MAM2046W - 2ND Research Paper Assignment

Caleb Bessit

October 2023

1 Introduction

The use of chaotic systems to aid in encryption has been an active topic of research for decades. However, finding systems "chaotic" enough to produce strongly encrypted data while not so chaotic as to make the decryption process impractically sensitive is a key issue in this field.

In their paper, Zhao, Meng, Zhang, and Yang propose a two-dimensional map that they claim meets these criteria and detail an image encryption algorithm they developed that makes use of this system. They outline the properties of the map they use and further steps they take to ensure the usability of the algorithm, as well as compare it to existing approaches.

The rest of this document is structured as follows: section 2 rederives the dynamical system used in the paper and provides a brief motivation on the choices made. Section 3 discusses and reproduces the methods used in the paper to analyze the suitability of the dynamical system for image encryption from various perspectives. Section 4 discusses the implementation of the algorithm at a high level. Section 5 shows the results of implementing the algorithm as well as one of the safety evaluation tests performed. Lastly, section 6 discusses the results and conclusion of the paper, leading on to section 7 which discusses my insights, and possible future work.

2 Rederivation of Dynamical System

The authors of the paper reason that by their very nature, chaotic systems are a natural choice for use in encryption. However, there is a delicate and elusive balance between systems that display chaotic behaviour for a broad enough parameter range, while keeping computational complexity to a minimum.

On this basis, they develop a two-dimensional system derived from a composition of two relatively simple one-dimensional maps.

They begin with the Logistic map,

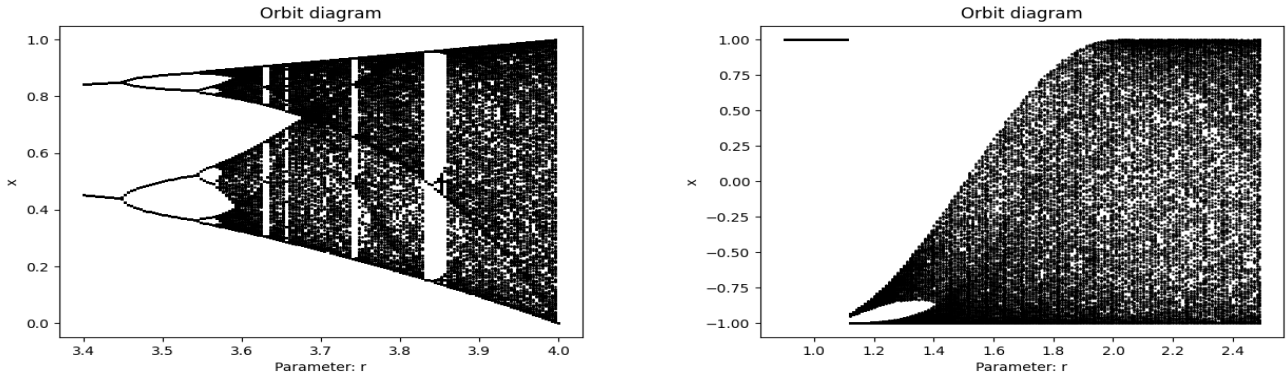
$$x_{n+1} = \mu x_n(1 - x_n) \tag{1}$$

where $\mu \in [0, 4]$ is a parameter. They note that the system maps the interval $[0, 1]$ to itself, and that the system displays chaotic behavior for $\mu \in (3.569945, 4]$. See figure 1a for a numerically generated orbit diagram confirming this.

They also consider the Chebyshev map, defined as

$$x_{n+1} = cos(\beta arccos(x_n)) \tag{2}$$

where $\beta \in [0, 4]$ is a parameter. They also note that this system maps the interval $[-1, 1]$ to itself. This system displays chaotic behavior for $\beta > 1$, confirmed in the figure 1b below.



(a) Logistic map, for $3.4 \leq \mu \leq 4$

(b) Chebyshev map, for $0.9 \leq \beta \leq 2.5$

Figure 1: Orbit diagrams for 1D basis maps

Notice that in both cases, the single dimensionality of the system limits the parameter range. Ideally, we would like to have an arbitrarily large parameter space, as this gives us more control and freedom in applications to real-world problems.

The key idea is to then compose these two maps. In particular, the output of the Logistic map is used as input (in the form of a dynamic parameter) to the Chebyshev map. A few algebraic tricks need to be performed to ensure this is a sound composition.

