Image Cryptography with Chaos: A Survey of Existing Methods and a Proposed Implementation Incorporating Fluid Dynamics Approaches

By

Gary L. Hammock

A thesis submitted in partial fulfillment

of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

Middle Tennessee State University

December 2016

ACKNOWLEDGMENTS

To my patient wife, Tabitha: Thank you for your understanding and your endless support. To my wonderful daughter, Kora, and my forthcoming son: I hope you always endeavor to learn something new. I love you all.

I would like to thank the members of my peer review team for your proofreading and valued inputs.

Additionaly, I would like to extend my thanks to the faculty and staff of MTSU for your support and understanding for all these years.

ABSTRACT

Cryptography has uses in everyday applications ranging from e-commerce transactions to secure communications. There is current research in encrypting images in their native two-dimensional form. To do this, deterministic chaos maps have been explored for their use in providing the operations required to transform a plaintext image into a ciphertext encrypted image and *vice versa*. This research implements existing bio-inspired and cellular automata image encryption techniques and shows that the bio-inspired approach is better than the cellular automata approach. A weakness in the cellular automata approach is also highlighted that was previously undiscovered. This research also explores a novel application of analogies from the field of Computational Fluid Dynamics to generate deterministic chaos. A cryptanalysis GUI was developed to quantitatively show that the proposed technique is superior to both the bio-inspired and cellular automata techniques using metrics including luminance histograms, pixel covariant dependence, chi-squared tests, and information entropy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I.
INTRODUCTION

Cryptography has many important uses in everyday applications. Such everyday usage includes e-commerce transaction security, military and diplomatic communications, and computer filesystem encryption. Traditional digital encryption falls into two categories:

1. Stream ciphers in which each bit/byte is processed sequentially

2. Block ciphers where encryption operations are processed on “blocks” of *n* bytes

In each of these two types, the input is processed as a one-dimensional vector of bytes.

There is current interest in exploring chaos-based cryptographic algorithms for secure image encryption[12]. One potential application of this venue of research is for Full Disk Encryption (FDE) of holographic storage. These image encryption techniques may be envisioned as two-dimensional (2D) symmetric key cryptosystems. In these systems, a plaintext image (this could represent a true image that is decoded visually, such as a photograph, or a matrix of binary data that is parsed algorithmically, such as a 2D barcode) is transformed into a ciphertext image where the hidden data should be otherwise unrecoverable without knowledge of the secret key.

Research is ongoing to explore how to produce more robust image encryption techniques without compromising the system to forms of cryptanalysis attacks. To this end, the common approach is to use chaos-based approaches to key generation.

For this introduction, a distinction must be made between the statistical definition of chaos and the type of amorphous chaos that is commonly a result of natural phenomena. The mathematical definition of chaos is deterministic oscillations about a mean[4, 11]. This *ergodicity* has been exploited in everything from Edward Lorenz’s weather predictions[13] to Benoit Mandelbrot’s fractal studies[15] to modern Computational Fluid Dynamics (CFD) turbulence models[10].

The first proposal for transmitting signals using chaos was published in a research paper by Pecora and Carroll in 1990[18]. In their proposal, a sender and receiver can have chaotic circuits that are synchronized and a message is passed in one of the masked chaotic signals. Research is ongoing to find new ways to exploit deterministic chaos for the purposes of encrypting messages. It is the aim of this effort to examine existing techniques for generating the chaos maps for image encryption and to present cryptanalysis metrics of these techniques.

The two existing image encryption algorithms to be implemented are:

1. A bio-inspired technique that mimics the evolutionary mechanisms of crossover and mutation as the chaos generator[2].

2. A Cellular Automata (CA) technique where the CA ruleset produces a chaos map[9].

A final goal is the development of a new chaos generator derived from fluid mechanics and heat transfer where an input image (or matrix) is subject to the *continuity equations* of CFD as the chaos inducing mechanism. The three techniques will be compared using a Graphical User Interface (GUI) incorporating digital image analysis metrics including luminance histograms, Fourier spectra, chi-squared tests, pixel correlation/covariance, and information entropy.

CHAPTER II.
BACKGROUND

A Primer on Cryptosystems

The most commonly envisioned cryptographic systems (as a portmanteau these are called *cryptosystems*) have employed symmetric key encryption. For such a cryptosystem $(\mathbf{C})$ there are three components: the *plaintext* message (m), the *ciphertext* message (c), and the *secret key* (k) that is used to perform the manipulations required to transform the plaintext to the ciphertext and *vice versa*.

$$\forall m, \exists k \text{ where:}$$

$$\text{Encryption:} \quad c = f(m,k) \tag{2.1}$$

$$\text{Decryption:} \quad m = f^{-1}(c,k) = f^{-1}(f(m,k),k) \tag{2.2}$$

That is to say a message, m, produces an encrypted ciphertext, c, when the encryption function $f(m,k)$ is used with secret key, k. Likewise, a decryption function, $f^{-1}(c,k)$, exists that performs the inverse operation returning the plaintext message, m, when given the same secret key, k.

Metrics for Cryptosystems and Cryptanalysis

To help describe key points in this section, a 512×512 pixel image shown in Figure 1 will be used. This image was created to have properties that highlight specific metrics or unique features for discussion.

An important metric for cryptosystems is the measure of Shannon entropy which couples a metric called *confusion* (the statistical relation between the ciphertext and the key) and a metric called *diffusion*. Diffusion is the statistical "dissipation" of a plaintext entity over many ciphertext entities—which means that an attacker must intercept a tremendous amount of material to infer the plaintext message structure[19]. The Shannon (also called information

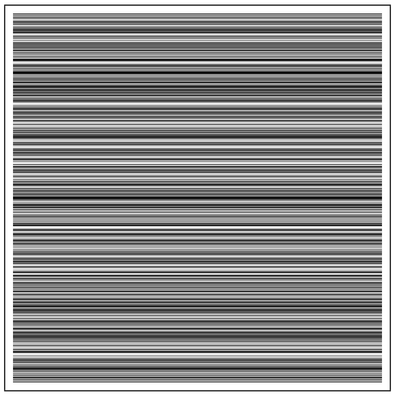


Figure 1: The Test Image for Describing the Cryptanalysis Metrics

theory) entropy for a discrete random variable $X \in \{x_1, x_2, \ldots, x_n\}$ is defined[19] in Equation 2.3 as:

$$\text{Shannon Entropy:} \quad h(X) = E\left[I\left(X\right)\right] \tag{2.3}$$

$$h(X) = \sum_{i=1}^{n} P(x_i) I(x_i) \tag{2.4}$$

$$h(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i) \tag{2.5}$$

Where $E(X)$ is the expected value function, $P(X)$ is the probability distribution, and $I(X)$ is the information content of X. The form of Equation 2.5 is the typically used form for bitwise entropy metrics and is reported in units: *bits of entropy*. The calculated Shannon entropy for the test image shown in Figure 1 is $h = 8$ bits because the image has a uniform distribution over the full luminous domain 0 to 255.

Another important cryptanalysis tool for observing cryptosystems is the frequency analysis histogram. This plot graphically shows the probability distribution of pixel intensity values. Ideally, the pixel intensity of the ciphertext is uniformly distributed. An image histogram shows the counts of pixels for each luminous value in the set [0, 255]. While histograms cannot uniquely define an image, they can be used as a type of signature or *crib* because each image is directly linked to its histogram. The histogram of the test image is shown in Figure 2. In this figure, it is apparent that there are 1024 pixels at each intensity value in the set $\{p_{x,y} | 0 \leq p_{x,y} \leq 255\}$. This is a uniform probability distribution.

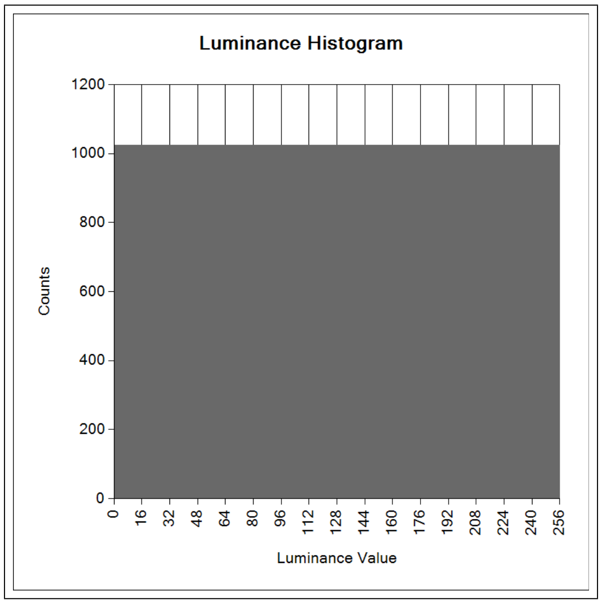


Figure 2: The Histogram of the Test Image Showing the Uniform Probability Distribution

As in text-based cryptosystems where there is a probability distribution for digraphs (such as the digraphs “th”, “er”, and “nd”) there also exists an image analogue. For images, it is necessary to calculate the horizontal and vertical pixel correlations. These correlations can show that a direct correlation exists between a randomly selected pixel and its adjacent pixel. In photographs, there tends to be a linear correlation in these pixel data (i.e. there does not tend to be severe discontinuities in luminance in adjacent pixels). This is analogous to having the prescient knowledge that the digraph “th” is the most common digraph in the English language, randomly selecting the English letter “t” out of a body of text, and predicting that the letter “h” will follow it as in the English word “the”. However, the following letter could—with nearly $\frac{1}{3}$ of the probability[14] of “h”—be the letter “o” as in “toe”.

The ideal scenario for encrypted images is to diffuse the luminance value of the plaintext image into adjacent pixels such that no discernible patterns can be retrieved from the ciphertext. The metric of correlation is calculated by Pearson's correlation coefficient, $R_{x|y}$, using Equations 2.6–2.9 where x and y are the luminance intensity values of adjacent pixels. From this result, the coefficient of determination, R^2 is also calculated.

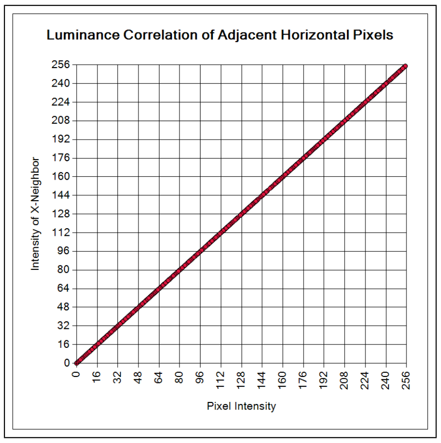
$$\text{Expected Value:} \quad E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.6}$$

$$\text{Variance of x:} \quad D(x) = \frac{1}{N}\sum_{i=1}^{N} [x_i - E(x)]^2 \tag{2.7}$$

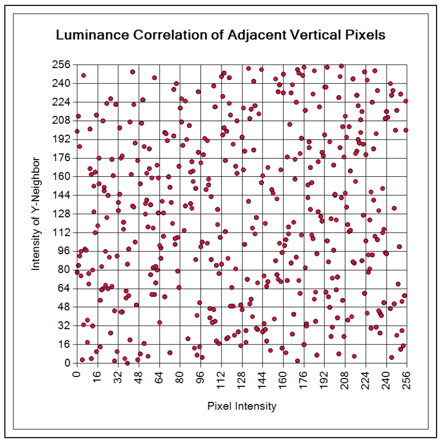
$$\text{Covariance of x and y:} \quad Cov(x,y) = \sum_{i=1}^{N} [x_i - E(x)]\,[y_i - E(y)] \tag{2.8}$$

$$\text{Correlation Coefficient of x and y:} \quad R_{x|y} = \frac{Cov(x,y)}{\left(\sqrt{D(x)}\sqrt{D(y)}\right)} \tag{2.9}$$

Figures 3A and 3B show the pixel horizontal and vertical adjacency correlations, respectively. One may see a strong linear correlation ($R_x^2 = 1$) in the horizontal correlation since the image is entirely composed of horizontal lines. However, the vertical adjacent pixel correlation has a low ($R_y^2 = 0.002$) correlation coefficient since the intensities of each row were determined at random (i.e. a randomly chosen differential exists between adjacent rows). The differences in these two figures show the extrema of the correlation possibilities.



(A) Horizontal Correlation



(B) Vertical Correlation

Figure 3: The Adjacent Pixel Correlations of the Test Image

An important cryptanalysis metric is Pearson's χ^2 statistic (also spelled out "chi-squared"). Succinctly, this is the measure of the similarity between two distributions. More formally, the χ^2 statistic is defined by assuming the null-hypothesis of a given probability distribution function (such as a uniform probability distribution) and calculating the "degree of disagreement" between the data and the null-hypothesis[16]. The χ^2 statistic for a grayscale image is mathematically defined by Equation 2.11.

$$\text{For probability distribution:} \quad P_0, P_1, \ldots, P_{255}$$

$$\text{The expected values (counts) are:} \quad E_i = NP_i \tag{2.10}$$

$$\text{Where} \quad N \equiv \text{Total number of pixels}$$

$$\text{and the Test Statistic is:} \quad \chi^2 = \sum_{i=0}^{255} \frac{[n_i - E_i]^2}{E_i} \tag{2.11}$$

When $\chi^2 = 0$ the observed distribution is exactly equal to the expected distribution. This is important when comparing an encrypted image's histogram to a uniform distribution or to the histogram of the plaintext image. If $\chi^2_{uniform} \to 0$, the encrypted image's histogram approaches that of a uniform distribution and has lost any "tells" that betray the plaintext image. If the value of the test statistic using the plaintext image's histogram, $\chi^2_{reference} \to 0$, the histogram of the encrypted image is *very* similar to the source plaintext image and can be used to identify the source image. **An ideally encrypted image will have $\chi^2_{uniform}$ minimized and $\chi^2_{reference}$ maximized**.

In formal statistical hypothesis testing, a level of significance, α, is predefined and the inverse cumulative distribution function, $\chi^2_{\alpha,k}$, is calculated for the predetermined significance and degrees of freedom, df. One then rejects the null-hypothesis when $\chi^2_{uniform|reference} > \chi^2_{\alpha,k}$.

For example, if one assumes the null-hypothesis that an encrypted image is statistically similar to a uniform distribution with a level of significance $\alpha = 0.05$ (this implies a confidence of 95%). The critical value $\chi^2_{\alpha=0.05,df=511} \approx 565$. If $\chi^2_{uniform} = 200$, $\chi^2_{uniform} < \chi^2_{\alpha=0.05,df=511}$ since $200 < 565$ and there is **insufficient evidence to reject the null-hypothesis** leading to the conclusion that the encrypted image has a distribution that is statistically similar to the uniform distribution[1]. If $\chi^2_{uniform}$ had the value $20,000$, one would be forced to reject the null hypothesis since $20,000$ is greater than the critical value for the chosen level of significance which would imply that there is a statistically significant deviation from the uniform distribution.

Figure 4 helps to better visualize the metric of the χ^2 statistic. In the figure, a uniform distribution is given by the horizontal, dashed, black line with a value of 1024 for all observations. A normal/Gaussian distribution is shown as a solid, blue line with a maximum at observation 127. Additionally, a periodic distribution given by $y = 512\sin(2\pi x/64) + 1024$ is shown by the dot-dashed, red line. Assuming the null-hypothesis that each distribution is identical to the null hypothesis, the χ^2 test statistics for the uniform distribution, periodic distribution, and normal distribution are 0, 3.2×10^4, and 5.7×10^4, respectively. This shows that the χ^2 statistic of the periodic distribution is appreciably closer to the uniform distribution than the normal distribution—an observation that is readily apparent in Figure 4.

As it relates to cryptography, the most secure scenario is that a randomly chosen byte from a ciphertext has an equal probability to be in the set $[0,255]$. If the probability distribution function were a Gaussian distribution as that shown in Figure 4, one could expect a higher probability of choosing a byte near the mean value (in this case, 127) and a significantly lower probability of randomly selecting a byte outside of the 80%

[1]In formal statistics, one must be careful to observe the distinction between “fail to reject the null hypothesis” and “accept the null hypothesis”. However, for the purposes of image cryptanalysis one may view nonrejection and acceptance as synonyms.

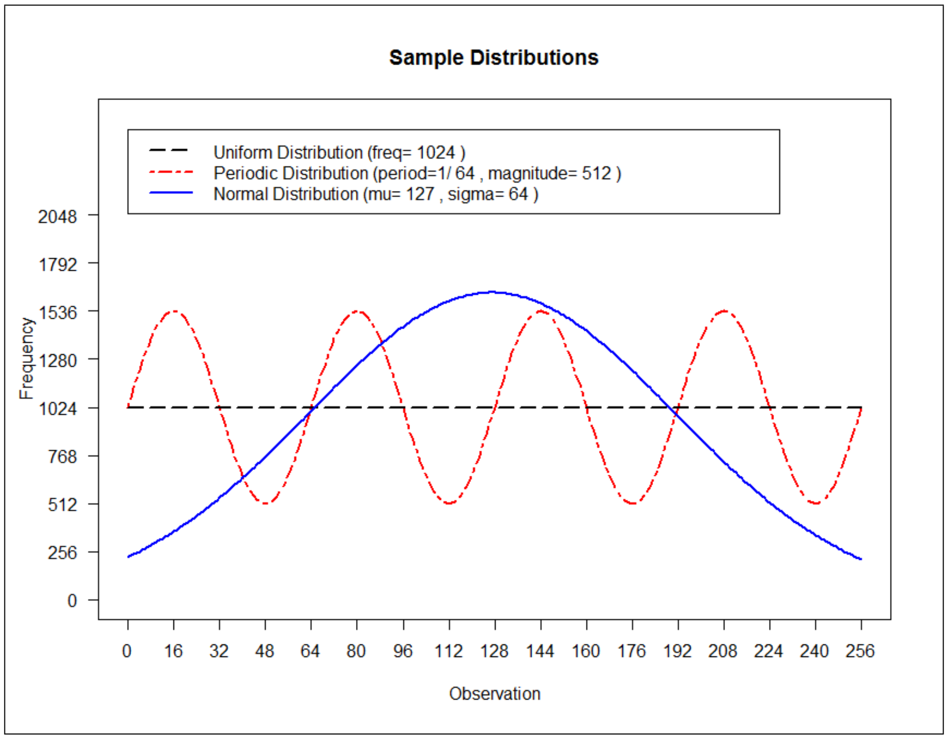


Figure 4: Sample Distributions for Demonstrating the χ^2 Statistic

confidence interval band (1.282σ) containing pixel values in the set [46, 208]. For the periodic distribution, a cryptanalyst could expect a higher probability of randomly selecting a byte near the crests defined by $64n+16$, for $n \in [0,4]$. These side-channel leaks may betray underlying structure to the ciphertext such as a cyclic state diagram or a short key that has been expanded. The interested reader is directed to advanced cryptanalysis texts such as Anderson's *Security Engineering*[3].

