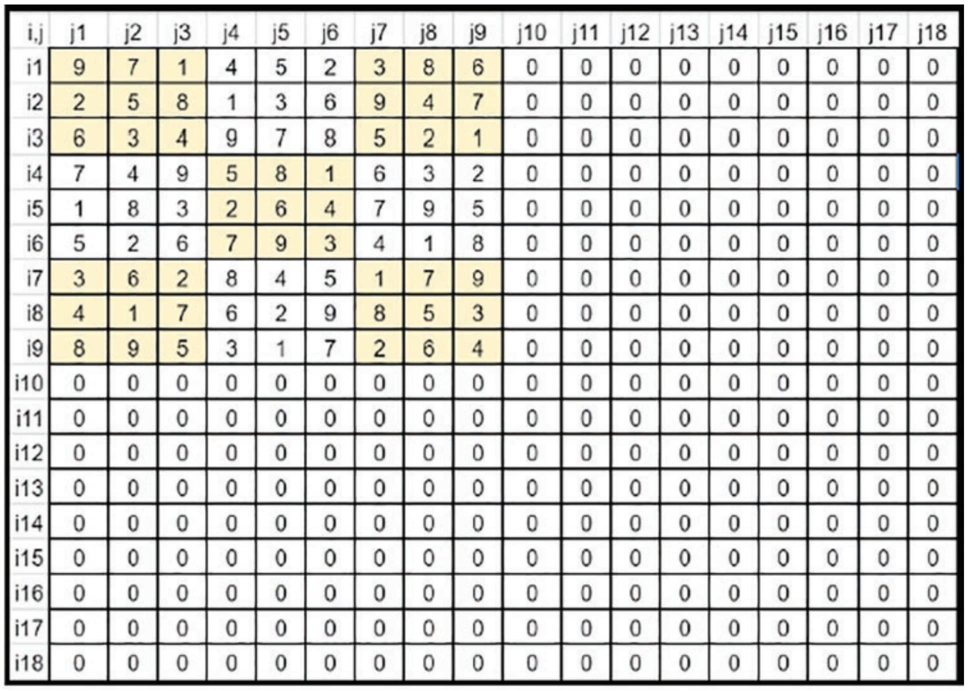
(b)

Figure 3b: Flowchart-image encryption

5 Crypto Key Generation–Case 1

5.1 Expansion

- The image size (N × N) is a multiple of the key's size (n × n). Hence the key's row and column need not be truncated or expanded by any offsets (Refer to Fig. 4). Scaled Expansion of the key is required to match the image size.
- The count the key needs to be scaled is stored as 'ifactor' along the Vertical axis (Top to down) and as 'jfactor' along the Horizontal axis (Left to right).
- Key is expanded along the Horizontal axis in each row by filling right rotated (single shift) values of the previous set of values. This method fills all elements of the base key rows. (Refer to Fig. 5).
- The key is expanded along the vertical axis in each column by filling down rotated (single shift) values of the previous set of values. This method fills all elements of the matrix (Refer to Fig. 6).
- In all the rows, the elements are scaled by the corresponding 'jfactor' values so that all row elements range from 1 to N (Refer to Fig. 7).

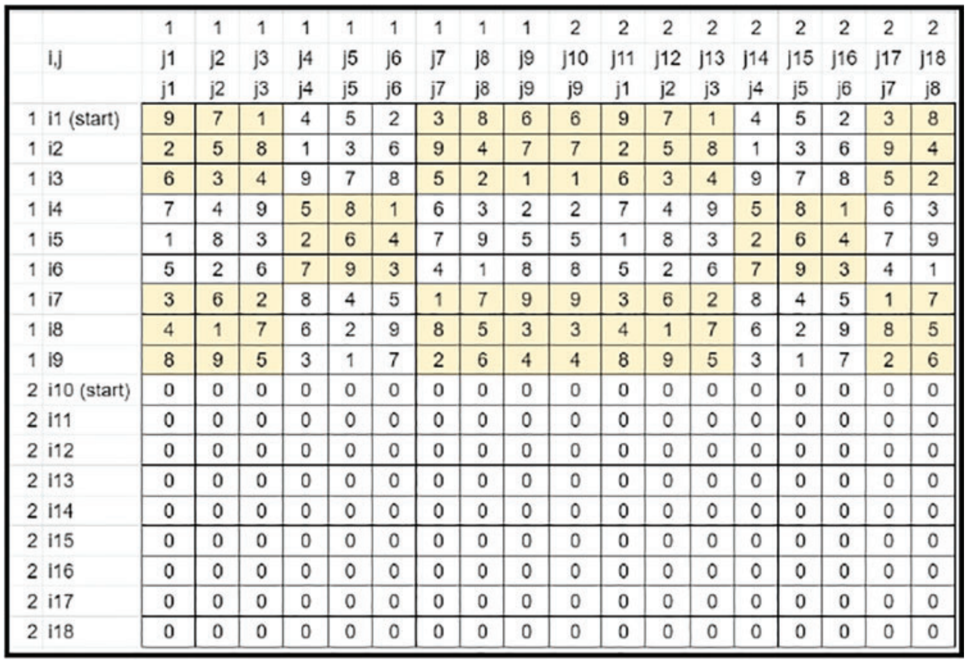


i,j	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14	j15	j16	j17	j18
i1	9	7	1	4	5	2	3	8	6	0	0	0	0	0	0	0	0	0
i2	2	5	8	1	3	6	9	4	7	0	0	0	0	0	0	0	0	0
i3	6	3	4	9	7	8	5	2	1	0	0	0	0	0	0	0	0	0
i4	7	4	9	5	8	1	6	3	2	0	0	0	0	0	0	0	0	0
i5	1	8	3	2	6	4	7	9	5	0	0	0	0	0	0	0	0	0
i6	5	2	6	7	9	3	4	1	8	0	0	0	0	0	0	0	0	0
i7	3	6	2	8	4	5	1	7	9	0	0	0	0	0	0	0	0	0
i8	4	1	7	6	2	9	8	5	3	0	0	0	0	0	0	0	0	0
i9	8	9	5	3	1	7	2	6	4	0	0	0	0	0	0	0	0	0
i10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4: Sudoku matrix (Case 1) fit into key's matrix

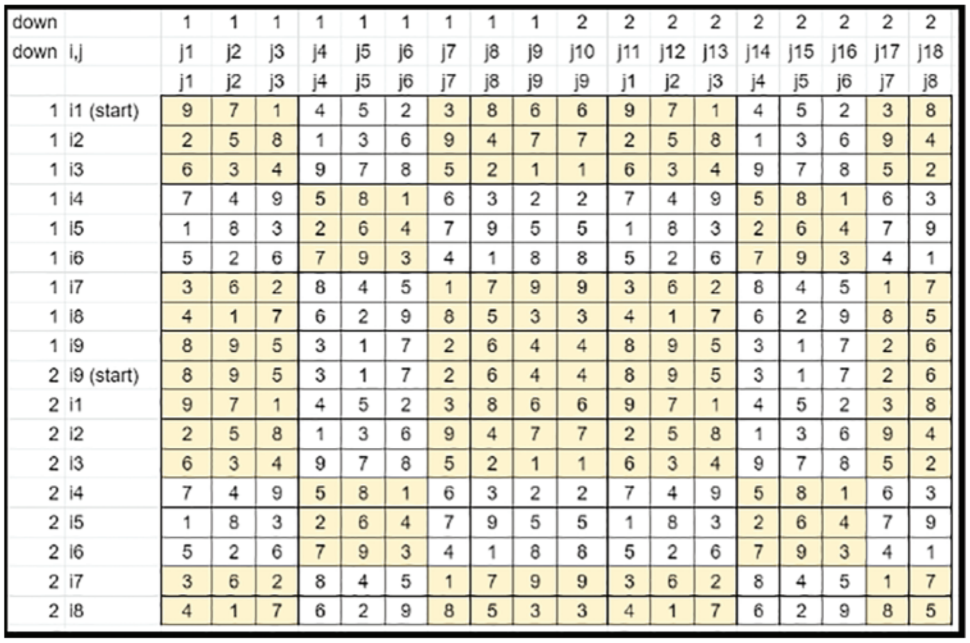
5.2 Sequencing

- The first element (0, 0) of the matrix dictates the factor for scaling and shuffling and is saved as prime.
- The transpose of the prime row element defines the scaling of the matrix elements along each row. This way, the entire matrix elements are scaled and range from 1 to N × N. This matrix becomes the Derived key matrix 1 (Refer to Fig. 8).
- The transpose of prime column elements of Derived key matrix 1 defines the downward rotational shifts of the derived matrix elements along each column.



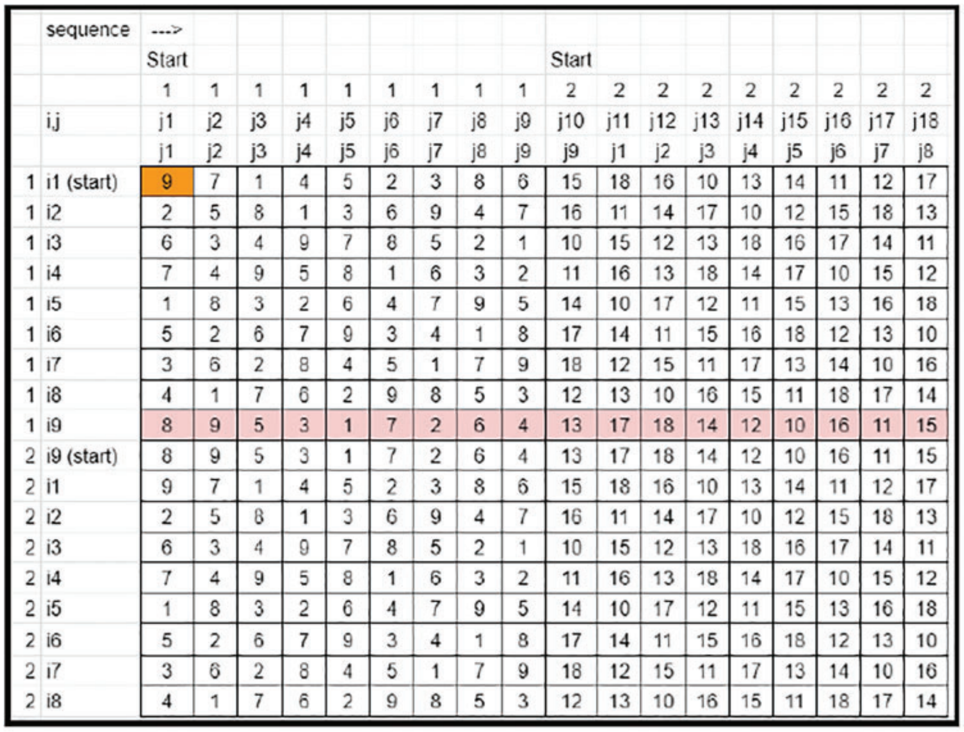
		1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
	i,j	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14	j15	j16	j17	j18
		j1	j2	j3	j4	j5	j6	j7	j8	j9	j9	j1	j2	j3	j4	j5	j6	j7	j8
1	i1 (start)	9	7	1	4	5	2	3	8	6	6	9	7	1	4	5	2	3	8
1	i2	2	5	8	1	3	6	9	4	7	7	2	5	8	1	3	6	9	4
1	i3	6	3	4	9	7	8	5	2	1	1	6	3	4	9	7	8	5	2
1	i4	7	4	9	5	8	1	6	3	2	2	7	4	9	5	8	1	6	3
1	i5	1	8	3	2	6	4	7	9	5	5	1	8	3	2	6	4	7	9
1	i6	5	2	6	7	9	3	4	1	8	8	5	2	6	7	9	3	4	1
1	i7	3	6	2	8	4	5	1	7	9	9	3	6	2	8	4	5	1	7
1	i8	4	1	7	6	2	9	8	5	3	3	4	1	7	6	2	9	8	5
1	i9	8	9	5	3	1	7	2	6	4	4	8	9	5	3	1	7	2	6
2	i10 (start)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	i11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	i12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	i13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	i14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	i15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	i16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	i17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	i18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5: Key's matrix-column filling



