Chaos and Fractals
Cutting-Edge Scientific Solutions

e-ISSN: 3023-8951
RESEARCH ARTICLE
Vol.1 / No.1 / 2024 / pp.6-18
https://doi.org/10.69882/adba.chf.2024072

A Novel Chaos-Based Encryption Technique with Parallel Processing Using CUDA for Mobile Powerful GPU Control Center

Harun Emre Kıran *,1
*Department of Computer Engineering, Hitit University, 19030, Corum, Turkiye.

ABSTRACT Chaotic systems possess unique properties that can be leveraged for cybersecurity. These properties stem from the complex and unpredictable nature of images, which makes it challenging for systems to interpret them. When combined with CUDA, chaotic systems benefit from high-efficiency parallel processing capabilities, allowing for the rapid and secure handling of large data sets. Therefore, chaotic systems can be effectively used to securely store and conceal images. In this study, a CUDA-supported chaos-based parallel processing encryption mechanism for mobile control centers is developed. Encryption processes leverage the powerful GPU of the control center. This allows for the fast encryption and decryption of image data received from multiple devices connected to the control center via wired or wireless connections. For encryption, the Logistic Map is used to generate random numbers. Using this map, image data is subjected to XOR operations, encrypting the R, G, B, and Gray Scale channels of the images. Initially, an analysis of the numbers generated from this map is conducted, followed by a detailed explanation of the encryption technique. The technique is then applied to image data, and image analyses are performed. Finally, the performance of the encryption technique is compared with other studies, and encryption speeds are examined. The results show that the new encryption technique provides significantly fast encryption and security levels comparable to other studies. The key discovery of this research is that the devised mechanism is well-suited for parallel processing, allowing for rapid image encryption using the proposed method. For encrypting large IoT files, random number generation is initially performed, followed by statistical tests. Subsequently, encryption is executed using the developed algorithm, and security analyses are conducted. The performance of the proposed mechanism is compared with other studies in the literature, and the results from image analysis and encryption performance demonstrate that the developed mechanism can be effectively used with high security for IoT applications.

KEYWORDS
Chaos-based encryption
CUDA programming
Logistic map
Image encryption
Internet of things

INTRODUCTION

Chaos theory is a field of study that explains nonlinear phenomena that appear disordered but have an underlying order. This theory addresses two concepts: determinism and randomness. Specifically, deterministic chaos theory explains that complex systems governed by deterministic rules exhibit behaviors that appear random due to their sensitivity to initial conditions. Due to the deterministic nature of this chaos, it is used in various fields such as encryption (Clemente-Lopez *et al.* 2024), motor control (Mai *et al.* 2015), image processing (Boyraz *et al.* 2022), random number generation (Tutueva *et al.* 2020), and the Internet of Things (Kumari and Mondal 2023).

Chaos theory plays a significant role in cryptography by explaining the mathematical order behind seemingly random events, thus ensuring data security (Naik and Singh 2024). This characteristic has enabled the development of innovative encryption methods in various fields. Furthermore, the inherent randomness of chaotic systems has proven to be a valuable resource for generating strong random numbers, which are a fundamental component of many cryptographic algorithms (Man *et al.* 2021).

The arrival of the Fourth Industrial Revolution has brought the Internet of Robotic Things to the forefront by combining the capabilities of the Internet of Things with autonomous robots, initiating a new era of interconnectedness (Romeo *et al.* 2020). This transformative technology has revolutionized industries such

Manuscript received: 3 May 2024,
Revised: 28 July 2024,
Accepted: 30 July 2024.

[1] harunemrekiran@hitit.edu.tr (**Corresponding author**)

as manufacturing (Singh *et al.* 2021), agriculture (Kashyap *et al.* 2021), healthcare (Rajendran and Doraipandian 2021), education (Francisti *et al.* 2020), and surveillance (Ghosh *et al.* 2021), leading to limitations in processing power and storage capacity commonly encountered in robots that necessitate efficient methods to encrypt large data sets. To overcome this challenge, a novel discrete-time chaotic encryption mechanism is proposed to securely store and conceal large data sets in robots (Kiran *et al.* 2023; Erkan *et al.* 2023).

Compute Unified Device Architecture (CUDA), the parallel computing platform and application programming interface developed by NVIDIA, has emerged as a game changer in high-performance computing. Its ability to leverage the general-purpose processing power of Graphics Processing Units has led to significant advancements in various fields, including cryptography (Jadhav *et al.* 2023). CUDA's parallel processing capabilities have been particularly useful for accelerating computationally intensive encryption and decryption tasks, making it an attractive option for real-time encryption and decryption applications.

Today, there are many studies related to parallel encryption using CUDA, especially in conjunction with chaos. The paper presented by Bezerra *et al.* (2024) introduces a novel single-core parallel image encryption scheme based on chaotic maps, and this method, optimized for GPU architectures, provides significantly higher efficiency than existing methods. The proposed scheme offers a robust solution for real-time image encryption by demonstrating strong resistance to various types of attacks. The paper by Elrefaey *et al.* (2021) introduces a parallel implementation of a chaotic map-based image encryption algorithm using the Baker map and the Chen map. This parallel implementation, using GPUs, significantly speeds up the encryption and decryption processes for high-resolution images and long videos, making it suitable for real-time applications. The work by You *et al.* (2020) introduces a new algorithm based on hybrid chaotic maps for image encryption. By using a 1D logistic map and a 2D logistic chaotic dynamic system, the algorithm provides high security by scrambling pixel positions and values of images and accelerates the process using OpenCL. Song *et al.* (2022) proposes a fast and secure algorithm using intrinsic properties of chaotic systems, reversible steganography, and parallel computing for batch image encryption. The algorithm distributes batch images equally to each thread, applies Cipher Block Chaining (CBC) mode among neighboring images, and encrypts and embeds thread identifiers and CBC indexes into the encrypted images. By using the logistic map and the Chen system as chaotic systems in the encryption process, it enhances resistance to chosen-plaintext attacks. The paper by Bharadwaj *et al.* (2021) presents a GPU-accelerated implementation of an image encryption algorithm optimized with genetic algorithms. The algorithm provides high performance and security by using a modified XOR cipher to encrypt images and a genetic algorithm-based pseudo-random number generator with CUDA programming.