For any k-dimensional tensor, the maximum Pearson's χ^2 statistic is equal to:

$$\chi^2_{max} = N\left[\min\left(k_{\dim j}\right) - 1\right] \tag{2.12}$$

Note: the χ^2 distribution has $\min\left(k_{\dim j}\right) - 1$ degrees of freedom

From this, a [0, 1] bounded statistic can be derived from $\frac{\chi^2}{\chi^2_{max}}$ where $\frac{\chi^2}{\chi^2_{max}} = 0$ implies that an image is maximally similar to a reference histogram (i.e. the null-hypothesis is not rejected) and $\frac{\chi^2}{\chi^2_{max}} = 1$ implies that an image is maximally dissimilar to the presumed probability distribution.

Using the test image of Figure 1, $\chi^2_{uniform} = 0$ since the histogram is a perfectly uniform distribution; likewise, $\frac{\chi^2_{uniform}}{\chi^2_{max}} = 0$. Additionally, $\chi^2_{max} = (512 \times 512)[512 - 1] = 2^{18}\left(2^9 - 1\right) \approx 1.34 \times 10^8$.

Approach

As mentioned previously, it is ideal for the cryptosystem algorithms to ensure a high degree of confusion between the key and a ciphertext produced using this key. To this end, research is ongoing to explore how to produce higher degrees of confusion without compromising the system to forms of cryptanalysis attacks. This field of research is currently exploring the use of chaos-based approaches to key generation.

The first chaos model to be implemented for comparison will be a biologically-inspired model proposed by Al-Utaibi and El-Alfy[2]. This method accepts a grayscale *MxN* pixel image as input and implements crossover and mutation elements to produce a ciphertext image as output. In their paper, the authors present histograms of the pixel intensity values for both the plaintext image and the produced ciphertext.

The second chaos model that is to be implemented as part of this effort is the cellular automata method proposed by Jin[9]. This grayscale image encryption algorithm uses a scanline (a line of pixels) and predefined toggle cellular automata rules to produce the output scanlines. Under this scheme the ciphertext image output is determined from the pattern of source plaintext pixels and a secret ruleset. Jin shows (using pixel intensity histogram and vertically adjacent pixel correlation) that the encrypted plaintext image has adequate key diffusion and confusion.

Finally, a new image encryption model is proposed using techniques from fluid dynamics as a chaos model. This approach will be shown to produce a better cryptosystem than either the bio-inspired or cellular automata technique. Additionally, the time performance is better than the bio-inspired approach. The metrics shown in the following chapters will show that the encrypted images produced by the proposed fluid-dynamics-inspired method has great key diffusion and a more uniform distribution of pixel intensity.

CHAPTER III.
METHODS

The Bio-Inspired Technique

The first image encryption technique for comparison is a biologically-inspired approach presented by Al-Utaibi and El-Alfy[2]. In this approach, the biological processes of crossover and mutation are implemented for both the rows and columns of an input image. This means that four arrays of double precision values are needed (crossover and mutation arrays for both the rows and columns). These arrays are initialized by the shared secret key:

$$\text{Key:}k = \{(r,x_0)_0\,,(r,x_0)_1\,,(r,x_0)_2\,,(r,x_0)_3\}$$

Where:

$$r_i \in \{r_i \mid 3.569955672 \leq r_i < 4\}$$

$$x_i \in \{x_i \mid 0 \leq x_i \leq 1\}$$

$$i \in [0,3]$$

Where the subscript index 0 implies the column crossover array, index 1 implies the row crossover array, index 2 implies the column mutation array, and index 3 implies the row mutation array. According to the paper's authors, the effective key space for this encryption technique is $\approx 2^{398}$—i.e. an effective key length of 398 bits. These inputs are given to the discrete logistic map function[7, 20]:

$$x_{i,n+1} = r_i x_{i,n}\left(1 - x_{i,n}\right) \tag{3.1}$$

As may be seen in Figure 5, this iterative function is chaotic for $r_i \geq 3.569955672$. The bifurcation diagram of the logistic map may be reproduced using the R script in Appendix A.

To produce an array of unique random integers of size n, the logistic map is "seeded" with the r_i and $x_{i,0}$ values and each $x_{i,j}$ is stored. The array is sorted and the original *indices* form a set of random integers in $[0,n]$. These arrays of integers are used to perform row and column swaps (simulating crossover) and XOR-ing (simulating mutation).

Algorithm 3.1: Creating a Randomized Index Array using the Logistic Map

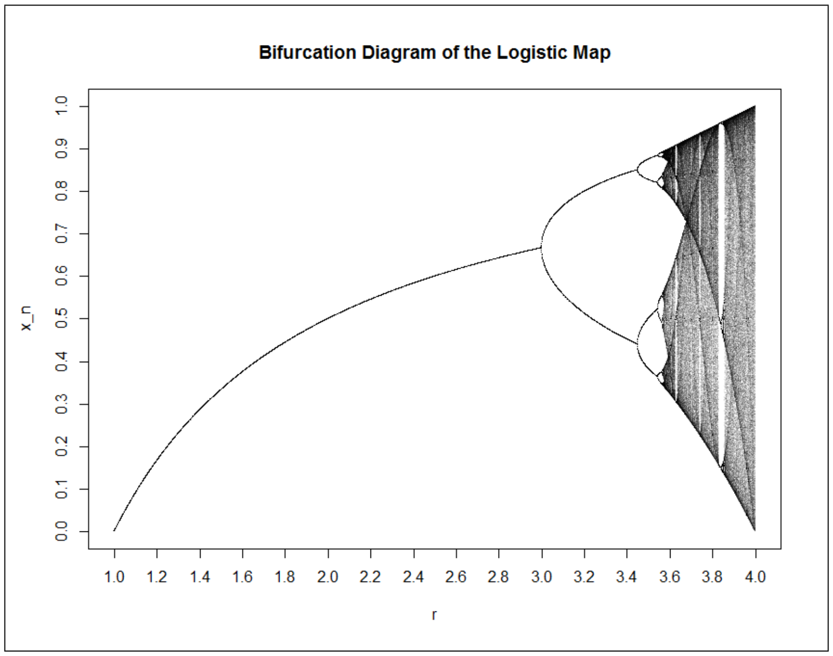
```
input: Array logMap, int size, double $x_0$, double r
output: Array indices
begin
    $logMap_0 \leftarrow x_0$
    $indices_0 \leftarrow 0$
    $i \leftarrow 1$
    while $i < size$
        $indices_i \leftarrow i$
        $logMap_i \leftarrow r logMap_{i-1}(1 - logMap_{i-1})$
    end

    # Sort the logMap array and swap the indices in the indices array.
    sort (logMap, indices)
    return indices
end
```

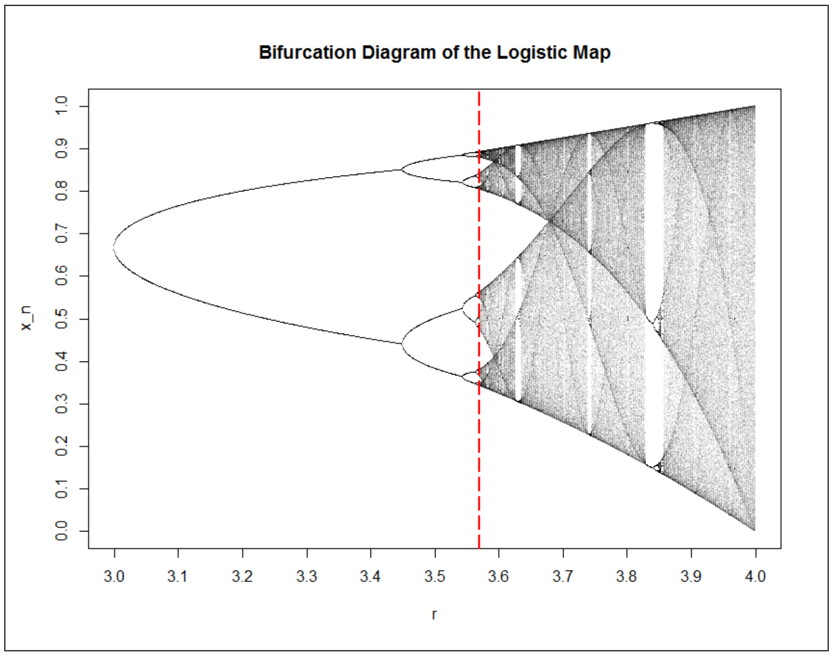
Lastly, a secret image is XOR-ed with the intermediate row- and column-shuffled (permuted) image to normalize the image histogram.

The Elementary Cellular Automata Technique

The second encryption technique that was implemented is an Elementary Cellular Automata (ECA) approach proposed by Jin[9]. Here, the Wolfram Cellular Automata (CA) rule[21] is used in conjunction with an initial state and a Pseudo-Random Number Generator (PRNG) fixed seed integer to initialize the key. The effective key length given the member constraints is effectively 48 bits yielding a key space of 2^{48} which is significantly less than the full key space of the bio-inspired method mentioned previously. This technique takes advantage of the rotor-like effect of the state attractors (i.e. cyclic state machine diagrams) in cellular automata to provide permutation of the input while also allowing the algorithm to easily return the original image when decrypted with the same input key. An example of



(A) The Bifurcation Diagram for $r \in [1,4]$



(B) The Bifurcation Diagram for $r \in [3,4]$ Showing Chaos for $r > 3.569955672$

Figure 5: The Bifurcation Diagram of the Logistic Map Showing Deterministic Chaos (n=2000 iterations, Generated using the script in Appendix A)

this cyclic attractor for Wolfram rule 42 and initial state 53 may be seen in Figure 6.

$$\text{Key:}k = \{rule, state_1, seed\}$$

Where:

$$rule \in \{rule \mid 0 \leq rule < 256\}$$

$$state_1 \in \{state_1 \mid 0 \leq state_1 < 256\}$$

$$seed \in \left[0, 2^{32} - 1\right]$$

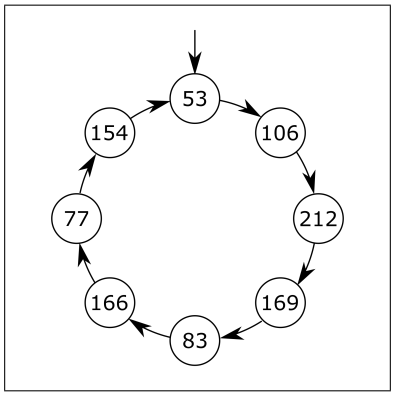


Figure 6: The Cellular Automata State Attractor for Initial State 53 and Wolfram Rule 42

The Fluid-Dynamics Inspired Technique

This work introduces an additional image encryption technique that borrows equations and analogies from the fields of Computational Fluid Dynamics and Heat Transfer. In this approach, each scanline (row) of pixels is envisioned as a circular pipe of an incompressible (i.e. constant density) fluid. An array of random integers (generated by the logistic map) is generated and is envisioned as a mass flow rate column vector. For each “pipe” (row) the governing one-dimensional (1D) continuity equation for incompressible, steady flow is derived from the simple schematic shown in Figure 7 as:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0 \tag{3.2}$$

$$\text{Incompressible} \implies \frac{\partial \rho}{\partial x} = 0$$

$$\text{Steady} \implies \frac{\partial \rho}{\partial t} = 0$$

$$\therefore \frac{\partial u}{\partial x} = 0 \tag{3.3}$$

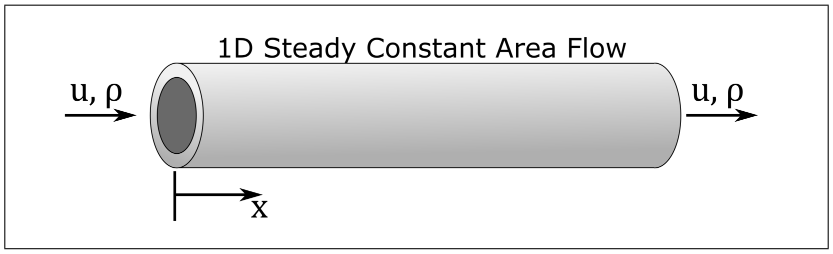


Figure 7: A 1D, Steady, Incompressible, Constant Area Streamline used to derive the Continuity Equation

This means that the “mass flow rate” of a uniform pipe must be constant (i.e. the volume dilation rate is zero). This fundamental observation from fluid mechanics (“what goes in, must come out”) provides permutation of the input image by right-circular shifting the pixels of each row of the image by the integer “mass flow rate”.

Diffusion and obfuscation of the input image is provided by a Fourier conduction analogy.

The conduction mode of heat transfer is governed by the equation:

$$\dot{q}'' = \kappa \frac{\partial T}{\partial x} \tag{3.4}$$

$$\text{Let Thermal Conductivity,} \quad \kappa = 1$$

$$\text{Let} \quad \partial x = 1$$

$$\therefore \dot{q}'' = \partial T \approx (T_{i+1} - T_i)$$

$$\text{Rearranging:} \quad T_{i+1} = T_i + \dot{q}'' \tag{3.5}$$

$$\text{In digital logic:} \quad T_{i+1} = T_i \oplus \dot{q}'' \tag{3.6}$$

In this step, each pixel represents a “temperature” and a “heat flux”, $\dot{q}''$, is transferred along each column. To ensure the pixels remain within the appropriate 8-bit range, an XOR operation is performed rather than addition so the final equation becomes $T_{i+1} = T_i \oplus \dot{q}''$ as shown by Equation 3.6. The “heat flux” array is also produced using the logistic map as discussed previously using Algorithm 3.1.

To further obfuscate the image and extend the key space, a secret image is also used along with the turbulent energy decay function (see Figure 8) given by the equation[10]:

$$k(t) = k_0 \left[1 + \frac{0.92\varepsilon}{k_0} t\right]^{\frac{-1}{0.92}} \tag{3.7}$$

Where:

$$k_0 \equiv \text{Initial turbulent kinetic energy}$$

$$\varepsilon \equiv \text{Turbulent kinetic energy dissipation rate}$$

A user-input initial turbulence energy (k_0), turbulence energy dissipation rate (ε), and time frame (t) generate a scalar turbulence energy value $k(t)$. The final step of obfuscation is to

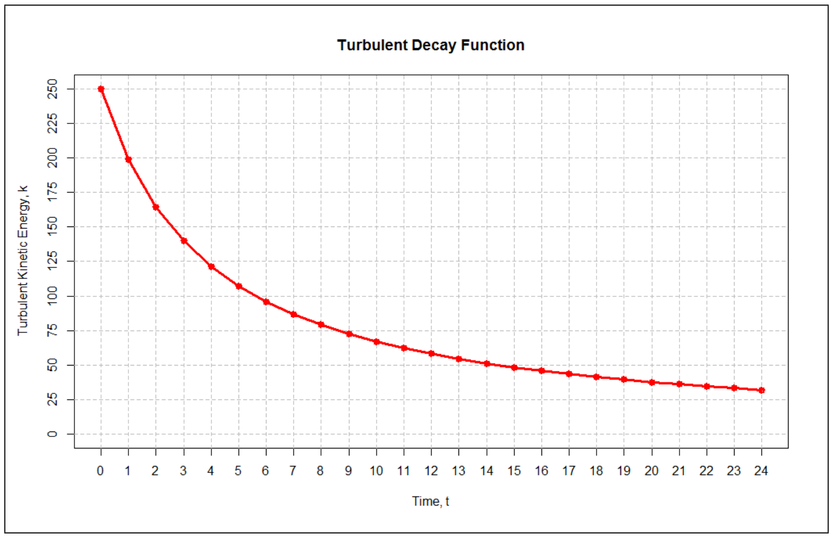


Figure 8: A Representative Turbulent Decay Function with $k = 250$ and $\varepsilon = 64$

apply the XOR relation:

$$p_{i,j,encrypted} = p_{i,j,intermediate} \oplus p_{i,j,secret} \oplus k(t) \tag{3.8}$$

for each pixel $p_{i,j}$ of the intermediate image.

