A Chaotic Image Encryption

Katherine Struss
Mathematics Senior Seminar, 4901
Advisor: Barry McQuarrie
Second Reader: Byungik Kahng
University of Minnesota, Morris
Spring 2009

Abstract

Throughout the years, there has been significant technological expansion, and with this the need for keeping information secure has also grown. The Internet has made it easy to send and receive pictures, but it has also made it relatively uncomplicated for others to find and view those images. This creates a problem with security, and therefore the issue of encoding pictures has increased in popularity. This presentation will focus on a greyscale image encryption. The techniques involved include the Arnold Cat Map and Chen’s chaotic system which are based on ideas from linear algebra and dynamical systems. The Arnold Cat Map will rearrange the order of the pixel values so that each one will be shuffled around, and the Chen’s chaotic map will change the greyscale values of the pixels. We will demonstrate an actual example of the encryption method using Mathematica.

1. Introduction

Technology is continuously expanding and as it is there are more and more techniques discovered that attempt to improve already existing works. An interesting field to expand upon is that of encryption. This paper is concerned with an applied mathematical example of using chaos to encrypt greyscale images.

The first part of the paper will be to describe dynamical systems; then we will move on to the concepts of chaos. That section will provide a brief background on chaos; including what it is, some history, and finally some examples that will clarify the definition. After that the specific parts of the encryption method will be analyzed and discussed. There are two major parts of the encryption, namely the Arnold Cat Map and Chen’s chaotic system. The Arnold Cat Map takes concepts from linear algebra and uses
them to change the positions of the pixel values of the original image. The result after applying the Arnold Cat Map will be a shuffled image that contains all of the same pixel values of the original image. Chen’s chaotic system will then take the image produced from the Arnold Cat Map and change the actual greyscale values of the pixels, the result will be the final encrypted image. Finally, we will come to an explanation of the encryption method and an example implementing it through *Mathematica.*

2. Dynamical Systems

**Definition 1:** A dynamical system is a smooth *action* of the reals or the integers on another object, [7].

Dynamical systems are time dependent; therefore they depict the position of a point depending on time. There are two types of dynamical systems, differential equations and iterated maps. The difference in the two is that differential equations are concerned with continuous time and iterated maps deal with time that is discrete, [7]. The following examples will provide further understanding about the definition and types of dynamical systems.

Example 1.1: Flow (Differential equations)
Let $X$ be a phase space. Let $\Phi : X \times \mathbb{R} \rightarrow X$ be the following smooth map satisfying the condition that:
$\Phi(\Phi(x,t_1),t_2) = \Phi(x,t_1 + t_2)$. The space $X$ with the flow map $\Phi$ is called a smooth (continuous) dynamical system.

Example 1.2: Iterated Dynamical System
Let $X \subseteq \mathbb{R}$ be a phase space, and define a map $\varphi : X \rightarrow X$. The space $X$ with the map $\varphi$ is called the iterated dynamical system.

In the following section a definition of chaos will be given, but it technically only applies to the latter of the two dynamical systems, i.e. iterated maps. However, it is possible to split up the continuous time of the smooth dynamical systems into discrete time intervals. For example if we look at a solution to a differential equation at discrete time intervals, i.e. $t = 0,1,2,3\ldots$ then we basically have an iterative process. If we do that then we are really looking at an iterated dynamical system, so in this way the definition will apply to both. Before we move on to chaos let’s take a look at the following theorem, which will give us a better idea about when dynamical systems can be chaotic. First of all a linear equation cannot be chaotic therefore we need nonlinear equations to create a chaotic system. Also, by the Poincaré–Bendixson theorem continuous dynamical systems on a plane cannot be chaotic [7].
Theorem 1: The Poincaré–Bendixson theorem states suppose that:
1.) $R$ is a closed, bounded subset of the plane
2.) $\frac{d}{dt} = f(x)$ is a continuously differentiable vector field on an open set containing $R$
3.) $R$ does not contain any fixed points
4.) There exists a trajectory $C$ that is confined in $R$, in the sense that it starts in $R$ and stays in $R$ for all future time.

Then either $C$ is a closed orbit, or it spirals toward a closed orbit as $t \to \infty$. In either case, $R$ contains a closed orbit.

So in conclusion, non-planar differential equations and both planar and non-planar discrete dynamical systems (iterated maps) have the possibility of being chaotic. Next we will look at what it means to be chaotic and the conditions needed for a system to exhibit the signs of chaos.

3. Chaos

Usually when the word chaos is heard it is associated to words like disorder, disarray, or randomness; however, scientifically there is much more to it. There are many definitions of chaos; the one we choose is from Robert Devaney ([1], [4]).