The main contributions of this paper to the literature are listed in order below:

- A new and simple encryption mechanism for three-dimensional images using the Logistic Map has been presented.
- The mobile computer version of the flagship GPU with CUDA support, which was released almost in the last year, has been used for the first time in the control center role and its performance has been tested.
- The importance of CUDA-supported encryption has been once again highlighted and proven with tests in the paper.
- The developed encryption technique has been implemented and the resulting outputs have been analyzed.

The remainder of this paper is organized as follows: Section 2 addresses the chaoticity analysis of the Logistic Map. Section 3 explains the generation of random numbers and randomness tests. Subsequently, Section 4 introduces the chaos-based encryption technique. Section 5 presents the analysis of images encrypted with this technique. Section 6 compares the proposed technique with other existing works. Finally, Section 7 discusses the conclusions and future work.

CHAOTIC ANALYSIS OF THE LOGISTIC MAP

Discrete-Time Logistic Map Chaotic System

In the recommended light encryption mechanism, the discrete-time chaotic system chosen for data encryption is the Logistic Map. The Logistic Map is particularly used to generate random numbers, which are crucial for encrypting image data through XOR operations. This chaotic system is preferred due to its ability to efficiently generate pseudo-random sequences necessary for the encryption process. The equation for the Logistic Map is presented in Equation 1. Here, X_n represents the current state, and r is the control parameter that dictates the system's behavior. By adjusting the value of r, the Logistic Map can produce a range of behaviors from steady-state to chaotic.

$$X_{n+1} = rX_n(1 - X_n) \tag{1}$$

The parameters necessary for the Logistic Map in this study are provided in Table 1. According to the table, the initial value $X(0)$ is set to 0.61, and the control parameter r is set to 3.9. For the random number generation process during encryption, 6,291,457 iterations were performed, and the least significant 4 bits of the binary representation of the number obtained in each iteration were used.

Time Series Analysis of the Values Generated by the Logistic Map

The time series analysis of the values generated by the Logistic Map is shown in Figure 1. The graph represents the values obtained from 250 iterations of the Logistic Map. The absence of periodicity in the time series demonstrates the chaotic behavior of the Logistic Map, which is essential for its effectiveness in encryption processes. This chaotic nature ensures that the values are unpredictable, thus enhancing the security of the encryption mechanism.

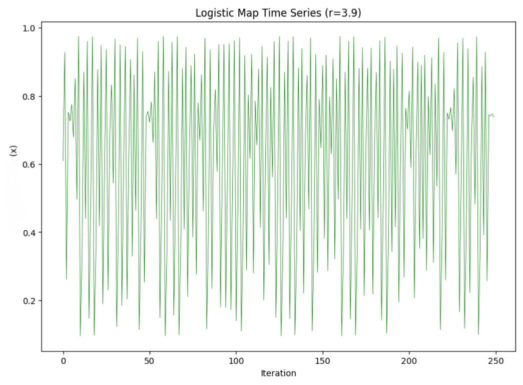


Figure 1 Time series of the values generated by the Logistic Map over 250 iterations. The plot illustrates the absence of periodicity, indicating the chaotic nature of the Logistic Map for $r = 3.9$.

Table 1 Logistic Map Parameters

Chaotic Systems	Logistic Map
X(0)	0.61
r value	3.9
Number of Iteration	6291457
Number of Binary	4

Initial Condition Sensitivity Analysis of the Logistic Map

The initial condition sensitivity analysis of the Logistic Map is depicted in Figure 2. The graph presents the time series of values generated over 250 iterations for two slightly different initial conditions: $X(0) = 0.61$ and $X(0) = 0.61 + 10^{-10}$. This analysis demonstrates that even a minute change in the initial condition results in significantly different trajectories, which is a hallmark of chaotic systems. The red and green lines represent the two initial conditions, respectively, and their rapid divergence emphasizes the sensitive dependence on initial conditions, underscoring the importance of precision in chaotic system applications.

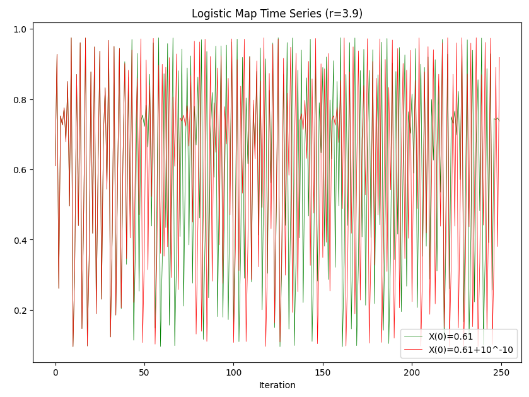


Figure 2 Time series of the Logistic Map values over 250 iterations for two slightly different initial conditions: $X(0) = 0.61$ and $X(0) = 0.61 + 10^{-10}$. The plot demonstrates the significant divergence in values due to the small change in the initial condition, illustrating the sensitive dependence on initial conditions characteristic of chaotic systems.

Logistic Map Chaotic System Function

The analysis utilizes a single map function due to the one-dimensional nature of the two nonlinear systems under consideration. The map functions are represented with X_n on the x-axis and X_{n+1} on the y-axis. Figure 3 illustrates the map function derived from data over 100,000 iterations. In this figure, there is a specific relationship between the values of X_n and X_{n+1}. These nonlinear systems demonstrate chaotic behavior as the values at each step differ from one another.

The graph in Figure 3 illustrates the relationship between X_n and X_{n+1} for the Logistic Map with $r = 3.9$. The plot shows how the population ratio changes iteratively, reflecting the quadratic nature of the Logistic Map. As the iterations proceed, the values generated by the map function demonstrate chaotic behavior, as evidenced by the lack of a repeating pattern and the sensitivity to initial conditions. This characteristic is essential for encryption processes, where unpredictability and complexity of the sequence are crucial for security.

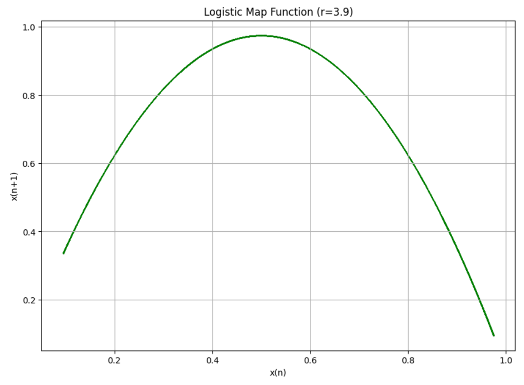


Figure 3 The map function of the Logistic Map for $r = 3.9$ over 100,000 steps. The x-axis represents X_n and the y-axis represents X_{n+1}.