Given these data, the shared secret key is represented by the structure:

$$\text{Key:}k = \{(r,x_0)_{\dot{m}}, (r,x_0)_{\dot{q}''}, k_0, \varepsilon, t_{frame}\}$$

Where:

$$r_{\dot{m}|\dot{q}''} \in \{r_{\dot{m}|\dot{q}''} \mid 3.569955672 \leq r_{\dot{m}|\dot{q}''} < 4\}$$

$$x_{\dot{m}|\dot{q}''} \in \{x_{\dot{m}|\dot{q}''} \mid 0 \leq x_{\dot{m}|\dot{q}''} \leq 1\}$$

$$k_0 \in \{k_0 \mid 0 \leq k_0 < 256\}$$

$$\varepsilon \in \{\varepsilon \mid 0 \leq \varepsilon < 256\}$$

$$t_{frame} \in \{t_{frame} \mid 0 \leq t_{frame} < 256\}$$

The author evaluates the key space of this process to be $\approx 2^{223}$, equivalent to a 223-bit secret key excluding the XOR operation with the secret image. Since the logistic map growth factors $r_{\dot{m}|\dot{q}''}$ is in the range $(3.569955672\ldots,4]$, only 14 bits are needed to represent this range. Therefore, $(10^{14})^2 \times (10^{16})^2 \times 2^8 \times 2^8 \times 2^8 \approx 2^{223}$. This represents a key space that is 4.7 times larger than the cellular automata technique, but 56% of the length of the bio-inspired approach. Though this is a smaller key space than the bio-inspired approach, this technique provides more key diffusion than the bio-inspired approach.

A pictorial representation of the overall process is given by Figure 9.

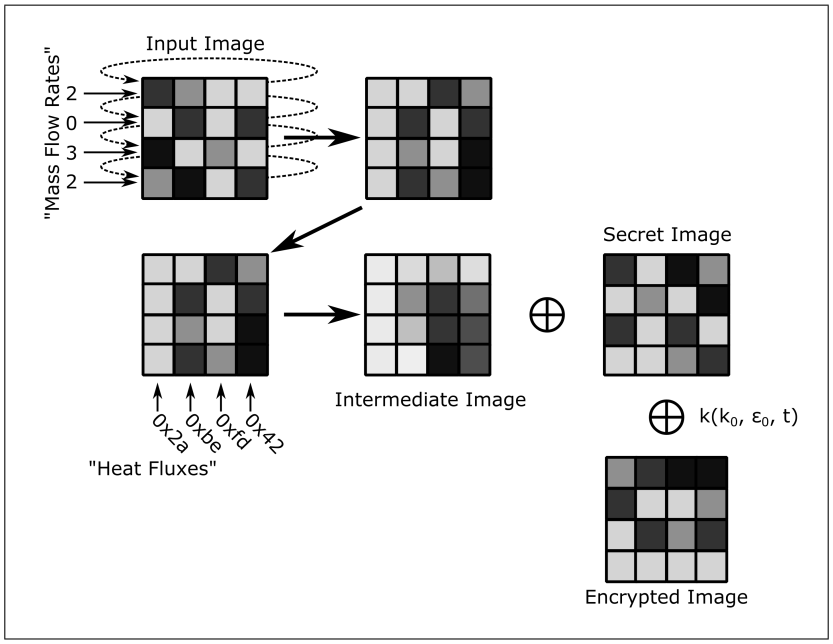


Figure 9: A schematic of the Fluid Dynamics Inspired Image Encryption Process

The Graphical User Interface for Evaluations

A Graphical User Interface (GUI)[1] was developed for Microsoft Windows® using the C# language and Windows Forms. The AForge.NET machine learning framework[1] was used for generating Fourier spectra and the edge detected images. An image of the main GUI window is shown in Figure 10.

The GUI provides a fast and easy way to observe the metrics and characteristics of an

[1]The GUI source code and all files may be found on the author's GitHub® repository at: `https://github.com/ghammock/ImageEncryptionSuite/`

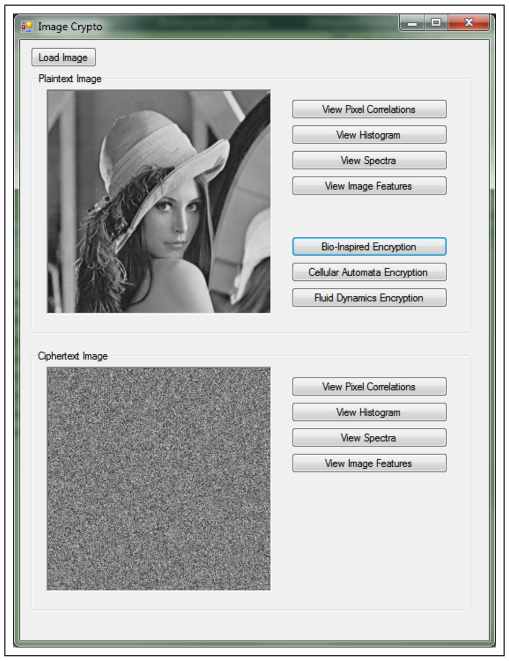


Figure 10: The Main Window of the GUI

input image as well as the images returned by the selected encryption technique. As may be seen in Figure 11, the option to view and record the pixel adjacency correlations may be performed by viewing the correlations child window (Figure 11A). This is the plot produced using Equations 2.6 through 2.9 for an input number of random pixel samples.

Figure 11B shows the luminance histogram window. The histogram is useful in determining the number of pixels at each visible intensity level over the 8-bit domain. This effectively provides a "signature" of the image and is analogous to a frequency histogram in traditional one-dimensional cryptosystems.

Figure 11C displays the Fourier magniutude spectra of the given image. For example, in Figure 11C, the *Lenna* image produces the two-dimensional Discrete Fourier Transform (2D-DFT) shown on the right side of the figure. The 2D-DFT was implemented as a 2D Fast Fourier Transform (2D-FFT) to maximize speed. An image spectra is useful as another signature type since it may reveal underlying structure in the image that is not readily apparent in the spatial domain. It may also be used to link a known image to a given spectra analogous to using rainbow tables[8] for exploiting weak hash algorithms.

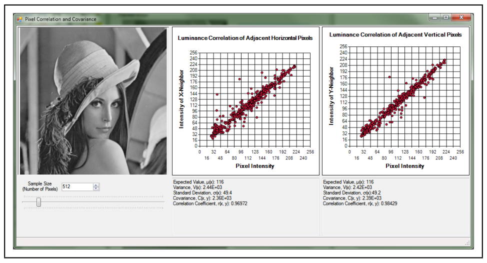
Finally, Figure 11D shows the detected edges of an input image. The two edge detection algorithms used were the Sobel edge detection algorithm and the Canny edge detection algorithm. The Sobel edge detection technique uses a 3x3 convolution kernel to amplify the high spatial frequency regions typical of edges[6]. The Canny edge detection technique tracks intensity discontinuities using a first derivative operator[6]. Both techniques are good at finding edges but do so in different ways and with varying sensitivities.

For comparing and contrasting the bio-inspired, cellular automata, and fluid dynamics inspired image encryption algorithms, the image and tabular data produced by the GUI is recorded and is presented in Chapter IV.

Additionally, a high precision stopwatch was programmed into the GUI to accurately measure the encryption time for each of the image encryption techniques. Ten tests were conducted at different times and at varying computational load to build the encryption time dataset. The specifications and hardware configuration for the computer system under which these tests were performed is given in Table 1. The average benchmark time for each of the techniques is also presented in Chapter IV.

Table 1: Benchmark Computer Configuration

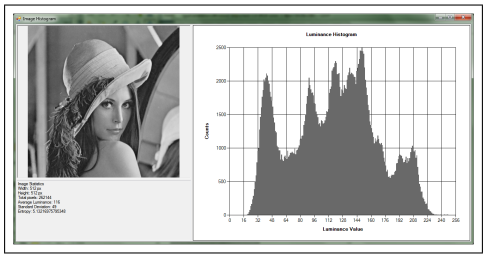
Operating System	Windows 7
Architecture	64-bit
Processor	Intel® Core™ i5-3450 (Quad-Core, 3.10 GHz)
Processor Architecture	“Ivy Bridge”, 22 nm
Memory	16 GB DDR3 (9-9-9-24)


Pixel Correlation and Covariance
Luminance Correlation of Adjacent Horizontal Pixels
Intensity of X-Neighbor
Pixel Intensity
Luminance Correlation of Adjacent Vertical Pixels
Intensity of Y-Neighbor
Pixel Intensity
Sample Size (Number of Pixels)
512

```
Expected Value, μ(x): 116
Variance, V(x): 2.44E+03
Standard Deviation, σ(x): 49.4
Covariance, C(x, y): 2.36E+03
Correlation Coefficient, r(x, y): 0.96972
```

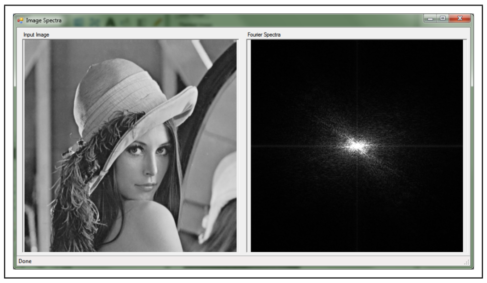
```
Expected Value, μ(x): 116
Variance, V(x): 2.42E+03
Standard Deviation, σ(x):49.2
Covariance, C(x, y): 2.39E+03
Correlation Coefficient, r(x, y): 0.98429
```

(A) The Correlations Window

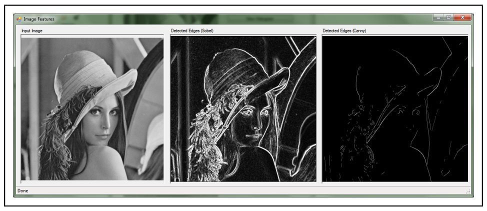


```
Image Statistics
Width: 512 px
Height: 512 px
Total pixels: 262144
Average Luminance: 116
Standard Deviation: 49
Entropy: 5.13216975795348
```

(B) The Histogram Window



(C) The Spectra Window



(D) The Edge Detection Window

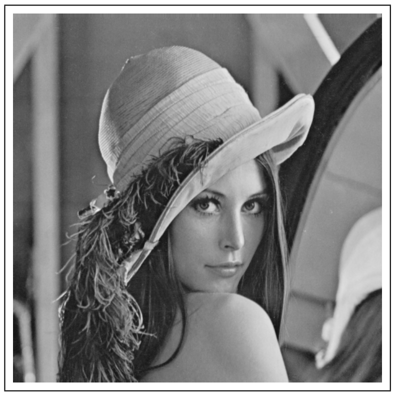
Figure 11: The Various Child Windows of the GUI

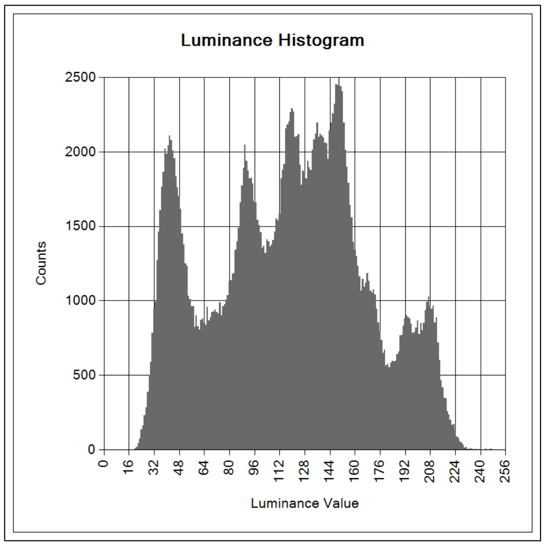
CHAPTER IV.

RESULTS

The Test Image

Before displaying the results of each of the three image encryption algorithms, it is prudent to examine the metrics of the source input image. Figure 12A is a classical photograph of Lena Söderberg (the photograph is often referred to as *Lenna* or *Lena*). This is a beautiful—albeit somewhat controversial—image that is used as a standard image analysis input because, as the editor-in-chief of *IEEE Transactions on Image Processing*, David C. Munson Jr. states, "the image contains a nice mixture of detail, flat regions, shading, and texture that do a good job of testing various image processing algorithms"[17]. The image also has a well defined histogram (Figure 12B) and Fourier spectra (Figure 14).


(A) *Lenna*



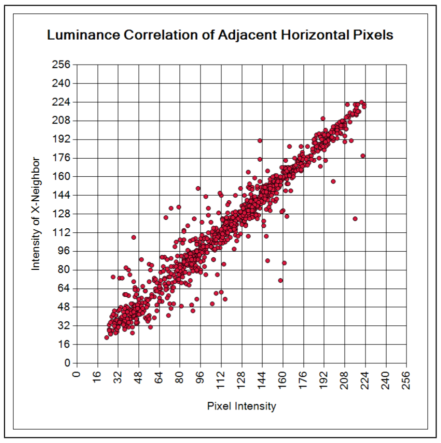
(B) Histogram

Figure 12: The Standard Image Analysis Input Image *Lenna* and it's Pixel Luminance Histogram

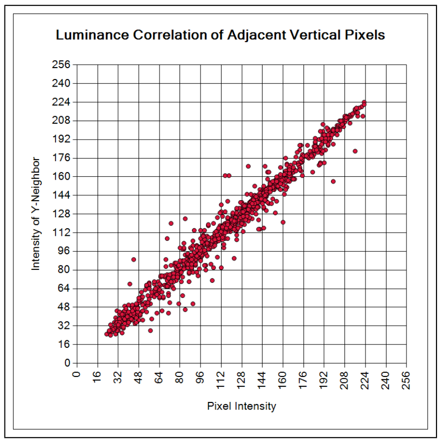
The image has a width of 512 pixels and a height of 512 pixels. This gives $(512-1) = 511$ degrees of freedom. Assuming a level of significance $\alpha = 0.05$ (alternatively, a confidence of $(1-\alpha) = 0.95$), the inverse cumulative χ^2 distribution function has the critical value

$\chi^2_{\alpha=0.05,df=511} \approx 565$. To statistically be able to reject a null-hypothesis (either that an image has a roughly uniform histogram or that an image has a histogram roughly equal to *Lenna*) a image's χ^2 test statistic must be greater than this critical $\chi^2_{\alpha,df}$ value. If χ^2 is less than this value, we fail to reject the null-hypothesis and are able to conclude the that the histogram of the image and the assumed reference are roughly identical. Additionally, recall from Chapter II. that for a 512 pixel $\times$ 512 pixel image, $\chi^2_{max} \approx 1.34 \times 10^8$ which is used when calculating the ratio χ^2/χ^2_{max}.

Figures 13A and 13B show the pixel horizontal and vertical adjacency correlations, respectively. It is readily apparent that a linear correlation exists between adjacent pixels in the image meaning that discernible photographic image data is present. For all correlations in this report, 1024 randomly chosen pixels were used as input into Equation 2.9 to define the correlation coefficients.



(A) Horizontal Correlation



(B) Vertical Correlation

Figure 13: The Adjacent Pixel Correlations of the *Lenna* Test Image

Additionally, the calculated Shannon entropy for the *Lenna* image is $h = 7.498$. Table 2 shows the calculated Shannon (information) entropy for the *Lenna* image as well as other quantitative metrics including the expected pixel intensity value (average), standard deviation, and χ^2 test statistics.

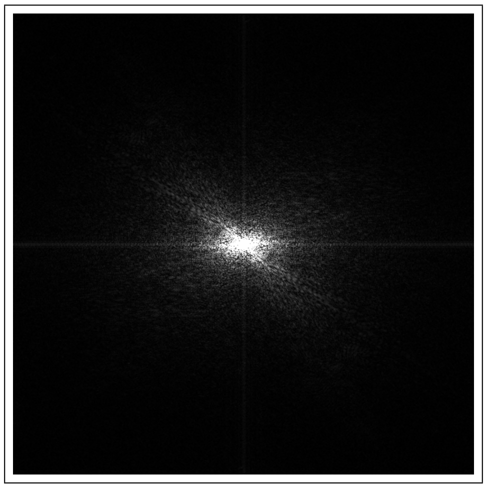


Figure 14: Fourier spectra of *Lenna*

Table 2: Image Metrics for *Lenna*

Metric	**Value**
Adjacent Horizontal Pixel Correlation, R_x	0.968
Adjacent Horizontal Coefficient of Determination, R_x^2	0.937
Adjacent Vertical Pixel Correlation, R_y	0.986
Adjacent Vertical Coefficient of Determination, R_y^2	0.972
Shannon Entropy, h	7.498
Average Pixel Value, E	116
Pixel Standard Deviation, σ	49
$\chi^2_{uniform}$	1.41×10^5
$\chi^2_{uniform}/\chi^2_{max}$	1.05×10^{-3}
χ^2_{Lenna}	0 (by definition)
$\chi^2_{Lenna}/\chi^2_{max}$	0 (by definition)

The Bio-Inspired Technique

The same key data from the Al-Utaibi and El-Alfy source (shown in Table 3) have been used so that meaningful comparisons may be made with Ref. [2]. As mentioned in Chapter III., the bio-inspired method is sensitive to the choice of secret image for normalizing the histogram data as well as obfuscating the horizontal and vertical pixel correlations. This sensitivity will be examined in the following sections.

Table 3: Secret Key for the Bio-Inspired Technique

$(r,x)_0$	$(3.7158, 0.11)$
$(r,x)_1$	$(3.89858, 0.25)$
$(r,x)_2$	$(3.76158, 0.35)$
$(r,x)_3$	$(3.8458, 0.552)$

The Bio-Inspired Technique with a Randomly Generated Secret Image

Figure 15 shows a secret image that was randomly generated using the C# `Random` class. The calculated Shannon entropy for this secret key image is $h = 7.999$ bits. It has an average luminance of $E = 126$, a standard deviation of $\sigma = 74$, and a uniform distribution test statistic, $\chi^2_{uniform} = 226.590$. Observe that for the random secret image, $\chi^2_{uniform} < \chi^2_{\alpha,df}$ (i.e. $226.590 < 565$) so one may conclude that the histogram of this random secret image is statistically indistinguishable from a uniformly distributed histogram with a confidence interval of 95%.