Definition 2: Let $(X, d)$ be a metric space. Then a map $f : X \to X$ is said to be (Devaney) chaotic on $X$ if it satisfies the following conditions:

1. $f$ exhibits sensitive dependence upon its initial conditions
2. $f$ is topologically transitive.

As mentioned previously, chaos is sometimes seen as meaning random or unstable, but it is important to make sure that the randomness also exhibits the conditions from the definition of chaos. The conditions are important to understand, so let’s take a deeper look at them.

Definition 3: Sensitive dependence on initial conditions means that there exists a certain $\delta > 0$ such that, for any $x \in X$ and $\epsilon > 0$, there exists some $y \in X$, where $d(x, y) < \epsilon$ and $n \in \mathbb{N}$ so that $d(f^n(x), f^n(y)) > \delta$ [4].

The dependence on initial conditions is very important in chaos; it is what makes it hard to determine long term behavior of dynamical systems which show signs of chaos [7]. If a chaotic output is generated by one set of initial conditions and then they are changed, even just a little bit, the output will change over time. This means that the trajectories that start out close together will after a long time venture apart.
**Definition 4:** To be topologically transitive means that for any pair of open sets $U, V \subset X$, there exists a certain $n \in \mathbb{N}$ such that $f^n(U) \cap V \neq \emptyset$, [4].

In other words, given our two open sets if we apply $f$ to one of them, say $U$, after awhile $U$ will overlap $V$.

3.1 History of Chaos

Henri Poincaré was the first to notice the possibility of chaos [7]. This occurred in the late 1800s, when he was considering qualitative questions compared to quantitative questions. Although Poincaré did notice the possibility, Lorenz was really the first one to discover chaotic motion, and he did this in 1963. He wanted to be able to predict weather by looking at the convection rolls in the atmosphere, [7]. Lorenz found that his equations continued to oscillate in an irregular pattern, and that if he changed his initial conditions the results would be completely different. However, he also noticed that there was some order to them when he plotted the equations in three dimensions. Lorenz’s equations for predicting the weather are the first example of a dynamical system that exhibit chaos. Although this happened in the early 1960s it wasn’t until the 1970s that the mathematical understanding of chaos came into full swing.

In 1975, James Yorke and T.Y. Li wrote the paper, “Period Three Implies Chaos”, in which they first coined the term chaos. Yorke was also the first person to attempt to mathematical define chaos; he did this in the 1970s. Some other contributors to the rise of popularity in chaos include Ruelle and Takens, May, and Feigenbaum. In particular, Ruelle and Takens looked into the connection between chaos and turbulence. They proposed a theory on the onset of turbulence in fluids using strange attractors, [7]. A few years later, May discovered some examples of iterated maps that demonstrated chaos in the topic of population biology. This then lead to an article by May that stressed the importance of educating pre-college students on nonlinear systems as well as linear systems; instead of only focusing on the latter [7]. Then in the late 1970’s, Feigenbaum discovered that there are universal laws that govern the transition from regular to chaotic behavior, [7]. According to Storgatz, “His (Feigenbaum’s) work established a link between chaos and phase transitions, and enticed a generation of physicists to the study of dynamics.” Along with these discoveries the invention of high-speed computers contributed to the increased study of chaos. The computer made it possible to do many things that people weren’t able to do, and in a much faster way; for example graphing fractals and taking a deeper look into nonlinear systems. These discoveries were contributors to the gigantic leaps that the mathematical study of chaos made in the 1970’s.

Even though the concept of chaos has been around for awhile, defining it is still somewhat of a challenge. This is because of the difficulty in coming up with one set definition that everyone agrees on. Today most agree that the definition put forth by Robert Devaney, (Definition 2), is the most acceptable [4]; however, there is no set universal definition for it.

There are chaotic examples in economics, fluid dynamics, optics, chemistry, changing weather, and even in the swirling patterns of cream being stirred into a cup of coffee [7]. One of the most unique examples pertains to a butterfly flapping its wings in one part of the world and that having an effect on if a storm arises one year later on the
other side of the world. This has led to the phenomena known as the butterfly effect. Uncertainties, no matter how small, would eventually overwhelm any calculations and defeat the accuracy of your prediction is known as the butterfly effect. The butterfly effect is usually mentioned with the topic of chaos, and it might be because it is such an interesting question to ponder.