Bifurcation Diagram of the Logistic Map

Bifurcation diagrams are a crucial tool for understanding the chaotic properties of a system. These diagrams illustrate the dynamic changes that occur as system parameters vary, allowing for an in-depth analysis of the system's chaotic behavior. Additionally, they help identify the parameter values at which the system exhibits chaotic or regular behavior.

For the Logistic Map system, the parameter r can range from 2 to 4, and the bifurcation diagram for this interval is presented in Figure 4. As depicted in the diagram, the system shows chaotic behavior within the range of 3.5 to 4.

RANDOM NUMBER GENERATION AND RANDOMNESS TESTS GENERATED BY THE LOGISTIC MAP

The Process of Generating Random Numbers

In this section, the process of generating random numbers used in the study is discussed. The Logistic Map chaotic system, introduced and analyzed for chaotic behavior in the previous section, will be utilized here.

The random number generator based on the Logistic Map is presented in Algorithm 1. Additionally, the parameters used in the algorithm are presented in Table 1. The algorithm starts with the parameter `r` and the initial condition `x0`. In the first step, the initial condition is used to calculate the first value with the Logistic Map

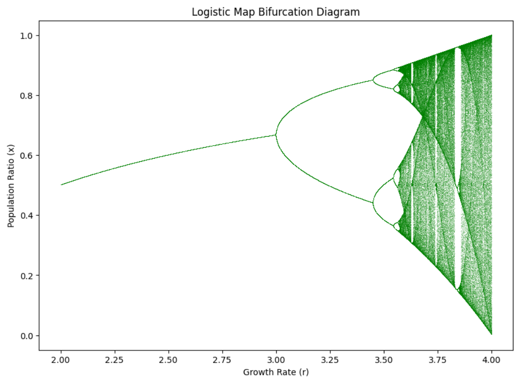


Figure 4 Bifurcation diagram of the Logistic Map for r values ranging from 2 to 4. The population ratio x is plotted against the growth rate r.

equation and assigned to `data[0]`. Then, for the specified number of steps, a loop is run where the Logistic Map equation is used in each step to calculate new values and add them to the list. The initial value is deleted from the list since it is used as the starting condition.

Subsequently, the number of binary digits to be obtained from each floating point number is specified, and these binary digits are added to the list called `binary_data_array`. This process continues by obtaining and adding the j-th lowest binary value of each floating point number to the list.

The obtained binary data list is divided into the specified number of parts (`num_parts`), and the size of each part (`part_size`) is calculated. Here, the number of parts is set to 8. These parts are assigned to the `parts` list, and each part is then added to the newly created `binary_data_array_new` list.

As a result, the random numbers generated using the Logistic Map are converted into binary values, which are then organized for later use. This algorithm creates an effective random number generator by leveraging the randomness characteristic of the Logistic Map chaotic system, thereby ensuring reliable random number generation.

Randomness Analysis of the Generated Random Numbers

In this subsection, the randomness analysis of the random numbers generated by the algorithm introduced in the previous subsection is conducted. The randomness analysis was performed using the NIST-800-22 and ENT (Beirami *et al.* 2012) tests.

The NIST-800-22 test, which is widely used worldwide, consists of 15 different statistical tests (Koyuncu 2014). For the test to be considered successful, all these statistical tests must pass. This provides evidence that the generated numbers are random. The NIST-800-22 test produces results based on the P-value, and for the results to be considered successful, the P-value must be greater than the threshold value of 0.001. In this study, the first 3,000,000 binary values generated by the Logistic Map were used. The results obtained are presented in Table 2. As seen in the table, all tests are considered successful as they meet the required conditions.

Another significant test for assessing the randomness in this study is the ENT test. Developed by John Walker, this test includes five different statistical evaluations. The entropy test measures the information density of the sequence, expressed as the number of bits per byte, with an ideal entropy level of 8. Entropy indicates the amount of information contained in the generated sequence. The chi-square test assesses the randomness of the sequence and

Algorithm 1 Pseudorandom number generation pseudo code

```
1: Result: Random Binary Numbers
2: Start
3: Enter parameter r=3.9
4: Enter initial condition x0 $\leftarrow$ 0.1
5: Number of steps number_of_steps
6: data[0] $\leftarrow$ equation(x0)
7: for $i \leftarrow 1$ to number_of_steps do
8:     data[i] $\leftarrow$ equation(data[i-1])
9: delete(data[0])
10: binary_data_array $\leftarrow$ []
11: Binary number to get from float number_of_binary
12: for $i \leftarrow 0$ to number_of_steps-1 do
13:     for $j \leftarrow 1$ to number_of_binary do
14:         Get $j$-th lowest value binary of data[i] $\rightarrow$ binary_data
15:         Append binary_data to binary_data_array
16: Number of parts to divide the list into
17: num_parts $\leftarrow$ 8
18: part_size $\leftarrow$ len(binary_data_array) // num_parts
19: parts $\leftarrow$ [binary_data_array[i*part_size:(i+1)*part_size]
    for $i$ in range(num_parts)]
20: Initialize the new list binary_data_array_new $\leftarrow$ []
21: for $i \leftarrow 0$ to part_size-1 do
22:     for part in parts do
23:         Append part[i] to binary_data_array_new
24: End
```

is highly sensitive to errors in the PRNG. The arithmetic mean is computed by dividing the sum of all bytes in the generated file by the file length, with a value close to 127 indicating randomness. The Monte Carlo test demonstrates that the sequence converges to π with a minimal error of 0.01 percent, suggesting the sequence's proximity to randomness. The serial correlation coefficient test ensures that there is no correlation between each byte and the previous byte in the sequence. All generated binary random numbers were subjected to this test, and the results are presented in Table 3. As shown in the table, the results are close to the optimal values that define randomness. Therefore, the outcomes from both tests confirm that the numbers can be used reliably.

OVERVIEW OF THE NEW CHAOS-BASED ENCRYPTION TECHNIQUE

In this section, the encryption technique will be introduced. This technique will be used in the control center that controls the devices. The communication structure of the control center with other devices is shown in Figure 5. When the mobile control center communicates with other devices, the data is encrypted and decrypted with the support of the GPU. Thus, although the mobile control center has a weaker processing capacity compared to other advanced control centers, the delay in data encryption and decryption is minimized thanks to the CUDA support of the GPU.