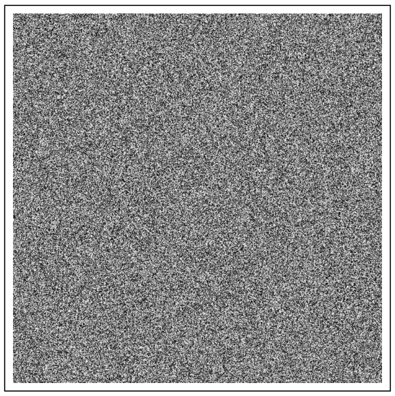
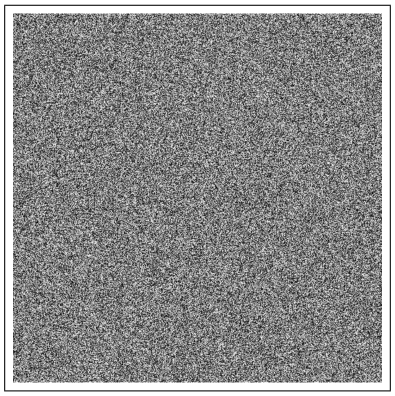
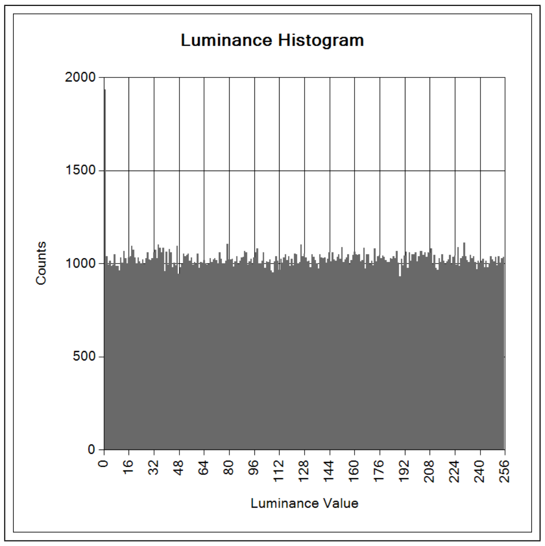


Figure 15: A Randomly Generated Secret Image

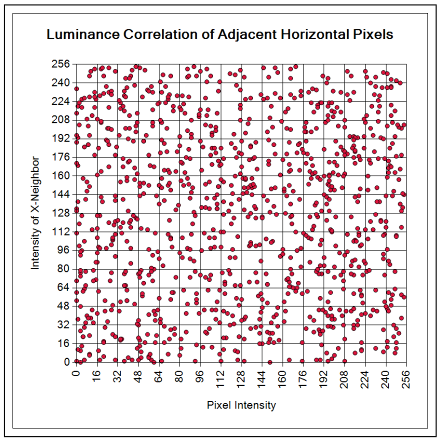
Figure 16A shows the input *Lenna* image encrypted using the bio-inspired technique of Ref. [2] and the randomly generated secret key Figure 15. Observe from Figure 16B that the secret image XOR process normalized the histogram of Figure 12B to disintegrate any "tells" of the image contents. Also observe from Figure 17 that the horizontal and vertical adjacency correlations have been uniformly distributed. In addition, the Fourier spectra of the encrypted image (Figure 18) appears to be nothing more than white noise. These are hallmarks of a good encryption technique and it is worth noting that the results of this implementation agree well with those published by Al-Utaibi and El-Alfy.


(A) Encrypted *Lenna*

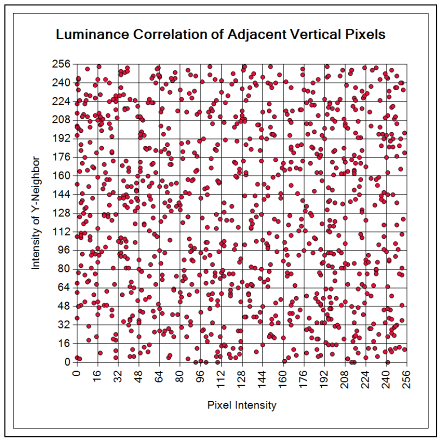


(B) Histogram

Figure 16: *Lenna* as Encrypted Using the Bio-Inspired Technique and the Encrypted Image's Histogram



(A) Horizontal Correlation



(B) Vertical Correlation

Figure 17: The Adjacent Pixel Correlations of Encrypted *Lenna* Test Image Encrypted via the Bio-Inspired Method

The calculated Shannon information entropy for the encrypted image (Figure 16A) is $h = 7.992$ (see Table 4). This means that the full eight bits of intensity encoding is needed to represent the image. Note that the uniform distribution test statistic

$$\chi^2_{uniform} = 2,070 > \chi^2_{\alpha=0.05,df=511}$$

meaning that the assumption of a uniform distribution with 95% confidence must be rejected. Upon closer inspection of the histogram data, there were no pixels in the encrypted image with the luminous intensity value 255; however, there were nearly twice as many pixels with the luminous intensity value of 0 than would be typical for a uniform distribution. This small deviation in the data manifests in the χ^2 value, which makes the χ^2 test statistic ideal for inferential statistics in cryptanalysis. Also observe that

$$\chi^2_{Lenna} = 7.44 \times 10^6 \ggg \chi^2_{\alpha=0.05,df=511}$$

meaning that the histogram of the encrypted image cannot be statistically correlated to the plaintext image's histogram.

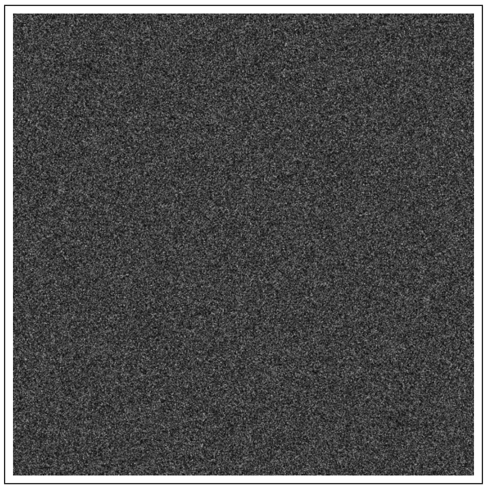


Figure 18: Fourier spectra of the Bio-Inspired Encrypted Image

The Bio-Inspired Technique with a Pure Black Secret Image

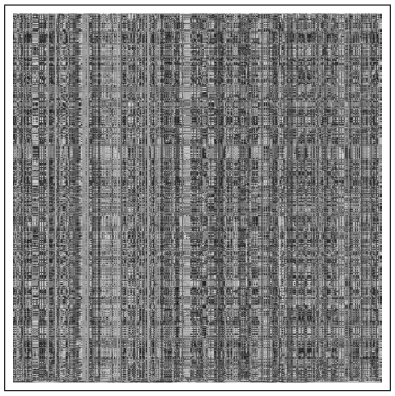
The worst-possible choice of secret image to use with this technique is a pure black image in which every pixel has the RGB value 0x000000. For a secret image such as this,

Table 4: Image Metrics for the Bio-Inspired Encrypted Image using the Randomly Generated Secret Key

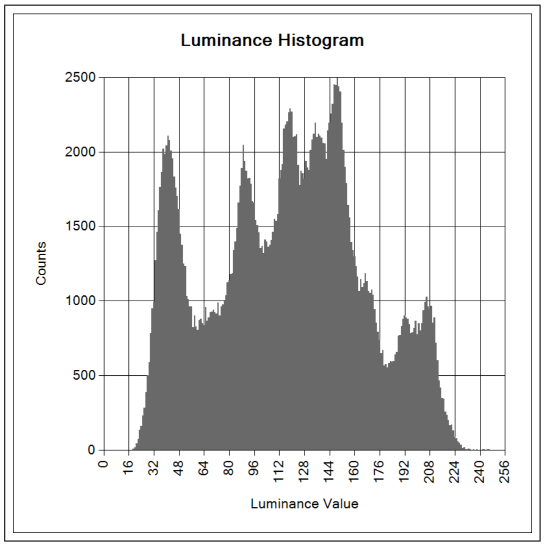
Metric	**Value**
Adjacent Horizontal Pixel Correlation, R_x	0.015
Adjacent Horizontal Coefficient of Determination, R_x^2	$\approx$0.000
Adjacent Vertical Pixel Correlation, R_y	-0.053
Adjacent Vertical Coefficient of Determination, R_y^2	0.003
Shannon Entropy, h	7.992
Average Pixel Value, E	126
Pixel Standard Deviation, σ	74
$\chi^2_{uniform}$	2,070
$\chi^2_{uniform}/\chi^2_{max}$	1.55×10^{-5}
χ^2_{Lenna}	7.44×10^{6}
$\chi^2_{Lenna}/\chi^2_{max}$	5.56×10^{-2}
Average Encryption Time, t_{encr}	1330 ms $\pm$ 11 ms

the Shannon entropy, h, is zero. Since the algorithm uses an XOR process to normalize the histogram and reduce adjacent pixel correlations, using this image is analogous to, at best, completely skipping the XOR process since for every pixel $p_{x,y} \oplus 0\text{x}000000 = p_{x,y}$. At worst, this would have the same effect as not using a secret image at all!

Figure 19A shows the encrypted *Lenna* image using the same key values from Table 3 and the pure black secret image. Observe that the histogram of the encrypted image (Figure 19B) is almost exactly the same as the histogram of the original plaintext *Lenna* image (Figure 12B) for the reason mentioned above. This shows the weakness of the bio-inspired approach in that a poor selection of secret key allows for the side channel leakage of histogram data. Also observe from Figure 20 that there is a linear clustering of the horizontal and vertical adjacency correlations as the correlation coefficients are 0.260 and 0.334, respectively. Additionally, the Fourier spectra of the encrypted image (Figure 21) has a tell-tale sign of hidden image data because of the strong center peak (the DC value) and the “plus” shape which is indicative of strong row and column information being present in the encrypted image.

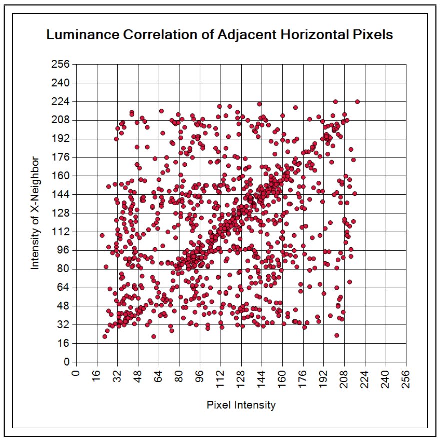


(A) Encrypted *Lenna*

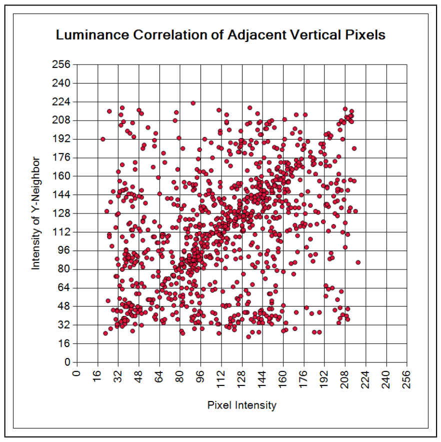


(B) Histogram

Figure 19: *Lenna* as Encrypted Using the Bio-Inspired Technique with a Pure Black Secret Image and the Corresponding Encrypted Image's Histogram



(A) Horizontal Correlation



(B) Vertical Correlation

Figure 20: The Adjacent Pixel Correlations of Encrypted *Lenna* Test Image Encrypted via the Bio-Inspired Method with a Pure Black Secret Image

The calculated Shannon information entropy for the encrypted image given in Figure 19A is $h = 7.498$ (see Table 5). Observe that this entropy value is identical to the entropy value of the source *Lenna* image. Note that even though the difference is small between this entropy value and the entropy value of the well-chosen secret image (i.e. $7.992 - 7.498 = 0.494$)

this difference is appreciable when considering that these are the results of the relation:

$$h = \log_2(\text{number of gray levels})$$

and $2^{7.992} - 2^{7.498} \approx 74$. This means that the encrypted image with a pure black secret image can be expressed with 74 fewer pixel values than the encrypted image with a randomly generated secret image—this is equivalent to a nearly 30% reduction ($100\% \times 74 \div 256$) in the available character space!

The uniform distribution test statistic

$$\chi^2_{uniform} = 1.41 \times 10^5 \gg \chi^2_{\alpha=0.05,df=511}$$

meaning that the assumption of a uniform distribution with 95% confidence must be rejected. However, note that

$$\chi^2_{Lenna} = 1,490 > \chi^2_{\alpha=0.05,df=511}$$

implying that although one must reject the hypothesis that this is statistically identical to the *Lenna* histogram with a 95% confidence interval though the value of the test statistic is only 2.6 times the cutoff value.

The Bio-Inspired Technique with a Pure White Secret Image

Another poor choice of secret key is a pure white image where every pixel has the RGB value 0xffffff and also has a Shannon entropy value of zero. The XOR process of the bio-inspired algorithm with this secret image may be represented by $p_{x,y} \oplus \text{0xffffff} = \neg p_{x,y}$.

The key values from Table 3 were used along with a pure white secret image to generate the encrypted *Lenna* image shown in Figure 22A. One may readily see that the histogram of the encrypted image given in Figure 22B is the horizontally-flipped mirror image of the histogram produced by the pure black secret key version (Figure 19B) because the XOR-ing process produced the complement of each pixel intensity. For the same reason

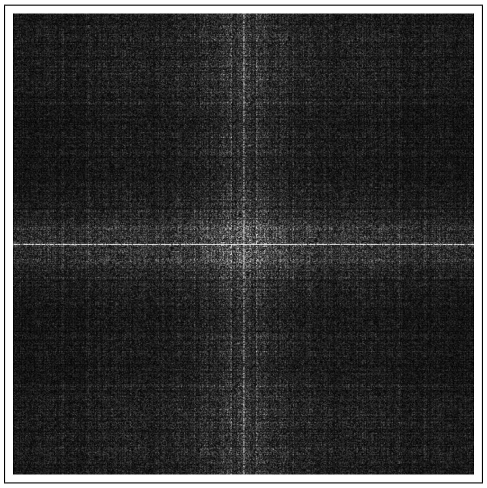
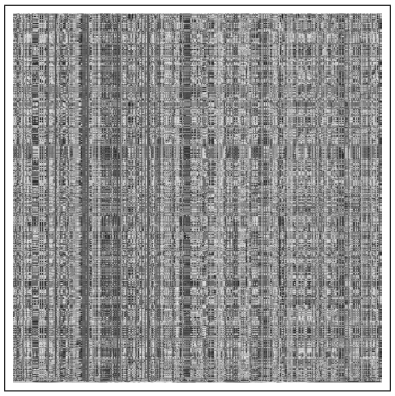


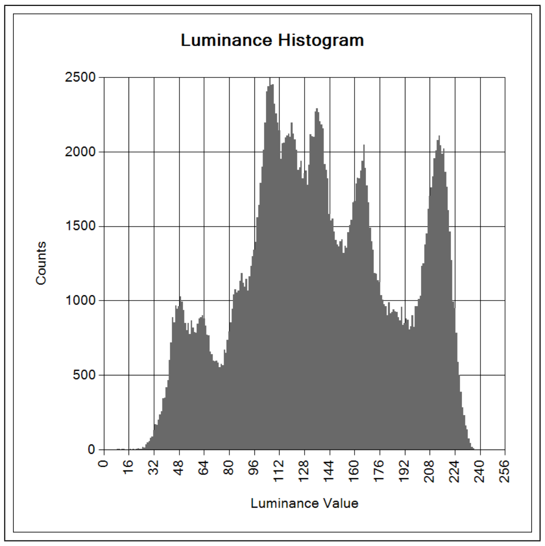
Figure 21: Fourier spectra of the Bio-Inspired Encrypted Image with the Pure Black Secret Key

Table 5: Image Metrics for the Bio-Inspired Encrypted Image using the Pure Black Secret Image

Metric	**Value**
Adjacent Horizontal Pixel Correlation, R_x	0.260
Adjacent Horizontal Coefficient of Determination, R_x^2	0.068
Adjacent Vertical Pixel Correlation, R_y	0.334
Adjacent Vertical Coefficient of Determination, R_y^2	0.112
Shannon Entropy, h	7.498
Average Pixel Value, E	115
Pixel Standard Deviation, σ	49
$\chi^2_{uniform}$	1.41×10^5
$\chi^2_{uniform}/\chi^2_{max}$	1.05×10^{-3}
χ^2_{Lenna}	1,490
$\chi^2_{Lenna}/\chi^2_{max}$	1.11×10^{-5}
Average Encryption Time, t_{encr}	1330 ms $\pm$ 10 ms

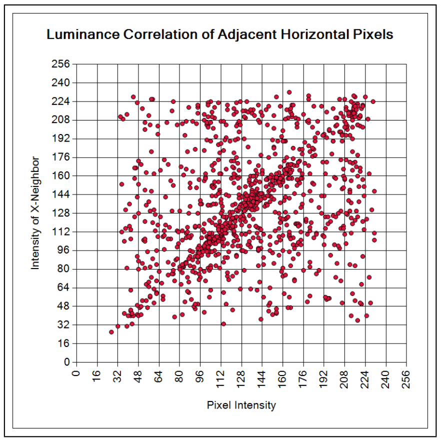
mentioned previously, this weakness can allow an adversary to deduce the contents of the image by comparing the histogram of the encrypted image to a database of known histogram signatures—similar to the use of rainbow tables for exploiting weak hash algorithms. The adjacency correlations also show linear clustering as may be seen in Figure 23. There is no difference in the Fourier spectra of the encrypted images using the pure white secret image and the pure black secret image (Figures 24 and 21, respectively) because the Fourier images are only the magnitude plots. The spectral differences would be captured by the Fourier phase plots (which were not computed).