They note that the output of the Logistic map is in the range $[0, 1]$, and the Chebyshev map displays chaotic behavior for $\beta > 1$, so the output of the Logistic map can be used as an argument for an exponential function, which they define as

$$\beta(y_n) = e^{\mu y_n (1-y_n)}. \tag{3}$$

Then they define the two-dimensional system

$$\begin{cases} x_{n+1} = cos(\beta(y_n) arccos(x_n)) \\ y_{n+1} = cos(\beta(x_n) arccos(y_n)) \end{cases} \tag{4}$$

Notice that as in the Chebyshev map 2, the output of these are in the range $[-1, 1]$, but since at the next iteration it is input to the function β, we need to ensure the output is in the range $[0, 1]$. To this end, they introduce a gain and the floor function:

$$\begin{cases} x_{n+1} = cos(\beta(y_n) arccos(x_n)) \times 10^k - \lfloor cos(\beta(y_n) arccos(x_n)) \times 10^k \rfloor \\ y_{n+1} = cos(\beta(x_n) arccos(y_n)) \times 10^k - \lfloor cos(\beta(x_n) arccos(y_n)) \times 10^k \rfloor \end{cases} \tag{5}$$

where now we have two control parameters, μ (in the β function) and k, whose ranges are both $[0, \infty)$ The introduction of the gain (multiplication by 10^k) may seem arbitrary, but its benefit will be shown later. Note that the function $\lfloor x \rfloor$ here is the standard round-down operation. It is also easy to see now that the outputs are now in the range $[0, 1]$ (if a value is negative, the "floored" quantity is the same or larger in magnitude but positive. Similarly, if it is positive, the "floored" quantity is less than or equal to it in magnitude.)

This system 5 is their proposed map, which they call the *Two-Dimensional Logistic-Chebyhev Map*, or **2D-LCCM** for short.

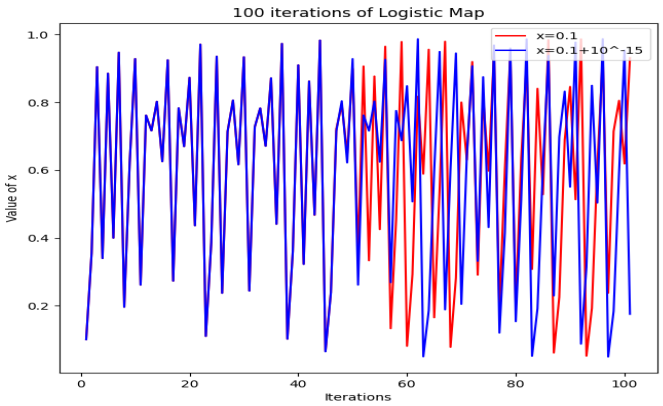
3 Analysis of dynamical system

In this section, there is a reproduction of 3 of the analytical procedures on the system that were done in the paper. Thereafter, there is a brief discussion on other methods of analysis performed (which are not reproduced.)

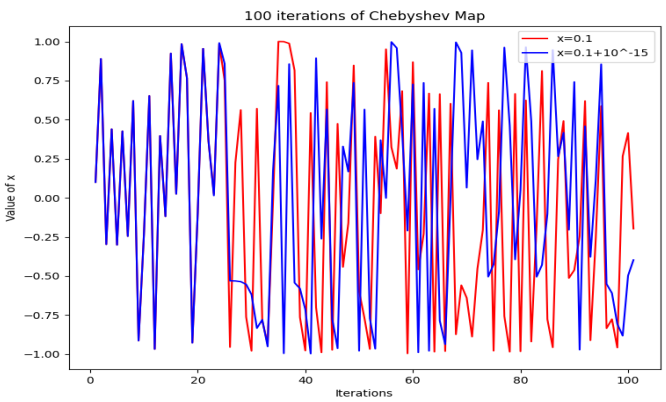
3.1 Sensitivity analysis

One method of analyzing the "chaoticness" of a dynamical system is by performing a sensitivity analysis. This involves selecting two initial conditions which are extremely close to each other, and monitoring the difference in value between

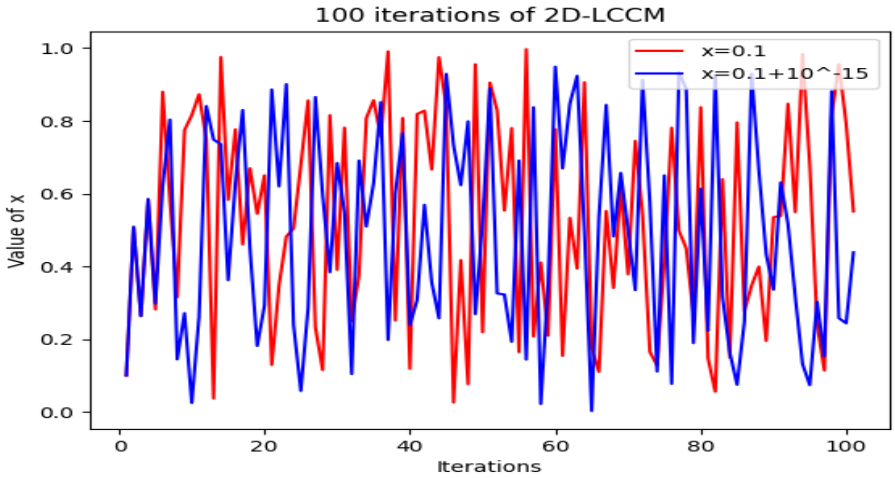
the values of the map at each iteration. Below are plots of 100 iterations of two initial conditions for each of the maps, where their parameters are in the required range for chaos.



