Secure GIF Files Based on ZUC Stream Cipher and Present Algorithm / Original article

Suhad Fakhri Hussein
Ministry of Education / Al-Rusafa 1, Baghdad / Iraq
suhad7242@gmail.com

Abstract

Some important security needs include authentication, confidentiality, integrity, non-repudiation, and user privacy. Many security systems include these required protections for information transmission. The encryption process is one of the most important security measures. Many secure encryption algorithms are based on different keys and key lengths to ensure a high degree of security. GIF file format is a common file format that is used in several applications, and securing these files through transmission is imperative. In this paper, an efficient method for encryption GIF files is proposed based on using the modified present algorithm, modified ZUC stream cipher, and an efficient method for key generation based on the simulation of a dynamic billiard table to generate a number that controls the encryption results. The proposed method tests through experiments on tests the randomness of generated sequences, the quality of encrypted image MSE, PSNR, SSIM, entropy, the time-consuming for encryption, and the differential attack test through NPCR and UCAI. All experiments satisfy the standard results, and the method could be used in all types of multimedia file formats.

Keywords: GIF Files. ZUC Stream Cipher, Present Algorithm

1. Introduction

The recent advances in information technology and the rise of smart devices have led to an exponential increase in the need for information exchange [1]. Images, being significant sources of information, are frequently shared over the internet by users via web browsing, social networking websites, email, messages, cloud systems, etc. [2]. However, there is a need to secure this information to avoid unauthorized access [3]. The application of conventional security techniques on images has been found inadequate due to their intrinsic properties, such as high redundancy, high correlation, high dimensionality, etc. [4]. Plain images make it easy for attackers to perform statistical analyses to retrieve crucial information [6]. Recently, digital images have been chosen as cover objects to hide secret information in steganography since the redundancy in images is very high [7]. Among various image formats, the GIF format is very popular due to its various features, such as small file size, support for animation, support for lossless compression, and usability in HTML. A lot of important and sensitive information is present in GIF images that should be protected from unauthorized access [8]. Nevertheless, GIF, being a very popular file format, is very vulnerable to attacks [9]. Many conventional algorithms have been proposed to secure the GIF images. However, conventional block ciphers and algorithms such as are not suitable for securing GIF files as they faced several limitations in computation time, key size, and security level, such as the upper limit of the key space available to possible key search, security do not guarantee chaos, not good sensitivity and so on [10]. Also, these algorithms needed complicated calculations, which increases the complexity [11]. To overcome these problems, a simple and robust technique has been proposed

to restore the GIF image file using the present RC4 algorithm. This method is highly feasible, fast, and effective [12].

GIF Images contain a lot of redundancy, which may be used by an attacker to recover the plaintext from the ciphertext. The objective of securing Graphic Interchange Format (GIF) files by using the lightweight algorithm Present-ZUC is to design a lightweight algorithm that ensures confidentiality, integrity, and authenticity for GIF image files. Proposing a solution for an optimized and secure algorithm that can save RAM utilization by enhancing the algorithm.

2. Overview of GIF Files

A GIF (Graphics Interchange Format) file can contain multiple images or frames that are displayed in sequence to create an animation [13]. Each frame in a GIF represents a distinct image within the animation sequence [14]. Working with GIF frames involves extracting, manipulating, and possibly saving individual frames. Extracting Frames from a GIF To extract frames from a GIF, you need to read the GIF file and retrieve each frame's image data. This can be done using libraries that support GIF format manipulation [15].

4. ZUC Stream Cipher

ZUC is a secure and effective stream cipher that satisfies the requirements of contemporary communication networks. Because of its performance-focused architecture and strong security features, it is a good option for applications in wireless and mobile networks [16]. Several encryption systems, notably LTE (Long-Term Evolution) for mobile communications, use the lightweight ZUC stream cipher [17]. The ZUC

algorithms are thought to be robust and appropriate for LTE after being assessed by two more teams of distinguished specialists in addition to the algorithm standardization committee ETSI SAGE. ZUC is a stream cipher that is word-oriented [18]. A keystream Of 32-bit word is produced by using an initial vector and an initial key of 128 bits as input. It is possible to encrypt and decrypt data using this keystream [19].