(A) Encrypted *Lenna*

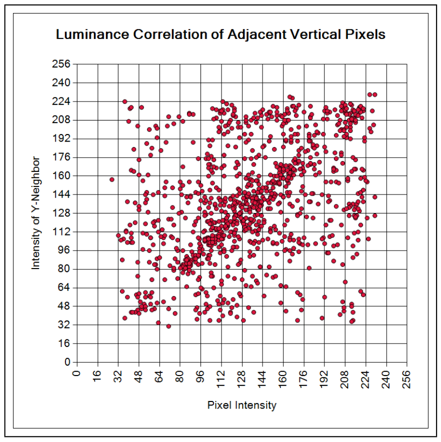


(B) Histogram

Figure 22: *Lenna* as Encrypted Using the Bio-Inspired Technique with a Pure White Secret Image and the Corresponding Encrypted Image's Histogram



(A) Horizontal Correlation



(B) Vertical Correlation

Figure 23: The Adjacent Pixel Correlations of Encrypted *Lenna* Test Image Encrypted via the Bio-Inspired Method with a Pure White Secret Image

The calculated Shannon information entropy for the image encrypted with a pure white secret image is $h = 7.498$ (see Table 6). As with the analysis with the pure black secret image, the uniform distribution test statistic

$$\chi^2_{uniform} = 1.41 \times 10^5 \gg \chi^2_{\alpha=0.05,df=511}$$

so the null hypothesis of having a uniform distribution with 95% confidence must be rejected. In addition,

$$\chi^2_{Lenna} = 1.46 \times 10^5 \gg \chi^2_{\alpha=0.05,df=511}$$

meaning that a χ^2 test would fail to conclude the histograms of the encrypted image and the histogram of *Lenna* are as similar as they are. If a cryptanalysis algorithm checked for histogram transposition, it would discover the same χ^2 value as the histogram from the pure black secret image analysis.

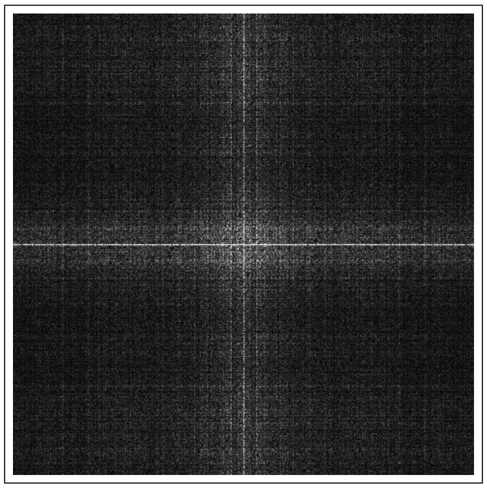


Figure 24: Fourier spectra of the Bio-Inspired Encrypted Image with a Pure White Secret Image

The Elementary Cellular Automata Technique

The second form of image encryption implemented in this work is an Elementary Cellular Automata approach that was proposed by Jin[9]. The same key values from Jin's work is used here and is shown in Table 7. The circular state attractor for Wolfram rule 42 and this key's initial state, S_0, was shown previously in Figure 6. The Elementary Cellular Automata

Table 6: Image Metrics for the Bio-Inspired Encrypted Image using the Pure White Secret Key

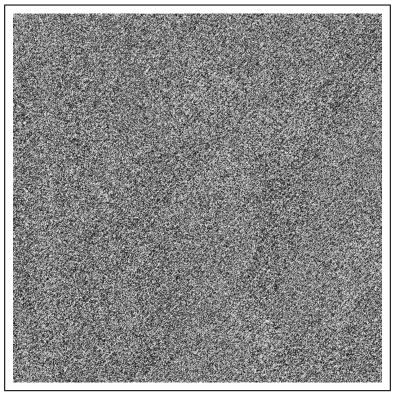
Metric	**Value**
Adjacent Horizontal Pixel Correlation, R_x	0.324
Adjacent Horizontal Coefficient of Determination, R_x^2	0.105
Adjacent Vertical Pixel Correlation, R_y	0.384
Adjacent Vertical Coefficient of Determination, R_y^2	0.147
Shannon Entropy, h	7.498
Average Pixel Value, E	137
Pixel Standard Deviation, σ	49
$\chi^2_{uniform}$	1.41×10^5
$\chi^2_{uniform}/\chi^2_{max}$	1.05×10^{-3}
χ^2_{Lenna}	1.46×10^5
$\chi^2_{Lenna}/\chi^2_{max}$	1.09×10^{-3}
Average Encryption Time, t_{encr}	1330 ms $\pm$ 16 ms

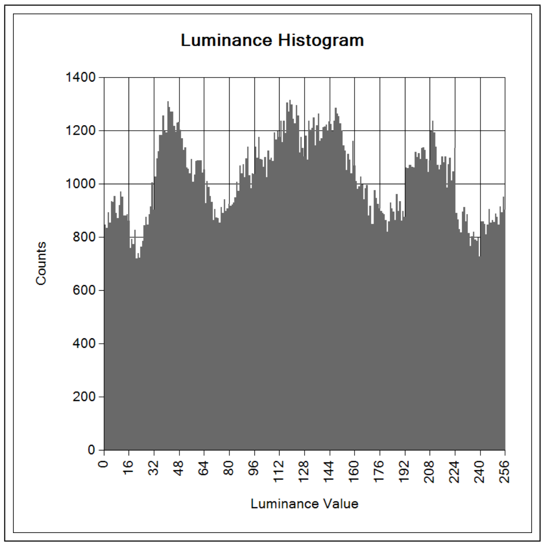
(CA) approach does not perform as well as the bio-inspired technique. At first glance, the CA encrypted image of Figure 25A does not reveal any underlying structure. The histogram (Figure 25B) presents a non-uniform distribution with a mean near the luminous center; however, this histogram is distinct from the histogram of the source *Lenna* histogram (Figure 12B). The pixel adjacency correlations in Figure 26 show a desirable uniform distribution with a correlation coefficient near zero. However, the Fourier spectra in Figure 27 presents a definite structure. There are multiple spectral peaks along a 45° line through the DC coefficient. This is indicative of some underlying information in the encrypted image!

Table 7: Secret Key for the Elementary Cellular Automata Technique

Wolfram Rule Number	42
Initial State, S_0	53
PRNG Seed	15

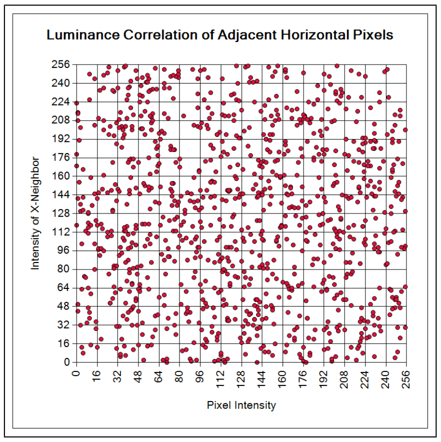
After this "side-channel spillage" from the Fourier spectra, a closer look at the encrypted image betrays some of the plaintext image contents. Figure 28A is the original CA encrypted image. Figure 28B shows the CA encrypted image with the Sobel detected edges of the plaintext *Lenna* image overlaid. From this one can see a "shadow" image of *Lenna* in the


(A) Encrypted *Lenna*

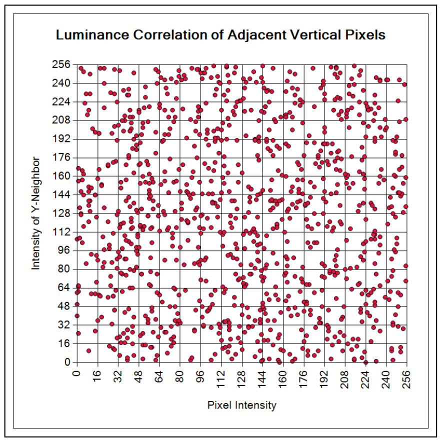


(B) Histogram

Figure 25: *Lenna* as Encrypted Using Cellular Automata Technique and the Encrypted Image's Histogram



(A) Horizontal Correlation



(B) Vertical Correlation

Figure 26: The Adjacent Pixel Correlations of Encrypted *Lenna* Test Image Encrypted via the Cellular Automata Method

encrypted image! (It may be useful to zoom out or stand farther away from Figure 28A to see this effect.) Figure 28C has been pseudo-colored with edges overlaid and callouts to emphasize this phenomenon.

The CA encrypted image, histogram, and correlations presented here agree very well with those presented by Jin in Ref. [9]. In fact, the intensity spillage may be observed in the

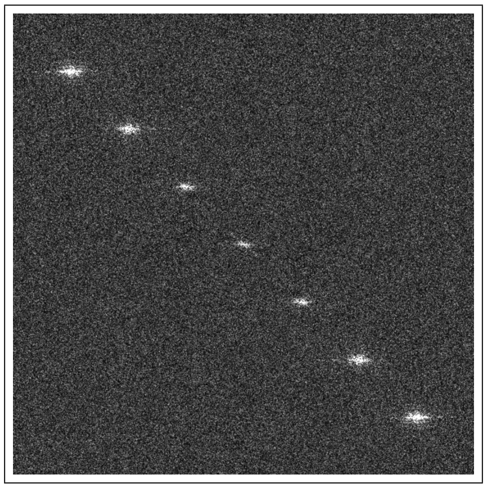
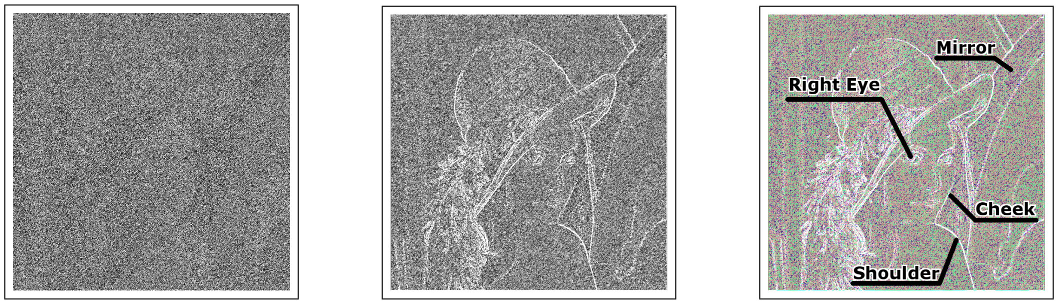


Figure 27: The Fourier Spectra of the Cellular Automata Encrypted Image



(A) CA Encrypted Image

(B) CA Encrypted Image with Sobel Edges Overlay

(C) CA Encrypted Image in Pseudocolor with Edges and Feature Callouts

Figure 28: The Cellular Automata (CA) Method has a Weakness by Inadvertently Preserving Luminous Features

Jin paper directly (after “zooming out”).

As may be seen in Table 8, the Shannon entropy for the CA encrypted image was calculated to be $h = 7.985$ which is good since it shows that, effectively, the entire gray-level space must be used to encode the encrypted image. The calculated uniform distribution χ^2 test statistic

$$\chi^2_{uniform} = 5,450 > \chi^2_{\alpha=0.05,df=511}$$

implying that the assumption of a uniform distribution with 95% confidence must be rejected.

The comparison χ^2 test statistic shows,

$$\chi^2_{Lenna} = 5.05 \times 10^6 \ggg \chi^2_{\alpha=0.05,df=511}$$

which implies that the encrypted image cannot statistically be correlated to the *Lenna* input image.

Table 8: Image Metrics for the Cellular Automata Encrypted Image

Metric	**Value**
Adjacent Horizontal Pixel Correlation, R_x	-0.038
Adjacent Horizontal Coefficient of Determination, R_x^2	0.001
Adjacent Vertical Pixel Correlation, R_y	$\approx$0.00
Adjacent Vertical Coefficient of Determination, R_y^2	0.00
Shannon Entropy, h	7.985
Average Pixel Value, E	126
Pixel Standard Deviation, σ	71
$\chi^2_{uniform}$	5,450
$\chi^2_{uniform}/\chi^2_{max}$	4.07×10^{-5}
χ^2_{Lenna}	5.05×10^6
$\chi^2_{Lenna}/\chi^2_{max}$	3.77×10^{-2}
Average Encryption Time, t_{encr}	446 ms $\pm$ 5 ms

The Fluid-Dynamics Inspired Technique

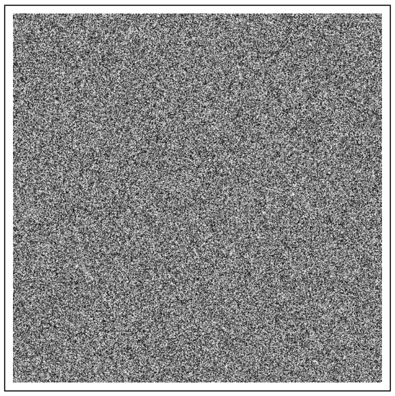
This section introduces the results of the author's proposed fluid-dynamics inspired technique. The key data used for the analyses of the proposed technique are shown in Table 9. The following sections have analyses using the randomly generated secret image (Figure 15), a pure black secret image, and a pure white secret image.

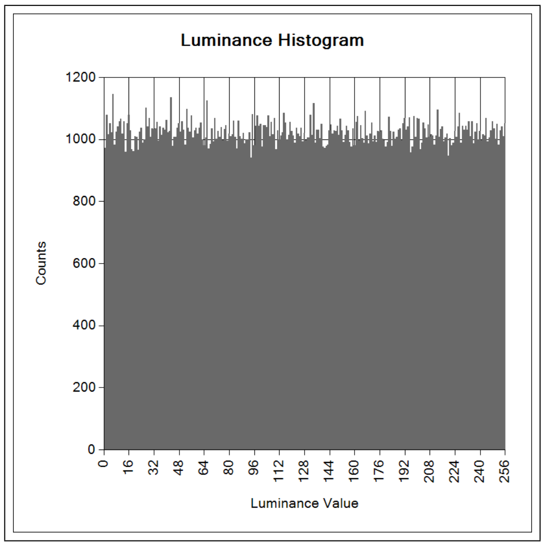
Table 9: Secret Key for the Fluid Dynamics Inspired Technique

Mass Flow Rate Initializer, $(r,x_0)_{\dot{m}}$	$(3.717, 0.55)$
Heat Flux Initializer, $(r,x_0)_{\dot{q}''}$	$(3.8222, 0.18)$
Turbulent Kinetic Energy, k_0	199
Turbulent Kinetic Energy Decay Rate, ε	17
Time Frame Slice, t_{frame}	32

The Fluid-Dynamics Inspired Technique with a Randomly Generated Secret Image

Figure 29A shows the input *Lenna* image encrypted with the proposed fluid dynamics inspired approach and the randomly generated secret image shown in Figure 15. It is apparent by observing Figure 29B that the luminance histogram is completely normalized. The horizontal and vertical adjacency correlations of Figure 30 show a uniform dissipation of adjacent pixel intensities. In addition, the Fourier spectra (Figure 31) appears to be white noise—not leaking any underlying image structure as the CA approach did.


(A) Encrypted *Lenna*



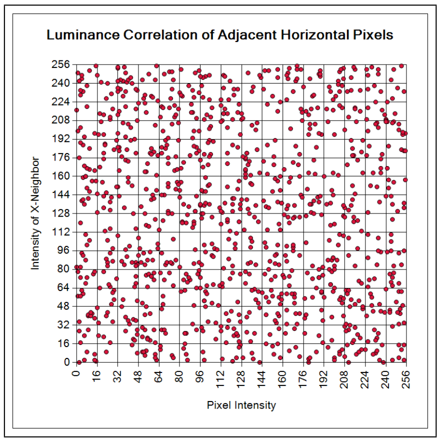
(B) Histogram

Figure 29: *Lenna* as Encrypted Using the Proposed Fluid Dynamics Inspired Technique and the Encrypted Image's Histogram

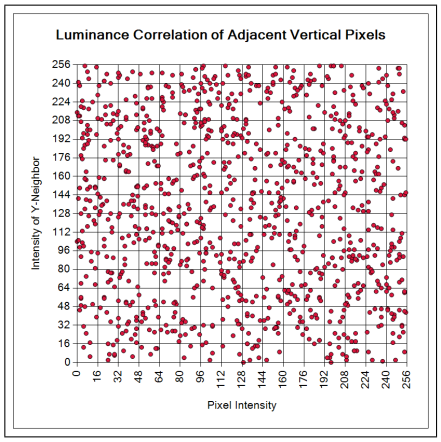
The calculated information entropy for the fluid dynamics inspired method with a randomly generated secret image was $h = 7.999$ which agrees well with the entropy calculated by the bio-inspired approach. Observe from Table 10 that the uniform distribution test statistic

$$\chi^2_{uniform} = 280 < \chi^2_{\alpha=0.05,df=511}$$

which means that the encrypted image's histogram is statistically indistinguishable from a uniform distribution with a 95% confidence interval. This suggests that an adversary would



(A) Horizontal Correlation



(B) Vertical Correlation

Figure 30: The Adjacent Pixel Correlations of Encrypted *Lenna* Test Image Encrypted via the Fluid Dynamics Inspired Method

not have a probabilistic advantage in determining the contents of any randomly selected set of pixels. For example, if this scheme were used in holographic filesystems, a adversary would have difficulty in selecting a randomly chosen set of bytes and be able to determine with confidence the contents of the selected byte array. Also observe that

$$\chi^2_{Lenna} = 7.62 \times 10^6 \ggg \chi^2_{\alpha=0.05,df=511}$$

which forces the adversary to infer that the histogram of the encrypted image cannot be statistically correlated to the plaintext image's histogram. The full list of metrics for the encrypted image output of this technique is presented in Table 10.