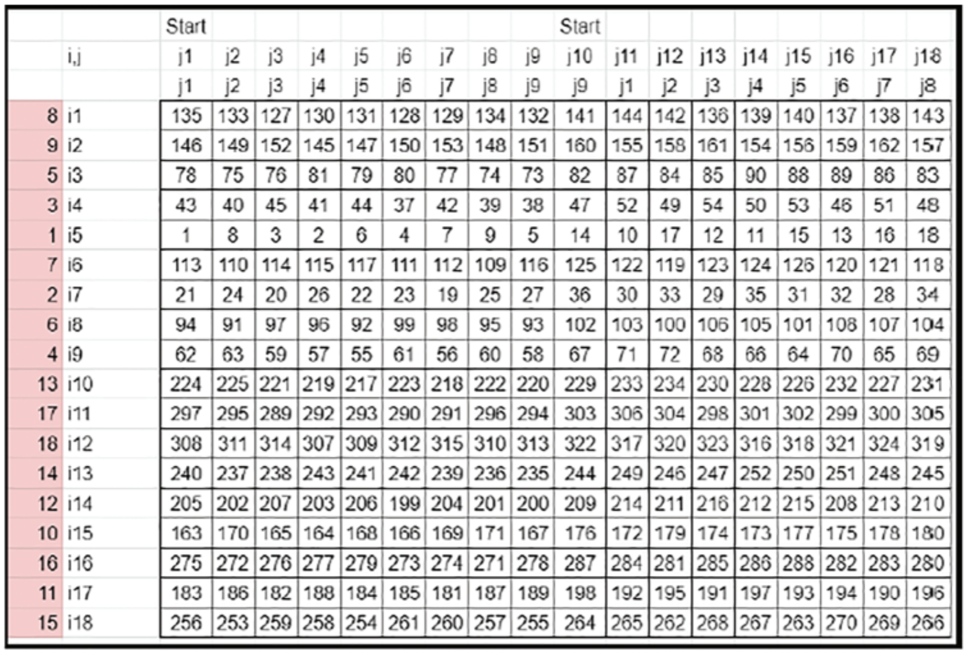
down		1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
down	i,j	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14	j15	j16	j17	j18
		j1	j2	j3	j4	j5	j6	j7	j8	j9	j9	j1	j2	j3	j4	j5	j6	j7	j8
1	i1 (start)	9	7	1	4	5	2	3	8	6	6	9	7	1	4	5	2	3	8
1	i2	2	5	8	1	3	6	9	4	7	7	2	5	8	1	3	6	9	4
1	i3	6	3	4	9	7	8	5	2	1	1	6	3	4	9	7	8	5	2
1	i4	7	4	9	5	8	1	6	3	2	2	7	4	9	5	8	1	6	3
1	i5	1	8	3	2	6	4	7	9	5	5	1	8	3	2	6	4	7	9
1	i6	5	2	6	7	9	3	4	1	8	8	5	2	6	7	9	3	4	1
1	i7	3	6	2	8	4	5	1	7	9	9	3	6	2	8	4	5	1	7
1	i8	4	1	7	6	2	9	8	5	3	3	4	1	7	6	2	9	8	5
1	i9	8	9	5	3	1	7	2	6	4	4	8	9	5	3	1	7	2	6
2	i9 (start)	8	9	5	3	1	7	2	6	4	4	8	9	5	3	1	7	2	6
2	i1	9	7	1	4	5	2	3	8	6	6	9	7	1	4	5	2	3	8
2	i2	2	5	8	1	3	6	9	4	7	7	2	5	8	1	3	6	9	4
2	i3	6	3	4	9	7	8	5	2	1	1	6	3	4	9	7	8	5	2
2	i4	7	4	9	5	8	1	6	3	2	2	7	4	9	5	8	1	6	3
2	i5	1	8	3	2	6	4	7	9	5	5	1	8	3	2	6	4	7	9
2	i6	5	2	6	7	9	3	4	1	8	8	5	2	6	7	9	3	4	1
2	i7	3	6	2	8	4	5	1	7	9	9	3	6	2	8	4	5	1	7
2	i8	4	1	7	6	2	9	8	5	3	3	4	1	7	6	2	9	8	5

Figure 6: Key matrix-complete filling



	sequence	--->																	
		Start									Start								
		1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
	i,j	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14	j15	j16	j17	j18
		j1	j2	j3	j4	j5	j6	j7	j8	j9	j9	j1	j2	j3	j4	j5	j6	j7	j8
1	i1 (start)	9	7	1	4	5	2	3	8	6	15	18	16	10	13	14	11	12	17
1	i2	2	5	8	1	3	6	9	4	7	16	11	14	17	10	12	15	18	13
1	i3	6	3	4	9	7	8	5	2	1	10	15	12	13	18	16	17	14	11
1	i4	7	4	9	5	8	1	6	3	2	11	16	13	18	14	17	10	15	12
1	i5	1	8	3	2	6	4	7	9	5	14	10	17	12	11	15	13	16	18
1	i6	5	2	6	7	9	3	4	1	8	17	14	11	15	16	18	12	13	10
1	i7	3	6	2	8	4	5	1	7	9	18	12	15	11	17	13	14	10	16
1	i8	4	1	7	6	2	9	8	5	3	12	13	10	16	15	11	18	17	14
1	i9	8	9	5	3	1	7	2	6	4	13	17	18	14	12	10	16	11	15
2	i9 (start)	8	9	5	3	1	7	2	6	4	13	17	18	14	12	10	16	11	15
2	i1	9	7	1	4	5	2	3	8	6	15	18	16	10	13	14	11	12	17
2	i2	2	5	8	1	3	6	9	4	7	16	11	14	17	10	12	15	18	13
2	i3	6	3	4	9	7	8	5	2	1	10	15	12	13	18	16	17	14	11
2	i4	7	4	9	5	8	1	6	3	2	11	16	13	18	14	17	10	15	12
2	i5	1	8	3	2	6	4	7	9	5	14	10	17	12	11	15	13	16	18
2	i6	5	2	6	7	9	3	4	1	8	17	14	11	15	16	18	12	13	10
2	i7	3	6	2	8	4	5	1	7	9	18	12	15	11	17	13	14	10	16
2	i8	4	1	7	6	2	9	8	5	3	12	13	10	16	15	11	18	17	14

Figure 7: Key matrix-sequencing across columns

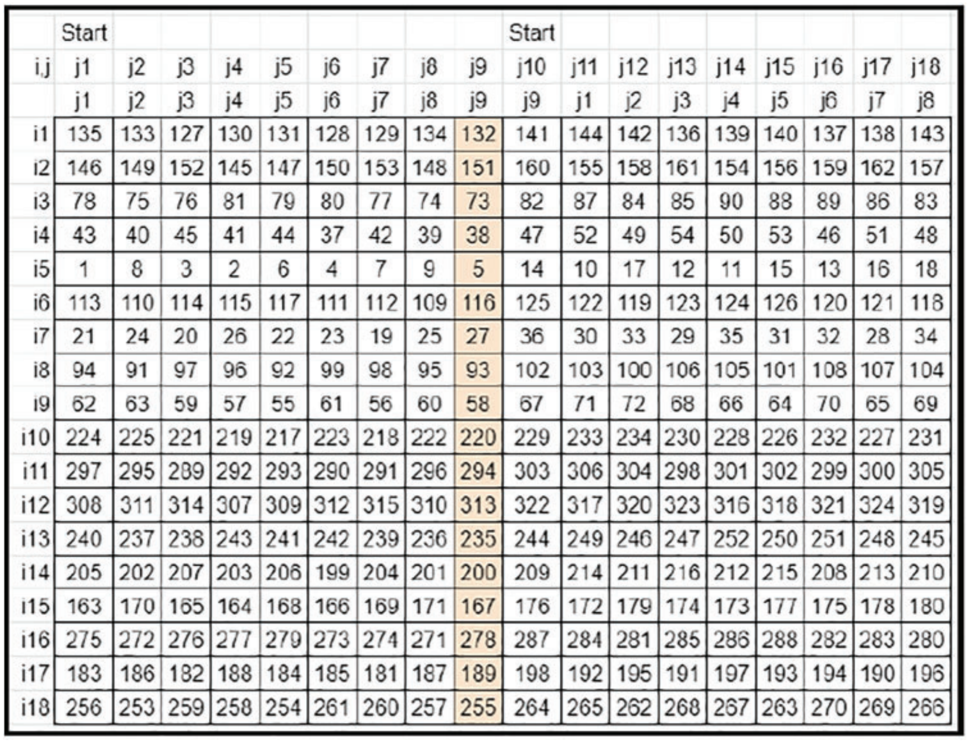


		Start									Start								
	i,j	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14	j15	j16	j17	j18
		j1	j2	j3	j4	j5	j6	j7	j8	j9	j9	j1	j2	j3	j4	j5	j6	j7	j8
8	i1	135	133	127	130	131	128	129	134	132	141	144	142	136	139	140	137	138	143
9	i2	146	149	152	145	147	150	153	148	151	160	155	158	161	154	156	159	162	157
5	i3	78	75	76	81	79	80	77	74	73	82	87	84	85	90	88	89	86	83
3	i4	43	40	45	41	44	37	42	39	38	47	52	49	54	50	53	46	51	48
1	i5	1	8	3	2	6	4	7	9	5	14	10	17	12	11	15	13	16	18
7	i6	113	110	114	115	117	111	112	109	116	125	122	119	123	124	126	120	121	118
2	i7	21	24	20	26	22	23	19	25	27	36	30	33	29	35	31	32	28	34
6	i8	94	91	97	96	92	99	98	95	93	102	103	100	106	105	101	108	107	104
4	i9	62	63	59	57	55	61	56	60	58	67	71	72	68	66	64	70	65	69
13	i10	224	225	221	219	217	223	218	222	220	229	233	234	230	228	226	232	227	231
17	i11	297	295	289	292	293	290	291	296	294	303	306	304	298	301	302	299	300	305
18	i12	308	311	314	307	309	312	315	310	313	322	317	320	323	316	318	321	324	319
14	i13	240	237	238	243	241	242	239	236	235	244	249	246	247	252	250	251	248	245
12	i14	205	202	207	203	206	199	204	201	200	209	214	211	216	212	215	208	213	210
10	i15	163	170	165	164	168	166	169	171	167	176	172	179	174	173	177	175	178	180
16	i16	275	272	276	277	279	273	274	271	278	287	284	281	285	286	288	282	283	280
11	i17	183	186	182	188	184	185	181	187	189	198	192	195	191	197	193	194	190	196
15	i18	256	253	259	258	254	261	260	257	255	264	265	262	268	267	263	270	269	266

Figure 8: Key matrix-complete sequencing

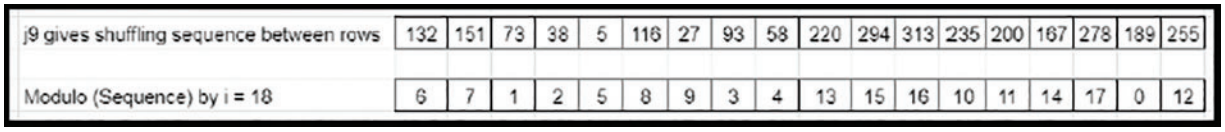
5.3 *Shuffling*

- Since the maximum rotational downshifting is limited by N, the prime column element factors (Refer to Fig. 9) are modulo operated by N (Refer to Fig. 10) and then applied to the matrix.
- The entire matrix elements are shuffled along the vertical axis (Refer to Fig. 11). This matrix becomes the derived key matrix 2 (Refer to Fig. 12).
- The prime column elements of derived key matrix 1 define the rightward rotational shifts of derived matrix elements along each row (Refer to Fig. 13).
- Since the maximum shift is limited by N, the prime column element factors are modulo operated by N and applied to the matrix.
- The entire matrix elements are shuffled along the Horizontal axis. This matrix becomes the derived key matrix 3.
- The linear down and right rotational shifts are balanced by the standard zig-zag up and left rotational shifts (Refer to Fig. 14).
- The iteration of shifts is defined by the first element (0, 0) of the Derived key matrix 3. The resultant matrix becomes the final key's matrix for Image encryption.
- The zig-zag shift of the colored elements can be compared (Refer to Figs. 15 and 16).
- The generated key is then written into a text file and saved for further encryption procedures.