(a) Logistic map



(b) Chebyshev map



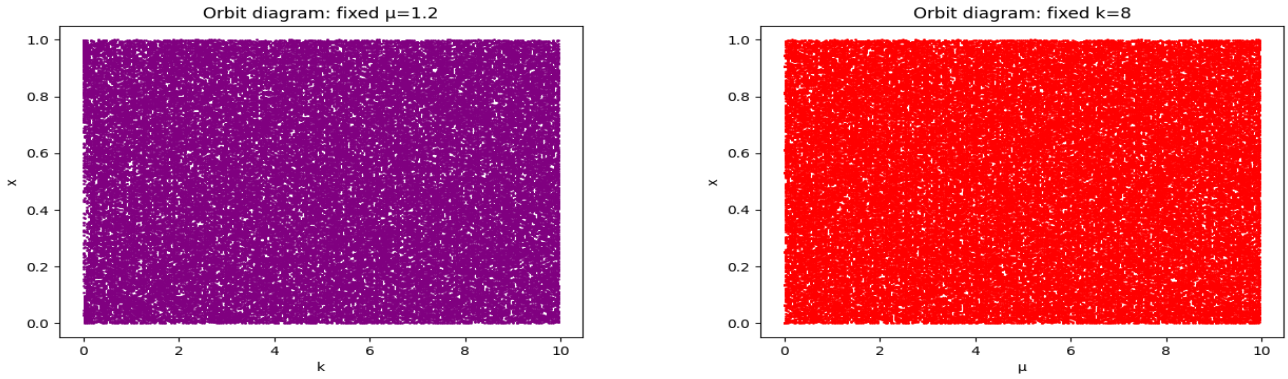
(c) 2D-LCCM

Figure 2: 100 iterations of each of the one-dimensional maps

Notice that for the two one-dimensional maps in figure 2a and 2b, the values at each iteration only show significant separation after about 30 iterations, whereas the 2D-LCCM shows separation between the sequences from as few as 5 iterations. This shows that the system proposed in the paper is quite sensitive to small changes in initial conditions, much more so than the two 1D maps individually.

3.2 Orbit diagrams and value distribution

Another method of analyzing the chaotic behavior of dynamical systems is to look at the orbit diagrams, as in figure 1a and 1b. Orbit diagrams are similar to bifurcation diagrams except they only display the attractors in the system as a function of some parameter, not any repellers or any other strange structures in phase space. Ideally, we would like many attractors for a large enough range of the parameter.



(a) Varying k, fixed $\mu = 1.2$

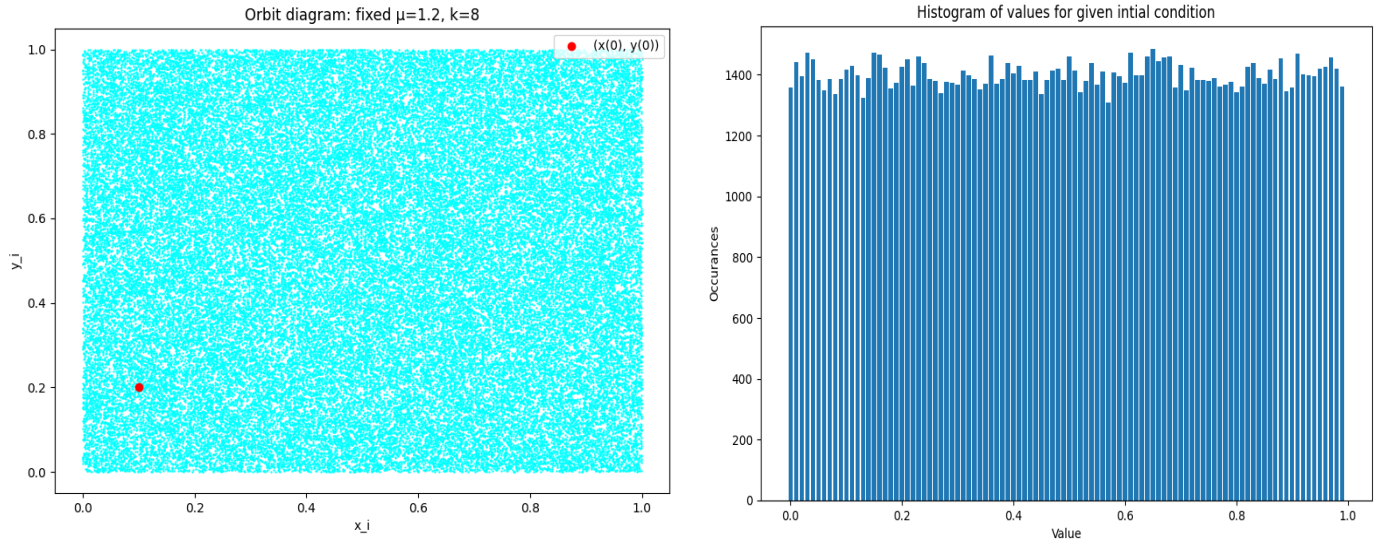
(b) Varying μ, fixed $k = 8$

Figure 3: Orbit diagrams 2D-LCCM for varying parameter values

Comparing the orbit diagrams of the 2D-LCCM to that of the 1D maps (figures 1a and 1b), the lack of periodic windows is evident. The authors use this as a point of motivation that the system is quite "chaotic", and thus suitable for use in encryption.