There exists a sensitivity to the selection of the secret key image for the XOR process, but the sensitivity is less than that of the bio-inspired approach as is shown in the following sections using pure black and pure white secret images.

The Fluid-Dynamics Inspired Technique with a Pure Black Secret Image

The proposed fluid dynamics inspired approach has some sensitivity to the choice of secret image. Figure 32A shows the input *Lenna* image encrypted with a pure black

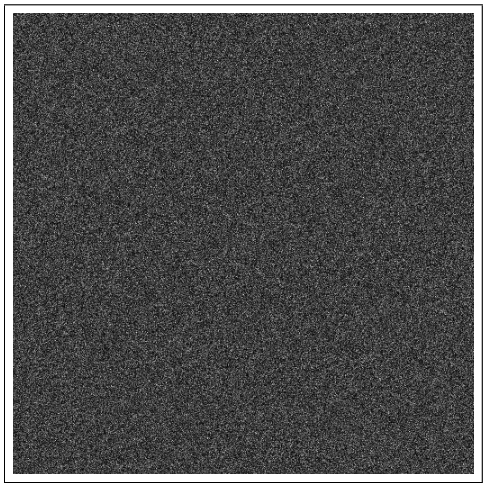
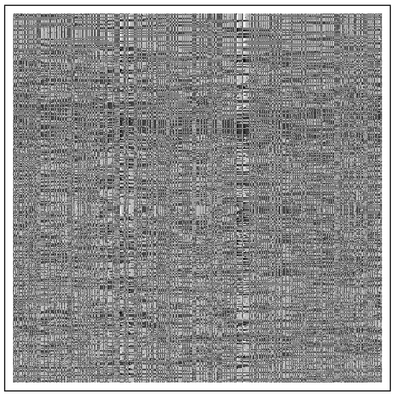


Figure 31: The Fourier spectra of the Fluid Dynamics Inspired Encrypted Image

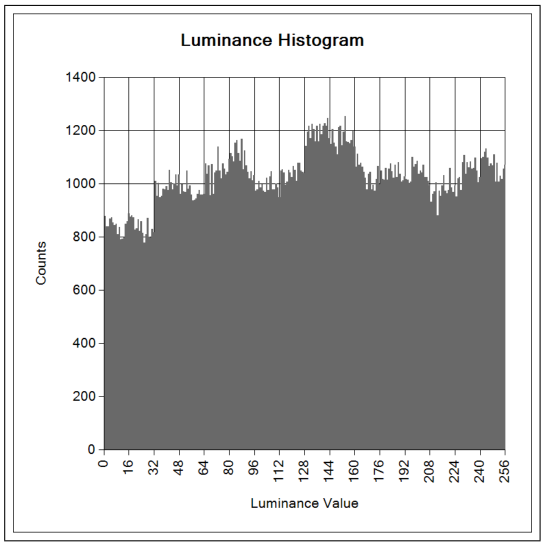
Table 10: Image Metrics for the Fluid Dynamics Inspired Encrypted Image with a Randomly Generated Secret Image

Metric	**Value**
Adjacent Horizontal Pixel Correlation, R_x	0.020
Adjacent Horizontal Coefficient of Determination, R_x^2	$\approx$0.00
Adjacent Vertical Pixel Correlation, R_y	-0.018
Adjacent Vertical Coefficient of Determination, R_y^2	$\approx$0.00
Shannon Entropy, h	7.999
Average Pixel Value, E	127
Pixel Standard Deviation, σ	74
$\chi^2_{uniform}$	280
$\chi^2_{uniform}/\chi^2_{max}$	2.09×10^{-6}
χ^2_{Lenna}	7.62×10^{6}
$\chi^2_{Lenna}/\chi^2_{max}$	5.69×10^{-2}
Average Encryption Time, t_{encr}	860 ms $\pm$ 6 ms

secret image where every pixel has the RGB value 0x000000. This would be equivalent to only outputting the intermediate image produced by the permutation and transformation functions (refer to Figure 9). Unlike the bio-inspired approach with a pure black secret image, Figure 32B shows that the fluid-dynamics inspired approach does not produce a discernable link to the plaintext image's histogram; however, the horizontal and vertical adjacency correlations of Figure 33 show some clustering. The "diamond shaped" pattern of the horizonal adjacency correlation is a result of the right circular rotation of the scanlines. These features do suggest image data that exists in the underlying permutations. In addition, the Fourier spectra (Figure 34) has three vertical “crests”. This type of structure is due to the prominence of the vertical columns in the encrypted image.

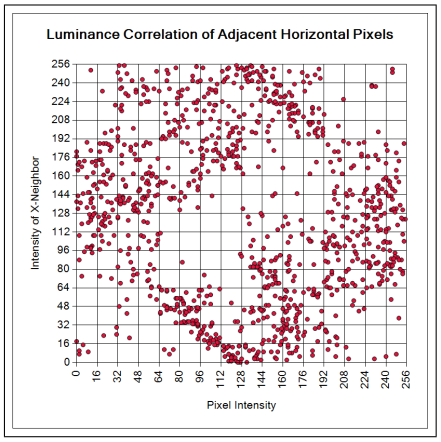


(A) Encrypted *Lenna*

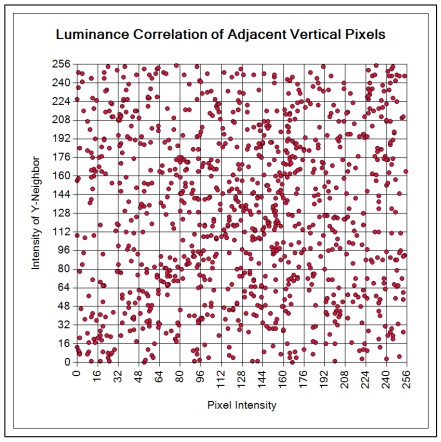


(B) Histogram

Figure 32: *Lenna* as Encrypted Using the Proposed Fluid Dynamics Inspired Technique with a Pure Black Secret Image and the Encrypted Image’s Histogram



(A) Horizontal Correlation



(B) Vertical Correlation

Figure 33: The Adjacent Pixel Correlations of Encrypted *Lenna* Test Image Encrypted via the Fluid Dynamics Inspired Method using a Pure Black Secret Image

The calculated information entropy for the fluid dynamics inspired method with the minimum entropy secret image was $h = 7.993$ which is notably greater than the calculated entropy $h = 7.498$ of the bio-inspired technique with the same secret image (recall that the differences are logarithmic). Compared to the bio-inspired technique, the proposed fluid-dynamics inspired technique yields a uniform distribution test statistic that is markedly nearer the critical χ^2 value; although

$$\chi^2_{uniform} = 2,430 > \chi^2_{\alpha=0.05,df=511}$$

so the distribution cannot be assumed uniform within a 95% confidence interval. Additionally,

$$\chi^2_{Lenna} = 8.34 \times 10^6 \ggg \chi^2_{\alpha=0.05,df=511}$$

which is nearly 5,600 times the χ^2_{Lenna} statistic of the bio-inspired approach with the pure black (weak) secret image.

The full list of metrics for the encrypted image output of this technique is presented in Table 11.

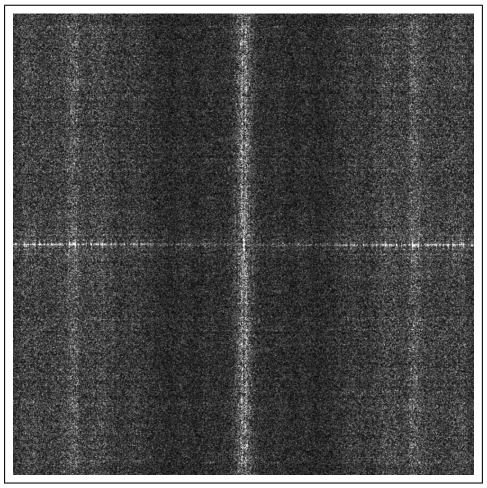


Figure 34: The Fourier spectra of the Fluid Dynamics Inspired Encrypted Image using the Pure Black Secret Image

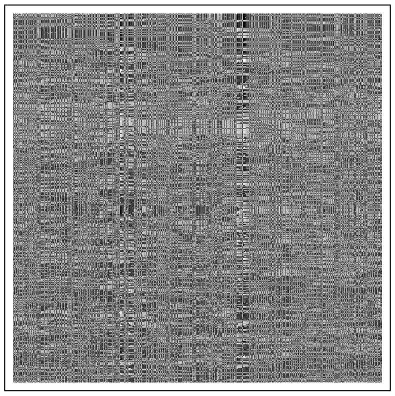
Table 11: Image Metrics for the Fluid Dynamics Inspired Encrypted Image using a Pure Black Secret Image

Metric	**Value**
Adjacent Horizontal Pixel Correlation, R_x	-0.124
Adjacent Horizontal Coefficient of Determination, R_x^2	0.015
Adjacent Vertical Pixel Correlation, R_y	0.056
Adjacent Vertical Coefficient of Determination, R_y^2	0.003
Shannon Entropy, h	7.993
Average Pixel Value, E	130
Pixel Standard Deviation, σ	72
$\chi^2_{uniform}$	2,430
$\chi^2_{uniform}/\chi^2_{max}$	1.81×10^{-5}
χ^2_{Lenna}	8.34×10^{6}
$\chi^2_{Lenna}/\chi^2_{max}$	6.23×10^{-2}
Average Encryption Time, t_{encr}	858 ms $\pm$ 4 ms

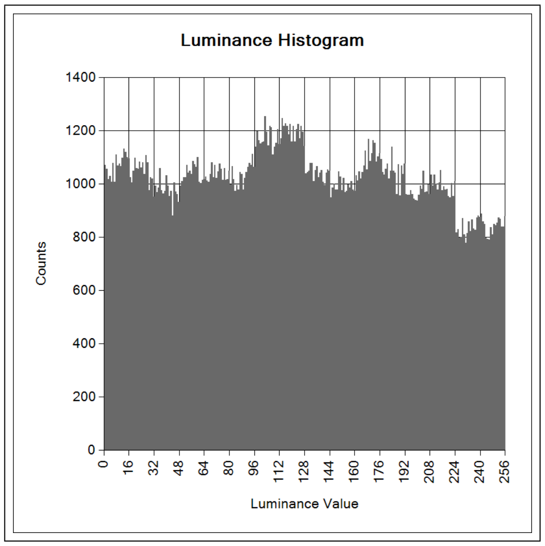
The Fluid-Dynamics Inspired Technique with a Pure White Secret Image

Figure 35A shows the input *Lenna* image encrypted with a pure white secret image where every pixel has the RGB value 0xffffff. This would be equivalent to outputting the color-inverted (complement, photo-negative, etc.) intermediate permutation image. It may be observed that the histogram of this image (Figure 35B) is the mirror image of Figure 32B which does not have a discernable link to the *Lenna* histogram. The horizontal and vertical adjacency correlations of Figure 36 show clustering similar to that discussed in the previous section and for the same reasons presented there. The Fourier spectra given in Figure 37 has the same “crests” as that given by Figure 34. The reasons why the same spectra can be produced for these two different encrypted images is that Figures 37 and 34 are the Fourier *magnitude* plots. The Fourier phase images carry the differences between the two encrypted images.

As expected from the previous section, the calculated information entropy for the fluid dynamics inspired method with the pure white secret image was $h = 7.993$ (the same as the entropy using the pure black secret image). The calculated uniform distribution χ^2 test

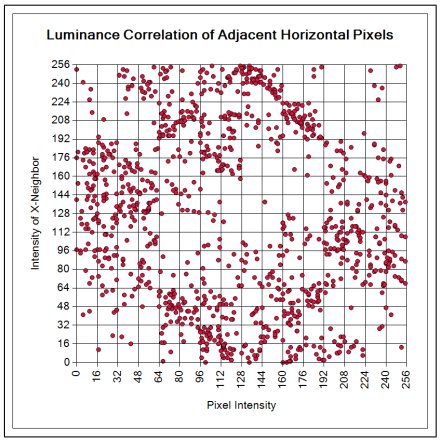


(A) Encrypted *Lenna*

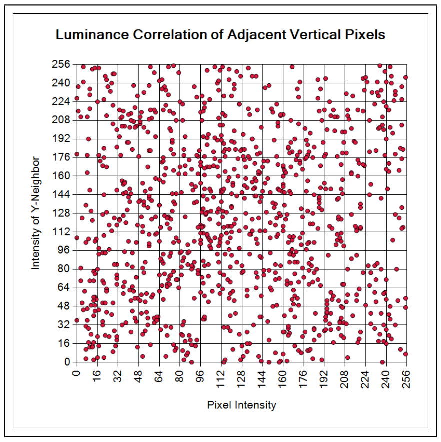


(B) Histogram

Figure 35: *Lenna* as Encrypted Using the Proposed Fluid Dynamics Inspired Technique with a Pure White Secret Image and the Encrypted Image's Histogram



(A) Horizontal Correlation



(B) Vertical Correlation

Figure 36: The Adjacent Pixel Correlations of Encrypted *Lenna* Test Image Encrypted via the Fluid Dynamics Inspired Method using a Pure White Secret Image

statistic for this encrypted image is

$$\chi^2_{uniform} = 2{,}430 > \chi^2_{\alpha=0.05,df=511}$$

meaning that the assumption of a uniform distribution with 95% confidence must be rejected.

The comparison χ^2 test statistic shows,

$$\chi^2_{Lenna} = 5.15 \times 10^6 \ggg \chi^2_{\alpha=0.05,df=511}$$

which implies that the encrypted image cannot statistically be correlated to the *Lenna* input image. Additionally, this value is 35 times greater than the χ^2_{Lenna} test statistic for the bio-inspired technique with a pure white secret image.

The full list of metrics for the encrypted image output of this technique is presented in Table 12.

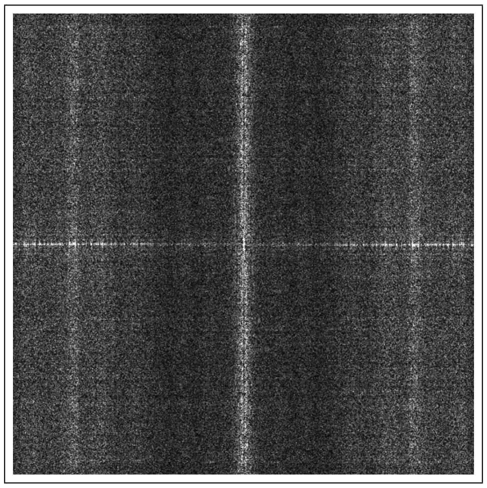


Figure 37: The Fourier spectra of the Fluid Dynamics Inspired Encrypted Image using the Pure White Secret Image

Tests with a QR Code

In this section a QR Code will be used to briefly contrast the image encryption methods using a low-entropy plaintext image. A QR Code (also called a Quick Response Code) is a two-dimensional, machine-readable code that can store nearly 2.9 kilobytes of binary information and has error correction features[5]. Figure 38A shows the 200 pixel $\times$ 200 pixel QR Code that will be used in this section. The only luminous values of a typical QR Code are 0 (black) and 255 (white); because of this, the histograms and correlation maps are not shown. However, the full list of image metrics for the QR Code is given

Table 12: Image Metrics for the Fluid Dynamics Inspired Encrypted Image using a Pure White Secret Image

Metric	**Value**
Adjacent Horizontal Pixel Correlation, R_x	-0.132
Adjacent Horizontal Coefficient of Determination, R_x^2	0.017
Adjacent Vertical Pixel Correlation, R_y	0.024
Adjacent Vertical Coefficient of Determination, R_y^2	$\approx$0.001
Shannon Entropy, h	7.993
Average Pixel Value, E	124
Pixel Standard Deviation, σ	72
$\chi^2_{uniform}$	2,430
$\chi^2_{uniform}/\chi^2_{max}$	1.81×10^{-5}
χ^2_{Lenna}	5.15×10^{6}
$\chi^2_{Lenna}/\chi^2_{max}$	3.84×10^{-2}
Average Encryption Time, t_{encr}	864 ms $\pm$ 6 ms

in Table 13. Additionally, Figure 38B shows the secret image that is used for the XOR steps of the bio-inspired and fluid-dynamics methods. For all three methods the key values given in Tables 3, 7, and 9 were used. The encrypted images as produced by the three

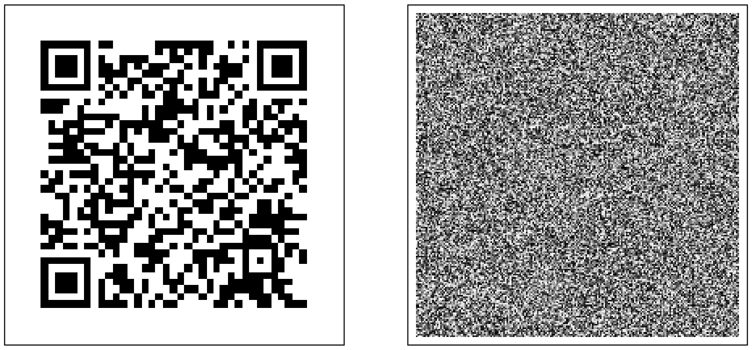

(A) The QR-Code (B) The Secret Image

Figure 38: The QR-Code used as the Plaintext Image and the Secret Image for the Bio-Inspired and Fluid-Dynamics-Inspired Techniques

different techniques are shown in Figure 39 and though the images may not appear to look different, the calculated metrics show vastly different results. As may be observed by comparing Tables 14, 15, and 16, the proposed fluid-dynamics-inspired approach greatly outperforms the cellular automata approach and has better statistics than the bio-inspired

Table 13: Image Metrics of the QR Code in Figure 38A

Metric	Value
Adjacent Horizontal Pixel Correlation, R_x	0.723
Adjacent Horizontal Coefficient of Determination, R_x^2	0.523
Adjacent Vertical Pixel Correlation, R_y	0.785
Adjacent Vertical Coefficient of Determination, R_y^2	0.616
Shannon Entropy, h	0.917
Average Pixel Value, E	170
Pixel Standard Deviation, σ	120
$\chi^2_{uniform}$	5.66×10^6
$\chi^2_{uniform}/\chi^2_{max}$	0.711
χ^2_{QRCode}	0 (by definition)
$\chi^2_{QRCode}/\chi^2_{max}$	0 (by definition)

approach in every category (recall that it is ideal for $\chi^2_{uniform}$ to be minimized and χ^2_{QRCode} to be maximized). These data show the efficacy of the proposed image encryption algorithm over both the cellular automata technique and the bio-inspired approach.