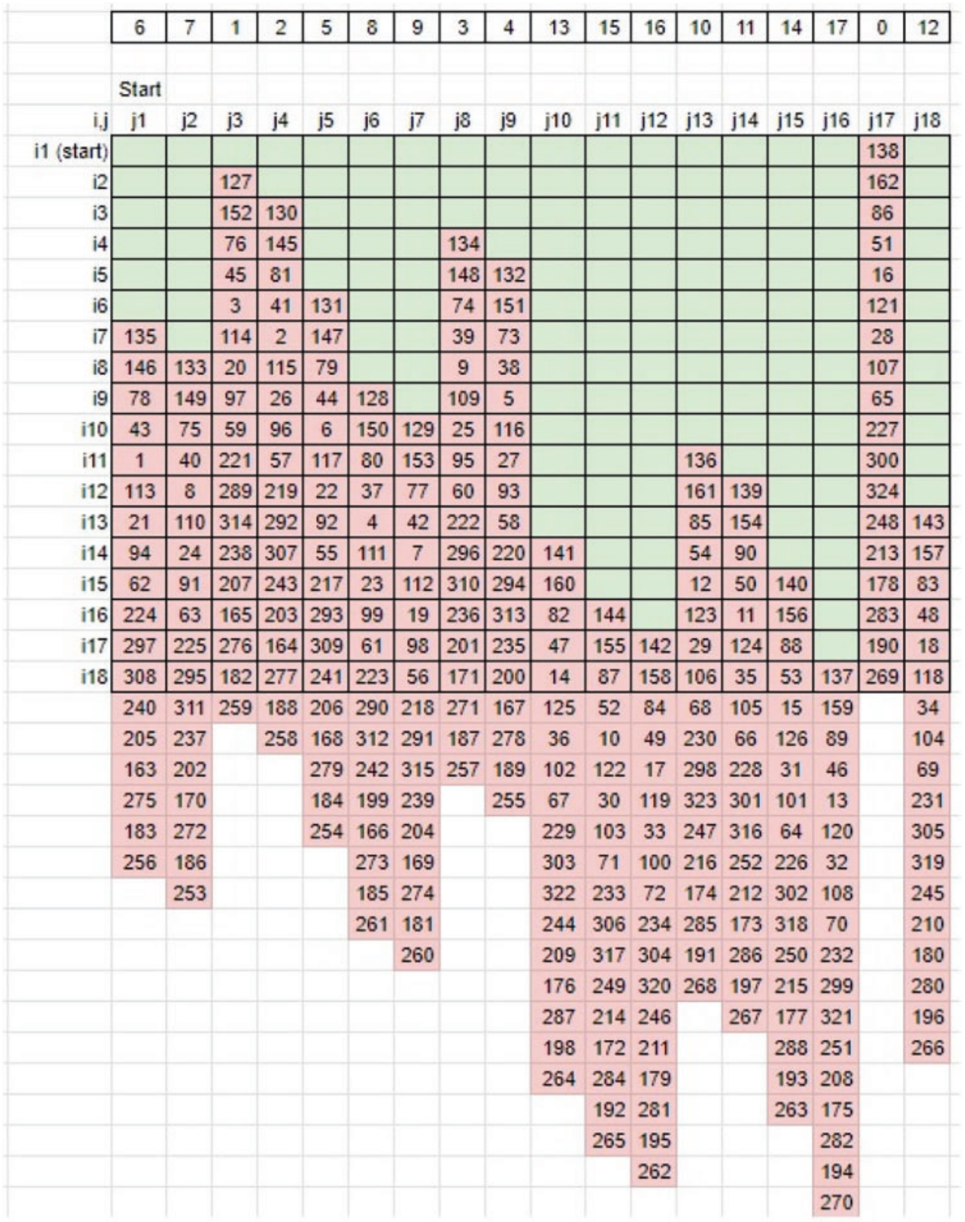
	Start									Start								
i,j	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14	j15	j16	j17	j18
	j1	j2	j3	j4	j5	j6	j7	j8	j9	j9	j1	j2	j3	j4	j5	j6	j7	j8
i1	135	133	127	130	131	128	129	134	132	141	144	142	136	139	140	137	138	143
i2	146	149	152	145	147	150	153	148	151	160	155	158	161	154	156	159	162	157
i3	78	75	76	81	79	80	77	74	73	82	87	84	85	90	88	89	86	83
i4	43	40	45	41	44	37	42	39	38	47	52	49	54	50	53	46	51	48
i5	1	8	3	2	6	4	7	9	5	14	10	17	12	11	15	13	16	18
i6	113	110	114	115	117	111	112	109	116	125	122	119	123	124	126	120	121	118
i7	21	24	20	26	22	23	19	25	27	36	30	33	29	35	31	32	28	34
i8	94	91	97	96	92	99	98	95	93	102	103	100	106	105	101	108	107	104
i9	62	63	59	57	55	61	56	60	58	67	71	72	68	66	64	70	65	69
i10	224	225	221	219	217	223	218	222	220	229	233	234	230	228	226	232	227	231
i11	297	295	289	292	293	290	291	296	294	303	306	304	298	301	302	299	300	305
i12	308	311	314	307	309	312	315	310	313	322	317	320	323	316	318	321	324	319
i13	240	237	238	243	241	242	239	236	235	244	249	246	247	252	250	251	248	245
i14	205	202	207	203	206	199	204	201	200	209	214	211	216	212	215	208	213	210
i15	163	170	165	164	168	166	169	171	167	176	172	179	174	173	177	175	178	180
i16	275	272	276	277	279	273	274	271	278	287	284	281	285	286	288	282	283	280
i17	183	186	182	188	184	185	181	187	189	198	192	195	191	197	193	194	190	196
i18	256	253	259	258	254	261	260	257	255	264	265	262	268	267	263	270	269	266

Figure 9: Sampled derived key's matrix 1 with prime column



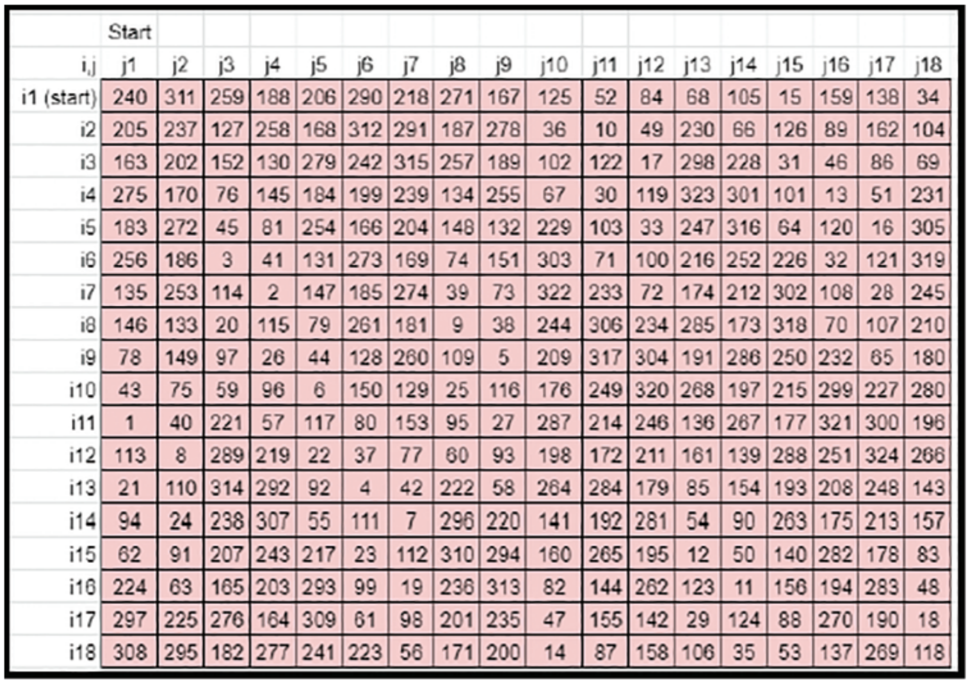
j9 gives shuffling sequence between rows	132	151	73	38	5	116	27	93	58	220	294	313	235	200	167	278	189	255
Modulo (Sequence) by i = 18	6	7	1	2	5	8	9	3	4	13	15	16	10	11	14	17	0	12

Figure 10: Modulo calculation



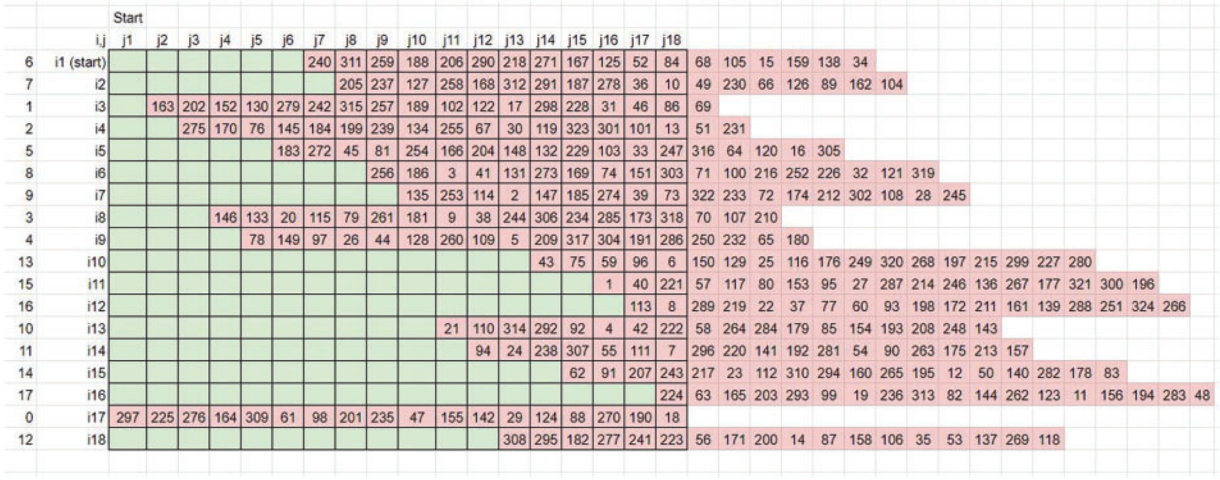
	6	7	1	2	5	8	9	3	4	13	15	16	10	11	14	17	0	12
	Start																	
i,j	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14	j15	j16	j17	j18
i1 (start)																	138	
i2			127														162	
i3			152	130													86	
i4			76	145				134									51	
i5			45	81				148	132								16	
i6			3	41	131			74	151								121	
i7	135		114	2	147			39	73								28	
i8	146	133	20	115	79			9	38								107	
i9	78	149	97	26	44	128		109	5								65	
i10	43	75	59	96	6	150	129	25	116								227	
i11	1	40	221	57	117	80	153	95	27				136				300	
i12	113	8	289	219	22	37	77	60	93				161	139			324	
i13	21	110	314	292	92	4	42	222	58				85	154			248	143
i14	94	24	238	307	55	111	7	296	220	141			54	90			213	157
i15	62	91	207	243	217	23	112	310	294	160			12	50	140		178	83
i16	224	63	165	203	293	99	19	236	313	82	144		123	11	156		283	48
i17	297	225	276	164	309	61	98	201	235	47	155	142	29	124	88		190	18
i18	308	295	182	277	241	223	56	171	200	14	87	158	106	35	53	137	269	118
	240	311	259	188	206	290	218	271	167	125	52	84	68	105	15	159		34
	205	237		258	168	312	291	187	278	36	10	49	230	66	126	89		104
	163	202			279	242	315	257	189	102	122	17	298	228	31	46		69
	275	170			184	199	239		255	67	30	119	323	301	101	13		231
	183	272			254	166	204			229	103	33	247	316	64	120		305
	256	186				273	169			303	71	100	216	252	226	32		319
		253				185	274			322	233	72	174	212	302	108		245
						261	181			244	306	234	285	173	318	70		210
							260			209	317	304	191	286	250	232		180
										176	249	320	268	197	215	299		280
										287	214	246		267	177	321		196
										198	172	211			288	251		266
										264	284	179			193	208		
											192	281			263	175		
											265	195				282		
												262				194		
																270		

Figure 11: Pattern-based downward rotation



	Start																	
i,j	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14	j15	j16	j17	j18
i1 (start)	240	311	259	188	206	290	218	271	167	125	52	84	68	105	15	159	138	34
i2	205	237	127	258	168	312	291	187	278	36	10	49	230	66	126	89	162	104
i3	163	202	152	130	279	242	315	257	189	102	122	17	298	228	31	46	86	69
i4	275	170	76	145	184	199	239	134	255	67	30	119	323	301	101	13	51	231
i5	183	272	45	81	254	166	204	148	132	229	103	33	247	316	64	120	16	305
i6	256	186	3	41	131	273	169	74	151	303	71	100	216	252	226	32	121	319
i7	135	253	114	2	147	185	274	39	73	322	233	72	174	212	302	108	28	245
i8	146	133	20	115	79	261	181	9	38	244	306	234	285	173	318	70	107	210
i9	78	149	97	26	44	128	260	109	5	209	317	304	191	286	250	232	65	180
i10	43	75	59	96	6	150	129	25	116	176	249	320	268	197	215	299	227	280
i11	1	40	221	57	117	80	153	95	27	287	214	246	136	267	177	321	300	196
i12	113	8	289	219	22	37	77	60	93	198	172	211	161	139	288	251	324	266
i13	21	110	314	292	92	4	42	222	58	264	284	179	85	154	193	208	248	143
i14	94	24	238	307	55	111	7	296	220	141	192	281	54	90	263	175	213	157
i15	62	91	207	243	217	23	112	310	294	160	265	195	12	50	140	282	178	83
i16	224	63	165	203	293	99	19	236	313	82	144	262	123	11	156	194	283	48
i17	297	225	276	164	309	61	98	201	235	47	155	142	29	124	88	270	190	18
i18	308	295	182	277	241	223	56	171	200	14	87	158	106	35	53	137	269	118

Figure 12: Sample derived key matrix 2



		Start																																		
	i,j	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14	j15	j16	j17	j18																	
6	i1 (start)							240	311	259	188	206	290	218	271	167	125	52	84	68	105	15	159	138	34											
7	i2								205	237	127	258	168	312	291	187	278	36	10	49	230	66	126	89	162	104										
1	i3		163	202	152	130	279	242	315	257	189	102	122	17	298	228	31	46	86	69																
2	i4			275	170	76	145	184	199	239	134	255	67	30	119	323	301	101	13	51	231															
5	i5						183	272	45	81	254	166	204	148	132	229	103	33	247	316	64	120	16	305												
8	i6									256	186	3	41	131	273	169	74	151	303	71	100	216	252	226	32	121	319									
9	i7										135	253	114	2	147	185	274	39	73	322	233	72	174	212	302	108	28	245								
3	i8				146	133	20	115	79	261	181	9	38	244	306	234	285	173	318	70	107	210														
4	i9					78	149	97	26	44	128	260	109	5	209	317	304	191	286	250	232	65	180													
13	i10														43	75	59	96	6	150	129	25	116	176	249	320	268	197	215	299	227	280				
15	i11																1	40	221	57	117	80	153	95	27	287	214	246	136	267	177	321	300	196		
16	i12																	113	8	289	219	22	37	77	60	93	198	172	211	161	139	288	251	324	266	
10	i13											21	110	314	292	92	4	42	222	58	264	284	179	85	154	193	208	248	143							
11	i14												94	24	238	307	55	111	7	296	220	141	192	281	54	90	263	175	213	157						
14	i15															62	91	207	243	217	23	112	310	294	160	265	195	12	50	140	282	178	83			
17	i16																		224	63	165	203	293	99	19	236	313	82	144	262	123	11	156	194	283	48
0	i17	297	225	276	164	309	61	98	201	235	47	155	142	29	124	88	270	190	18																	
12	i18													308	295	182	277	241	223	56	171	200	14	87	158	106	35	53	137	269	118					