As an extension of this, the authors show that given a particular initial condition, the system produces a well-distributed range of values in the range $[0, 1]$; that is, the system is said to have good ergodicity.

In one case, the initial condition $(x_0, y_0) = (0.1, 0.2)$ was chosen, with $\mu = 1.2$, and $k = 8$. The values of 7×10^4 iterations were calculated and plotted in figure 4a below. The first 300 values were discarded to allow the initial transient to decay. Figure 4b shows the exact distribution of values in the range of the map.



(a) Iteration values on $[0, 1] \times [0, 1]$

(b) Histogram of values produced

Figure 4: Distribution of values produced by the 2D-LCCM

3.3 Lyapunov exponents

Lyapunov exponents are a formal method of quantifying the chaos of the system. The exponents indicate the rate at which trajectories that start infinitesimally near each other diverge.

There is a standard definition of Lyapunov exponents for a one-dimensional map, but since this system 5 is two-dimensional, we need to rederive an equivalent form. To do so, we first write it in the form

$$f(x, y) = \begin{Bmatrix} x_{n+1} = f(x_n, y_n) \\ y_{n+1} = g(x_n, y_n) \end{Bmatrix}.$$

Next, we compute the Jacobian matrix for the system at the n^{th} iteration

$$J = \begin{pmatrix} \frac{\partial f}{\partial x_n}|_{x_n,y_n} & \frac{\partial f}{\partial y_n}|_{x_n,y_n} \\ \frac{\partial g}{\partial x_n}|_{x_n,y_n} & \frac{\partial g}{\partial x_n}|_{x_n,y_n} \end{pmatrix}.$$

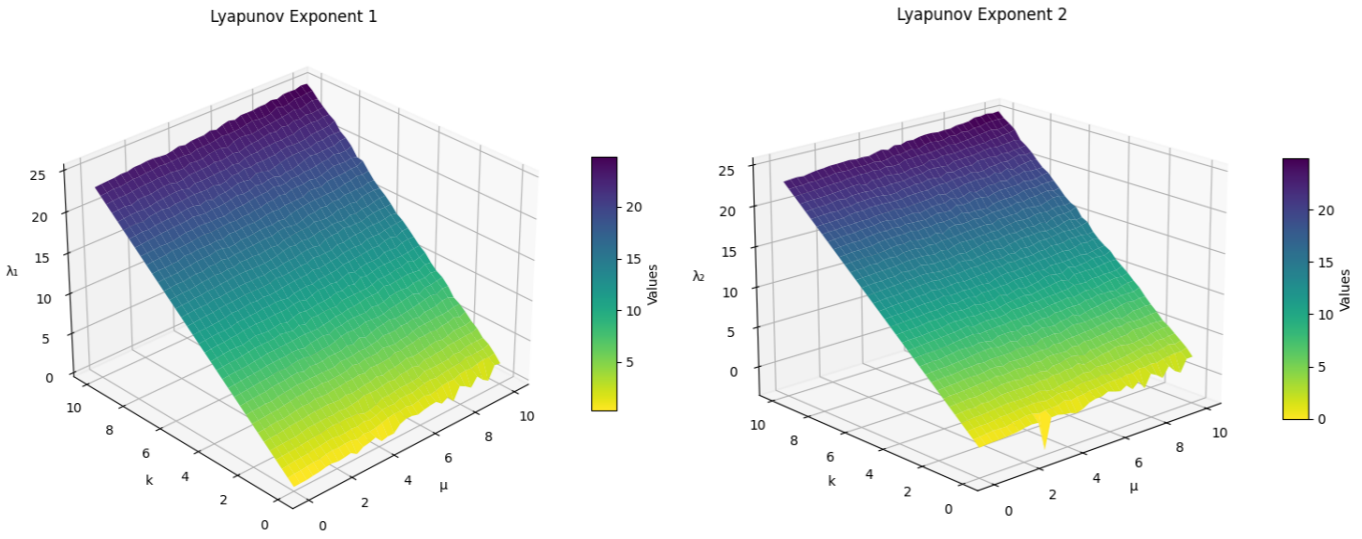
At the j^{th} iteration, the two eigenvalues of this matrix, λ_{1j} and λ_{2j} are calculated, and then the two Lyapunov exponents for the system are then calculated by

$$LE_1 = \lim_{n\to\infty} \left\{ \frac{1}{n} \sum_{j=1}^{n} \ln|\lambda_{1j}| \right\}$$

$$LE_2 = \lim_{n\to\infty} \left\{ \frac{1}{n} \sum_{j=1}^{n} \ln|\lambda_{2j}| \right\}.$$

It is known that if the largest Lyapunov exponent is positive, the system shows chaotic behavior. In particular, the larger the Lyapunov exponents, the "more chaotic" the system is, since this indicates a much faster divergence of nearby trajectories. (There are finer details to this, but

Below is a plot of the Lyapunov exponents of the system for varying values of μ and k.