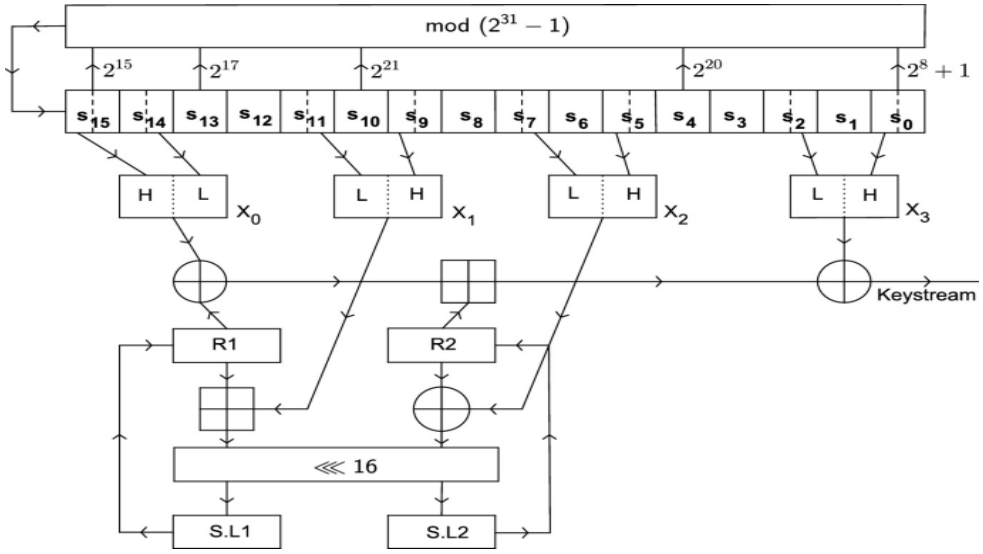

Figure 1: ZUC Stream Cipher

5. Present Encryption Algorithm

Andrey Bogdanov and associates developed the Present encryption method [20], which is a thin-film symmetric-key block cipher. It can use an 80-bit or a 128-bit key and runs on 64-bit blocks. As seen in Figure 2, Present uses a substitution permutation network topology with 31 rounds total, including a final crucial addition round [21]. Every round has these three primary purposes:

- Including a circular key: In this phase, the round key and the state data block are subjected to a straightforward XOR operation [22].
- Substitution box layer (S-Box layer): This layer uses a corresponding Substitution Box (S-Box) to transfer a four-bit input to a four-bit output, introducing nonlinearity [23].

6. Dynamical Billiards for Key Generation

A dynamical system that represents the inertial motion of a point mass inside an area with a piecewise smooth boundary and elastic reflections is called a "dynamical billiard [24]. The angle of incidence from the border is equal to the angle of reflection. In many optical, acoustic, and classical mechanical difficulties, pool tables seem like natural models [25]. It is easy to reduce the Boltzmann gas of elastically colliding hard balls in a box—the most well-known statistical mechanics model to a pool. The theory of billiards was largely sparked by the Boltzmann-Sinai hypothesis, which has yet to be proven, regarding the ergodicity of the gas of hard balls in a torus that interact elastically. [26].