Figure 13: Pattern-based rightward rotation

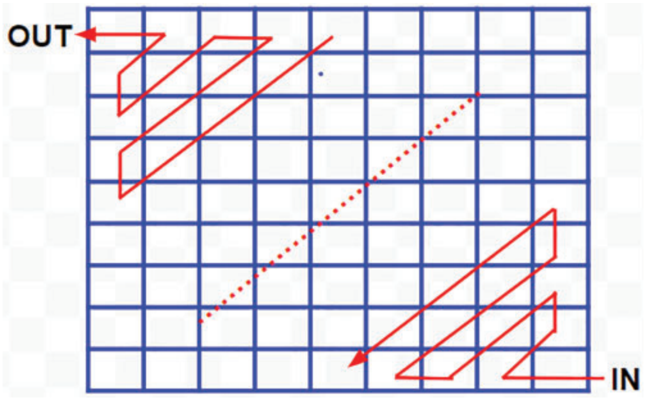
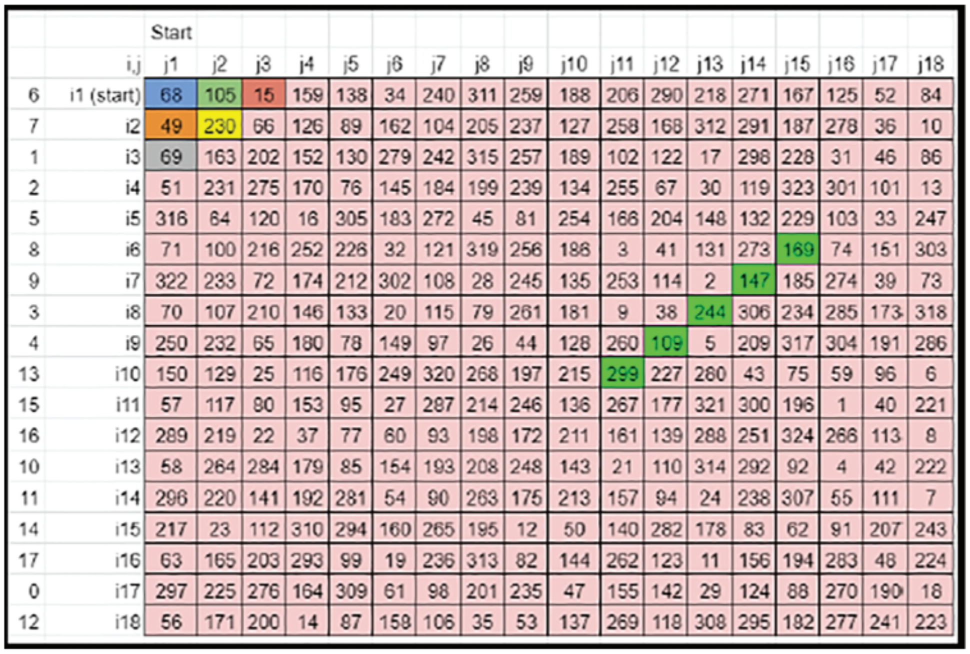
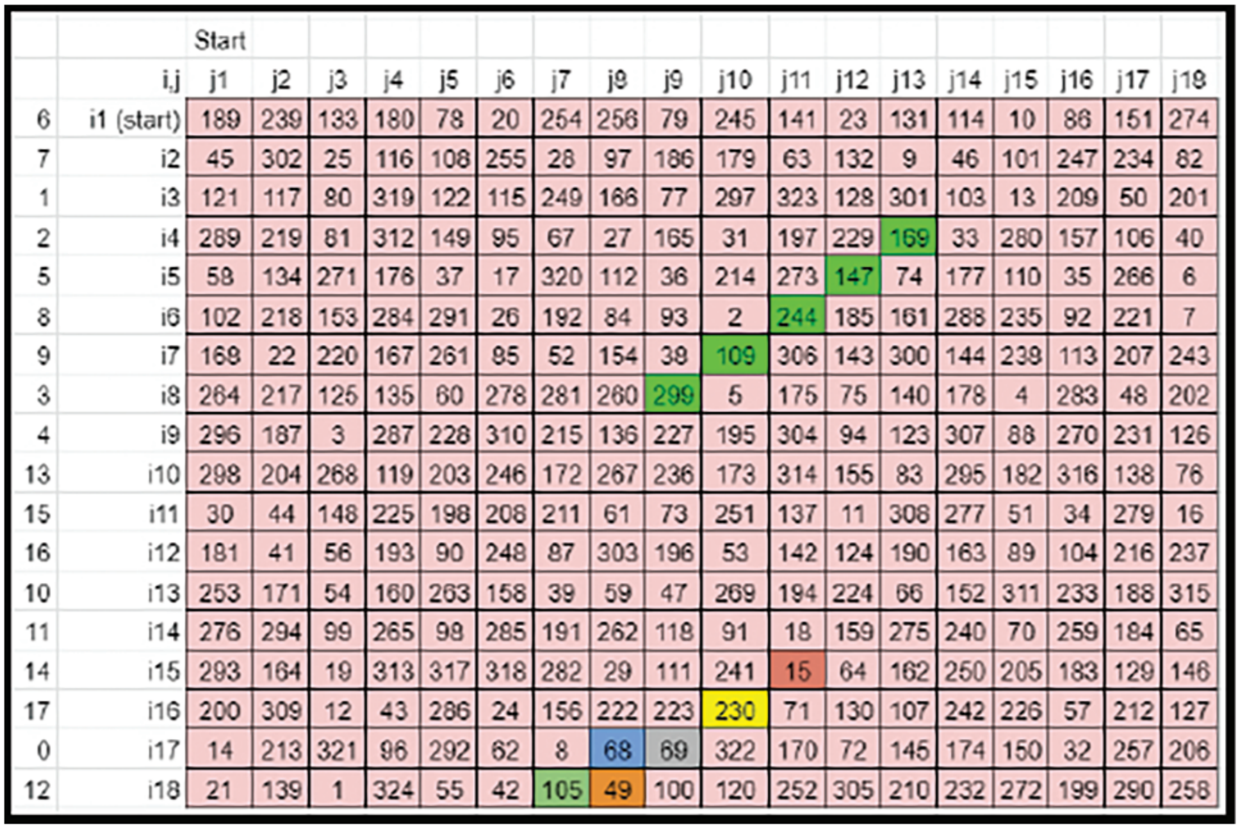


Figure 14: Count-based up-left zigzag shifts



		Start																	
	i,j	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14	j15	j16	j17	j18
6	i1 (start)	68	105	15	159	138	34	240	311	259	188	206	290	218	271	167	125	52	84
7	i2	49	230	66	126	89	162	104	205	237	127	258	168	312	291	187	278	36	10
1	i3	69	163	202	152	130	279	242	315	257	189	102	122	17	298	228	31	46	86
2	i4	51	231	275	170	76	145	184	199	239	134	255	67	30	119	323	301	101	13
5	i5	316	64	120	16	305	183	272	45	81	254	166	204	148	132	229	103	33	247
8	i6	71	100	216	252	226	32	121	319	256	186	3	41	131	273	169	74	151	303
9	i7	322	233	72	174	212	302	108	28	245	135	253	114	2	147	185	274	39	73
3	i8	70	107	210	146	133	20	115	79	261	181	9	38	244	306	234	285	173	318
4	i9	250	232	65	180	78	149	97	26	44	128	260	109	5	209	317	304	191	286
13	i10	150	129	25	116	176	249	320	268	197	215	299	227	280	43	75	59	96	6
15	i11	57	117	80	153	95	27	287	214	246	136	267	177	321	300	196	1	40	221
16	i12	289	219	22	37	77	60	93	198	172	211	161	139	288	251	324	266	113	8
10	i13	58	264	284	179	85	154	193	208	248	143	21	110	314	292	92	4	42	222
11	i14	296	220	141	192	281	54	90	263	175	213	157	94	24	238	307	55	111	7
14	i15	217	23	112	310	294	160	265	195	12	50	140	282	178	83	62	91	207	243
17	i16	63	165	203	293	99	19	236	313	82	144	262	123	11	156	194	283	48	224
0	i17	297	225	276	164	309	61	98	201	235	47	155	142	29	124	88	270	190	18
12	i18	56	171	200	14	87	158	106	35	53	137	269	118	308	295	182	277	241	223

Figure 15: Case 1 sample derived key matrix 3-before zig-zag shift



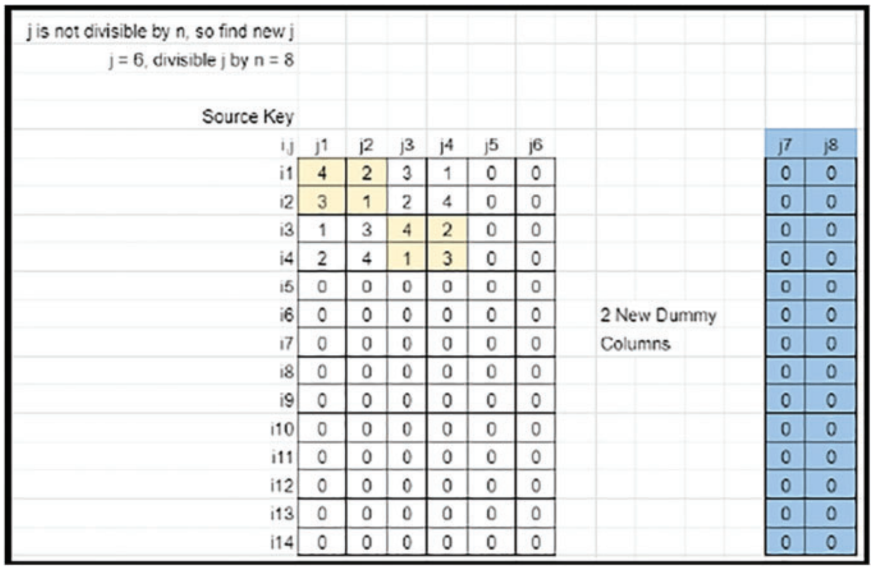
		Start																
	i,j	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14	j15	j16	j17	j18
6	i1 (start)	189	239	133	180	78	20	254	256	79	245	141	23	131	114	10	86	151	274
7	i2	45	302	25	116	108	255	28	97	186	179	63	132	9	46	101	247	234	82
1	i3	121	117	80	319	122	115	249	166	77	297	323	128	301	103	13	209	50	201
2	i4	289	219	81	312	149	95	67	27	165	31	197	229	169	33	280	157	106	40
5	i5	58	134	271	176	37	17	320	112	36	214	273	147	74	177	110	35	266	6
8	i6	102	218	153	284	291	26	192	84	93	2	244	185	161	288	235	92	221	7
9	i7	168	22	220	167	261	85	52	154	38	109	306	143	300	144	238	113	207	243
3	i8	264	217	125	135	60	278	281	260	299	5	175	75	140	178	4	283	48	202
4	i9	296	187	3	287	228	310	215	136	227	195	304	94	123	307	88	270	231	126
13	i10	298	204	268	119	203	246	172	267	236	173	314	155	83	295	182	316	138	76
15	i11	30	44	148	225	198	208	211	61	73	251	137	11	308	277	51	34	279	16
16	i12	181	41	56	193	90	248	87	303	196	53	142	124	190	163	89	104	216	237
10	i13	253	171	54	160	263	158	39	59	47	269	194	224	66	152	311	233	188	315
11	i14	276	294	99	265	98	285	191	262	118	91	18	159	275	240	70	259	184	65
14	i15	293	164	19	313	317	318	282	29	111	241	15	64	162	250	205	183	129	146
17	i16	200	309	12	43	286	24	156	222	223	230	71	130	107	242	226	57	212	127
0	i17	14	213	321	96	292	62	8	68	69	322	170	72	145	174	150	32	257	206
12	i18	21	139	1	324	55	42	105	49	100	120	252	305	210	232	272	199	290	258

Figure 16: Case 1 sample derived key matrix 3-after zig-zag shift

6 Crypto Key Generation–Cases 2 & 3

6.1 Expansion

- The image size (M × N) may not be a multiple of the key's size (n × n). Hence the key rows and columns need to be truncated or expanded by adequate offsets. If N is not divisible by n, expand the column size from N to N_{new}, so that it is divisible by n (Refer to Figs. 17 and 18)
- The iteration the key needs to be scaled is stored as 'ifactor' along the Vertical axis (Top to down) and as 'jfactor' along the Horizontal axis (Left to right).
- The key is expanded along the Horizontal axis in each row by filling right rotated (single shift) values of the previous set of values. This method fills all elements of the base key rows.
- The key is expanded along the vertical axis in each column by filling down rotated (single shift) values of the previous set of values. This method fills all elements of the matrix.
- If N is expanded to N_{new}, the Last segment in each row needs to be sorted. The Highest $N_{new}-N$ elements gets cornered in the last column and removed (Refer to Figs. 19–21).
- Also, while sorting the patterns of segments, the leftover elements in each row must be assigned by numbers 1 to the max. The $N_{new}-N$ columns are removed (Refer to Figs. 22 and 23).
- All the row elements are scaled by the corresponding 'jfactor' values so that the values get limited within 1 to N.