(a) Lyapunov Exponent 1

(b) Lyapunov Exponent 2

Figure 5: Lyapunov Exponents for 2DLCCM

Figure 5 shows that the system has large, positive Lyapunov exponents for sufficiently large parameter values. It is unclear to me whether the Lyapunov exponents for this system can be made arbitrarily large simply by increasing the parameter values, but playing around with the values numerically indicates that this may be the case.

The authors of the paper also urge readers to notice that the Lyapunov exponents do not increase significantly for a fixed value of k and increasing values of μ. A more prominent increase in the Lyapunov exponents is seen for a fixed value of μ and increasing values of k.

This now serves to motivate the introduction of the gain (multiplication by 10^k in the system 5.) This allows us to have a powerful control parameter that can be used to amplify the chaoticness of the system with very little change in another parameter. Not introducing the gain would be equivalent to setting $k = 0$. As shown in figure 5, doing so would then require large values of μ to obtain Lyapunov exponents which may still be relatively small.

3.4 Discussion of other analysis methods

The authors employed two other techniques part of their analysis of of the system. The first of these was *sample entropy*. Briefly, the idea of sample entropy is similar to that of linear stability analysis. With stability analysis, we ask how a small perturbation in initial conditions evolves over time. This gives an idea about the stability of the system in some

context. Similarly, sample entropy works by considering subsequences of the chaotic sequence of a particular length, say 5. These are then compared to subsequences with one more data point (i.e., 6.) This increase in data points can be likened to the small perturbation in a dynamical system. The sample entropy method then uses these subsequences (characterized by vectors of dimension n and $n+1$, or dimensions 5 and 6 in our example) to calculate a value that measures the rate at which the similarity of patterns in the subsequences breaks down as the sequences become longer. Note that here, "similarity" is quantified by checking whether the values of the subsequences are within a particular tolerance of each other.

The authors showed that the values generated by the 2D-LCCM have a relatively high sample entropy, which indicates that the values of the sequence are quite irregular or unpredictable. This is desirable in the context of using these values for encryption.

Related to this is a standard NIST randomness test. This test evaluates the randomness of the sequence generated based on 15 standard tests that measure a variety of factors similar to sample entropy. In either of these cases, the authors show that the sequence meets the requirements to be described as having "good randomness".

4 Discussion of algorithm implementation

Figure 6 below is extracted from the paper by Zhao et al. [1] themselves, detailing the encryption process.

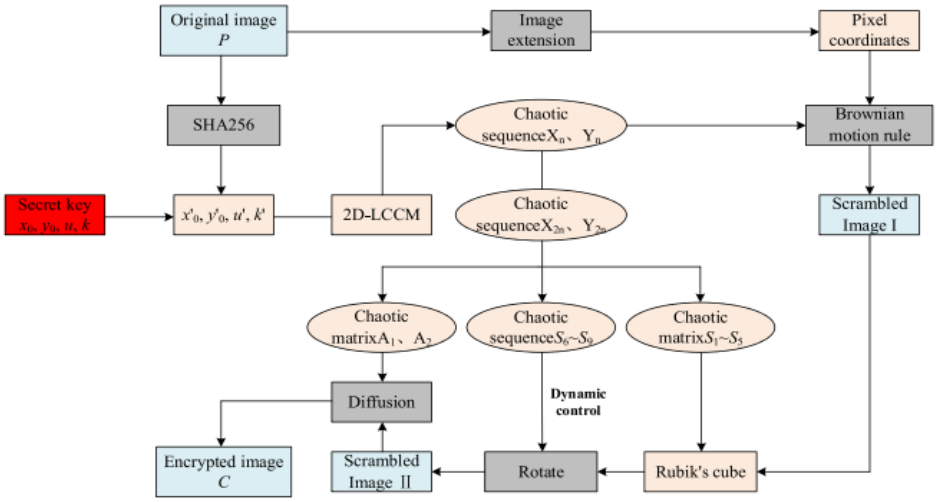


Figure 6: Algorithm implemented in the paper

The algorithm begins with the original image, P, and the secret key (the values used as the basis for the parameters.) The secret key is the initial values of the system, (x_0, y_0) and initial parameters (μ, k).The image P is used to generate a 256-bit hash value, which transforms the secret key values to get the transformed parameters (x'_0, y'_0, μ', k').

For an image P of height M pixels and width N pixels, they let $K = \max\{M, N\}$, which is used later.