So far all that has been mentioned for mathematical examples of chaos are natural events that happen to have chaotic properties. There are also examples where chaos can be used and manipulated to create something. One such example is that of encryption; which is the topic of this paper. Specifically, the encryption algorithm that uses the Arnold Cat Map and Chen’s chaotic system will be discussed. The mathematical properties of the Arnold Cat Map will be discussed next.

4. Arnold Cat Map

Mathematically the Arnold Cat Map, (ACM), is represented as the following:

**Definition 5:** The Arnold Cat Map is a discrete system that stretches and folds its trajectories in phase space.

Let \( X = \begin{bmatrix} x \\ y \end{bmatrix} \), where \( X \) is a vector, then the ACM transformation is,

\[
\Gamma: \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} 1 & p \\ q & pq + 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \mod n
\]

Some conditions for the map is that \( p \) and \( q \) are positive integers and

\[
\det \begin{bmatrix} 1 & p \\ q & pq + 1 \end{bmatrix} = 1,
\]

which makes the map area-preserving.

Vladimir Arnold discovered the ACM in the 1960s, and he used the image of a cat while working on it [5].

An interesting property of the ACM is the Poincaré Recurrence Theorem [2]:

**Theorem 2:** The Poincaré Recurrence Theorem states that certain systems will, after a sufficiently long time, return to a state very close to the initial state.

This means that after a certain number of iterations the ACM will return to its original state. For example if we are looking at an image and the ACM is applied to it, after a certain number of iterations the original image will be produced. This implies that the ACM is periodic and that has implications on it being a chaotic map. Some references state that the ACM is a chaotic map, which would imply that the discrete system that makes it up exhibits the dynamics of chaos. An example of this is Peterson who states
that the ACM “is a simple and elegant demonstration and illustration of some of the
principles of chaos – namely, underlying order to an apparently random evolution of a
system.” The ACM will be influenced by its initial conditions and it is also true that the
outputs will appear to be random. Although it does have some properties of chaos it isn’t
truly chaotic; this is due to the fact that it is periodic.

This section provided a mathematical background about the ACM, which is the
first part of the encryption method put forth in Section 5; the next task will be to describe
the mathematical properties of the second major part of the encryption, chaotic systems.

5. Chaotic Systems

This section will provide a deeper mathematical understanding of chaotic
systems, which is the second main part of the encryption method. A particularly common
dynamical system that demonstrates chaos is the Lorenz chaotic system. As mentioned
previously, in the early 1960s, Edward Lorenz started looking for equations that could
predict the weather [7]. After he discovered the equations, he noticed that when you
varied the initial conditions, $x_0$, $y_0$, and $z_0$, the weather patterns would change
dramatically. Out of this interest came the Lorenz’s equations, which make up the most
common chaotic system and they are:

$$\begin{align*}
\frac{dx}{dt} &= \sigma (y - x) \\
\frac{dy}{dt} &= rx - y - xz \\
\frac{dz}{dt} &= xy - bz
\end{align*}$$

Here $\sigma$, $r$, and $b$ are parameters, where $\sigma$ is the Prandtl number and $r$ is the Rayleigh
number, [7].

As well as being completely chaotic throughout the whole system some
dynamical systems are chaotic within a certain subset of phase space; one way that this
can happen is if a dynamical system is chaotic in a certain attractor.

**Definition 6:** An attractor is a closed set $M$ that satisfies:

1.) $M$ is an invariant set: any trajectory $x(t)$ that starts in $M$ stays in $M$ for all time.
2.) $M$ attracts an open set of initial conditions, so $M$ attracts all trajectories that
start sufficiently close to it.
3.) $M$ is minimal: there is no proper subset of $M$ that satisfies conditions 1 and 2.

**Definition 7:** A strange attractor is an attractor that exhibits sensitive dependence on
initial conditions.

An attractor is what the behavior of a system settles down to or is attracted to. Although
the chaotic dynamical system doesn’t ever set into a certain pattern or converge to a
certain point it does develop a pattern once it is plotted in 3d. Again, the Lorenz
equations provide the most common example of a strange attractor, the Lorenz attractor.
A particular case that Lorenz studied had the parameters $\sigma = 10$, $b = (8/3)$, and $r = 28$ creating the equations:

\[
\begin{align*}
\frac{dx}{dt} &= 10(y - x) \\
\frac{dy}{dt} &= 28x - y - xz \\
\frac{dz}{dt} &= xy - \frac{8}{3}z
\end{align*}
\]

All attractors have an underlying shape; the Lorenz attractor’s shape is referred to as a butterfly, and it can be seen in Figure 1.

As can be seen in Figure 1 there isn’t a set pattern that the system follows, it doesn’t loop from one side to the other and then back again. This is made pretty clear because the loop on the right has been traced over a lot more than the loop on the left.