j is not divisible by n, so find new j

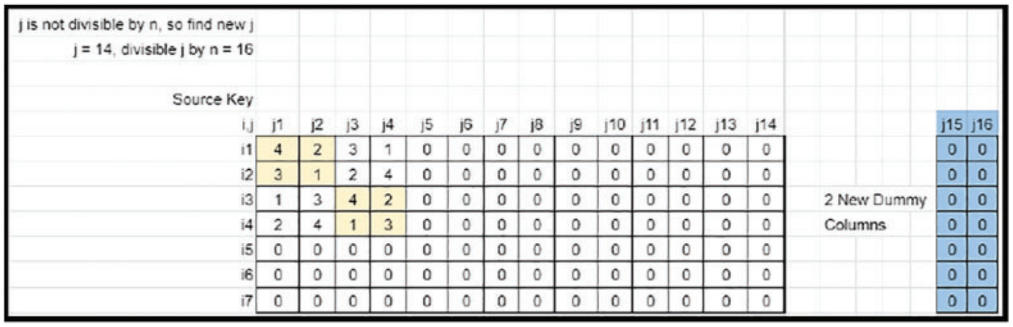
j = 6, divisible j by n = 8

Source Key

2 New Dummy Columns

i,j	j1	j2	j3	j4	j5	j6	j7	j8
i1	4	2	3	1	0	0	0	0
i2	3	1	2	4	0	0	0	0
i3	1	3	4	2	0	0	0	0
i4	2	4	1	3	0	0	0	0
i5	0	0	0	0	0	0	0	0
i6	0	0	0	0	0	0	0	0
i7	0	0	0	0	0	0	0	0
i8	0	0	0	0	0	0	0	0
i9	0	0	0	0	0	0	0	0
i10	0	0	0	0	0	0	0	0
i11	0	0	0	0	0	0	0	0
i12	0	0	0	0	0	0	0	0
i13	0	0	0	0	0	0	0	0
i14	0	0	0	0	0	0	0	0

Figure 17: Sudoku matrix (Case 2) fit into key matrix with column mapping



j is not divisible by n, so find new j

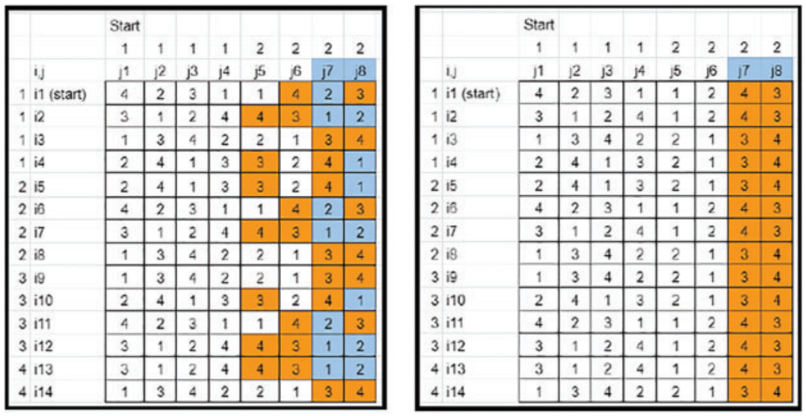
j = 14, divisible j by n = 16

Source Key

2 New Dummy Columns

i,j	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14	j15	j16
i1	4	2	3	1	0	0	0	0	0	0	0	0	0	0	0	0
i2	3	1	2	4	0	0	0	0	0	0	0	0	0	0	0	0
i3	1	3	4	2	0	0	0	0	0	0	0	0	0	0	0	0
i4	2	4	1	3	0	0	0	0	0	0	0	0	0	0	0	0
i5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

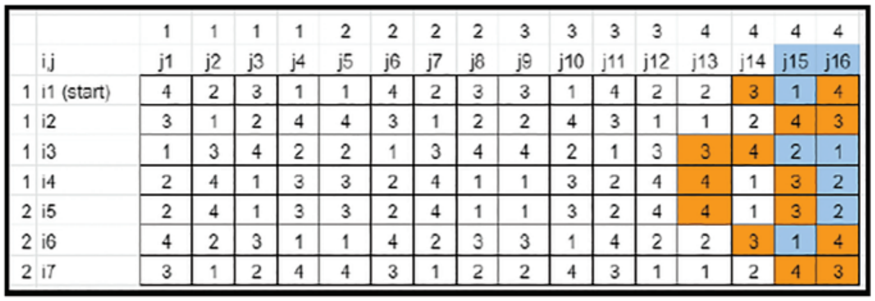
Figure 18: Sudoku matrix (Case 3) fit into key matrix with column mapping



		Start							
		1	1	1	1	2	2	2	2
	i,j	j1	j2	j3	j4	j5	j6	j7	j8
1	i1 (start)	4	2	3	1	1	4	2	3
1	i2	3	1	2	4	4	3	1	2
1	i3	1	3	4	2	2	1	3	4
1	i4	2	4	1	3	3	2	4	1
2	i5	2	4	1	3	3	2	4	1
2	i6	4	2	3	1	1	4	2	3
2	i7	3	1	2	4	4	3	1	2
2	i8	1	3	4	2	2	1	3	4
3	i9	1	3	4	2	2	1	3	4
3	i10	2	4	1	3	3	2	4	1
3	i11	4	2	3	1	1	4	2	3
3	i12	3	1	2	4	4	3	1	2
4	i13	3	1	2	4	4	3	1	2
4	i14	1	3	4	2	2	1	3	4

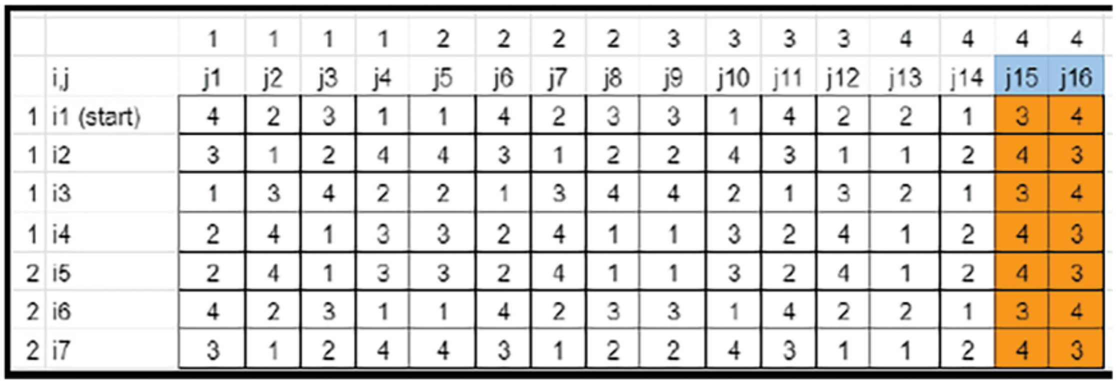
		Start							
		1	1	1	1	2	2	2	2
	i,j	j1	j2	j3	j4	j5	j6	j7	j8
1	i1 (start)	4	2	3	1	1	2	4	3
1	i2	3	1	2	4	1	2	4	3
1	i3	1	3	4	2	2	1	3	4
1	i4	2	4	1	3	2	1	3	4
2	i5	2	4	1	3	2	1	3	4
2	i6	4	2	3	1	1	2	4	3
2	i7	3	1	2	4	1	2	4	3
2	i8	1	3	4	2	2	1	3	4
3	i9	1	3	4	2	2	1	3	4
3	i10	2	4	1	3	2	1	3	4
3	i11	4	2	3	1	1	2	4	3
3	i12	3	1	2	4	1	2	4	3
4	i13	3	1	2	4	1	2	4	3
4	i14	1	3	4	2	2	1	3	4

Figure 19: Key matrix-element sorting and isolation (Case 2)



		1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4
	i,j	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14	j15	j16
1	i1 (start)	4	2	3	1	1	4	2	3	3	1	4	2	2	3	1	4
1	i2	3	1	2	4	4	3	1	2	2	4	3	1	1	2	4	3
1	i3	1	3	4	2	2	1	3	4	4	2	1	3	3	4	2	1
1	i4	2	4	1	3	3	2	4	1	1	3	2	4	4	1	3	2
2	i5	2	4	1	3	3	2	4	1	1	3	2	4	4	1	3	2
2	i6	4	2	3	1	1	4	2	3	3	1	4	2	2	3	1	4
2	i7	3	1	2	4	4	3	1	2	2	4	3	1	1	2	4	3

Figure 20: Key matrix-element sorting (Case 3)



		1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4
	i,j	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14	j15	j16
1	i1 (start)	4	2	3	1	1	4	2	3	3	1	4	2	2	1	3	4
1	i2	3	1	2	4	4	3	1	2	2	4	3	1	1	2	4	3
1	i3	1	3	4	2	2	1	3	4	4	2	1	3	2	1	3	4
1	i4	2	4	1	3	3	2	4	1	1	3	2	4	1	2	4	3
2	i5	2	4	1	3	3	2	4	1	1	3	2	4	1	2	4	3
2	i6	4	2	3	1	1	4	2	3	3	1	4	2	2	1	3	4
2	i7	3	1	2	4	4	3	1	2	2	4	3	1	1	2	4	3

Figure 21: Key matrix-element isolation (Case 3)

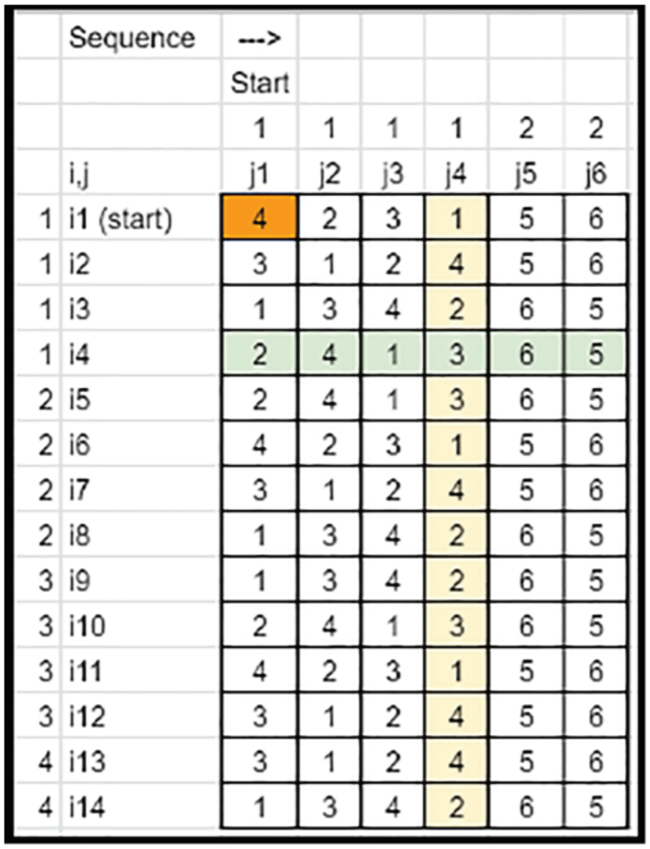
6.2 Sequencing

- The first element (0, 0) of the matrix is the factor for scaling and shuffling in the further steps. Hence, it is saved as prime.
- The Transposition of prime row elements defines the scaling of the matrix elements along each row. Since M is greater than N, the size of the elements does not match (undersized) with the row count. Hence, the prime column end elements are appended to compensate for the shortage in row scaling factor (Refer to Fig. 24). Case-2 involves appending. In contrast, for Case-3, Since $M < N$, the size of elements does not match (oversized) with the row count and hence it involves truncation (Refer to Fig. 25).
- Since the sequence is disturbed due to the appending, the expanded prime row elements are sorted from 1 to maximum value, based on the sequence in the expanded elements.
- With the conditioned prime row elements, all the matrix elements get scaled and range from 1 to $M \times N$. This matrix becomes the Derived key matrix 1.

6.3 Shuffling

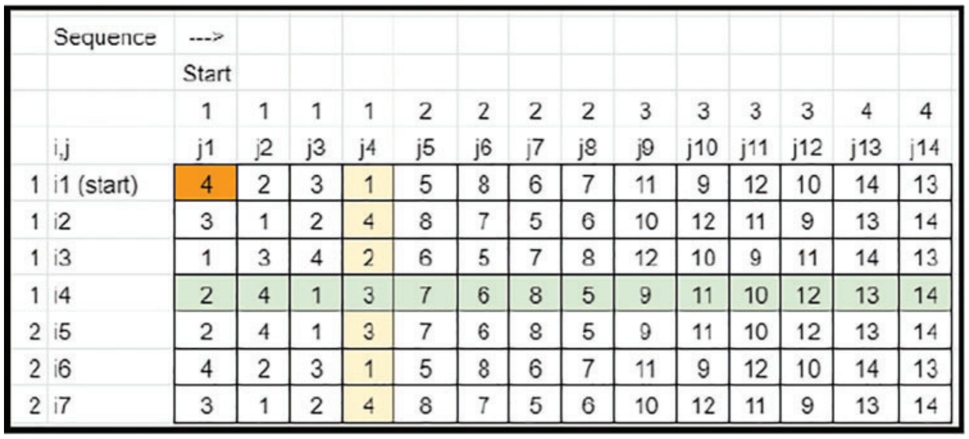
- The Transpose of prime column elements of Derived key matrix 1 defines the downward rotational shifts of the derived matrix elements along each column. Since M is greater than N, the size of the elements does not match (oversized) with the column count. Hence, the end elements of the transposed prime column of derived key matrix 1 are truncated to compensate for the excess in column shuffling factor (Refer to Fig. 26), which is applicable for Case-2. Expansion is involved by appending the earlier truncated content (Refer to Fig. 27) for case-3.

- Since the maximum rotational downshifting is limited by M, the truncated prime column element factors are modulo operated by M.
- With the conditioned prime column elements, all the matrix elements get shuffled along the vertical axis. This matrix becomes the derived key matrix 2.
- Since the maximum shift is limited by N, the conditioned prime column element factors are modulo operated by N.
- These derived prime column elements of derived key matrix 1 define the rightward rotational shifts of the derived matrix elements along each row. The entire matrix elements get shuffled along the horizontal axis. This matrix becomes the derived key matrix 3.
- The linear down and right rotational shifts are balanced by the standard zig-zag up and left rotational shifts. The shift count is defined by the first element (0, 0) of the Derived key matrix 3. The Resulting matrix becomes the final key matrix for image encryption for case 2 (Refer to Fig. 28) and case 3 (Refer to Fig. 29).
- The generated key is then written into a text file and saved for further encryption procedures.