The transformed parameters are used as input to the 2D-LCCM, for which $(K * K * n) + 1000$ terms are generated (the first 1000 terms are discarded to allow initial transients to decay.) Let that sequence of terms be $(x_i)_{i=1}^{n \cdot K^2}$. In the paper, the authors set $n = 20$.

The first main part of the encryption involves scrambling the image data. The justification for this is provided by looking at test cases later. For ease of calculation, the image is extended from a $M \times N$ pixel image to a square $K \times K$ pixel image by filling the remaining space with pure black or white pixels.

This image is then scrambled on the basis of a model of Brownian motion. The details are omitted, but a sample of the code is available in the notebook for this assignment here. This step justifies the generation of $K^2 \cdot n$ terms instead of simply $K \cdot K$ terms.

The second main part of the algorithm involves using two subsequences of $(x_i)_{i=1}^{n \cdot K^2}$ to scramble the image again using an algorithm that splices the image data into a virtual Rubik's cube. The rotation of the Rubik's cube is dynamically controlled by the terms of the subsequences.

Lastly, the scrambled image is diffused with matrices generated from the chaotic subsequences. This is done on the basis of another chaotic map, the Hénon map

$$\begin{cases} x_{n+1} = y_n + 1 - ax_n^2 \\ y_{n+1} = bx_n \end{cases}$$

with $a = 1.2,\ b = 0.3$. This produces the final encrypted image.

The decryption process essentially does this process in reverse. To do so, the initial key, image hash, and encrypted image are necessary.

5 Some results of implementing the algorithm

The algorithm was implemented in Python to work with greyscale `.ppm` files. Due to time constraints, part of the algorithm (involving the splicing into a virtual Rubik's cube) was not implemented. The diagram below shows the parts of the algorithm that have been implemented.

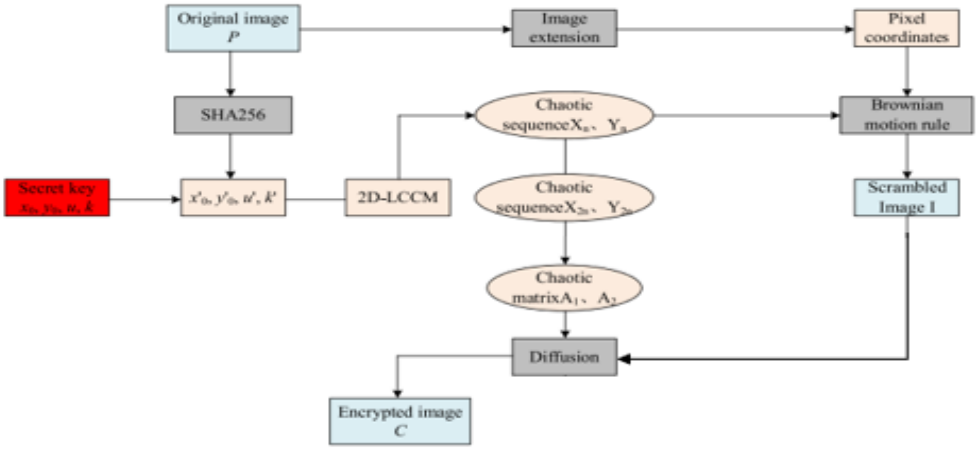
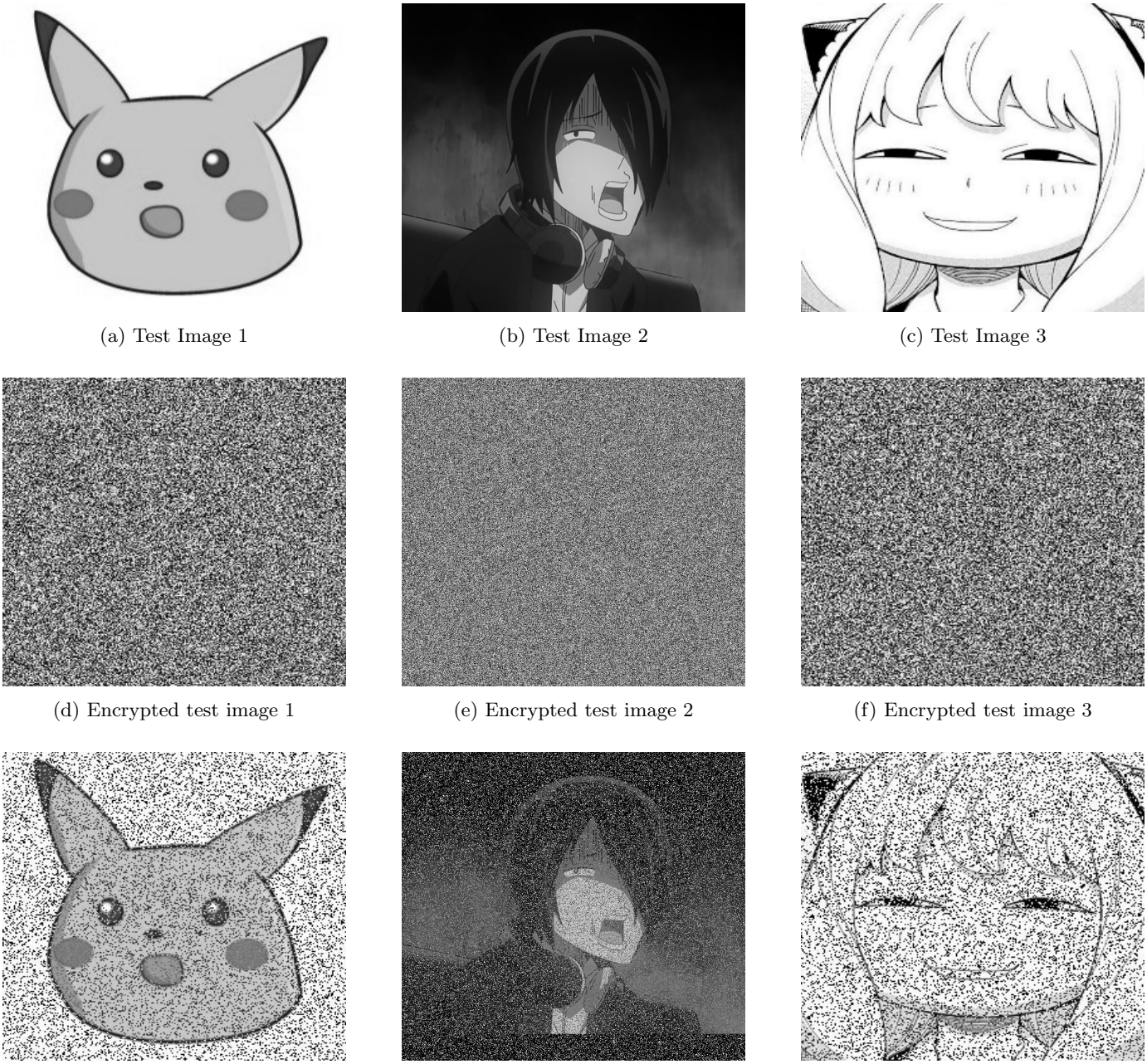