In the study, encryption initially starts with the generation of random numbers as described in the previous section. The generated random numbers are then applied to RGB or Gray-Scale images to encrypt these images. The encryption of each channel is done separately as shown in Figure 6. The process of encrypting the random numbers for each of the RGB channels is presented in Algorithm 2. The first phase of the algorithm involves loading the data into the GPU and performing the XOR operation. Initially, the generated random numbers are split into three lists using

Table 2 Logistic Map P-Values and Test Results

Test	P-Value (Logistic Map)	Result
Frequency Monobit	0.60712	Passed
Frequency Test (M = 128)	0.03616	Passed
Run Test	0.72421	Passed
Test for The Longest	0.44117	Passed
Binary Matrix Rank	0.62683	Passed
Discreet Fourier Transform	0.84356	Passed
Non-overlapping (Single, B = 111 111 111)	0.00069	Passed
Overlapping Temp (B = 111 111 111)	0.07939	Passed
Maurier's Universal	0.20237	Passed
Linear Complexity (M = 500)	0.15630	Passed
Serial Test-1 (m = 16)	0.08933	Passed
Serial Test-2 (m = 16)	0.19621	Passed
Approximate Entropy (m = 10)	0.74849	Passed
Cumulative Sums Test (Forward)	0.57423	Passed
Random Excursion Test (x = -1)	0.62257	Passed
Random Excursion Variant Test (x = -6)	0.65029	Passed

Table 3 ENT Test Results

Test Name	Average	Ideal Results	Result
Arithmetic Mean	127.45217	127.5	Success
Entropy	7.99993	8	Success
Correlation	0.00032	0	Success
Chi Square	278.19856	10% and 90% between	Success
Monte Carlo	3.13186	3.1415	Success

the `split_list_into_three` function. Then, the CUDA kernel is defined, and the necessary kernel code for the XOR operation is created. This kernel code is a simple CUDA function that performs the XOR operation between two lists at each index. After the kernel code is compiled, preparations are made to run the function in the CUDA context. In this phase, the lists are converted to numpy arrays, and empty arrays are created to store the results. Subsequently, these numpy arrays are copied to the GPU memory. The block and grid sizes are defined, and the XOR operation is executed on the GPU. Finally, the calculated results are copied from the GPU memory to the CPU memory, completing the process. Thus, the encryption data for each channel is obtained.

The process of applying the encryption data generated for each channel to each channel is shown in Algorithm 3. The algorithm involves applying the encryption data to the RGB or Gray-Scale image channels. First, a new CUDA kernel code is defined, and each channel of the images is flattened. Then, these channels are divided into 64 parts, and each part is assigned as a new channel list. After creating empty arrays to store the results, the data is copied to the GPU memory. The block and grid sizes are defined, and the XOR operation is performed on the GPU. Once the process is complete, the calculated results are copied from the GPU

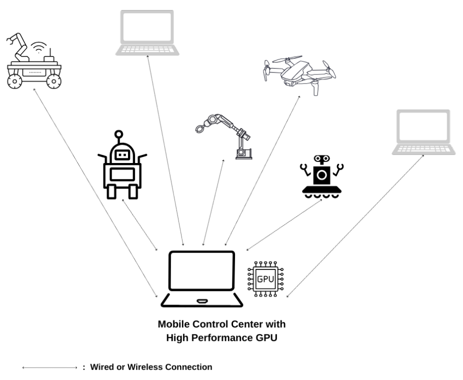

GPU
Mobile Control Center with High Performance GPU
: Wired or Wireless Connection

Figure 5 The wired or wireless connection structure of the mobile control center with other devices.

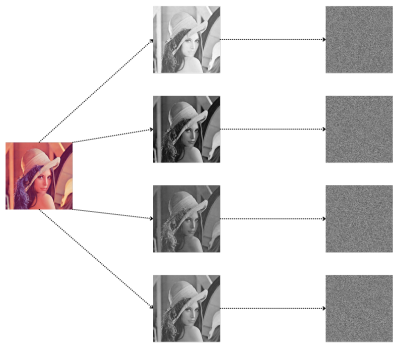


Figure 6 Encryption of each channel in RGB images or Gray-Scale image.

memory to the CPU memory. Finally, the results obtained for each channel are combined to produce the final encrypted image.

TESTS OF IMAGES ENCRYPTED WITH THE NEW TECHNIQUE

Image Analysis between Encrypted Image and Original Image

This section examines the differences between the original image and the encrypted image using the Lena image, a standard in image processing. Histogram, correlation, and entropy analyses were conducted on both the original and encrypted images, and differential attack analyses (NPCR and UACI) were performed on the encrypted image. NPCR evaluates the percentage of differing pixels between two encrypted images generated by altering a single pixel in the plaintext image, while UACI measures the average of the absolute differences between corresponding pixels in two encrypted images created by altering a single pixel in the plaintext image. In the NPCR test, two nearly identical plaintext images, differing by only one pixel, are encrypted, with a high NPCR value indicating strong resistance to differential attacks. Similarly, in the UACI test, two nearly identical plaintext images, differing by only one pixel, are encrypted, with a low UACI value indicating strong resistance to differential attacks. For the encryption method to be considered resistant to differential attacks, the NPCR value should be at least 99.9% and the UACI value should be less than 0.01. Additionally, to assess the performance of the new encryption method, histogram, correlation, and entropy analyses were carried out on both the original and encrypted images. The histogram displays the distribution of pixel intensities in an image. The formula for calculating the histogram of an image is:

$$H(i) = \sum_{j=0}^{L-1} \delta(i - f(j)) \tag{2}$$

Correlation measures the linear relationship between two variables. The formula for calculating correlation is:

$$C = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} [(f(i,j) - \bar{f})(g(i,j) - \bar{g})]}{\sqrt{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (f(i,j) - \bar{f})^2 \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (g(i,j) - \bar{g})^2}} \tag{3}$$

Entropy measures the amount of information in an image and expresses the degree of randomness or uncertainty in the data input. High entropy indicates that the data is more unpredictable