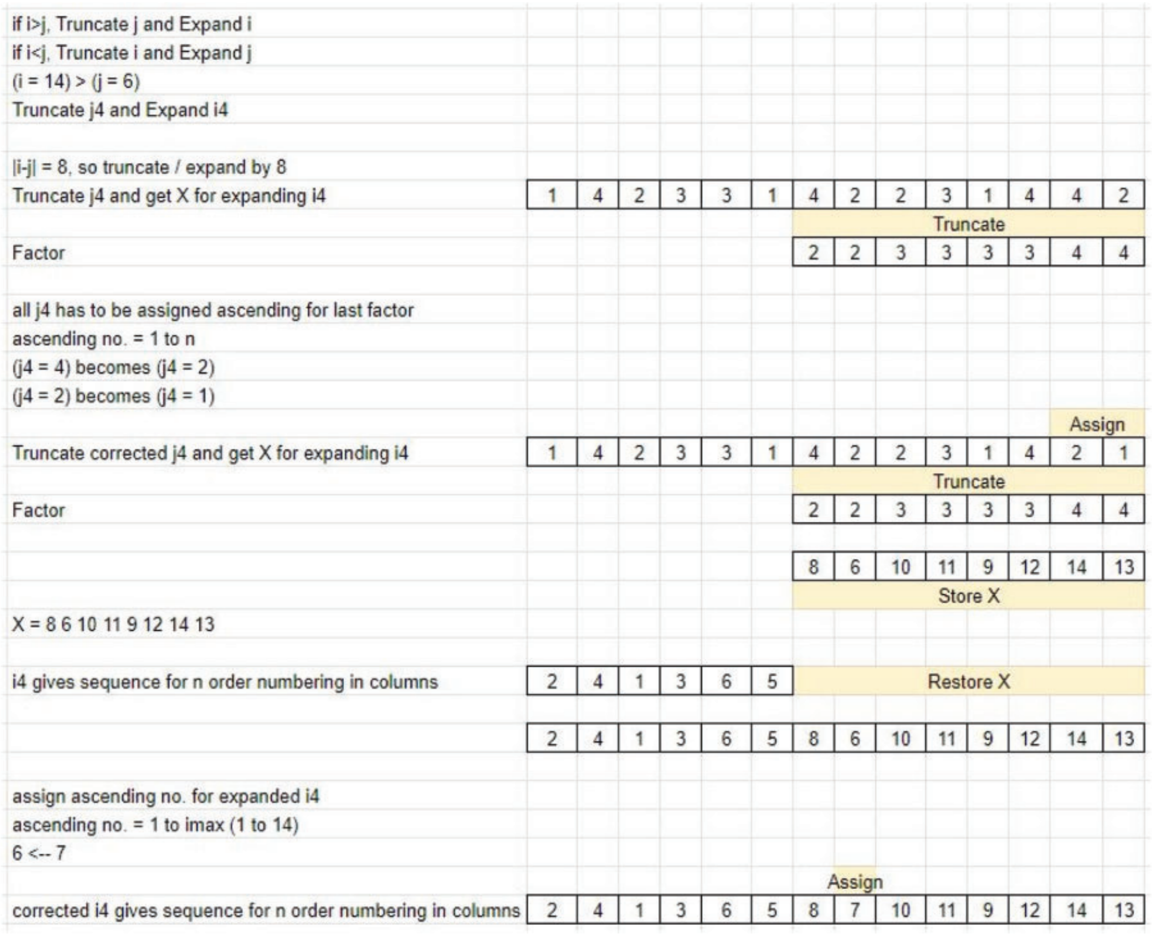
	Sequence	--->					
		Start					
		1	1	1	1	2	2
	i,j	j1	j2	j3	j4	j5	j6
1	i1 (start)	4	2	3	1	5	6
1	i2	3	1	2	4	5	6
1	i3	1	3	4	2	6	5
1	i4	2	4	1	3	6	5
2	i5	2	4	1	3	6	5
2	i6	4	2	3	1	5	6
2	i7	3	1	2	4	5	6
2	i8	1	3	4	2	6	5
3	i9	1	3	4	2	6	5
3	i10	2	4	1	3	6	5
3	i11	4	2	3	1	5	6
3	i12	3	1	2	4	5	6
4	i13	3	1	2	4	5	6
4	i14	1	3	4	2	6	5

Figure 22: Key matrix-columns removed (Case 2)



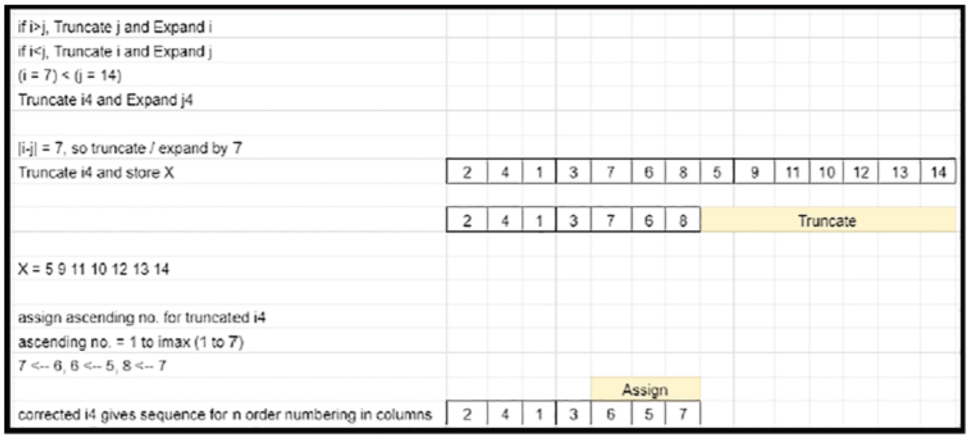
	Sequence	--->													
		Start													
		1	1	1	1	2	2	2	2	3	3	3	3	4	4
	i,j	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14
1	i1 (start)	4	2	3	1	5	8	6	7	11	9	12	10	14	13
1	i2	3	1	2	4	8	7	5	6	10	12	11	9	13	14
1	i3	1	3	4	2	6	5	7	8	12	10	9	11	14	13
1	i4	2	4	1	3	7	6	8	5	9	11	10	12	13	14
2	i5	2	4	1	3	7	6	8	5	9	11	10	12	13	14
2	i6	4	2	3	1	5	8	6	7	11	9	12	10	14	13
2	i7	3	1	2	4	8	7	5	6	10	12	11	9	13	14

Figure 23: Key matrix-columns removed (Case 3)



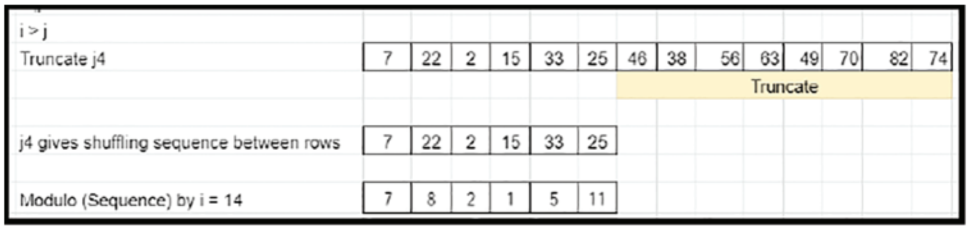
if i>j, Truncate j and Expand i														
if i<j, Truncate i and Expand j														
(i = 14) > (j = 6)														
Truncate j4 and Expand i4														
|i-j| = 8, so truncate / expand by 8														
Truncate j4 and get X for expanding i4	1	4	2	3	3	1	4	2	2	3	1	4	4	2
							Truncate							
Factor							2	2	3	3	3	3	4	4
all j4 has to be assigned ascending for last factor														
ascending no. = 1 to n														
(j4 = 4) becomes (j4 = 2)														
(j4 = 2) becomes (j4 = 1)														
													Assign	
Truncate corrected j4 and get X for expanding i4	1	4	2	3	3	1	4	2	2	3	1	4	2	1
							Truncate							
Factor							2	2	3	3	3	3	4	4
							8	6	10	11	9	12	14	13
							Store X							
X = 8 6 10 11 9 12 14 13														
i4 gives sequence for n order numbering in columns	2	4	1	3	6	5	Restore X							
	2	4	1	3	6	5	8	6	10	11	9	12	14	13
assign ascending no. for expanded i4														
ascending no. = 1 to imax (1 to 14)														
6 <-- 7														
								Assign						
corrected i4 gives sequence for n order numbering in columns	2	4	1	3	6	5	8	7	10	11	9	12	14	13

Figure 24: Sequencing-arrival of pattern-Case 2



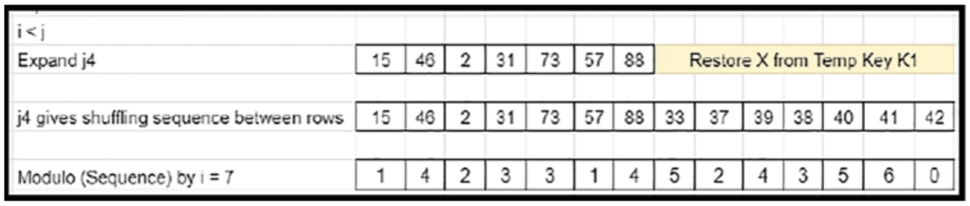
if i>j, Truncate j and Expand i														
if i<j, Truncate i and Expand j														
(i = 7) < (j = 14)														
Truncate i4 and Expand j4														
|i-j| = 7, so truncate / expand by 7														
Truncate i4 and store X	2	4	1	3	7	6	8	5	9	11	10	12	13	14
	2	4	1	3	7	6	8	Truncate						
X = 5 9 11 10 12 13 14														
assign ascending no. for truncated i4														
ascending no. = 1 to imax (1 to 7)														
7 <-- 6, 6 <-- 5, 8 <-- 7														
					Assign									
corrected i4 gives sequence for n order numbering in columns	2	4	1	3	6	5	7							

Figure 25: Sequencing-arrival of pattern-Case 3



i > j														
Truncate j4	7	22	2	15	33	25	46	38	56	63	49	70	82	74
							Truncate							
j4 gives shuffling sequence between rows	7	22	2	15	33	25								
Modulo (Sequence) by i = 14	7	8	2	1	5	11								

Figure 26: Shuffling-arrival of pattern-Case 2



i < j														
Expand j4	15	46	2	31	73	57	88	Restore X from Temp Key K1						
j4 gives shuffling sequence between rows	15	46	2	31	73	57	88	33	37	39	38	40	41	42
Modulo (Sequence) by i = 7	1	4	2	3	3	1	4	5	2	4	3	5	6	0

Figure 27: Shuffling-arrival of pattern-Case 3

7 Image Encryption

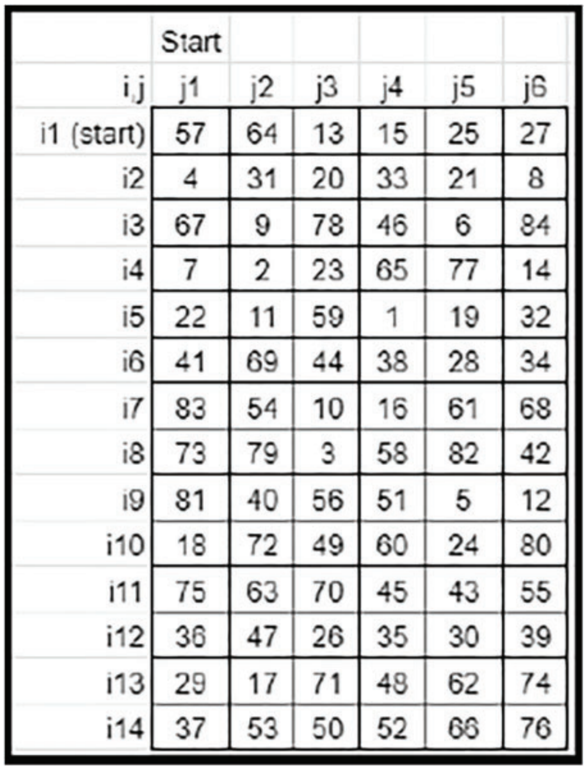
Since the crypto key and the image are of the same size, Encryption is a direct position transformation of the pixels (Type-A) or the RGB sub-pixels (Type-B).

7.1 Type-A Encryption

This type of image encryption is a direct image scrambling. In image encryption (Type-A), RGB sub-pixels are jointly position-transformed in the pattern defined by the crypto key. The Steps involved in Type-A encryption are given below.

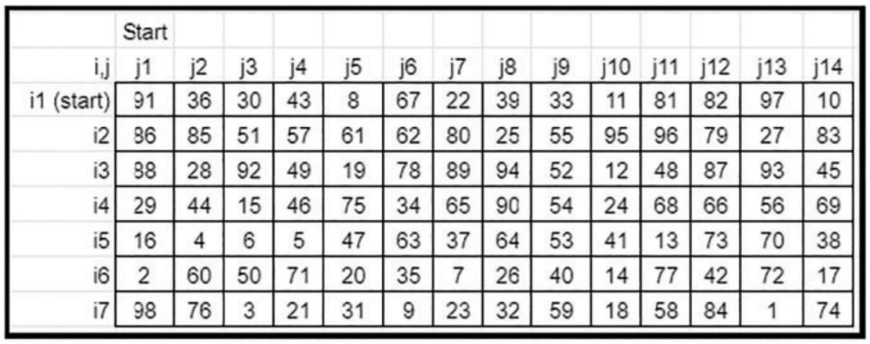
- The crypto key is read from a Text file and stored in a two-dimensional matrix of size M × N.
- The image file to be encrypted is read. The pixel data is stored in a three-dimensional matrix of size M × N × 3. M × N represents the row width (Breadth) and column width (Length), while the pixel color information is available as three parametric values corresponding to R, G and B.

