Analysis of Chaotic Cellular Automata for Use with Symmetric Key Cryptography

Sean Wiese

April 30, 2012

1 Introduction

Over the past few decades the field of cryptography has become an increasingly important part of modern life. Tools that were once only employed by governments to protect state secrets are now used to protect everything from personal financial data to email logins. In the field of cryptography old encryptions systems are broken as new code breaking techniques are developed and computer performance increases. Consequently, Cryptographers are continuously looking for new algorithms to use in encryption systems.

One of the backbones of many major cryptography systems is a pseudorandom number generator (PRNG). These functions work to create a deterministic output that appears random based on a specific seed. A good PRNG is a deterministic mathematical function with high sensitivity to initial conditions, through key dependence, and an output that appears random. The field of Chaos Theory stresses a high sensitivity to initial conditions with outputs that can appear to be random, making chaotic functions an excellent candidate for a PRNG. A popular chaotic function, the logistic equation, has already been shown to be an effective PRNG[5]. A set of discrete dynamical systems called Cellular Automata (CA) have also been shown to be an effective PRGN for use in encryption systems [3][4]. Although the use of CAs in cryptography is not new, this paper seeks to compare the effectiveness of elementary cellular automata and life like cellular automata for use in encryption, as well as, the difference between a CA based stream cipher and CA based block cipher.

In Section 2, we will discuss the various types of cellular automata used in our encryption systems. In Section 3, a stream cipher based on an elementary cellular automaton will be described and analyzed. In Section 4, a stream cipher based on a 2 dimensional life like cellular automaton will be described and analyzed. In Section 5, a block cipher encryption system
based on a 2 dimensional life like cellular automaton will be proposed and tested in detail. Finally, in Section 6, the paper will end with a comparison of the advantages and disadvantages of the three encryption systems, possible weaknesses of CA based encryption, and potential future extensions.
2 Cellular Automata

In this section we will provide a definition of a cellular automaton and describe the two different types of CAs that will be used in this paper.

2.1 CA Definition

A cellular automaton is a discrete model studied in computability theory, mathematics, physics, complexity science, theoretical biology and microstructure modeling. It consists of a regular grid of cells, each in one of a finite number of states, such as "On" and "Off". The grid can be in any finite number of dimensions. For each cell we will refer to the surrounding cells in the grid as neighbor cells. These neighbor cells can be defined to be any discrete distance away from the center cell. An initial state is selected by assigning a state for each cell. A new generation is created, according to some fixed rule that determines the new state of each cell in terms of the current state of the cell and the states of the cells in its neighborhood. Typically, the rule for updating the state of cells is the same for each cell and does not change over time.

2.2 Elementary 1D CA

Elementary 1D CAs are the simplest form of cellular automata. They exist in only two states, On or Off, and the rules that govern the system only depend on the nearest neighbor values. This gives a possible $2^3$ possibly binary states, thus, producing $2^8$ possibly 1D elementary CAs each of which can be represented by an 8-bit binary number. These cellular automata were extensively studied and classified by Stephen Wolfram. Stephen classified elementary cellular automata into four different classifications. [8]

- class 1: Nearly all initial patterns evolve quickly into a stable, homogeneous state. Any randomness in the initial pattern disappears.
- class 2: Nearly all initial patterns evolve quickly into stable or oscillating structures. Some of the randomness in the initial pattern may filter out, but some remains. Local changes to the initial pattern tend to remain local.
- class 3: Nearly all initial patterns evolve in a pseudo-random or chaotic manner. Any stable structures that appear are quickly destroyed by the surrounding noise. Local changes to the initial pattern tend to spread indefinitely.
- class 4: Nearly all initial patterns evolve into structures that interact in complex and interesting ways. Class 2 type stable or oscillating structures may be the eventual out-
come, but the number of steps required to reach this state may be very large, even when
the initial pattern is relatively simple. Local changes to the initial pattern may spread
indefinitely.

For purpose of this paper we will only focus on class 3 CAs.