Figure 7: Parts of algorithm implemented

Below are some of the results and the parameter values used to generate them.



(a) Test Image 1 (b) Test Image 2 (c) Test Image 3

(d) Encrypted test image 1 (e) Encrypted test image 2 (f) Encrypted test image 3

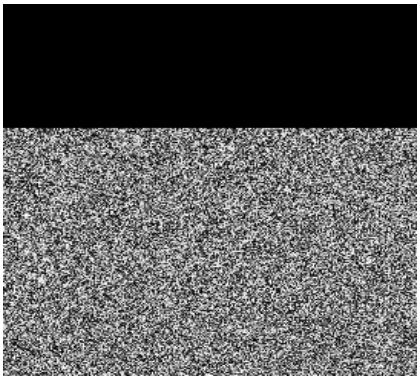
(g) Decrypted test image 1 (h) Decrypted test image 2 (i) Decrypted test image 3

Figure 8: Test Images, encrypted version (with parameters $\mu = 10$, and $k = 10$), and decrypted versions

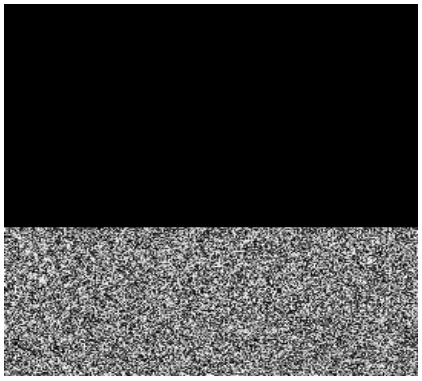
This implementation shows quite a lot of noise produced when the images are decrypted. Part of this is due to Python's implementation of bitwise operations: built-in support is only available for integer bitwise operations, so floating point values are truncated. There is a way to work around this, but it would involve modifying part of the algorithm (for which the potential benefits did not seem worth it, given time constraints.)

Given this, I was quite satisfied with the results, though there is a lot of room for improvement.

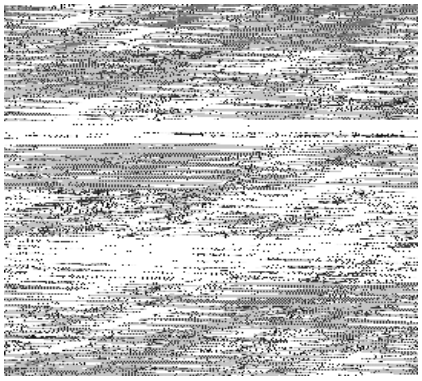
Below are tests that have been done by encrypting test image 1 and then chopping off 30% and 60% of the encrypted image in figures 9a and 9b respectively. Figure 9c shows the result of encrypting the image with parameters $\mu = 300$ and $k = 300$.