- This image matrix is converted into a two-dimensional matrix of size $M * N \times 3$ by stretching each row one after another without disturbing the RGB data.
- The position of the pixel data is shuffled in a single dimension ranging from 1 to $M * N$, based on the crypto key.
- The shuffled image matrix is converted back to a 3-dimensional matrix of size $M \times N \times 3$ by linear sequencing in steps of M.
- The encrypted images (Refer to Figs. 30 to 33) are saved for analysis and reference.



	Start					
i,j	j1	j2	j3	j4	j5	j6
i1 (start)	57	64	13	15	25	27
i2	4	31	20	33	21	8
i3	67	9	78	46	6	84
i4	7	2	23	65	77	14
i5	22	11	59	1	19	32
i6	41	69	44	38	28	34
i7	83	54	10	16	61	68
i8	73	79	3	58	82	42
i9	81	40	56	51	5	12
i10	18	72	49	60	24	80
i11	75	63	70	45	43	55
i12	36	47	26	35	30	39
i13	29	17	71	48	62	74
i14	37	53	50	52	66	76

Figure 28: Sample final crypto key-Case 2



	Start													
i,j	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14
i1 (start)	91	36	30	43	8	67	22	39	33	11	81	82	97	10
i2	86	85	51	57	61	62	80	25	55	95	96	79	27	83
i3	88	28	92	49	19	78	89	94	52	12	48	87	93	45
i4	29	44	15	46	75	34	65	90	54	24	68	66	56	69
i5	16	4	6	5	47	63	37	64	53	41	13	73	70	38
i6	2	60	50	71	20	35	7	26	40	14	77	42	72	17
i7	98	76	3	21	31	9	23	32	59	18	58	84	1	74

Figure 29: Sample final crypto key-Case 3

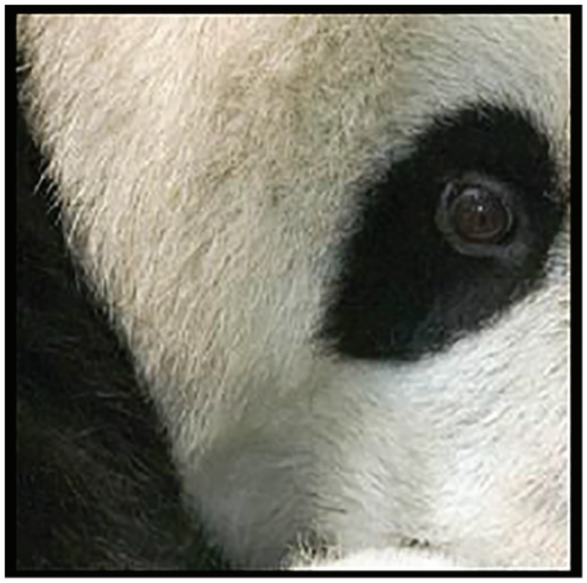


Figure 30: Source image-1 (270 × 270 pixels) -scaled down

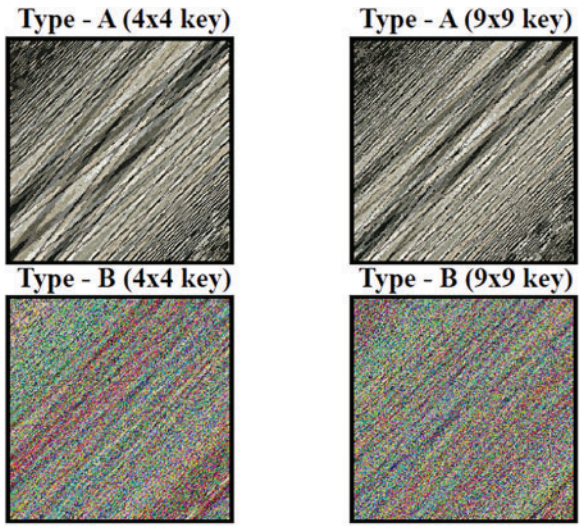


Figure 31: Encrypted images (Type-A and B) for source image-1-scaled down

7.2 Type-B Encryption

This type of image encryption is image scrambling with partial diffusion. In the image encryption (Type-B), RGB sub-pixels are differently position-transformed in the pattern defined by the crypto key. The Steps involved in Type-B encryption are given below.

- The crypto key is read from a Text file and stored in a two-dimensional matrix of size M × N. Three keys corresponding to R, G and B, each of one-dimensional matrix of size M ∗ N, are derived from the crypto key.
- The crypto key for B-pixels is obtained by simple conversion of the crypto key from a two-dimensional matrix of size M × N to one-dimensional matrix of M ∗ N by stretching each row one after another without disturbing the key.
- The crypto key for B-pixels is read from end to start. The reversed pattern key is stored as the crypto key for G-pixels.
- The crypto key for B-pixels is shifted by half its length and the shuffled pattern key is stored as the crypto key for R-pixels.
- The image file to be encrypted is read and stored in 3 separate one-dimensional matrices corresponding to R, G and B, each of size M × N. M × N represents the row width (Breadth) and column width (Length) and remains the same for all the color matrices.
- Each color matrix is converted into a one-dimensional matrix of size M ∗ N by stretching each row one after another without disturbing the color data.
- The positions of the pixel data in all the color matrices are shuffled based on the respective crypto keys in a single dimension ranging from 1 to M ∗ N.
- Each color matrix is converted from a one-dimensional matrix of size M ∗ N into a two-dimensional matrix of size M × N by linear sequencing in steps of M.
- All 2D matrices are merged by weighting the R, G and B values to form the net encrypted image.
- The encrypted images (Refer to Figs. 30 to 33) are saved for analysis and reference.

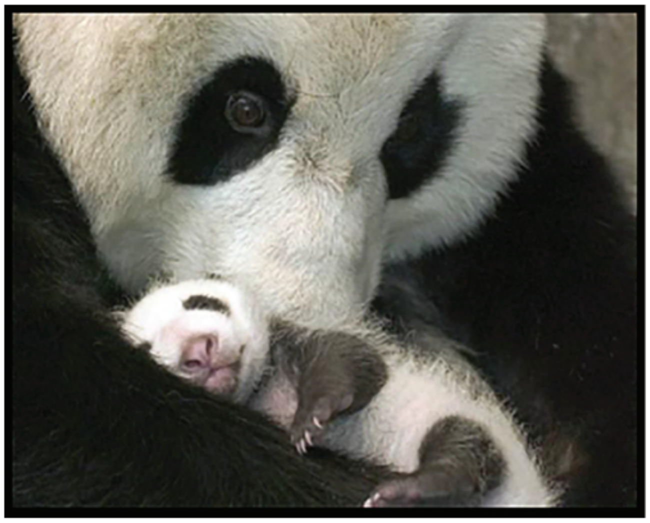


Figure 32: Source image-2 (480 × 600 pixels) -scaled down

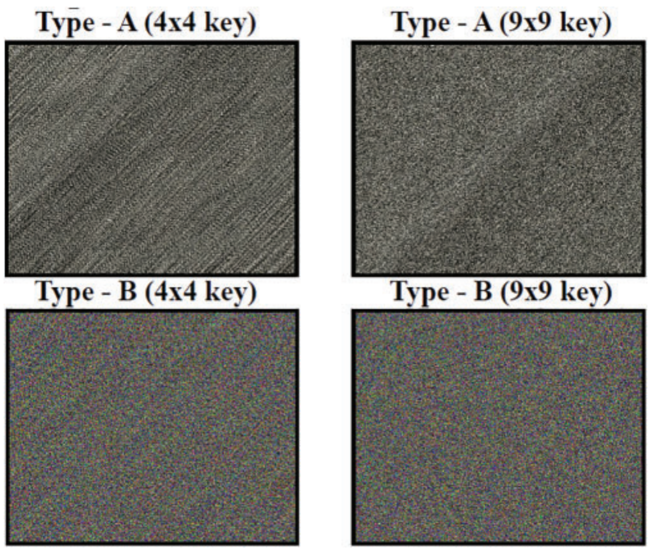


Figure 33: Encrypted images (Type-A and B) for source image-2-scaled down

8 Performance Metrics

The performance metrics are grading the encryption technique. Many methods are available for assessing the encryption efficiency. However, few of them are used in these sections for validation, namely the histogram analysis, correlation coefficient analysis, differential measure analysis and Noise analysis.

8.1 Histogram Analysis

This method is employed as a performance metric to assess and derive a graphical outcome on the Diffusion characteristics of the encryption technique [23]. The following steps are involved in the analysis.

- The source and the encrypted images are converted into equivalent grayscale images. The absolute difference between the converted grayscale image data is calculated for each pixel position. $M \times N$ data, each with an intensity value ranging from 0 to 255 in grayscale, will be obtained.
- Two hundred fifty-six grayscales are weighted by the data points containing respective intensities.
- The average is found for the pixel count with grayscale values of 0 and 255. The result is summed with the sum of products of all remaining grayscale values and corresponding pixel count. The result is divided by the total pixel count $(M * N)$.
- The Histogram deviation [25] is estimated using the equation, $HD = \left\{[(d0 + d255)/2] + \sum di\right\} / (M * N)$, with 'i' ranging from 1 to 255, di representing the amplitude of the absolute difference at the 'i'th gray level and $M * N$ representing the size of the input image.

8.2 Correlation Coefficient

One of the performance metrics to assess the correlation between the adjacent pixels in an image is the correlation coefficient. For an actual image, there is no sudden change in the adjacent pixels, hence the correlation coefficient is almost unity. This metric must be as low as possible for better encryption [23]. The following steps are involved in this analysis.

- The mean grayscale value is found for the source image by adding the grayscale values and dividing by the total pixel count ($M * N$) in an image.
- The variance for each grayscale value is found by calculating the difference between each value and the derived mean grayscale value. Difference values are squared, summed and then divided by the total pixel count ($M * N$). This calculation is done for encrypted images.
- The difference between each grayscale value and the derived mean grayscale value of the source image is multiplied by that of the encrypted image. The products obtained for $M * N$ pixels are summed and divided by the total pixel count ($M * N$). Cross-correlation value divided by the product of square roots of autocorrelation values of the source and the encrypted images gives the correlation coefficient).
- The Correlation coefficient [25] is estimated using the equation, $CC = \mathrm{cov}(x, y)/\sqrt{D(x)} * \sqrt{D(y)}$, where $D(x)$ is given by $\left\{\sum n\{xn - (\sum k\,(xk/L))\}2\right\}/L$ and $\mathrm{cov}(x, y)$ is given by $\left\{\sum n[\{xn - (\sum k\,(xk/L))\} * \{yn - (\sum k\,(yk/L))\}]\right\}/L$, with n ranging from 1 to L and k ranging from 1 to L. The x represents the plain image, y represents the cipher image and L represents the number of pixels involved in the calculation.

8.3 *Differential Measure (NPCR)*

For overcoming a differential attack, the original image could result in an encrypted image with notable impact. The Number of Pixel Change Rate (NPCR) is a standard measure to assess such impacts. NPCR is used to measure the percentage of the pixel count changed in ciphertext after making a slight change (one-pixel change) in plaintext. T, the theoretical greatest upper boundary of the NPCR is 100% [1]. The following steps are involved in the NPCR analysis.

- The encrypted image for the actual image is saved as a grayscale matrix.
- Range number is chosen between 1 and $M * N$ positions. Random values are chosen between 0 and 255 for RGB color intensities of the chosen pixel.
- The actual image is subjected to a single random pixel change based on the pixel position and values. The 1-pixel variation in the actual image is subjected to Encryption.
- A comparison is made between the encrypted image of the actual image and that of the 1-pixel varied actual image. The number of unchanged pixels is divided by the total pixel count.
- The final result is multiplied by 100 for percentage representation. The obtained result gives the number of pixel change rate (NPCR).
- The NPCR [25] is estimated using the equation, $NPCR = \left\{\sum i \sum j[D(i,j)]/(M * N)\right\} * 100\%$, where $D(i, j)$ is assigned '1' whenever $C1(i, j) = C2(i, j)$ and '0' otherwise.

8.4 *Visual Observation of Difference Image*

The pattern of the actual image remains the same when it is posed on uniform (almost) distributed data. The uniformity is tested with the encrypted data by computing the absolute difference between the actual and the encrypted images. If the resultant image still possesses the pattern of the actual image, the Encryption is a well-distributed uniform pattern (Refer to Fig. 34).