2.3 2D Life Like Cellular Automata

A 2D life like CA differs from elementary cellular automata in that it exists in a 2 dimensional
grid and must meet the following criteria:

- The array of cells of the automaton has two dimensions.
- Each cell of the automaton has two states (conventionally referred to as ”alive” and
  ”dead”, or alternatively ”on” and ”off”)
- The neighborhood of each cell consists of the eight adjacent cells and (possibly) the cell
  itself.
- In each time step of the automaton, the new state of a cell can be expressed as a function
  of the number of adjacent cells that are in the alive state and of the cell’s own state.

This class of cellular automata is named for the Conways Game of Life, the most famous
cellular automaton, which meets all of these criteria.

Throughout this paper we will be using B/S notation to define CAs. This notation system
indicates the number of neighbors that can result in a Birth, a change of a dead cell to an alive
cell, and the number of neighbors that can result in a Survival, an alive cell remaining alive.
For example, Conways Game of Life is denoted by (B3,S23) indicating that 3 neighbors allows
a cell to be born and 2 or 3 neighboring cells allows the cell to survive.
3 1D CA Stream Cipher

In this section the development of an encryption system using a chaotic elementary CA will be discussed.

3.1 Stream Cipher

In cryptography, a stream cipher is a symmetric key cipher where plaintext bits are combined with a pseudorandom cipher bits. In a stream cipher each plaintext bit is encrypted one at a time with the corresponding bit of the cipher stream, to give a bit of the ciphertext stream. The XOR operation is used to combine a plaintext bit and a cipher bit into the ciphertext bit.

3.2 Choice of 1 Dimensional CA

A good stream cipher will have the following characteristics.

1. Apparent Randomness: The stream must appear to be random. If a pseudorandom stream is XORed with a non-random stream the output will appear random.

2. Diffusion: A small change in the key used to generate the stream should provide a completely different result, i.e. half of the bits should be flipped.

A chaotic system satisfies the first condition. Therefore, we will look to select one of the chaotic elementary cellular automata identified by Wolfram. Wolfram identified rules 22, 30, 126, 150, and 182 as chaotic cellular automata [8]. For this paper we will use Rule 30 as the basis of our stream cipher. This rule was chosen for having the unique characteristic that the center column of the rule produces a statistically random stream of bits with a high sensitivity to initial conditions. Furthermore, this CA is currently used in the computational software program Mathematica as a large number random number generator [7].

3.3 Algorithm

To encrypt a data stream using this method the following technique can be employed.

1. Set the key as the initial condition of the CA.

2. Run the CA for n iterations where n is the number of bits in the key. The goal of this step is to increase diffusion for bit flip in an edge bit of the key. For example, if the initial condition starts with the 16 bit key 10000000100000001 and then changes to 00000000100000001 the first 8 bits of the center column will be the same.
3. Run the CA m iterations where m is the number of bits in the data source. The bit of the center column of the CA is saved for use in the stream cipher.

4. Take the resulting center column bits and xor with data.

To decrypt the ciphertext, the stream is generated using the same method and XORed with the encrypted data the result is the original plaintext. Figure 2 shows how the XOR operator can be used to encrypt and decrypt data.

**Encryption XOR**

<table>
<thead>
<tr>
<th>Data</th>
<th>01010111 01101001 01101011 01101001</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream</td>
<td>11110011 11110011 11110011 11110011</td>
</tr>
<tr>
<td>E Data</td>
<td>10100100 10011010 10011000 10011010</td>
</tr>
</tbody>
</table>

**Decryption XOR**

<table>
<thead>
<tr>
<th>E Data</th>
<th>10100100 10011010 10011000 10011010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream</td>
<td>11110011 11110011 11110011 11110011</td>
</tr>
<tr>
<td>Data</td>
<td>01010111 01101001 01101011 01101001</td>
</tr>
</tbody>
</table>

**Figure 1:** A image of rule 30 run for 15 iterations. The highlighted center column is chaotic and will be used for our stream cipher. Image courtesy of [7].