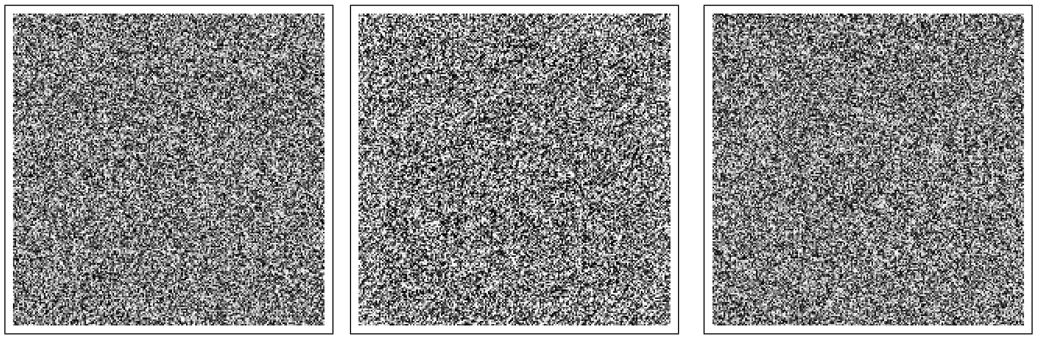

(A) Bio-Inspired (B) Cellular Automata (C) Fluid-Dynamics-Inspired

Figure 39: The Encrypted QR Code Using the Three Different Techniques

It should be noted that the huge value of the χ^2 statistic for the cellular automata technique in Table 15 is due to both the low entropy of the input plaintext image and the maximum of eight states in each cyclic state attractor (such as the one shown in Figure 6). In this case, the luminous intensity frequencies were isolated to the values of the state attractor with zero counts for nearly all other values. This same effect is responsible for the low Shannon entropy value $h = 5.854$ meaning that a palette with only $2^{5.854} \approx 59$ luminous values could be used to encode the entire encrypted image. These data lead one to conclude that the elementary cellular automata approach is ill-suited for low entropy plaintext images.

Table 14: Image Metrics of the QR Code Encrypted with the Bio-Inspired Method

Metric	**Value**
Adjacent Horizontal Pixel Correlation, R_x	0.065
Adjacent Horizontal Coefficient of Determination, R_x^2	0.004
Adjacent Vertical Pixel Correlation, R_y	-0.057
Adjacent Vertical Coefficient of Determination, R_y^2	0.003
Shannon Entropy, h	7.987
Average Pixel Value, E	126
Pixel Standard Deviation, σ	74
$\chi^2_{uniform}$	613
$\chi^2_{uniform}/\chi^2_{max}$	7.70×10^{-5}
χ^2_{QRCode}	3.93×10^4
$\chi^2_{QRCode}/\chi^2_{max}$	4.94×10^{-3}
Average Encryption Time, t_{encr}	319 ms $\pm$ 4.5 ms

Table 15: Image Metrics of the QR Code Encrypted with the Cellular Automata Method

Metric	**Value**
Adjacent Horizontal Pixel Correlation, R_x	0.045
Adjacent Horizontal Coefficient of Determination, R_x^2	0.002
Adjacent Vertical Pixel Correlation, R_y	-0.019
Adjacent Vertical Coefficient of Determination, R_y^2	$\approx$0.000
Shannon Entropy, h	5.584
Average Pixel Value, E	129
Pixel Standard Deviation, σ	83
$\chi^2_{uniform}$	2.34×10^5
$\chi^2_{uniform}/\chi^2_{max}$	0.029
χ^2_{QRCode}	3.65×10^4
$\chi^2_{QRCode}/\chi^2_{max}$	4.59×10^{-3}
Average Encryption Time, t_{encr}	80 ms $\pm$ 4.5 ms

Figure 40 shows a comparison of the three methods using the arbitrary metric:

$$\varepsilon \equiv \frac{2^{h-8}}{t_{proc} \times \log_{10} \chi^2} \tag{4.1}$$

Table 16: Image Metrics of the QR Code Encrypted with the Fluid-Dynamics Inspired Method

Metric	**Value**
Adjacent Horizontal Pixel Correlation, R_x	0.018
Adjacent Horizontal Coefficient of Determination, R_x^2	$\approx$0.000
Adjacent Vertical Pixel Correlation, R_y	$\approx$0.000
Adjacent Vertical Coefficient of Determination, R_y^2	$\approx$0.000
Shannon Entropy, h	7.995
Average Pixel Value, E	126
Pixel Standard Deviation, σ	74
$\chi^2_{uniform}$	281
$\chi^2_{uniform}/\chi^2_{max}$	3.53×10^{-5}
χ^2_{QRCode}	3.95×10^{4}
$\chi^2_{QRCode}/\chi^2_{max}$	4.96×10^{-3}
Average Encryption Time, t_{encr}	199 ms $\pm$ 3.4 ms

as the plotted statistic. This metric was chosen to highlight the benefit of larger information entropy values, smaller processing time, and statistically uniform luminance distributions. Using this as a comparison metric, the proposed fluid-dynamics method is a better image encryption technique because of the high amounts of information entropy, lower processing time, and relatively small $\chi^2_{uniform}$ test statistic. In all test cases, the cellular automata technique had the least processing time, but was encumbered by low information entropy values and very large $\chi^2_{uniform}$ values. Conversely, the bio-inspired technique generated large entropy values with decent $\chi^2_{uniform}$ values, but was overshadowed by lengthy processing time. The fluid-dynamics-inspired method almost always produces entropy values that were greater than the bio-inspired method with shorter processing times and test statistics that are within the 95% confidence interval of a uniform probability distribution.

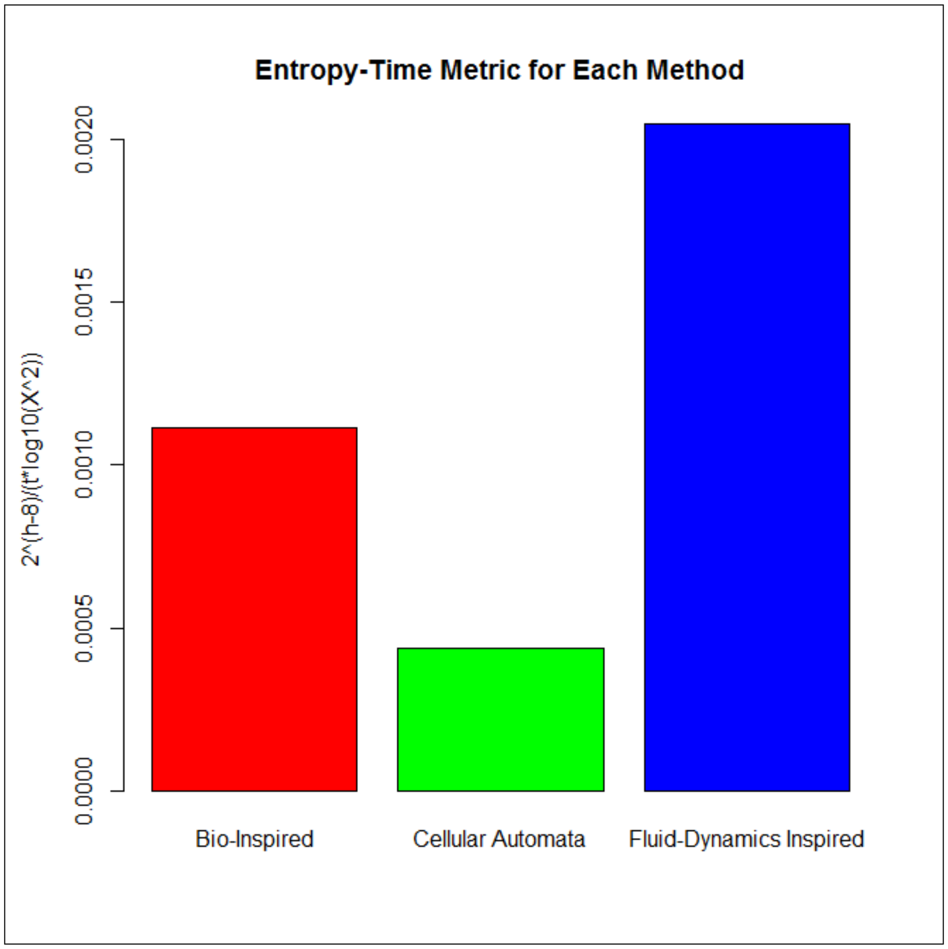


Figure 40: A Comparison of the Three Methods Using Equation 4.1

CHAPTER V. DISCUSSION

This research effort implemented two established image encryption techniques: a bio-inspired technique published by Al-Utaibi and El-Alfy[2] and an Elementary Cellular Automata technique published by Jin[9]. Additionally, a novel technique was developed based on analogies from the field of Computational Fluid Dynamics to implement a new form of image encryption.

The implementations of the established methods agree with those found in the literature. The results presented for the fluid dynamics inspired method show that it is better in both time and entropy/confusion performance to that of the bio-inspired technique and vastly better than the cellular automata approach. Additionally, the proposed fluid-dynamics inspired method does not present the weaknesses of the cellular automata approach where intensity values of the plaintext image “bleeds over” into the encrypted image. Though the proposed fluid-dynamics-based approach has a smaller keyspace than the bio-inspired approach, its key diffusion and encryption speed make it a viable technique for the secure storage and transmission of two-dimensional matrix and image data.

There are four areas for future research regarding the proposed method:

1. Altering the system to represent a 2D flowfield with flow disturbances representing the key data, and

2. The efficacy of using the fluid dynamics inspired approach for higher dimensional holographic image encryption (i.e. operations on rank *n* tensors rather than matrices/images)

3. Using Peak Signal-to-Noise Ratio (PSNR) measurements on the decrypted images to ensure decrypted image fidelity

4. Explore the use of Random Matrix Theory (RMT) in quantifying encryption robustness against brute force algorithms

each of these venues present challenges that are not discussed here, but can yield exciting developments for the future of both image encryption and holographic storage full disk encryption.

BIBLIOGRAPHY

[1] “AForge.NET Framework.” [Online]. Available: http://www.aforgenet.com/framework

[2] K. A. Al-Utaibi and E.-S. M. El-Alfy, “A bio-inspired image encryption algorithm based on chaotic maps,” in *IEEE Congress on Evolutionary Computation*. IEEE, jul 2010, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5586463

[3] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 2nd ed., C. Long, Ed. Indianapolis, IN, USA: Wiley Publishing, Inc., 2008.

[4] M. Baptista, “Cryptography with chaos,” *Physics Letters A*, vol. 240, no. 1-2, pp. 50–54, mar 1998. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0375960198000863

[5] Denso ADC, “QR Code Essentials,” 2011. [Online]. Available: http://www.nacs.org/LinkClick.aspx?fileticket=D1FpVAvvJuo%3D&tabid=1426&mid=4802

[6] R. Fisher, S. Perkins, A. Walker, and E. Wolfart, “The Hypermedia Image Processing Reference (HIPR).” [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2

[7] M. Gesmann, “Logistic map: Feigenbaum diagram in R,” 2012. [Online]. Available: http://www.magesblog.com/2012/03/logistic-map-feigenbaum-diagram.html

[8] M. E. Hellman, “A cryptanalytic time-memory trade-off,” in *IEEE Transactions on Information Theory*, vol. IT-26, no. 4. IEEE, July 1980. [Online]. Available: http://www-ee.stanford.edu/~hellman/publications/36.pdf

[9] J. Jin, “An image encryption based on elementary cellular automata,” *Optics and Lasers in Engineering*, vol. 50, no. 12, pp. 1836–1843, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.optlaseng.2012.06.002

[10] B. Launder and D. Spalding, “The numerical computation of turbulent flows,” *Computer Methods in Applied Mechanics and Engineering*, vol. 3, no. 2, pp. 269–289, mar 1974. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/0045782574900292

[11] T.-Y. Li and J. A. Yorke, “Period three implies chaos,” *The American Mathematical Monthly*, vol. 82, no. 10, pp. 985–992, dec 1975. [Online]. Available: http://www.jstor.org/stable/2318254

[12] S. Liu, J. Sun, Z. Xu, and J. Liu, “Analysis on an image encryption algorithm,” *2008 International Workshop on Education Technology and Training and 2008 International Workshop on Geoscience and Remote Sensing, Vol 1, Proceedings*, pp. 803–806, 2009.

[13] E. N. Lorenz, “Deterministic nonperiodic flow,” *Journal of the Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, 1963.

[14] J. Lyons, “English Letter Frequencies,” 2012. [Online]. Available: http://practicalcryptography.com/cryptanalysis/letter-frequencies-various-languages/english-letter-frequencies/

[15] B. B. Mandelbrot, *The Fractal Geometry of Nature*. New York: W. H. Freeman and Co., 1982.

[16] J. McClave and T. Sincich, *A First Course in Statistics*, 10th ed., D. Lynch, Ed. Upper Saddle River, NJ, USA: Pearson, 2009.

[17] D. Munson, “A note on Lena,” in *IEEE Transactions on Image Processing*, vol. 5, no. 1. IEEE, jan 1996. [Online]. Available: http://www.cs.cmu.edu/~chuck/lennapg/editor.html

[18] L. M. Pecora and T. L. Carroll, “Synchronization in chaotic systems,” *Physical Review Letters*, vol. 64, no. 8, pp. 821–824, feb 1990. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.64.821

[19] C. E. Shannon, “Communication theory of secrecy systems,” *Bell Systems Technical Journal*, vol. 28, pp. 656–715, 1949.

[20] E. Weisstein, “Logistic Map,” 2016. [Online]. Available: http://mathworld.wolfram.com/LogisticMap.html

[21] S. Wolfram, *A New Kind of Science*. Champaigne, IL: Wolfram Media, 2002.