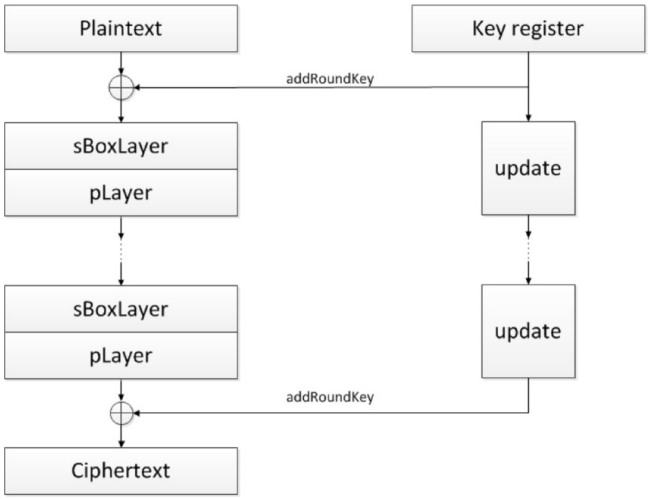

Figure 2: Present encryption algorithm

6. Proposed Method

The proposed method for encrypting a GIF image is applied by separating the frames from the GIF file. These frames are, in turn, divided into equal parts called blocks. Each block enters the encryption process. The blocks are collected in a three-dimensional matrix and permuted according to a specific key. The key generation based on dynamical billiard algorithm, the seeds of which will be used to generate keys and improve the ZUC stream cipher and Present algorithm to raise the level of its performance, then these blocks are merged into frames, and these frames are merged into a GIF file to display it as an encrypted GIF file as explain in figure 3.

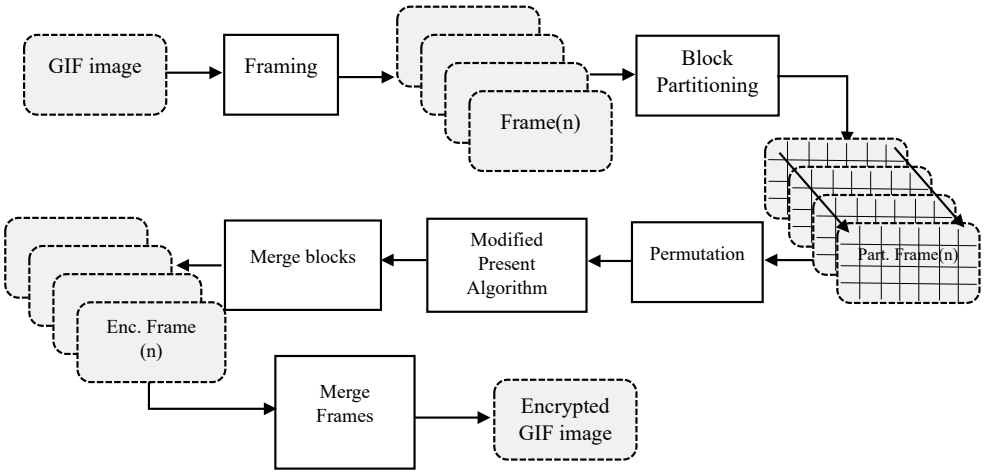

Figure 3: Present encryption algorithm

- **GIF Framing**

The framing process of the GIF file format is represented by splitting it into a sequence of slices, each one considered as an individual image. Creating many frames, controlling color tables, and specifying how each

frame should be shown are all part of the framing process in GIF files. Simple animations that are frequently used on the web can be produced using this method. Knowing these elements will help you produce and work with GIFs programmatically. GIF file framing is explained in Figure 4.

Figure 4: GIF framing process

- **Block Partitioning of Frames**

At this stage, the image is cut into equal parts called blocks. These blocks are square and of equal size. Each of these blocks is encrypted as an independent case. The block partitioning is explained in Figure 5.

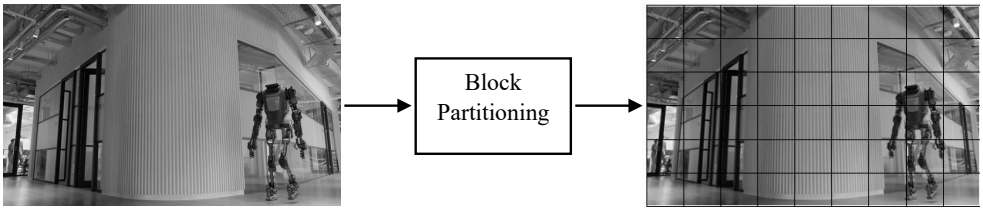

Figure 5: Partitioning of the frame into blocks process

- **Key Generation Based Dynamical Billiards**

The dynamical Billiards is used for key generation in the proposed encryption algorithm. The number generation is denoted by the following steps:

Step 1: Choose the shape of the billiard table (e.g., circular, polygonal).