Algorithm 2 Encryption data for R (or gray-scale), G, B channels

```
Result: Encryption data for R (or gray-scale), G, B channels
Start
part1, part2, part3 ← split_list_into_three(binary_data_array_new)
Define CUDA kernel code for XOR operation:
kernel_code ← "
__global__ void xor_lists(int *list1, int *list2, int *result, int size) {
int idx = threadIdx.x + blockIdx.x * blockDim.x;
if (idx < size) {
result[idx] = list1[idx] ^ list2[idx]; } }"
mod ← SourceModule(kernel_code)
xor_lists ← mod.get_function("xor_lists")
Convert lists to numpy arrays
part1_np ← part1, part2_np ← part2, part3_np ← part3
Allocate space for results
zeros_like(r_kanal_list←part1_np, g_kanal_list←part2_np, b_kanal_list←part3_np)
Copy lists to GPU memory
cuda.mem_alloc(part1_gpu←part1_np, part2_gpu←part2_np, part3_gpu←part3_np, r_kanal_gpu←r_kanal_list, g_kanal_gpu←g_kanal_list, b_kanal_gpu←b_kanal_list)
cuda.memcpy_htod(part1_gpu←part1_np, part2_gpu←part2_np, part3_gpu←part3_np)
Define block and grid sizes
block_size ← 256
grid_size ← ceil(len(part1_np) / block_size)
Perform XOR operation on GPU
xor_lists(part1_gpu, part2_gpu, r_kanal_gpu, len(part1_np), block=(block_size, 1, 1), grid=(grid_size, 1))
xor_lists(part1_gpu, part3_gpu, g_kanal_gpu, len(part1_np), block=(block_size, 1, 1), grid=(grid_size, 1))
xor_lists(part2_gpu, part3_gpu, b_kanal_gpu, len(part2_np), block=(block_size, 1, 1), grid=(grid_size, 1))
Copy results back to CPU
cuda.memcpy_dtoh(r_kanal_list←r_kanal_gpu, g_kanal_list←g_kanal_gpu, b_kanal_list←b_kanal_gpu)
End
```

Algorithm 3 CUDA based XOR Operation for Any Channel

```
1: Result: Encrypted Image Channel
2: Start
3: Define new CUDA kernel code for image XOR operation (The
   definition is the same as in the Algorithm 2.)
4: Load channel images and flatten them
5: load_image_as_array(any_channel ← any_channel_file)
6: Split the list into 64 parts
7: num_parts ← 64
8: part_size ← len(any_kanal_list) // num_parts
9: Assign the first part as new channel lists
10: any_kanal_new_list ← any_kanal_list[:part_size]
11: Allocate space for results
12: zeros_like(any_result ← any_kanal_new_list)
13: Copy data to the GPU
14: cuda.mem_alloc(any_channel_gpu ← any_channel)
15: cuda.mem_alloc(any_list_gpu ←
    any_kanal_new_list.nbytes, any_result_gpu ←
    any_result)
16: cuda.memcpy_htod(any_channel_gpu ← any_channel,
    any_list_gpu ← any_kanal_new_list)
17: Define grid and block sizes
18: block_size ← 256
19: grid_size ← ceil(any_channel.size / block_size)
20: Perform XOR operation on GPU
21: xor_lists(any_channel_gpu, any_list_gpu,
    any_result_gpu, any_channel.size, block=(block_size,
    1, 1), grid=(grid_size, 1))
22: Copy results back to CPU
23: cuda.memcpy_dtoh(any_result ← any_result_gpu)
24: Combine channels back into images
25: image_shape ← (image_size_1, image_size_2)
26: Process Any channel result
27: any_result_image ← any_result.reshape(image_shape)
28: End
```

and thus harder to compress or encrypt. In encryption, high entropy means more randomness in the input, which makes it harder for an attacker to detect patterns or weaknesses in the encryption algorithm. Low entropy indicates that there may be too many patterns or repetitions in the data, which can make it easier for an attacker to decrypt the message. Therefore, to ensure strong encryption, a balance between entropy and predictability must be established. The formula for calculating entropy is:

$$S = -\sum_{i=1}^{N} p_i \log_2(p_i) \tag{4}$$

Figure 7 displays the histogram analyses of the original image (shown on the left) in Figure 6, as well as its encrypted versions in the order of R, G, B, and grayscale channels. In Figure 7, the R, G, B, and grayscale channels are presented separately as histogram plots. Examining the histogram plots of the different channels of the original image, it is observable that each channel contains distinct information. However, when analyzing the histogram plots of images encrypted using new random numbers generated by the Logistic Map, these plots can be considered as purely noise data. This indicates that no meaningful information can be extracted from the histogram of the image.

Figure 8 presents the cross-correlation plots for the R, G, B, and gray-scale channels of the same image. As can be seen, while information can be extracted from the channels of the original image, the situation in the encrypted image channels appears spread out and homogeneous. This indicates that the image channels have been well encrypted.

Tables 4 and 5 provide the Horizontal and Vertical Correlation Coefficient values for the original image and its encrypted versions. Calculations have been performed separately for each channel (R, G, B, grayscale). As seen in the tables, for the original image in different dimensions, both horizontal and vertical correlation values for each channel range from 0.7 to 1.0, indicating the presence of information within the image. However, after encrypting the image channels using random numbers generated from the Logistic Map in different dimensions, these values have approached 0. This demonstrates that the encryption is robust.

Table 6 presents the NPCR, UACI, and Entropy values obtained from different sizes and channels of the Lena image after encryption (including the entropy values of the original images). Here, the NPCR value shows the normalized form of the changing values for 8-bit pixels between 0 and 1, while the UACI values indicate the normalized form of the intensity of changing pixels, also between 0 and 1. Considering all these values, it is evident that the discrete-time chaotic encryption technique is successful.

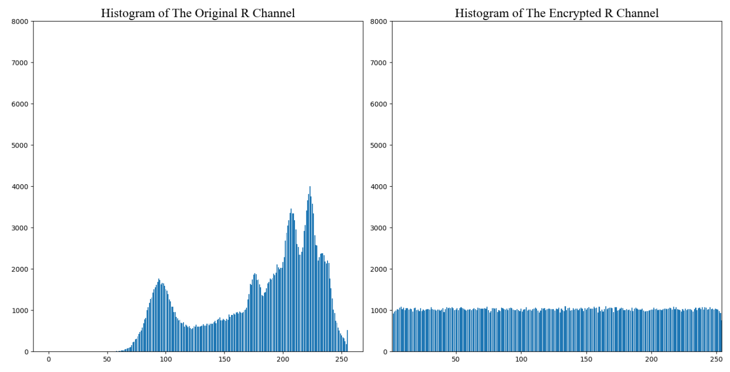
Processing Time of the Encryption Technique in Mobile Control Center for Video-Based Applications

In this sub-section, the encryption speeds of video frames from devices connected to the Mobile Control Center will be analyzed. Images with sizes of 128, 256, 512, and 1024 were used for encryption. In the simulation environment, the number of connected devices was set to 1, 5, 10, 20, 50, and 100. Additionally, the laptop version of the NVIDIA GeForce RTX 4090 model was used for encryption. The average number of frames encrypted by the control center from the image data received from each device was calculated. The results obtained are presented in Table 7. The table shows the number of frames processed according to image sizes and the number of devices.