The image encryption in Section 5 will use Chen’s chaotic system which was developed in 1999 by Professor G. Chen [10], and it is a three dimensional or third-order system. This dynamical system is made up of differential equations that are similar to Lorenz’s system; it differs mostly due to the $c$ in front of the $y$ in the second equation. Chen’s equations are shown below:

\[
\begin{align*}
\frac{dx}{dt} &= a(y - x) \\
\frac{dy}{dt} &= (c - a)x - xz + cy \\
\frac{dz}{dt} &= xy - bz
\end{align*}
\]
Chen’s attractor is the following with the parameters $a = 35$, $b = 3$, and $c = 28$:

![Image of Chen’s Attractor]

Figure 2: Chen’s Attractor with initial values, $x_0 = -10.058$, $y_0 = 0.368$, and $z_0 = 37.368$.

Taylor’s Method of order $n$ was used to solve the chaotic systems put forth in this section. This is a numerical technique that is used to solve differential equations with given initial values, and it is shown in Appendix A.

6. Image Encryption Algorithm

According to Guan et al. “With the rapid growth of multimedia production systems, electronic publishing and widespread dissemination of digital multimedia data over the internet, protection of digital information against illegal copying and distribution has become extremely important.” Chaotic encryptions are being used to protect this information; this is because of their dependence on initial conditions.

Before we move on to the proposed encryption, the way the image is stored must be discussed. An image $W$ is stored as a $n \times n$ matrix whose elements are the greyscale value of the pixels of $W$; the matrix represents the pixel values from right to left and then top to bottom. We will be dealing with grey scale images whose range is 0 to 256, so the elements of the matrix will be in this range as well. In the following step by step representation of the encryption we will refer to the matrix as the set of the image, so for example set $W$ is the $n \times n$ matrix whose elements are the pixel values of $W$.

The goal of the cryptosystem in this paper is to encrypt images by shuffling pixel values and then changing the grey scale values to create a ciphered image. The pixel values are rearranged using the ACM and then the greyscale values are changed using Chen’s chaotic system. Breaking down the encryption we have the following five steps:
Given original image $A$ and set $A$.

Step 1.) Take the original image $A$, and apply ACM to it, $M$ times. This creates a new picture, image $B$ and set $B$.

Step 2.) Iterate Chen’s chaotic system $N_0 = \frac{n^2}{3}$ times, using our chosen initial values, $x_0$, $y_0$, and $z_0$.

Step 3.) Each time that Chen’s is iterated we get three values $x_i$, $y_i$, and $z_i$.

Step 4.) Then use $x_i$, $y_i$, and $z_i$ by putting them together in set $C$ as follows:

\[ C_{x_i} = \text{IntegerPart}[(\text{Abs}(x_i) - \text{Floor}(\text{Abs}(x_i))) \times 10^{14}, 256] \]
\[ C_{y_i} = \text{IntegerPart}[(\text{Abs}(y_i) - \text{Floor}(\text{Abs}(y_i))) \times 10^{14}, 256] \]
\[ C_{z_i} = \text{IntegerPart}[(\text{Abs}(z_i) - \text{Floor}(\text{Abs}(z_i))) \times 10^{14}, 256] \]

Step 5.) Then the shuffled image from set $B$ will be encrypted as:

\[ L_{3(i-1)+1} = B_{3(i-1)+1} \oplus C_{x_i} \]
\[ L_{3(i-1)+2} = B_{3(i-1)+2} \oplus C_{y_i} \]
\[ L_{3(i-1)+3} = B_{3(i-1)+3} \oplus C_{z_i} \]

The set $L$ will be our encrypted pixel values, whose greyscale values have been changed; the image $L$ will be the final encrypted image.

The $\oplus$ is the bitwise “exclusive or” function. $\oplus$ takes two binary representations of equal length and then applies $\oplus$ to it. Say we have the number 43, its binary representation will be 101011 and that is because:

\[ [2^5 \times 1] + [2^4 \times 0] + [2^3 \times 1] + [2^2 \times 0] + [2^1 \times 1] + [2^0 \times 1] = 32 + 0 + 8 + 2 + 1 = 43 \]

The rules for $\oplus$ are the following:

| $0 \oplus 0$ | $0$ |
| $1 \oplus 1$ | $0$ |
| $1 \oplus 0$ | $1$ |

6.1 A Closer Look at the Encryption

The first step for the proposed encryption is applying the ACM to the original image, $A$. The first part of the map, $\begin{bmatrix} 1 & p \\ \frac{1}{q} & pq + 1 \end{bmatrix}$, shears the image so that the end result is a picture that looks similar to the original image but stretched. The next step is to evaluate the mod; this splits similar to the original image but stretched. The next step is to evaluate the mod; this splits similar to the original image but stretched.
The best way to show this is by the following figure:

This was done in Mathematica and the code for it is shown in Appendix B.