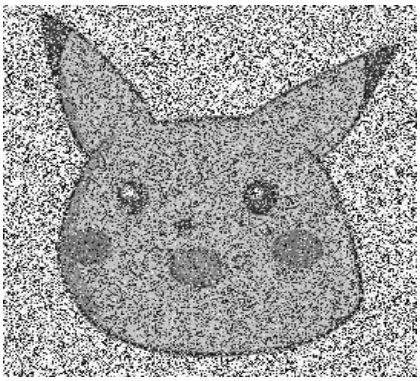
(a) Encrypted image 1, 30% clipped



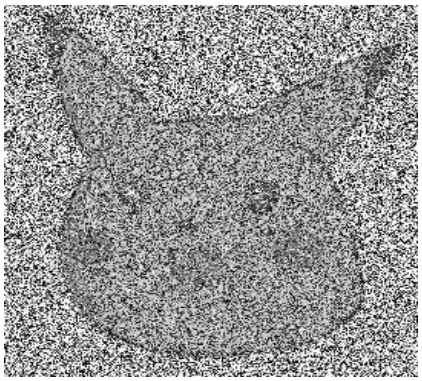
(b) Encrypted image 1, 60% clipped



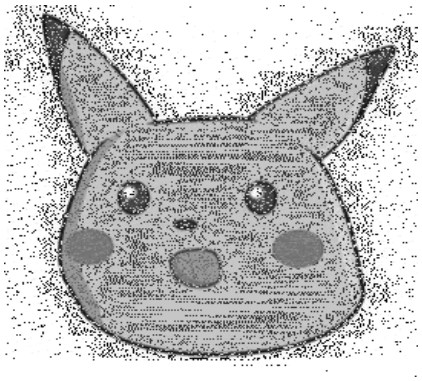
(c) Encrypted with $\mu = 300$, $k = 300$



(d) Decrypted image 1, 30% clipped



(e) Decrypted image 1, 60% clipped



(f) Decrypted with $\mu = 300$, $k = 300$

Figure 9: Encrypted and decrypted version of cut images; version with different parameters

Figures 9a and 9b show the robustness of the algorithm to data losses. This is mainly due to the scrambling process. The overall effect on the image due to chopping off part of it is the same as if it were not scrambled, but the scrambling distributes the loss of information, leading to a noisier, but still perceptible image.

Figure 9c shows that better decryption results can be obtained by using large parameter values (at least with this implementation.) It has not been included for the sake of space, but the encrypted version generated by using larger parameter values turns out to be more vulnerable to information loss. So for example, a loss of 30% of information in the encrypted image results in more defined differences in the encrypted image. I could not find a reasonable explanation of why this is the case, and this requires further investigation.

Another interesting test done by the authors is attempting to decrypt the image with a slight difference in parameter values (for example, encrypting the image with $x_0 = 0.1$ and decrypting it with $x_0 = 0.1 + 10^{-15}$.) This has also not been included, but the decrypted image is completely unrecognizable compared to the original image. This serves as an indication of the extremely sensitive dependence on the initial conditions of the map.

Please note that all the code for the diagrams and images produced in this assignment are available in the notebook here.

6 Discussion of result evaluation and conclusions

As with the dynamical system, the authors used a variety of standard tests to verify the safety of the algorithm to several attacks. In particular, they performed tests such as a sensitivity analysis, showing the sensitivity of the decryption on the keys used to decrypt it) and key space calculations (a standard test to ensure an algorithm is not vulnerable to brute force decryption.)

They also evaluated the efficacy of the algorithm by comparing the results of the algorithm to previous work by different authors. They made a point to evaluate the algorithms on the same devices with the same parameters to display a fair comparison.

The computational efficiency of the algorithm was briefly discussed, which they cited as the major shortcoming of this algorithm (compared to existing algorithms, at least.)

They conclude that given the high security of the algorithm, this is a fair compromise.

7 Insights and further research

Reading, thinking about, and reproducing the results from this paper was an insightful experience. I'd like to implement the algorithm in full and perform more of the safety tests involved.

The algorithm took about 40 seconds to a minute to encrypt or decrypt an image on my device (granted, the test images used in this write-up were larger than those used in the paper.) This is incredibly slow, so I'd like to look at ways of optimizing the current implementation. In particular, I'd like to explore the possibility of generating a large number of terms of a chaotic sequence in parallel. This seems immediately impractical given the very nature of chaotic maps, but I still think it is worth looking into.

Another question I had while working though the paper was the composition of chaotic maps. I'm interested in investigating the behaviour we could get from composing maps. Do we always get arbitrarily chaotic behaviour by composing chaotic maps, or could we dynamically control the parameter space where chaotic behaviour is exhibited through this composition? Are there even cases where the composed map displays less chaotic behavior than the two systems individually, and how could this be used in application?

References

[1] Yibo Zhao et al. "Image encryption algorithm based on a new chaotic system with Rubik's cube transform and Brownian motion model". In: *Optik* 273 (2023), p. 170342.