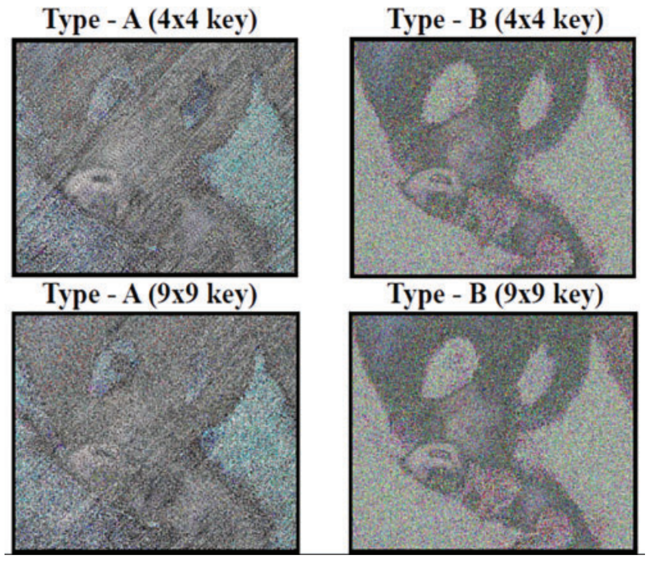


Figure 34: Difference images (Type-A and B) -scaled down

8.5 Peak Signal to Noise Ratio (PSNR)

The Peak Signal to Noise Ratio (PSNR) is a metric to assess the Noise immunity of the algorithm. PSNR measures the proportion of peak signal strength for peak corrupting noise strength. PSNR is expressed in decibels since the ratio covers a wide range of values. The Higher the PSNR, the Encryption quality is better. The following steps are involved in the PSNR analysis.

- The product of the row and column sizes and square of maximum intensity (255) is reserved as the Numerator.
- The Squares of Intensity differences between Source and Decrypted images are summed and reserved as the Denominator.
- The Ratio of the Numerator to Denominator is converted from linear to a logarithmic scale. The obtained result gives the Peak Signal to Noise Ratio (PSNR).
- The PSNR [25] is estimated using the equation, $\mathrm{PSNR} = 10 * \log 10 \left\{ (\mathrm{M} * \mathrm{N} * 2552) \left(\sum \mathrm{m} \sum \mathrm{n}[|\mathrm{f}(\mathrm{m},\mathrm{n}) - \mathrm{fd}(\mathrm{m},\mathrm{n})|]2 \right) \right\}$, where f(m, n) represents the original image and fd(m, n) represents the decrypted image.

Upon summarizing the performance metrics, pixel and sub-pixel transformations using 4×4 and 9×9 keys have different characteristics towards the plain text and differential attacks (Refer to Table 1).

Both encryption have good performance metrics. Type-B Encryption seems to have better shuffling than Encryption Type-A, which is visible in the histogram deviation, the difference image and the low correlation coefficient. Encryption Type-A is better than Type-B encryption for Differential attacks. Type B encryption is superior when used with sudoku keys. There are four classical types of attacks, namely, Cipher only attack, Known plaintext attack, Chosen cipher attack and Chosen plaintext attack.

Table 1: Performance metrics-summary

Encryption	Type-A using 4 × 4 Sudoku	Type-B using 4 × 4 Sudoku	Type-A using 9 × 9 Sudoku	Type-B using 9 × 9 Sudoku
Histogram deviation (Best = 1)	0.99608	**0.99751**	0.99606	**0.99759**
Correlation coefficient (Best = 0)	−0.01303	**0.00375**	0.01404	−**0.00034**
NPCR (Best = 100%)	**99.9802%**	99.9395	**99.9829%**	99.9583%
Difference image-Visual (Best = Good visible)	Poorly visible	**Fairly visible**	Poorly visible	**Fairly visible**

White and black images are used by attackers to find the keys [26]. The Sudoku-based encryption method is susceptible to plain images (Refer to Figs. 35 and 36). which is evident from the key-based analysis and differential attack analysis outcomes (Refer to Table 2).

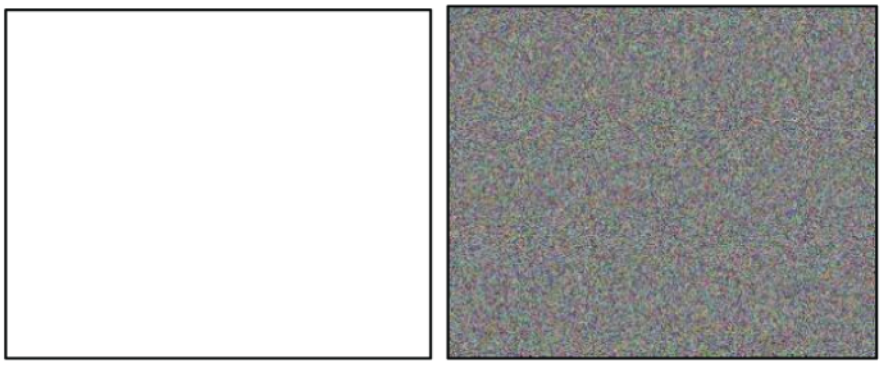


Figure 35: White input image and encrypted output-600 × 480 pixels (scaled down)

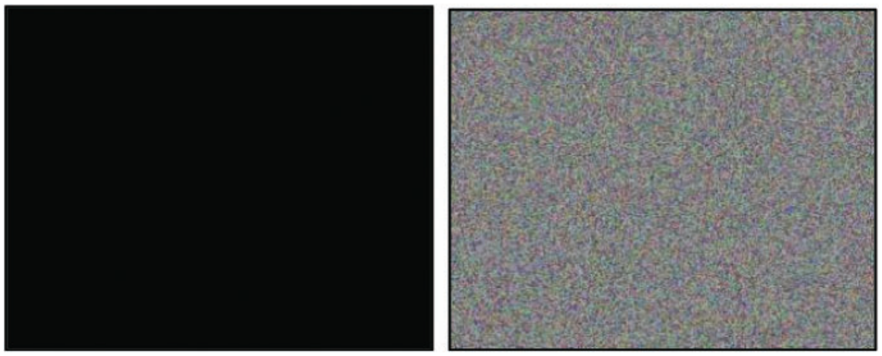


Figure 36: Black input image and encrypted output-600 × 480 pixels (scaled down)

Table 2: Performance metrics–summary

Parameters	Performance metrics	
Histogram deviation (Best = 1)	Tiffany (362 × 348 pixels)	0.99865
	Lena (512 × 512 pixels)	0.99734
	Baboon (512 × 512 pixels)	0.99718
	Image1 (600 × 480 pixels)	0.99835
	Image2 (3000 × 2400 pixels)	0.99827
	Black (600 × 480 pixels)	0.99999
	White (600 × 480 pixels)	0.99999
Correlation coefficient (Best = 0)	Tiffany (362 × 348 pixels)	0.00479
	Lena (512 × 512 pixels)	−0.01089
	Baboon (512 × 512 pixels)	0.01815
	Image1 (600 × 480 pixels)	0.00159
	Image2 (3000 × 2400 pixels)	8.7E-06
	Black (600 × 480 pixels)	nan
	White (600 × 480 pixels	nan
NPCR (Best = 100%)	Tiffany (362 × 348 pixels)	99.8627%
	Lena (512 × 512 pixels)	99.9584%
	Baboon (512 × 512 pixels)	99.9741%
	Image1 (600 × 480 pixels)	99.9517%
	Image2 (3000 × 2400 pixels)	99.9976%
	Black (600 × 480 pixels)	99.9590%
	White (600 × 480 pixels	99.9344%
PSNR (Best = Nan)	Tiffany (362 × 348 pixels)	40.7189 dB
	Lena (512 × 512 pixels)	39.7831 dB
	Baboon (512 × 512 pixels)	36.2643 dB
	Image1 (600 × 480 pixels)	42.2046 dB
	Image2(3000 × 2400 pixels)	42.9430 dB
	Black (600 × 480 pixels)	43.3596 dB
	White (600 × 480 pixels	43.3596 dB

9 Conclusion

In this paper, the attempt to apply the Sudoku key in the field of cryptography is interesting and efficient, without any compromise in Encryption. An additional advantage of using the sudoku solution as a crypto key is that the key need not be stored for Encryption or Decryption. The key can be reverse sudoku-ed. The clue is stored with details such as the element value and position. A Trade-off exists between the compacted key's size and the complexity. The main advantage of this algorithm is that this method is independent of Image size, resistant to all the classical attacks, has a high index of performance metrics and has compact key storage. The average throughput for this encryption algorithm approaches 1Mbps, which is very low due to the limitations of the conventional Google Colaboratory engine. Due to this limitation, comparative speed performance is not carried out. The actual throughput would be much better when executed in a proper processing engine. The future scope of this work is to attempt three-dimensional shifting of keys to perform RGB shuffling robustly. Multi-staged Encryption through index referencing and number flow pattern-based matrix shuffling could add more points to the efficiency of cryptography. Moving frame encryption can also be done using Sudoku solutions.

Acknowledgement: The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

Funding Statement: The work of U.F.-G. was supported by the government of the Basque Country for the ELKARTEK21/10 KK-2021/00014 and ELKARTEK22/85 research programs, respectively.

Author Contributions: M. A. P. Manimekalai-Conceptualization and Experimentation; M. Karthikeyan –Investigation and resource collection; I. Thusnavis Bella Mary-Framing of algorithm and formal analysis. K. Martin Sagayam-Manuscript review and editing; Ahmed A. Elngar-Statistical details of the existing work; Unai Fernandez-Gamiz-Language correction; Hatıra Günerhan-Plagiarism check and removal.

Availability of Data and Materials: There is no availability data and materials.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] X. Zhang, L. Wang, Y. Niu, G. Cui and S. Geng, “Image encryption algorithm based on the H-fractal and dynamic self-invertible matrix,” *Hindawi Journal-Computational Intelligence and Neuroscience*, vol. 12, pp. 1–12, 2019.
[2] S. Gao, R. Wu, X. Wang, J. Wang, Q. Li *et al.,* “A 3D model encryption scheme based on a cascaded chaotic system,” *Elsevier Signal Processing*, vol. 202, pp. 1–13, 2023.
[3] C. Wnag, B. Ma, Z. Xia, J. Li, Q. Li *et al.,* “Stereoscopic image description with Trinion fractional-order continuous orthogonal moments,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 4, pp. 1998–2012, 2022.
[4] G. S. Chandel, V. Sharma and U. P. Singh, “Different image encryption techniques-survey and overview,” *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, pp. 434–437, 2013.
[5] M. A. B. Younes and A. Jantan, “Image encryption using block-based transformation algorithm,” *IAENG International Journal of Computer Science*, vol. 35, no. 1, pp. 1–9, 2006.
[6] M. A. B. Younes and A. Jantan, “An image encryption approach using a combination of permutation technique followed by encryption,” *International Journal of Computer Science and Network Security*, vol. 8, no. 4, pp. 191–197, 2008.
[7] A. Nag, J. P. Singh, S. Khan, S. Ghosh, S. Biswas *et al.,* “Image encryption using affine transform and XOR operation,” in *IEEE Int. Conf. on Signal Processing, Communication, Computing and Networking Technologies*, Thuckalay, India, pp. 309–312, 2011.
[8] Q. Sun, W. Yan, J. Huang and W. Ma, “Image encryption based on bit-plane decomposition and random scrambling,” in *Proc. CECNet*, Yichang, China, pp. 2630–2633, 2012.

[9] Q. A. Kester, "A cryptographic image encryption technique based on the RGB PIXEL shuffling," *International Journal of Advanced Research in Computer Engineering and Technology (IJARCET)*, vol. 2, no. 2, pp. 848–854, 2013.

[10] M. M. Abbasi, "Solving sudoku by Sparse signal processing," Master's Degree Project Report, KTH Royal Institute of Technology, XR-EE-SB 2015:001, pp. 01–47, 2015.

[11] V. Kumar and A. Giridhar, "A 2D logistic map and Lorenz-Rossler chaotic system based RGB image encryption approach," *Multimedia Tools and Applications*, vol. 80, pp. 3749–3773, 2020.

[12] A. Vedaldi and B. Fulkerson, 2007. https://www.vlfeat.org/overview/plots-rank.html

[13] Auckland University, https://www.cs.auckland.ac.nz/∼rklette/CCV-CIMAT/pdfs/B04-AdvancedEdgeDetection.pdf

[14] S. Jain, 2015. https://www.geeksforgeeks.org

[15] Wikimedia Foundation, 2020. https://simple.wikipedia.org/wiki/Fingerprint

[16] Wikimedia Foundation, 2020. https://en.wikipedia.org/wiki/Wikimedia_Foundation

[17] Y. Xian and X. Wang, "Fractal sorting matrix and its application on chaotic image encryption," *Information Sciences*, vol. 547, pp. 1154–1169, 2021.

[18] L. Liu, Y. Lei and D. Wang, "A fast chaotic image encryption scheme with simultaneous permutation-diffusion operation," *IEEE Access*, vol. 8, pp. 27361–27374, 2021.

[19] J. Gunther and T. Moon, "Entropy minimization for solving sudoku," *IEEE Transactions on Signal Processing*, vol. 60, no. 1, pp. 508–513, 2012.

[20] K. A. Markus and D. Borsboom, 2010. https://psycnet.apa.org/record/2010-14802-043

[21] Astraware Limited, 2014. https://www.sudokuoftheday.com/about/difficulty

[22] Wikimedia Foundation, 2020. https://en.wikipedia.org/wiki/Glossary_of_Sudoku

[23] M. Bala Kumar, P. Karthikka, N. Dhivya and T. Gopalakrishnan, "A performance comparison of encryption algorithms for digital images," *IJERT*, vol. 3, no. 2, IJERTV3IS21325, pp. 2169–2174, 2014.

[24] Google LLC, 2017. https://colab.research.google.com

[25] C. Chattopadhyay, B. I. Sarkar and D. Mukherjee, "Encoding by DNA relations and randomization through chaotic sequences for image encryption," arXiv:1505.01795v1 [cs.CR], pp. 01–15, 2015.

[26] X. Zhang, L. Wang, Y. Wang, Y. Niu and Y. Li, "An image encryption algorithm based on hyperchaotic system and variable-step Josephus problem," *Hindawi International Journal of Optics*, vol. 2020, pp. 1–15, 2020.