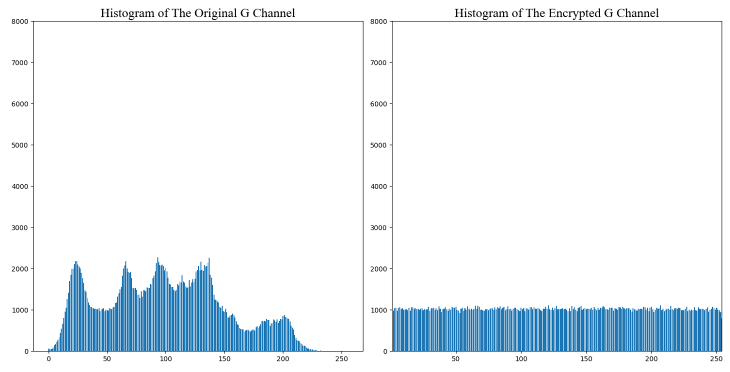
When examining the obtained results, it is observed that the significant drop in fps occurs with the increase in image sizes. Although the increase in the number of devices for each size decreases the fps value, this decrease is not significant. Particularly, the GPU used for encryption has not been heavily burdened during these processes. There has only been a busy state in CPU-GPU read/write operations. However, this busy state is not at a level that would affect other tasks. This indicates that the proposed technique is efficient.

COMPARISON OF THE PROPOSED TECHNIQUE WITH EXISTING WORKS

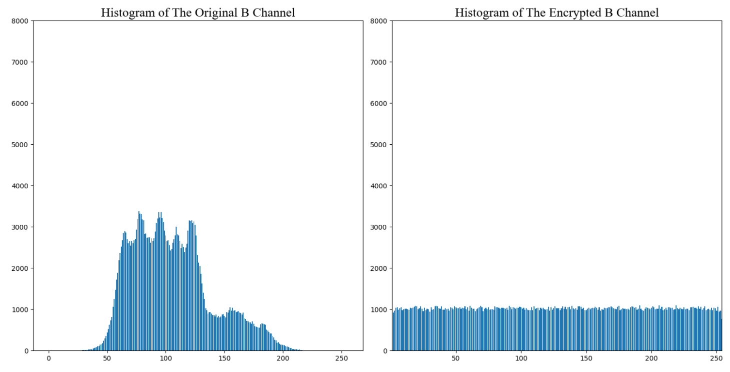
In this section, the performance of the Lena image encrypted with the proposed encryption technique is compared with that of the Lena image encrypted by other studies. Generally, recent and significant studies have been used for this comparison. The grayscale version of the Lena image was used, focusing on correlation coefficients, entropy, and NPCR values. The obtained test results are presented in Table 8. In the table, while the proposed study achieved very good results compared to some other studies, it obtained values very close to those of the studies with the best results. This demonstrates the effectiveness of the encryption technique.



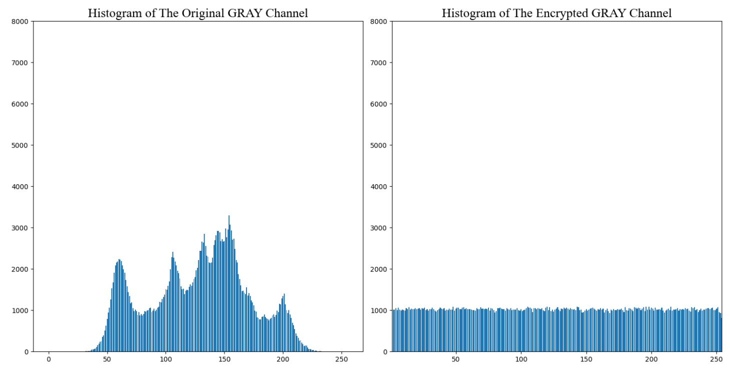
(a) Histogram of The Original and Encrypted R Channel



(b) Histogram of The Original and Encrypted G Channel

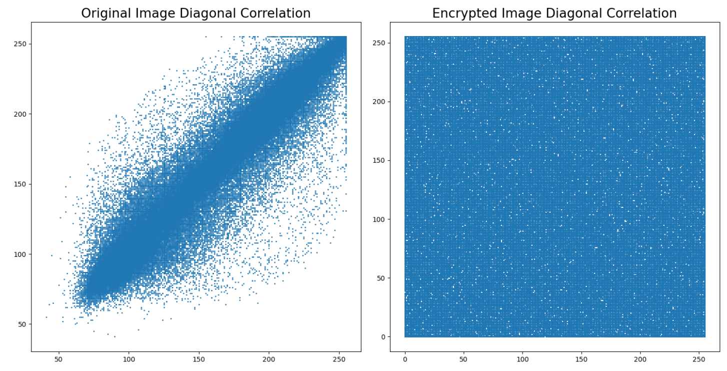


(c) Histogram of The Original and Encrypted B Channel

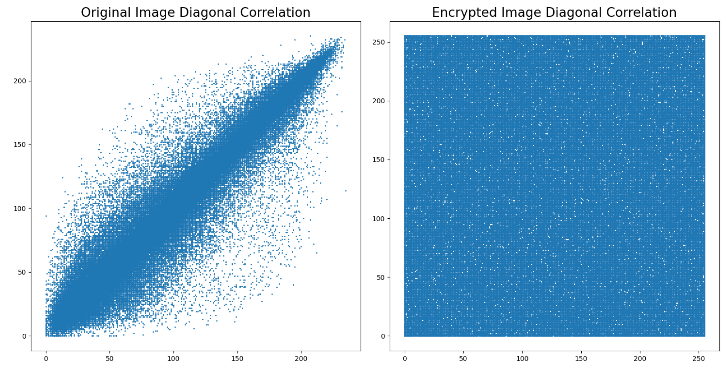


(d) Histogram of The Original and Encrypted Gray Channel

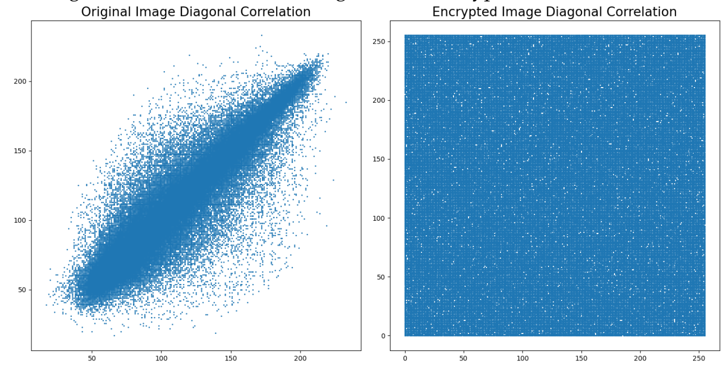
Figure 7 Comparative Histogram Analysis of Original and Encrypted Image Channels