Step 2: Define the boundaries of the table mathematically.

Step 3: Initialize the Particle:

Step 4: Set the initial position and velocity of the particle.

Step 5: Ensure the initial position is within the boundaries of the table.

Step 6: Simulation Loop:

Update the position of the particle over time based on its velocity.

Implement collision detection to check for interactions with the walls of the table

or obstacles.

Reflect the velocity of the particle upon collision according to the laws of physics.

(elastic collisions).

Store the trajectory data (positions and velocities) at each time step.

Step 7: Generate Randomness:

Use the stored trajectory data to create a source of randomness.

This can involve sampling the position, velocity, or angles of reflection.

Convert this randomness into a suitable format for key generation.

- **Blocks Permutation**

The permutation process is done by taking the numbers generated from the proposed method for seed number generation via a dynamic billiards table. The set of selected numbers is equal to the number of blocks in the GIF frames. These numbers are sorted in ascending or descending order, and the locations of the arranged numbers are taken as an index of the location of the block that will enter the encryption process.

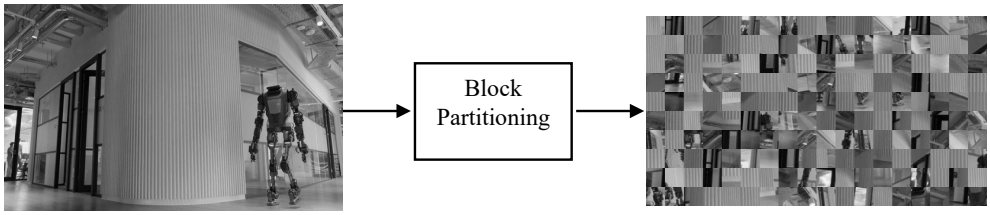

Figure 5: Partitioning of the frame into blocks process

- **Modified Present Algorithm**

The proposed method consists of two approaches for encryption; the selection of which one to apply depends on the texture complexities. If a uniform texture, a ZUC stream cipher is applied; otherwise, a modified Present algorithm is applied. The texture complexity is found by finding the variance of the block after converting it to a gray level.

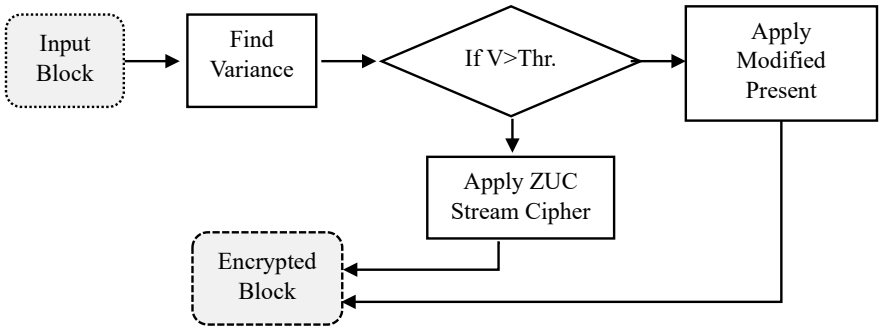

Figure 6: The proposed method for block encryption

- **Encrypted GIF reconstruction**

The encrypted blocks are merged into the matrix to reconstruct frames. The reconstruct frame's function takes the encrypted blocks and reshapes them into images using the specified frame shape. Then, the frames are merged to reconstruct the GIF file format.

7. Experimental Results

The improved efficiency of the proposed encryption algorithm is achieved through several tests applied on a set of GIF images that are used for applications such as “robot.gif”, “cars.gif”, “airplane.gif”, “forest.gif”, “sport.gif”, and “dog.gif”. These files have different frame sizes and numbers of frames. All of them are encrypted via the proposed algorithm, and the results as explained in the following sections.

The seed number generation in the proposed method uses a dynamic billiard table to find a real number that represents the position of the ball through time. The coordination of these numbers is a two-dimensional sequence and is plotted in Figure 7 as an individual curve.