**Figure 2:** Sample encryption and decryption of data using a stream composed of the repeated binary sequence 11110011
3.4 Results

Several tests can be run to evaluate the effectiveness of the proposed encryption system. These tests include the entropy, % of bits that are 1, Matlab randomness test, key diffusion rate, and plaintext diffusion rate [4]. Each of these focuses on detecting the weakness of the encryption system against different types of attacks.

<table>
<thead>
<tr>
<th></th>
<th>Original Image</th>
<th>Encrypted Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entropy (10 is perfectly random)</td>
<td>7.717</td>
<td>9.813</td>
</tr>
<tr>
<td>% of bits that are 1</td>
<td>52.3%</td>
<td>50.5%</td>
</tr>
<tr>
<td>Matlab runs test</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>1 bit flip key diffusion rate</td>
<td>NA</td>
<td>49.92%</td>
</tr>
<tr>
<td>NPCR (1 bit flip plaintext)</td>
<td>NA</td>
<td>0.00003%</td>
</tr>
</tbody>
</table>

Table 1: Results of Rule 30 1 Dimensional Stream Cipher Encryption. The final two tests are only valid for an encryption system since they measure how effective the system is key and input text diffusion. Therefore there are no results for the original image.

3.4.1 Brute Force Key Search

The most basic form of attack is a brute force key search. In this type of attack an attacker will systematically try to decrypt the ciphertext by checking permutations of the key until the correct key is found.

The security of this encryption system against such an attack is based on the length of the key. For the tests run in this section a key of length 256 bits was used. To brute force a 256 bit key would require an attacker to check $2^{256}$ different keys. This key space is too large for an exhaustive key search using even the most powerful modern day supercomputers.
3.4.2 Cipher Only Attacks

In cryptography, a ciphertext-only attack (COA) or known ciphertext attack is an attack model for cryptanalysis where the attacker is assumed to have access only to a set of ciphertexts. Over the years cryptographers have developed statistical techniques for attacking ciphertext, such as frequency analysis. These attacks attempt to identify an underlying pattern in the data and exploit that pattern in order to reduce the number of attempts a brute force attack must make before decrypting the data. In order to protect against these attacks the cipher text should appear to be a random distribution of bits. This ability to generate an apparently random ciphertext is a property of the encryption system referred to as confusion.

One indication of random data is a uniform distribution of values. This can be checked by viewing a histogram of the distribution of red, green, blue, and alpha values of the pixels in the encrypted vs. original image. From figures 10 and 11 it is clearly shown that the distribution of rgba values is relatively uniform and differs from the original image.

A random dataset of 1s and 0s should have an approximately equal number of each type of bit in the set. A larger value of either indicates that the PRNG favors one bit over the other. This behavior decreases the encryption system's resistance to frequency attacks. The value for our encryption system is with one half a percent of 50%, indicating a pseudorandom distribution.

Another indication of randomness is image entropy. Image entropy is a statistical measure of randomness that can be used to characterize the texture of the input image. Entropy is defined as

$$E = - \sum_i (p_i \cdot \log_2(p_i))$$

where p contains the histogram counts. A higher value of E is indicative of higher randomness in the image. From the results in table 1 we can see that entropy of the encrypted image is far greater than that of the plaintext image.

A more advanced test for randomness is the runs test used by Matlab. This test performs a runs test on the sequence of observations in the vector x. This is a test of the null hypothesis that the values in x come in random order, against the alternative that they do not. The test is based on the number of runs of consecutive values above or below the mean of x. Too few runs indicate a tendency for high and low values to cluster. Too many runs indicate a tendency for high and low values to alternate. The test returns the logical value h = 1 if it rejects the null hypothesis at the 5% significance level, and h = 0 if it cannot. From table 1 it is shown that the encrypted image passes the runs test while the original image fails the test.
3.4.3 Differential Attacks

Differential cryptanalysis is a general form of cryptanalysis applicable primarily to block ciphers, but also to stream ciphers and cryptographic hash functions. In the broadest sense, it is the study of how differences in an input can affect the resultant difference at the output. In the case of a block cipher, it refers to a set of techniques for tracing differences through the network of transformations, discovering where the cipher exhibits non-random behavior, and exploiting such properties to recover the secret key.