As stated before (in Section 3) according to the Poincare Recurrence Theorem, after a set number of iterations of the ACM the original image will return. The random relationship between the size of the image and how many iterations it takes to return to the original image is depicted in the following table:

<table>
<thead>
<tr>
<th>Dimension of $n \times n$ matrix (pixel values)</th>
<th>Number of iterations to return to original image</th>
</tr>
</thead>
<tbody>
<tr>
<td>300 $\times$ 300</td>
<td>300</td>
</tr>
<tr>
<td>183 $\times$ 183</td>
<td>60</td>
</tr>
<tr>
<td>124 $\times$ 124</td>
<td>15</td>
</tr>
<tr>
<td>100 $\times$ 100</td>
<td>150</td>
</tr>
<tr>
<td>150 $\times$ 150</td>
<td>300</td>
</tr>
<tr>
<td>257 $\times$ 257</td>
<td>258</td>
</tr>
<tr>
<td>157 $\times$ 157</td>
<td>157</td>
</tr>
<tr>
<td>147 $\times$ 147</td>
<td>56</td>
</tr>
</tbody>
</table>
The table shows that the relationship between the size of the image and how many iterations it takes to return to its original form appears to be random. According to a paper by Freeman J. Dyson and Harold Falk, the number of iterations will be less than $3n$, where $n$ is the dimension of the image, e.g. if you have a $50 \times 50$ image dimension then $n = 50$ [6]. This return to the original image makes it relatively easier for hackers to decipher the message through simple brute force, and therefore we apply Chen’s chaotic system to the image to make it harder to decode.

The first thing to note is that the parameters of Chen’s chaotic system were selected as $a = 35$, $b = 3$, and $c = 28$ which make the system chaotic. Another important part in step two is the $N_0$, which is the number of times we should iterate Chen’s chaotic system. The $N_0 = \frac{n^2}{3}$, where $n^2$ is the $n \times n$ dimension of the image, e.g. if we have a $100 \times 100$ matrix of pixel values the $n = 100$. The reasoning behind the $N_0$ is that Chen’s gives us three numbers each time, $x_i$, $y_i$, and $z_i$, and we have to compensate for that by dividing by three so that the result will be an image that preserves the original $n \times n$ dimension.

Step 4 gives three variations of the following,

$\text{Cx}_i = \text{IntegerPart}[\mod[(\text{Abs}(x_i) - \text{Floor}(\text{Abs}(x_i)))] \times 10^{14}, 256]$  

The reasoning for doing this is that the $x_i$, $y_i$, and $z_i$ (i.e. values from Chen’s chaotic system) need to be changed in order to use $\oplus$ with the set $B$. To break it down, $(\text{Abs}(x_i))$ returned the absolute value of $x_i$, so we get rid of negatives. The $\text{Floor}$ function rounded the element to the nearest integer less than or equal to it, for example $\text{Floor}[2.1345]$ would return 2. The next step was to multiply by $10^{14}$, in order to ensure the number that Chen’s gave us was large enough to correspond to the other part of the $\oplus$. The $\mod$ (modulo) made sure the result corresponded to 256, the highest possible greyscale value. The $\text{IntegerPart}$ command turned the Chen value into an integer, and this was done because the $\oplus$ function from Mathematica wouldn’t work if it wasn’t given integers. Basically, the point to this whole function was so that the values given by Chen’s chaotic system would work in step five.

6.2 Keys to the encryption

The keys are what will unlock the encryption, so if we send this to someone who already has the keys they will be able to decode the picture. If by some random reason the method of our encryption is known, the keys are what make it hard for others to know how to break the code. The ACM provides us with three keys to encode the picture, and they are the two parameters $p$ and $q$ and the number of iterations, $M$.