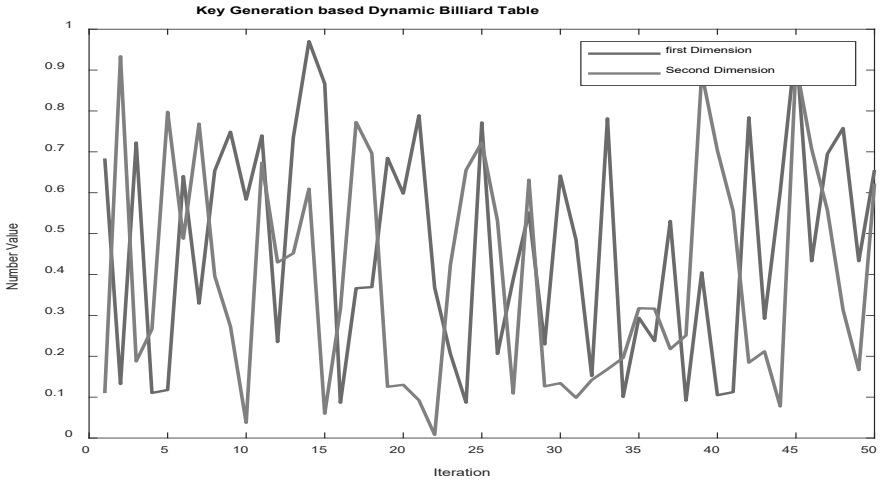

Figure 6: The Visualization of a dynamic billiard table.

The previous numbers are processed to find a hexadecimal number that is entered into the two proposed methods, Modified ZUC and Present algorithm, which need starting position (initial values) and the velocity

(control values), and the table dimension. Some samples of seed numbers are explained in Table 1.

Table 1: Generated numbers using a dynamic billiard table

No.	1st dimension	2nd dimension	No.	1st dimension	2nd dimension
1	0.791267678674	0.033726515230	7	0.50727098086981	0.25836350842357
2	0.523723688761	0.515946089995	8	0.42876735761504	0.48404762003481
3	0.591063640655	0.205753035517	9	0.26315682408109	0.09849427494733
4	0.266049358021	0.393284074524	10	0.32975535198079	0.12311848224074
5	0.504227801739	0.118147567635	11	0.44394351492245	0.10954989285146
6	0.251958023790	0.422023826478	12	0.29357599580899	0.10829997902249

These values from the previous table are converted to decimal values by getting the values of the digits after the floating point in fixed numbers for all values (10, 11, 12, 15, or more digits) as explained, then converted to a hexadecimal number as explained in Table 2.

Table 2: The Hexadecimal Form of Generated Numbers

No.	1st dimension	2nd dimension	No.	1st dimension	2nd dimension
1	'B83B3A99D2'	'761BB4E505'	7	'07DA41C81D'	'3C27AA46C7'
2	'79F05D4B38'	'63D486AAAF'	8	'7820C88E0B'	'70B37C87C2'
3	'899E236A4F'	'3D455E7811'	9	'2FE7D62AFD'	'16EEB75D83'
4	'3DF1C708C5'	'4CC6F35BAC'	10	'5B918EBC1C'	'1CAA6EF340'
5	'756651AA8A'	'675D18732A'	11	'1B8224D013'	'1981AEB4F3'
6	'3AA9DE566E'	'445A7E31A0'	12	'6242948C2E'	'19372E890E'

The standard NIST test is applied on the generated number after converted to binary numbers to find the randomness. Table 3. Explain the p-value of applying all tests on the generated seed numbers.

Table 3: The minimum, maximum, and mean p-value of generated seed numbers

Test #	Min. P-Value	Mean P-Value	Max. P-value	Test #	Min. P-Value	Mean P-Value	Max. P-value
1	0.0460	0.1297	0.0169	7	0.0011	0.1001	0.0012
2	0.0015	0.0315	0.0021	8	0.0063	0.1008	0.0009
3	0.1672	0.3486	0.0185	9	0.0175	0.0146	0.0004
4	0.5953	0.2725	0.1120	10	0.0021	0.0356	0.0094
5	0.1220	0.1363	0.1748	11	0.1820	0.1482	0.0385
6	0.0002	0.1016	0.0005	12	0.0042	0.1046	0.0009

From the previous table, these forms of numbers satisfy the requirements for generated keys used in the security requirements.