One indication of resilience to differential attacks is if an encryption system has good diffusion. Diffusion means that a change a bit in the plaintext leads to several changes in the encrypted image. Ideally one would have a diffusion rate of approximately 50%, meaning that any change in the original plaintext appears to produce a completely different ciphertext.

When encrypting images the NPCR test can be used to test diffusion. NPCR rate is the % of bits that change when a single input bit is changed.

\[
NPCR = \sum_{i} \frac{F(i)}{N} \times 100
\]

where \( F(i) \) is the the bit value, 0 or 1, at the \( i \) index and \( N \) is the number of bits in the data set. From table 1 it is clear that our cipher has an extremely poor NPCR rate. This is due to the fact that the encryption systems cipher stream is generated solely based on the key with no regard to the plain text. Thus, if one bit is flipped in the plain text, only one bit will be flipped in the ciphertext.

3.4.4 Key dependence analysis

Further attacks look at how the encryption system changes based on a change in the key. A good encryption system should produce a different output for different keys even if only one bit is flipped in the key.

To measure the key dependence of the encryption system a test was run where only one bit was flipped in the key and the number of flipped bits in the output was summed. From table 1 it is clear that even a small change in the initial key produces a completely different output.

3.4.5 Overview of Results

From table 1 and figure 3 the proposed encryption method was effective and producing a cipher image that differs from the original image and appears to be random noise.

Despite the success of this encryption method, there still exist some major drawbacks. The computation time of this encryption system is relatively large. This is due to the face that only
one column of each CA iteration is used. This restriction forces the system to run at least one iteration for every bit in the plaintext. This is extremely inefficient and could never be used for any large dataset.
4 2D Life Like CA Stream Cipher

In this section the development of an encryption system using a chaotic life like CA will be discussed.

4.1 Choosing Life Like Chaotic CA

Several papers have looked at how effective various life like cellular automata behave as PRNGs. In [3] an encryption system was designed uses life like cellular automata. [3] studied several life like CAs for use as a PRNG and found that a Fredkin (B4357/02468) and Amoeba (B357, S1358) performed the best in a Diehard and Ent randomness tests. In the next section we will examine both CAs for use in our stream cipher.

4.1.1 Testing Chaos of CA

As a test to ensure that the Fredkin and Amoeba CA acts as a PRNG and is chaotic, the Lyapunov Exponent of the CA can be estimated in addition to running randomness tests on the output of the CA. A positive Lyapunov exponent is indicative of chaos.

The nondirectional maximum Lyapunov exponent (MLE) of an elementary cellular automaton (CA) may be interpreted as the natural logarithm of the time averaged number of cells $c_j$ in a cell’s neighborhood $N(c_i)$ that is affected during each consecutive time step if the state of $c_j$ is perturbed. Results of testing Fredkin (B4357/02468) and Amoeba (B357, S1358) against the known chaotic rule 30 can be found in table 2.[1]

<table>
<thead>
<tr>
<th>Rule</th>
<th>Matlab Runs Test</th>
<th>Lyapunov Exponent</th>
<th>Lyapunov Exponent [3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rule 30</td>
<td>Pass</td>
<td>0.653</td>
<td>NA</td>
</tr>
<tr>
<td>Fredkin (B4357/02468)</td>
<td>Pass</td>
<td>0.999</td>
<td>0.999</td>
</tr>
<tr>
<td>Amoeba (B357, S1358)</td>
<td>Pass</td>
<td>0.821</td>
<td>0.8604</td>
</tr>
</tbody>
</table>

Table 2: Comparison of calculated MLE values for different chaotic CAs. For this paper the MLE value was determined from the average of 5 runs and the MLE calculation between the 200th and 201st iteration with different initial conditions

The Fredkin CA outperformed the Amoeba CA in the tests from table 2 and the Diehard randomness test from [3]. Consequently, for our stream cipher we will use the Fredkin CA.