Chen’s chaotic system also gives us three keys and they are the initial values $x_0$, $y_0$, and $z_0$. The reasoning behind this is that the system is chaotic and therefore it is sensitive to its initial conditions and will change dramatically if they do. We can also manipulate the $\text{Cx}_i = \text{IntegerPart}[\mod[(\text{Abs}(x_i) - \text{Floor}(\text{Abs}(x_i)))] \times Q, 256]$ which changes the values we get from Chen’s chaotic system to get another key. This can be done by changing the $Q$ (in the step by step explanation of the encryption we had it equal $10^{14}$). The only limitation to this key is that it needs to be a large number, otherwise the $\oplus$ won’t work.
7. Image Encryption Example
With Keys: \( p=1, q=1, M=5, Q=10^{14}, x_0 = -10.058, y_0 = 0.368, \) and \( z_0 = 37.368 \)

Our original image \( A \) is in Figure 4. The pixel values that make up our set \( A \) are represented by the histogram. The y-axis of the histogram represents the number of pixels in the image, while the x-axis is the greyscale value of the pixel, so what is being plotted is the number of pixels that have a particular greyscale value in the image. For example, in Figure 4 there are around 200 pixels in the image that have a greyscale value of 150. The whole encryption was run in Mathematica and the code can be found in Appendix B.

Step 1:
The first step is to apply the ACM to the original image; for this example let \( p=1 \) and \( q=1 \) so the ACM will be \( \Gamma: \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \). This was applied to image \( A \) five times (i.e. \( M=5 \)), so that it became the image you see in Figure 5.

Notice that the histogram in Figure 4 (from set \( A \)) is the same as the histogram in Figure 5 (from set \( B \)) this is due to the fact that the pixel values themselves didn’t change. Therefore set \( A \) and set \( B \) consist of the same pixel values but their order is different, so set \( A \) does not equal set \( B \).
Steps 2-4:
After the ACM was applied to the image, Chen’s chaotic system is run using Taylor’s Method, our initial values for Chen’s chaotic system are $x_0 = -10.058$, $y_0 = .368$, and $z_0 = 37.368$. The dimension of the picture was $361 \times 361$, so $N_0 = \frac{(361)^2}{3}$.

Step 5:
Next the “exclusive or” function was ran to form the new set $L$, which produced the encrypted image in Figure 6.

![Figure 6: The final encrypted image](image)

Notice how different the histogram is from the other two, this is due to the fact that the greyscale values of our image have changed. Now that we have this successful encryption, let’s try to decrypt it.

7.1 Decryption
With Keys: $p=1$, $q=1$, $M=5$, $Q=10^{14}$, $x_0 = -10.058$, $y_0 = .368$, and $z_0 = 37.368$

Now let’s decrypt the image $L$, which we got from the previous example. The first thing we want to do is run Chen’s system with the keys provided, ensuring that we have the same initial conditions that were used in the encryption. We have been given the keys so we know that $x_0 = -10.058$, $y_0 = .368$, and $z_0 = 37.368$, and we also know the parameters will be $a=35$, $b=3$, and $c=28$. After we iterate Chen’s via Taylor’s method of order $n$, we will get the $x_i$, $y_i$, and $z_i$ values. They will then be put into the following equations:

$$T \bar{x}_i = \text{IntegerPart}[\text{mod}((\text{Abs}(\bar{x}_i) - \text{Floor}(\text{Abs}(\bar{x}_i))) \times 10^{14}, 256]$$

$$T \bar{y}_i = \text{IntegerPart}[\text{mod}((\text{Abs}(\bar{y}_i) - \text{Floor}(\text{Abs}(\bar{y}_i))) \times 10^{14}, 256]$$

$$T \bar{z}_i = \text{IntegerPart}[\text{mod}((\text{Abs}(\bar{z}_i) - \text{Floor}(\text{Abs}(\bar{z}_i))) \times 10^{14}, 256]$$
Next put those new values into the bitwise “exclusive or” function so we get:

\[ F_{3(i-1)+3} = L_{3(i-1)+3} \oplus T_{x_i} \]

\[ F_{3(i-1)+3} = L_{3(i-1)+3} \oplus T_{y_i} \]

\[ F_{3(i-1)+3} = L_{3(i-1)+3} \oplus T_{y_i} \]

That will create our set \( F \) which will produce the following picture and histogram:

Figure 7: Image F, which is our encrypted image L, after applying Chen’s chaotic system. Hopefully it looks familiar.

Remember that earlier it was stated that the ACM is periodic; one way to execute the next step would be to iterate it until we got the original picture back. It is also possible to go the other direction by using a modification in Mathematica (Appendix C).

Now, run image \( F \) through the ACM to get:

Figure 8: Our decrypted image.