The complexity of any encryption algorithm depends on the technology used for encryption and decryption, and the size of the input data (frame size and number of frames) are some of the variables that can affect how long it takes to apply an encryption technique. to gauge how long it takes to encrypt and decrypt images of various sizes all frames are resized into (256x256, and 512x512) and get specific number of frames for standardization. The performance of the algorithm can then be evaluated by recording the time measurements for every image size. Table 2 measures and explains the average time to apply the suggested algorithm for the encryption and decryption of two different image sizes: (256 x 256) and (512 x 512).

Table 4: The average time-consuming for the proposed encryption/decryption algorithm

Image #	Encryption Time (256*256)	Decryption Time (256*256)	Encryption Time (512*512)	Decryption Time (512*512)
1	0.36358	0.77443	0.74989	0.11850
2	0.54773	0.38389	1.19114	0.16229
3	0.28991	1.38893	0.91124	0.11358
4	0.11385	0.91119	0.36396	0.13807
5	0.37205	0.88403	0.86212	0.05857
6	0.29301	1.39807	0.52619	0.02505

Several objective tests are used to assess the image quality and dissimilarity between original and encrypted images, these tests include MSE, PSNR, Entropy, and SSIM. The average of these tests is explained in Table 4.

Table 4: Image quality test using a proposed method

#	Average MSE	Average PSNR	Average SNR	Average SIM	Average Entropy
1	7884.4945	0.0689	1.9019	105.6666	7.7655
2	7241.9338	0.0610	1.6410	93.6049	7.4968
3	8897.1998	0.0982	1.4129	136.1678	7.7414
4	8377.9428	0.0912	1.7823	102.7638	7.5066
5	7455.5410	0.0571	1.7707	115.4101	7.5428
6	8656.1781	0.0700	1.5366	97.5147	7.4215

8. Conclusion

Authentication, confidentiality, integrity, non-repudiation, and user privacy are some of the most crucial security requirements. These necessary safeguards for information transfer are a part of many security systems. One of the most crucial security procedures is encryption. To guarantee a high level of security, many secure encryption techniques are based on various keys and key lengths. Common files in the GIF format are used in many applications; it is crucial to secure these files during transmission. In order to generate the numbers that control the encryption outcomes, this work proposes an efficient technique for encrypting GIF files utilizing a modified version of the current algorithm, a modified ZUC stream cipher, and an efficient approach for key generation based on dynamic billiard table simulation. The suggested approach evaluates the quality of the encrypted picture, MSE, PSNR, SSIM, entropy, the time required for encryption, the unpredictability of created sequences, and

the differential attack test using NPCR and UCAI through trials. Every experiment yielded conventional findings, and the technique could be used with any kind of multimedia file format.

7. References

[1] De Azambuja, Antonio João Gonçalves, *et al*. "Artificial intelligence-based cyber security in the context of industry 4.0—a survey." Electronics 12.8 (2023): 1920.

[2] Yadav, Uma Shree, *et al*. "Security and privacy of cloud-based online social media: A survey. "Sustainable management of manufacturing systems in industry 4.0. Cham: Springer International Publishing, 2022. 213-236.

[3] Achar, Sandesh. "Cloud computing security for multi-cloud service providers: Controls and techniques in our modern threat landscape." International Journal of Computer and Systems Engineering 16.9 (2022): 379-384.

[4] Karim, Shahid, *et al*. "Current advances and future perspectives of image fusion: A comprehensive review." Information Fusion 90 (2023): 185-217.

[5] Rehman, Mujeeb Ur, *et al*. "Efficient and secure image encryption using key substitution process with discrete wavelet transform." Journal of King Saud University-Computer and Information Sciences 35.7 (2023): 101613.