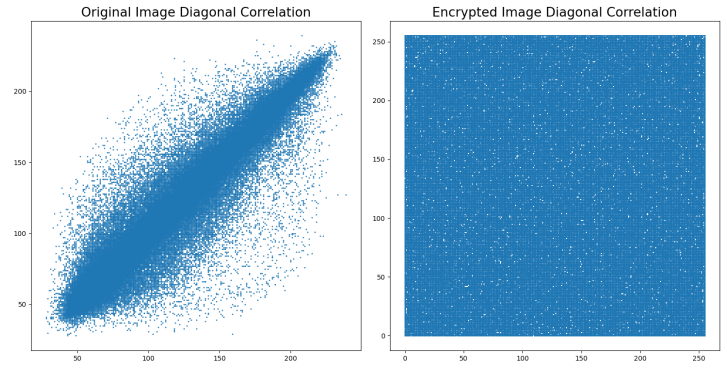
(a) Diagonal Correlation of The Original and Encrypted R Channel



(b) Diagonal Correlation of The Original and Encrypted G Channel



(c) Diagonal Correlation of The Original and Encrypted B Channel



(d) Diagonal Correlation of The Original and Encrypted Gray Channel

Figure 8 Comparative Diagonal Correlation Analysis of Original and Encrypted Image Channels

Table 4 Logistic Map Horizontal Correlation

Size	Channel	Horizontal Corr. (Original)	Horizontal Corr. (Encrypted)
128x128	Red	0.9539	0.0053
128x128	Green	0.9581	0.0074
128x128	Blue	0.9191	-0.0046
128x128	Gray-Scale	0.9442	0.0040
256x256	Red	0.9742	-0.0001
256x256	Green	0.9768	0.0072
256x256	Blue	0.9515	-0.0045
256x256	Gray-Scale	0.9684	0.0002
512x512	Red	0.9888	-0.0005
512x512	Green	0.9899	-0.0014
512x512	Blue	0.9783	-0.0001
512x512	Gray-Scale	0.9864	-0.0011

Table 5 Logistic Map Vertical Correlation

Size	Channel	Vertical Corr. (Original)	Vertical Corr. (Encrypted)
128x128	Red	0.8654	0.0005
128x128	Green	0.8463	0.0136
128x128	Blue	0.7802	-0.0145
128x128	Gray-Scale	0.8280	0.0064
256x256	Red	0.9255	-0.0043
256x256	Green	0.9144	0.0016
256x256	Blue	0.8736	0.0017
256x256	Gray-Scale	0.9035	0.0003
512x512	Red	0.9676	-0.0005
512x512	Green	0.9630	-0.0011
512x512	Blue	0.9448	-0.0010
512x512	Gray-Scale	0.9580	-0.0011

Table 6 Logistic Map NPCR, UACI, Entropy

Size	Channel	NPCR	UACI	Entropy (Original)	Entropy (Encrypted)
128x128	Red	0.9960	0.3246	7.2996	7.9875
128x128	Green	0.9959	0.3031	7.5558	7.9861
128x128	Blue	0.9961	0.2786	7.1104	7.9854
128x128	Gray-Scale	0.9967	0.2753	7.3255	7.9892
256x256	Red	0.9962	0.3296	7.2835	7.9961
256x256	Green	0.9958	0.3036	7.5801	7.9963
256x256	Blue	0.9962	0.2765	7.0603	7.9957
256x256	Gray-Scale	0.9958	0.2787	7.3176	7.9961
512x512	Red	0.9962	0.3309	7.2709	7.9981
512x512	Green	0.9961	0.3050	7.5860	7.9981
512x512	Blue	0.9958	0.2761	7.0022	7.9979
512x512	Gray-Scale	0.9960	0.2796	7.3015	7.9981

Table 7 FPS Test Results

Size	Devices	fps	Size	Devices	fps
128x128	1	3456	512x512	1	376
128x128	5	3280	512x512	5	360
128x128	10	2900	512x512	10	316
128x128	20	2404	512x512	20	284
128x128	50	1960	512x512	50	232
128x128	100	1284	512x512	100	168
256x256	1	1484	1024x1024	1	148
256x256	5	1424	1024x1024	5	144
256x256	10	1240	1024x1024	10	132
256x256	20	924	1024x1024	20	116
256x256	50	644	1024x1024	50	104
256x256	100	264	1024x1024	100	76

Table 8 Comparison of Studies

Studies	Correlation Coefficient				Entropy	NPCR (%)
	256		512		512	512
	Horizontal	Vertical	Horizontal	Vertical		
Shakir (2019)	N/A	N/A	0.1394	0.0339	7.7362	N/A
Kiran *et al.* (2023)	0.0022	0.0027	-0.0016	-0.0025	7.9981	99.606
You *et al.* (2020)	N/A	N/A	0.0149	0.0151	7.9457	99.65
Clemente-López *et al.* (2024)	N/A	N/A	0.0049	0.0038	7.9992	99.61
Our Study	0.0002	0.0003	-0.0011	-0.0011	7.9981	99.60

CONCLUSION

This study presents a novel chaos-based encryption technique leveraging the parallel processing capabilities of CUDA for mobile control centers with powerful GPUs. The proposed method utilizes the Logistic Map for random number generation, which is then applied to the RGB and grayscale channels of image data through XOR operations. The results of our experiments demonstrate that the technique provides fast encryption speeds and high security levels, making it suitable for real-time applications in IoT environments. The statistical analyses, including histogram, correlation, and entropy tests, confirm the robustness and effectiveness of the encryption mechanism against various types of attacks.