It is the original image!
7.2 Unsuccessful Decryption

With Keys: \( p = 1, \ q = 1, \ M = 5, \ Q = 10^{14}, \ x_0 = -10.058, \ y_0 = 0.3680000000000005, \) and \( z_0 = 37.368 \)

Let’s take advantage of the fact that dynamical systems that exhibit chaos have a dependence on their initial conditions. Remember that the initial values that were used to encrypt the image were \( x_0 = -10.058, \ y_0 = 0.368, \) and \( z_0 = 37.368 \). Let’s change them so that we get \( x_0 = -10.058, \ y_0 = 0.3680000000000005, \) and \( z_0 = 37.368 \), notice that only one of them is changed. Just for comparison, Figure 10 is the plot of the difference between the values that are produced by Chen’s chaotic system with the original initial values (the ones we encrypted with) compared to our new values:

Figure 9: The difference created by changing the initial values of Chen’s chaotic system.

If we do everything the same as we did in the previous example with the only change being the initial values we get the image in Figure 10, compare it to Figure 5, which is the image that is needed to succeed in the decryption.

Figure 10: An unsuccessful decryption.
This second decryption has failed us miserably, and the only part that we changed was one initial value. This is a prime example of how important the initial values are to a chaotic system and why they are such good keys for the encryption.

8. Conclusion

Earlier encryptions based their methods solely on shuffling the positions of the pixels around or only changing the greyscale values of the pixels, but both of these were relatively easy to break, [10]. Shuffling the positions and changing the pixel values is a great improvement to the earlier methods and according to Guan et al. doing both is now preferred by many as a more secure form of encryption. That is why this paper focused on using both the ACM and Chen’s chaotic system to encrypt the image. Although this is an improved encryption method there are some areas that could be expanded upon.

Studying the properties of the encryption in a more systematic way would be a possibility for future research. This could include something like breaking down the steps and looking at all the different possibilities for them. Another interesting topic for the future would be the periodicity of the Arnold Cat Map. It was mentioned briefly that it is less than 3N, but determining if there is any way to narrow that any further would be interesting. Another possibility would be to change the ACM into something that is chaotic and see if that changes the dynamics of the encryption at all. These are all interesting ideas for the future, but let’s get back to the main focus of this paper.

This paper provided an image encryption method that used the Arnold Cat Map and Chen’s chaotic system. It took advantage of the properties of both to manipulate an image’s pixel values. In particular it changed the position and the actual greyscale value of them. There was also an example given of an encryption, successful decryption, and unsuccessful decryption. They showed the importance of keys and exploited the fact that chaos has a dependence on its initial conditions. This is a very interesting topic but there are plenty of other possibilities for expansion.
**Appendix A: Taylor’s Method**

Taylor’s Method of order \( n \) is based on Taylor’s series:

Let \( f(x) \) be the function and \( x = a \) be the point then mathematically the Taylor series is represented as:

\[
f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \ldots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \ldots
\]

If we expand \( y(t_{i+1}) \) in a Taylor Series about \( t_i \):

\[
y(t_{i+1}) \sim \sum_{k=0}^{n} \frac{y^k(t_i)}{k!} (t_{i+1} - t_i)^k
\]

\[
\sim \sum_{k=0}^{n} \frac{y^k(t_i)}{k!} h^k, \text{ where } h = t_{i+1} - t_i
\]

Chen’s chaotic system is of order 3, thus Taylor’s Method of order 3 was used to solve it. This technique was implemented in Mathematica to solve Chen’s chaotic system; the code for it can be found in Appendix B.

**Appendix B: Mathematica Code for Encryption Algorithm**

**Arnold Cat Map:**

```mathematica
Needs["Histograms"]
A=Import["E:\giraffes_2.jpg"]; pixelvalues=A[[1,1]]; Dimensions[pixelvalues] lg=Length[pixelvalues] picture[0]=ArrayPlot[pixelvalues, DataReversed -> True]; hist[0]=Histogram[Flatten[pixelvalues]] Do[
  Do[
    newPIC[[Mod[x+y,lg,1],Mod[x+2*y,lg,1]]]=PIC[[x,y]]
    ,{x,lg},{y,lg}]; picture[k]=ArrayPlot[newPIC,DataReversed -> True]; hist[k]=Histogram[Flatten[newPIC]]; PIC=newPIC;
  ,{k,1,10}]
Manipulate[GraphicsGrid[{{picture[k],hist[k]}}],{k,0,29,1}]
```