APPENDICES

APPENDIX A.
LOGISTIC MAP BIFURCATION DIAGRAM USING R

```
# Logistic Map Bifurcation Diagram
# Author: Gary Hammock
# Date: 2016-07-14
# Ref:
# Gesmann, Markus. "Logistic map: Feigenbaum diagram in R,"
#   "Mage's Blog." 2012-03-17.
#   URL:
    http://www.magesblog.com/2012/03/logistic-map-feigenbaum-diagram.html
#   Last Access: 2016-07-14.

# @param r Bifurcation parameter.
# @param x0 Initial seed value.
# @param iter Number of iterations.
# @param req Number of iteration points to be returned.
LogisticMap <- function(r, x0, iter, req) {
    x <- 1:iter # Needed to size the x-array.
    x[1] <- x0
    for(i in c(1:(iter - 1))) {
        x[i + 1] <- r * x[i] * (1 - x[i])
    }

    # Return the last req number of iteration points
    output <- x[c((iter - req):iter)]
    return(output)
}

# Compile the function to byte code for a speed boost
library(compiler)
LogisticMap <- cmpfun(LogisticMap)

# Parametric inputs
start.r <- 1.0
end.r <- 4.0
iterations <- 2000
returnPts <- 500
start.x <- 0.1

# Generate the input array.
r_array <- seq(start.r, end.r, by = 0.001)

# Calculate the distribution of x's from the given parameters.
x_n <- sapply(r_array,
              LogisticMap,
             x=start.x,
             iter=iterations,
             req=returnPts)
x_n <- as.vector(x_n)

# Replicate the r_array elements for the returned points.
r <- sort(rep(r_array, (returnPts + 1)))

# Use the alpha channel to help show overlap.
# i.e. darker means more points
color <- rgb(0, 0, 0, 0.1)

# Plot the bifurcation diagram.
plot(r, x_n, pch=".",
     main="Bifurcation Diagram of the Logistic Map",
     col=color,
     axes=FALSE)
axis(1, seq(start.r, end.r, by=0.1))
axis(2, seq(0, 1, by=0.1))
box()

# Plot a line showing the generally accepted start
# of deterministic chaos at r=3.569955672.
abline(v=3.569955672, col="red", lty="longdash", lwd=2)
```

APPENDIX B
FLUID-DYNAMICS INSPIRED KEY CLASS

```
   #region filename
/// FluidDynamicsKey.cs
#endregion

#region description
/// <summary>
/// This file contains the implementation of the fluid-dynamic-inspired
/// image encryption key class.
///
/// ///
/// This file may be found at:
/// https://github.com/ghammock/ImageEncryptionSuite/
/// </summary>
#endregion

#region copyright
/// Copyright (c) Gary Hammock, 2016

/// Permission is hereby granted, free of charge, to any person obtaining a
/// copy of this software and associated documentation files (the
    "Software"),
/// to deal in the Software without restriction, including without
    limitation
/// the rights to use, copy, modify, merge, publish, distribute,
    sublicense,
/// and/or sell copies of the Software, and to permit persons to whom the
/// Software is furnished to do so, subject to the following conditions:
///
/// The above copyright notice and this permission notice shall be included
/// in all copies or substantial portions of the Software.
///
/// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
/// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
/// MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
/// IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
/// CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
/// TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
/// SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#endregion

using System;
```

This source code listing and any updates may be found at the author's GitHub® repository at: `https://github.com/ghammock/ImageEncryptionSuite/`.

```
using System.Drawing;

namespace ImageEncryption.FluidDynamicsInspired
{
    /// <summary>
    /// This class stores and provides accessors/mutators for the
    /// fluid-dynamics inspired image encryption technique.
    /// </summary>
    public class FluidDynamicsKey
    {
        /// <summary>
        /// This is the secret image that will be XOR-ed
        /// with the intermediate image.
        /// </summary>
        public Bitmap SecretImage;

        /***************************************************
        *                Private Fields                    *
        ***************************************************/

        // Growth rate coefficient for the mass flow rate logistic map.
        private double r_mdot;

        // Growth rate coefficient for the heat transfer logistic map.
        private double r_qdot;

        private byte k0;    // The initial "turbulence energy".
        private byte epsilon; // The "dissipation rate" for the decay
            function.
        private byte tFrame; // The "time snapshot" to read K.

        private double mDotKey; // Feeds the "mass flow rate" key expansion.
        private double qDotKey; // Feeds the "heat flux" key expansion.

        /***************************************************
        *                 Constructors                     *
        ***************************************************/

        /// <summary>
        /// Empty constructor for manual setup.
        /// </summary>
        public FluidDynamicsKey()
        { }

        /// <summary>
        /// Initializer constructor.
        /// </summary>
        /// <param name="r_mdot">
        /// The growth rate parameter for the mass flow rate logistic map.
        /// </param>
        /// <param name="massFlowRateInitializer">
        /// Initializes the "mass flow rate" key expansion.
        /// </param>
        /// <param name="r_qdot">
        /// Growth rate coefficient for the heat transfer logistic map.
        /// </param>
        /// <param name="heatTransferInitializer">
        /// Initializers the "heat flux" key expansion.
        /// </param>
        /// <param name="k0">
        /// The initial "turbulent kinetic energy".
        /// </param>
        /// <param name="epsilon">
        /// The "turbulent kinetic energy dissipation rate" for
        /// the decay function.
        /// </param>
        /// <param name="tframe">
        /// The "time snapshot" to read K.
        /// </param>
        /// <param name="secretImage">
        /// This is the secret image that will be XOR-ed
        /// with the intermediate image.
        /// </param>
        public FluidDynamicsKey(
            double r_mdot, double massFlowRateInitializer,
            double r_qdot, double heatTransferInitializer,
            byte k0, byte epsilon,
            byte tframe, Bitmap secretImage)
        {
            R_mdot = r_mdot;
            R_qdot = r_qdot;
            mDotKey = massFlowRateInitializer;
            qDotKey = heatTransferInitializer;
            K0 = k0;
            Epsilon = epsilon;
            TFrame = tframe;
            SecretImage = (Bitmap)secretImage.Clone();
        }

        /***************************************************
        *                  Properties                      *
        ***************************************************/
```

```
/// <summary>
/// The growth rate coefficient (3.569955672, 4]
/// of the row mutation logistic map.
/// </summary>
public Double R_mdot
{
    get { return r_mdot; }
    set
    {
        if (ValidDomain(value))
            r_mdot = value;
        else
            throw new System.ArgumentOutOfRangeException();
    }
}

/// <summary>
/// The growth rate coefficient (3.569955672, 4]
/// of the column mutation logistic map.
/// </summary>
public Double R_qdot
{
    get { return r_qdot; }
    set
    {
        if (ValidDomain(value))
            r_qdot = value;
        else
            throw new System.ArgumentOutOfRangeException();
    }
}

/// <summary>
/// The initializer for the "mass flow rate" key expansion.
/// </summary>
public Double MassFlowRateInitializer
{
    get { return mDotKey; }
    set
    {
        if (mDotKey < 0.0 || mDotKey > 1.0)
            throw new System.ArgumentOutOfRangeException();
        else
            mDotKey = value;
    }
```

```
}

/// <summary>
/// The initializer for the "heat flux" key expansion.
/// </summary>
public Double HeatTransferInitializer
{
    get { return qDotKey; }
    set
    {
        if (qDotKey < 0.0 || qDotKey > 1.0)
            throw new System.ArgumentOutOfRangeException();
        else
            qDotKey = value;
    }
}

/// <summary>
/// The initial "turbulent kinetic energy".
/// </summary>
public byte K0
{
    get { return k0; }
    set
    {
        if (value < 0 || value > 255)
            throw new System.ArgumentOutOfRangeException();
        else
            k0 = value;
    }
}

/// <summary>
/// The "turbulent kinetic energy dissipation rate" for
/// the decay function.
/// </summary>
public byte Epsilon
{
    get { return epsilon; }
    set
    {
        if (value < 0 || value > 255)
            throw new System.ArgumentOutOfRangeException();
        else if (k0 < value)
        {
            throw new System.ArgumentException(
                "Epsilon must be less than K0.");
        }
        else
            epsilon = value;
    }
}

/// <summary>
/// The "time snapshot" to read K.
/// </summary>
public byte TFrame
{
    get { return tFrame; }
    set
    {
        if (value < 0 || value > 255)
            throw new System.ArgumentOutOfRangeException();
        else
            tFrame = value;
    }
}

/***************************************************
*               Public Methods                  *
***************************************************/

/// <summary>
/// Checks that all of the stored values indicate a
/// valid FluidDynamicsKey object.
/// </summary>
/// <returns>
/// True if the values represent a valid key; false otherwise.
/// </returns>
public bool IsValidKey ()
{
    if (epsilon < k0 && ValidDomain(r_mdot) && ValidDomain(r_qdot))
        return true;
    else
        return false;
}

/// <summary>
/// Creates a clone of this FluidDynamicsKey object.
/// </summary>
/// <returns>
/// A new FluidDynamicsKey object whose values are
/// copied from the calling object.
/// </returns>
public FluidDynamicsKey Clone()
{
    FluidDynamicsKey newKey = new FluidDynamicsKey(
        this.r_mdot, this.mDotKey, this.r_qdot, this.qDotKey,
        this.k0, this.epsilon, this.tFrame, this.SecretImage);

    return newKey;
}

/// <summary>
/// Generates a new random key that is within the valid keyspace.
/// </summary>
/// <returns>
/// A new FluidDyanmicsKey object whose values are within the
/// correct logistic map keyspace.
/// </returns>
public static FluidDynamicsKey Generate()
{
    FluidDynamicsKey newKey = new FluidDynamicsKey();
    return newKey;
}

/**************************************************
*                Private Methods              *
**************************************************/

/// <summary>
/// Determines if a given growth rate coefficient is in the proper
/// domain for deterministic chaos to occur.
/// </summary>
/// <param name="mu">
/// The growth rate coefficient to check.
/// </param>
/// <returns>
/// True if mu is in (3.569955672, 4]; false otherwise.
/// </returns>
private bool ValidDomain (double mu)
{
    if (mu > 3.569955672 && mu <= 4.0)
        return true;
    else
        return false;
}
```

```
    }  // End class Fluid Dynamics Key.
}
```

APPENDIX C
FLUID-DYNAMICS INSPIRED ENCRYPTER CLASS

```
  #region filename
/// FluidDynamicsEncryptor.cs
#endregion

#region description
/// <summary>
/// This file contains the implementation of the fluid-dynamic-inspired
/// image encryption key class.
///
/// This file may be found at:
/// https://github.com/ghammock/ImageEncryptionSuite/
/// </summary>
#endregion

#region copyright
/// Copyright (c) Gary Hammock, 2016

/// Permission is hereby granted, free of charge, to any person obtaining a
/// copy of this software and associated documentation files (the
    "Software"),
/// to deal in the Software without restriction, including without
    limitation
/// the rights to use, copy, modify, merge, publish, distribute,
    sublicense,
/// and/or sell copies of the Software, and to permit persons to whom the
/// Software is furnished to do so, subject to the following conditions:
///
/// The above copyright notice and this permission notice shall be included
/// in all copies or substantial portions of the Software.
///
/// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
/// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
/// MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
/// IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
/// CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
/// TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
/// SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#endregion

using System;
using System.Drawing;
```

This source code listing and any updates may be found at the author's GitHub® repository at: `https://github.com/ghammock/ImageEncryptionSuite/`.

```
namespace ImageEncryption.FluidDynamicsInspired
{
    public class FluidDynamicsEncryptor : ImageEncryption
    {
        /***************************************************
        *                Public Fields                *
        ***************************************************/

        /// <summary>
        /// The key that is used for the encryption/decryption process.
        /// </summary>
        public FluidDynamicsKey Key;

        /***************************************************
        *                Constructors                 *
        ***************************************************/

        /// <summary>
        /// Empty default constructor.
        /// </summary>
        public FluidDynamicsEncryptor()
        { }

         /// <summary>
        /// Basic constructor with a predefined key.
        /// </summary>
        /// <param name="key">
        /// The FluidDynamicsKey object that will be used by the
            cryptosystem.
        /// </param>
        public FluidDynamicsEncryptor (FluidDynamicsKey key)
            : this(null, key)
        { }

        /// <summary>
        /// Basic initialization constructor with a predefined plaintext
            image.
        /// </summary>
        /// <param name="plaintextImage">
        /// The image that will be encrypted once a key is defined.
        /// </param>
        public FluidDynamicsEncryptor (Bitmap plaintextImage)
            : this(plaintextImage, FluidDynamicsKey.Generate())
        { }
```

```
/// <summary>
/// Full initialization constructor.
/// </summary>
/// <param name="plaintextImage">
/// The image that is to be encrypted.
/// </param>
/// <param name="key">
/// The FluidDynamicsKey object that is used to encrypt the image.
/// </param>
public FluidDynamicsEncryptor (Bitmap plaintextImage,
    FluidDynamicsKey key)
{
    this.PlaintextImage = (Bitmap)plaintextImage.Clone();
    Key = key.Clone();
}

/**************************************************
*                Public Methods                   *
**************************************************/

/// <summary>
/// Encrypt the plaintext image with the given key.
/// </summary>
public void Encrypt()
{
    // Check the input for the correct domain of values.
    if (Key == null || !Key.IsValidKey())
        throw new System.ArgumentException("The key is not valid.");

    // Ensure that a plaintext image is available for encryption.
    if (PlaintextImage == null)
    {
        throw new System.NullReferenceException(
            "The plaintext image is not defined.");
    }

    // Ensure that we have a 24bpp image to work with since
        SetPixel()
    // won't work with an indexed image.
    CiphertextImage = PlaintextImage.Clone(
        new Rectangle(0, 0, PlaintextImage.Width,
            PlaintextImage.Height),
        System.Drawing.Imaging.PixelFormat.Format24bppRgb);

    ApplyMomentumFlux();
    ApplyHeatFlux();
    ApplyDissipativeTurbulence();
}  // End method Encrypt().

/// <summary>
/// This method applies the row-wise (scanline) momentum (mass)
    fluxes.
/// </summary>
private void ApplyMomentumFlux()
{
    // Retrieve the size of the image in pixels.
    int n = PlaintextImage.Height;
    int m = PlaintextImage.Width;

    // Generate a logistic map to seed the mass flow rate vector.
    double[] logisticMap = GenerateLogisticMap(Key.R_mdot,
        Key.MassFlowRateInitializer, n);

    // The row-wise mass flow rates are a true set of integers [0,
        n].
    int[] rowMassFlowRate = new int[n];
    for (int i = 0; i < n; ++i)
        rowMassFlowRate[i] = i;

    // Randomize the mass flow rates by the sort order
    // of the logistic map vector.
    Array.Sort(logisticMap, rowMassFlowRate);

    // Right circular rotate each scanline in the image.
    int row = 0;
    for (int y = 0; y < n; ++y)
    {
        for (int x = 0; x < m; ++x)
        {
            int replacementIndex = (x + rowMassFlowRate[row]) % m;

            Color replaceWith
                = PlaintextImage.GetPixel(replacementIndex, y);

            CiphertextImage.SetPixel(x, y, replaceWith);
        }

        row++;
    }
}  // End method ApplyMomentumFlux().

/// <summary>
/// This method applies the column-wise heat fluxes.
/// </summary>
private void ApplyHeatFlux()
{
    // Retrieve the size of the image in pixels.
    int n = PlaintextImage.Height;
    int m = PlaintextImage.Width;

    // Generate a logistic map to seed the mass flow rate vector.
    double[] logisticMap = GenerateLogisticMap(Key.R_qdot,
        Key.HeatTransferInitializer, m);

    // The elements of the heat flux vector are in
    // the integer set [0, 255]. This is effectively an
    // initialization vector for the bottom scanline.
    int[] columnHeatFlux = new int[m];
    for (int i = 0; i < m; ++i)
        columnHeatFlux[i] = (int)(255 * logisticMap[i]);

    // For every pixel in the image, XOR its value with the
    // value of the pixel beneath it.
    for (int x = 0; x < m; ++x)
    {
        for (int y = 0; y < n; ++y)
        {
            Color source = CiphertextImage.GetPixel(x, y);

            int Tj = (source.ToArgb() & 0x000000ff);
            int Tj1 = Tj ^ columnHeatFlux[x];

            Color replaceWith = Color.FromArgb(0xff, Tj1, Tj1, Tj1);

            CiphertextImage.SetPixel(x, y, replaceWith);
        }
    }
} // End method ApplyHeatFlux().

/// <summary>
/// Retrieve the turbulent kinetic energy value using the
/// Reynolds k-epsilon equation. This value is applied to
/// the <c>GetDissipativeTurbulence()</c> method.
/// </summary>
/// <returns>
/// A kinetic turbulence energy value in the domain [0, 255].
/// </returns>
private byte GetTurbulenceMultiplier()
```

```
{
    // 0.92 is the turbulence destructive term. See Reynolds, 1987.
    double arg = 0.92 * (double)Key.Epsilon
        * (double)Key.TFrame / (double)Key.K0;

    double exponent = -1.0 / 0.92;

    double Kt = (double)Key.K0 * Math.Pow(1.0 + arg, exponent);

    return (byte)Kt;
}

/// <summary>
/// Applies the k-epsilon dissipative turbulence
/// value to the secret image.
/// </summary>
private void ApplyDissipativeTurbulence ()
{
    // Retrieve the size of the image in pixels.
    int m = PlaintextImage.Width;
    int n = PlaintextImage.Height;

    // This is the scalar value to apply to the XOR operation.
    byte scalar = GetTurbulenceMultiplier();

    // For every pixel in the image, XOR its value with the
    // value of the secret image and the turbulence scalar value.
    for (int y = 0; y < n; ++y)
    {
        for (int x = 0; x < m; ++x)
        {
            int pixel = CiphertextImage.GetPixel(x, y).ToArgb();
            int secPixel = Key.SecretImage.GetPixel(x, y).ToArgb();

            // The luminance value of the XOR of the
            // input image and the secret image.
            byte Y = (byte)((pixel & 0x000000ff)
                         ^ (secPixel & 0x000000ff));

            // Apply the scalar value.
            int xor = Y ^ scalar;

            Color newPix = Color.FromArgb(0xff, xor, xor, xor);

            CiphertextImage.SetPixel(x, y, newPix);
        }
```

```
            }
        }  // End method ApplyDissipativeTurbulence().

        /// <summary>
        /// Generate a vector of deterministic chaos values
        /// using the logistic function.
        /// </summary>
        /// <param name="r">
        /// The population growth coefficient, <c>r</c> in (3.569955672, 4].
        /// </param>
        /// <param name="x0">
        /// The population initializer, <c>x0</c> in [0, 1].
        /// </param>
        /// <param name="iterations">
        /// The number of iterations to perform which is in this case,
        /// the size of the vector.
        /// </param>
        /// <returns>
        /// An array of double precision values of length <c>iterations</c>.
        /// </returns>
        private double[] GenerateLogisticMap
            (double r, double x0, int iterations)
        {
            // Ensure that the values are in the correct domains.
            if (r <= 3.569955672 || r > 4.0)
                throw new ArgumentOutOfRangeException();
            if (x0 < 0.0 || x0 > 1.0)
                throw new ArgumentOutOfRangeException();

            // This is the vector of chaotic logistic map values.
            double[] logisticMap = new double[iterations];

            // The logistic map is defined as:
            //    x_n = r * x_(n-1) * (1 - x_(n-1))
            logisticMap[0] = x0;
            for (int x = 1; x < iterations; ++x)
            {
                logisticMap[x] = r * logisticMap[x - 1]
                    * (1.0 - logisticMap[x - 1]);
            }

            return logisticMap;
        }
    }  // End class FluidDynamicsEncryptor.
}
```