Future work will focus on several enhancements and extensions of the current study. One area of improvement involves optimizing the algorithm for different hardware configurations, including low-power devices, to ensure broader applicability. Additionally, integrating more sophisticated chaotic maps and exploring hybrid encryption techniques could further enhance security and performance. We also plan to conduct extensive testing in real-world IoT scenarios to validate the practicality and reliability of the proposed method. Furthermore, expanding the scope of the study to include other types of data, such as video and sensor data, will help in assessing the versatility and scalability of the encryption technique.

Availability of data and material

Not applicable.

Conflicts of interest

The author declares that there is no conflict of interest regarding the publication of this paper.

Ethical standard

The author has no relevant financial or non-financial interests to disclose.

LITERATURE CITED

Beirami, A., H. Nejati, and W. Ali, 2012 Zigzag map: a variability-aware discrete-time chaotic-map truly random number generator. Electronics Letters **48**: 1537–1538.

Bezerra, J. I. M., A. Molter, G. Machado, R. I. Soares, and V. V. d. A. Camargo, 2024 A novel single kernel parallel image encryption scheme based on a chaotic map. Journal of Real-Time Image Processing **21**: 1–13.

Bharadwaj, B., J. Saira Banu, M. Madiajagan, M. R. Ghalib, O. Castillo, *et al.*, 2021 Gpu-accelerated implementation of a genetically optimized image encryption algorithm. Soft Computing **25**: 14413–14428.

Boyraz, O. F., E. Guleryuz, A. Akgul, M. Z. Yildiz, H. E. Kiran, *et al.*, 2022 A novel security and authentication method for infrared medical image with discrete time chaotic systems. Optik **267**: 169717.

Clemente-Lopez, D., J. de Jesus Rangel-Magdaleno, and J. M. Muñoz-Pacheco, 2024 A lightweight chaos-based encryption scheme for iot healthcare systems. Internet of Things **25**: 101032.

Clemente-López, D., J. M. Munoz-Pacheco, and J. de Jesus Rangel-Magdaleno, 2024 Experimental validation of iot image encryption scheme based on a 5-d fractional hyperchaotic system and numba jit compiler. Internet of Things **25**: 101116.

Elrefaey, A., A. Sarhan, and N. M. El-Shennawy, 2021 Parallel approaches to improve the speed of chaotic-maps-based encryption using gpu. Journal of Real-Time Image Processing pp. 1–10.

Erkan, E., H. Oğraş, and Ş. Fidan, 2023 Application of a secure data transmission with an effective timing algorithm based on lora modulation and chaos. Microprocessors and Microsystems **99**: 104829.

Francisti, J., Z. Balogh, J. Reichel, M. Magdin, Š. Koprda, *et al.*, 2020 Application experiences using iot devices in education. Applied Sciences **10**: 7286.

Ghosh, G., f. Kavita, D. Anand, S. Verma, D. B. Rawat, *et al.*, 2021 Secure surveillance systems using partial-regeneration-based non-dominated optimization and 5d-chaotic map. Symmetry **13**: 1447.

Jadhav, S., U. Patel, A. Natu, B. Patil, and S. Palwe, 2023 Cryptography using gpgpu. In *Intelligent Communication Technologies and Virtual Mobile Networks*, pp. 299–313, Springer.

Kashyap, P. K., S. Kumar, A. Jaiswal, M. Prasad, and A. H. Gandomi, 2021 Towards precision agriculture: Iot-enabled intelligent irrigation systems using deep learning neural network. IEEE Sensors Journal **21**: 17479–17491.

Kiran, H. E., A. Akgul, O. Yildiz, and E. Deniz, 2023 Lightweight encryption mechanism with discrete-time chaotic maps for internet of robotic things. Integration **93**: 102047.

Koyuncu, İ., 2014 *Kriptolojik uygulamalar için FPGA tabanlı yeni kaotik osilatörlerin ve gerçek rasgele sayı üreteçlerinin tasarımı ve* *gerçeklenmesi*. Ph.D. thesis, Sakarya Universitesi (Turkey).

Kumari, P. and B. Mondal, 2023 An encryption scheme based on grain stream cipher and chaos for privacy protection of image data on iot network. Wireless Personal Communications **130**: 2261–2280.

Mai, X. H., B. Zhang, X. S. Luo, *et al.*, 2015 Controlling chaos in complex motor networks by environment. IEEE Transactions on Circuits and Systems II: Express Briefs **62**: 603–607.

Man, Z., J. Li, X. Di, X. Liu, J. Zhou, *et al.*, 2021 A novel image encryption algorithm based on least squares generative adversarial network random number generator. Multimedia Tools and Applications **80**: 27445–27469.

Naik, R. B. and U. Singh, 2024 A review on applications of chaotic maps in pseudo-random number generators and encryption. Annals of Data Science **11**: 25–50.

Rajendran, S. and M. Doraipandian, 2021 Chaos based secure medical image transmission model for iot-powered healthcare systems. In *IOP Conference Series: Materials Science and Engineering*, volume 1022, p. 012106, IOP Publishing.

Romeo, L., A. Petitti, R. Marani, and A. Milella, 2020 Internet of robotic things in smart domains: Applications and challenges. Sensors **20**: 3355.

Shakir, H. R., 2019 An image encryption method based on selective aes coding of wavelet transform and chaotic pixel shuffling. Multimedia Tools and Applications **78**: 26073–26087.

Singh, R., A. Gehlot, S. V. Akram, L. R. Gupta, M. K. Jena, *et al.*, 2021 Cloud manufacturing, internet of things-assisted manufacturing and 3d printing technology: reliable tools for sustainable construction. Sustainability **13**: 7327.

Song, W., C. Fu, M. Tie, C.-W. Sham, J. Liu, *et al.*, 2022 A fast parallel batch image encryption algorithm using intrinsic properties of chaos. Signal Processing: Image Communication **102**: 116628.

Tutueva, A. V., E. G. Nepomuceno, A. I. Karimov, V. S. Andreev, and D. N. Butusov, 2020 Adaptive chaotic maps and their application to pseudo-random numbers generation. Chaos, Solitons & Fractals **133**: 109615.

You, L., E. Yang, and G. Wang, 2020 A novel parallel image encryption algorithm based on hybrid chaotic maps with opencl implementation. Soft Computing **24**: 12413–12427.

How to cite this article: Kiran, H. E. A Novel Chaos-Based Encryption Technique with Parallel Processing Using CUDA for Mobile Powerful GPU Control Center. *Chaos and Fractals*, 1(1), 6-18, 2024.