*Image and set B:*
Chen’s Chaotic System via Taylor’s Method of order 3:
Clear[f1,f2,f3,t,x,y,z,h]
a=35.0;
b=3.0;
c=28.0;

f1[t_,x_,y_,z_]=a (y-x)
f2[t_,x_,y_,z_]=(c-a) x-x z+c y
f3[t_,x_,y_,z_]=x y-b z

h=0.001;

\[ t(0)=0; \]
\[ x(0)=-10.058; \]
\[ y(0)=0.368; \]
\[ z(0)=37.368 \]

\[ t[i_]:=t[i]=t[i-1]+h \]
\[ x[i_]:=x[i]=x[i-1]+h f1[t[i-1],x[i-1],y[i-1],z[i-1]]+1/2 h^2 (f1[1,0,0,0][t[i-1],x[i-1],y[i-1],z[i-1]]+f1[0,1,0,0][t[i-1],x[i-1],y[i-1],z[i-1]] f1[t[i-1],x[i-1],y[i-1],z[i-1]]+f1[0,0,1,0][t[i-1],x[i-1],y[i-1],z[i-1]] f1[t[i-1],x[i-1],y[i-1],z[i-1]]+f1[0,0,0,1][t[i-1],x[i-1],y[i-1],z[i-1]] f1[t[i-1],x[i-1],y[i-1],z[i-1]]) \]

\[ y[i_]:=y[i]=y[i-1]+h f2[t[i-1],x[i-1],y[i-1],z[i-1]]+1/2 h^2 (f2[1,0,0,0][t[i-1],x[i-1],y[i-1],z[i-1]]+f2[0,1,0,0][t[i-1],x[i-1],y[i-1],z[i-1]] f2[t[i-1],x[i-1],y[i-1],z[i-1]]+f2[0,0,1,0][t[i-1],x[i-1],y[i-1],z[i-1]] f2[t[i-1],x[i-1],y[i-1],z[i-1]]+f2[0,0,0,1][t[i-1],x[i-1],y[i-1],z[i-1]] f2[t[i-1],x[i-1],y[i-1],z[i-1]]) \]

\[ z[i_]:=z[i]=z[i-1]+h f3[t[i-1],x[i-1],y[i-1],z[i-1]]+1/2 h^2 (f3[1,0,0,0][t[i-1],x[i-1],y[i-1],z[i-1]]+f3[0,1,0,0][t[i-1],x[i-1],y[i-1],z[i-1]] f3[t[i-1],x[i-1],y[i-1],z[i-1]]+f3[0,0,1,0][t[i-1],x[i-1],y[i-1],z[i-1]] f3[t[i-1],x[i-1],y[i-1],z[i-1]]+f3[0,0,0,1][t[i-1],x[i-1],y[i-1],z[i-1]] f3[t[i-1],x[i-1],y[i-1],z[i-1]]) \]

taylor2=Table[\{x[k],y[k],z[k]\},\{k,0,length^2/3\}];
taylor2plot=ListPointPlot3D[taylor2,AxesLabel \{"x","y","z"\},PlotRange \{All,All\}]

BitXor to encrypt:
B=Flatten[B];
L=ConstantArray[0,length*length];
Do[
L[[3*(i-1)+1]]=Mod[BitXor[B[[3*(i-1)+1]],IntegerPart[Mod[(Abs[x[i]-Floor[Abs[x[i]]],256]*10^14]],[256]];L[[3*(i-1)+2]]=Mod[BitXor[B[[3*(i-1)+2]],IntegerPart[Mod[(Abs[y[i]-Floor[Abs[y[i]]],256]*10^14]],[256]];L[[3*(i-1)+3]]=Mod[BitXor[B[[3*(i-1)+3]],IntegerPart[Mod[(Abs[z[i]-Floor[Abs[z[i]]],256]*10^14]],[256]];,
{1,1,length^2/3-1}
]

L=Partition[L,length];
Appendix C: Mathematica code for Running ACM backwards

```
GraphicsGrid[{{ArrayPlot[L,Frame -> True,FrameTicks -> True,ImageSize -> {300,300},DataReversed -> True,ColorFunction -> GrayLevel],Histogram[Flatten[L]]},Spacings -> Scaled[.8],Frame -> All,FrameStyle -> Directive[Thickness[.001],Black]]

Appendix C: Mathematica code for Running ACM backwards

PIC = newPIC = 256 - F;
picture[0] = ArrayPlot[PIC, DataReversed -> True];
hist[0] = Histogram[Flatten[PIC]];
Do[
  Do[
    newPIC[[Mod[2 x - y, lg, 1], Mod[-x + y, lg, 1]]] = PIC[[x, y]]
    , {x, lg}, {y, lg}];
picture[k] = ArrayPlot[newPIC, DataReversed -> True];
hist[k] = Histogram[Flatten[newPIC]];
PIC = newPIC; , {k, 1, 10}]
Manipulate[GraphicsGrid[{{picture[k], hist[k]}}, {k, 0, 29, 1}]
```

References