[6] Evsutin, Oleg, Anna Melman, and Roman Meshcheryakov. "Digital steganography and watermarking for digital images: A review of current research directions." IEEE Access 8 (2020): 166589-166611.

[7] Jakopec, Tomislav, and Željana Hrkač. "Use of Image File Format WebP on Websites in Croatian top Domains." 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO). IEEE, 2021.

[8] Al-Sumaidaee, Ghassan, and Željko Žilić. "Sensing Data Concealment in NFTs: A Steganographic Model for Confidential Cross-Border Information Exchange." Sensors 24.4 (2024): 1264.

[9] Puchalski, Damian, *et al*. "Stegomalware detection through structural analysis of media files." Proceedings of the 15th International Conference on Availability, Reliability and Security. 2020.

[10] Mousavi, Seyyed Keyvan, *et al*. "Security of Internet of Things based on cryptographic algorithms: a survey." Wireless Networks 27.2 (2021): 1515-1555.

[11] Freire, Pedro, *et al*. "Computational complexity evaluation of neural network applications in signal processing." arXiv preprint arXiv:2206.12191 (2022).

[12] Lin, Jingzhi, *et al*. "A new steganography method for dynamic GIF images based on palette sort." Wireless Communications and Mobile Computing 2020.1 (2020): 8812087.

[13] Shu, Xinhuan, *et al*. "What makes a data-GIF understandable." IEEE Transactions on Visualization and Computer Graphics 27.2 (2020): 1492-1502.

[14] Xie, Liwenhan, *et al*. "Wakey-Wakey: Animate Text by Mimicking Characters in a GIF." Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology. 2023.

[15] Verhoeven, Geert J., Martin Wieser, and Massimiliano Carloni. "GRAPHIS—Visualise, Draw, Annotate, and Save Image Regions in Graffiti Photos." disseminate| analyse| understand gra ti-scapes (2024): 72.

[16] Hui, Shihao, *et al*. "Design and implementation of a physical layer optical fiber security communication system based on a ZUC stream cipher." Applied Optics 63.19 (2024): 5150-5158.

[17] Goulart, Ana, *et al*. "On wide-area IoT networks, lightweight security and their applications—a practical review." Electronics 11.11 (2022): 1762.

[18] Yang, Jing, Thomas Johansson, and Alexander Maximov. "Spectral analysis of ZUC-256." IACR transactions on symmetric cryptology (2020): 266-288.

[19] Tian, Hao, and Chao Wang. "A secure and lightweight implementation scheme for Internet of Things device management based on ZUC algorithm." Sixth International Conference on Computer Information Science and Application Technology (CISAT 2023). Vol. 12800. SPIE, 2023.

[20] Hoomod, Haider K., Jolan Rokan Naif, and Israa S. Ahmed. "A new intelligent hybrid encryption algorithm for IoT data based on modified PRESENT-Speck and novel 5D chaotic system." Period Eng Nat Sci 8.4 (2020): 2333-2345.

[21] Xue, Xianglian, Dongsheng Zhou, and Changjun Zhou. "New insights into the existing image encryption algorithms based on DNA coding." Plos one 15.10 (2020): e0241184.

[22] Kolivand, Hoshang, *et al*. "Image encryption techniques: A comprehensive review." Multimedia Tools and Applications (2024).

[23] Kaur, Mandeep, Surender Singh, and Manjit Kaur. "Computational image encryption techniques: a comprehensive review." Mathematical Problems in Engineering 2021.1 (2021): 5012496.

[24] Kaloshin, Vadim, and Alfonso Sorrentino. "Inverse problems and rigidity questions in billiard dynamics." Ergodic Theory and Dynamical Systems 42.3 (2022): 1023-1056.

[25] Clark, William, and Anthony Bloch. "Invariant forms in hybrid and impact systems and a taming of Zeno." Archive for Rational Mechanics and Analysis 247.2 (2023): 13.

[26] Hameedi, Balsam A., Muntaha A. Hatem, and Jamal N. Hasoon. "Dynamic Key Generation Using GWO for IoT System." JOIV: International Journal on Informatics Visualization 8.2 (2024): 819-825.