4.2 Proposed Algorithm

To encrypt a data set the following algorithm can be used.
1. Set the key as the initial condition of the CA by looping the key over a grid the size of the data set.

2. Run the CA for a number of iterations $n$.

3. Take the resulting array of bits and xor with data.

To decrypt the ciphertext, the stream is generated using the same method and XORed with the encrypted data, the result is the original plaintext.

### 4.3 Results

Testing of the effectiveness of a CA was discussed in section 3.4. In this section we will only summarize the results of using the life like CA.

<table>
<thead>
<tr>
<th></th>
<th>Original Image</th>
<th>Encrypted Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entropy (10 is perfectly random)</td>
<td>6.343</td>
<td>9.989</td>
</tr>
<tr>
<td>% of bits that are 1</td>
<td>47.05%</td>
<td>50.15%</td>
</tr>
<tr>
<td>Matlab runs test</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>1 bit flip key diffusion rate</td>
<td>NA</td>
<td>50.04%</td>
</tr>
<tr>
<td>NPCR (1 bit flip plaintext)</td>
<td>NA</td>
<td>0.000002%</td>
</tr>
</tbody>
</table>

**Table 3:** Results of Rule 30 1 Dimensional Stream Cipher Encryption. The final two tests are only valid for an encryption system since they measure how effective the system is key and input text diffusion. Therefore there are no results for the original image.

**Figure 4:** Original image on the left. Encrypted image 10 iterations in the center. Encrypted image 1000 iterations on the right.

The results displayed in table 3 are comparable to those generated by the 1 dimensional rule 30 stream cipher. However, the 2D cellular automaton was able to overcome many of
the problems presented by the rule 30 stream ciphers. Most notably this algorithm allows all the bits of the final cellular automata to be used. Additionally, less iteration are required to generate a usable data stream.

From testing 100 to 1000 iterations appears sufficient to generate a pseudorandom stream. However, there is a danger that insufficient iterations will be used and remnants of the initial data will still be visible. Figure 4 shows an image after only 10 iterations. The outline of the earth is still visible indicating that there was not enough randomness in the CA after only 10 runs. As a rule of thumb the PRNG should be run a minimum of 100 iterations and until 99% of the rgba frequency values are within 10% to 15% of the mean frequency.

There still remains a major drawback in the design of this algorithm. The encryption systems NPCR remains low remains low leaving the ciphertext vulnerable to differential attacks.
5 Block Cipher

In order to produce an encryption system that has diffusion, a block cipher can be used. In cryptography, a block cipher is a deterministic algorithm operating on fixed-length groups of bits, called blocks, with an unvarying transformation that is specified by a symmetric key.

The modern design of block ciphers is based on the concept of an iterated product cipher. In cryptography, a product cipher combines two or more transformations in a manner intending that the resulting cipher is more secure than the individual components to make it resistant to cryptanalysis. Iterated product ciphers carry out encryption in multiple rounds, each which uses a different subkey derived from the original key.

One of the main advantages of block ciphers is diffusion. In our two previous encryptions systems, the encrypted data was still vulnerable to a differential attack. Due to the use of S and P boxes, to be explained later, block ciphers are able to have a high sensitivity to the input text.

5.1 Block Cipher Algorithm

The proposed block cipher algorithm can be broken into several parts.

1. Generation of keys.
2. Generation of P and S Boxes
3. Block Cipher Encryption Algorithm
4. Block Cipher Chaining

An overview of the block cipher encryption system without chaining can be viewed in figure 5. This particular block cipher is a substitution permutation network. Each part of the system servers a particular focus.

**S-Box** In cryptography, an S-Box (Substitution-box) is a basic component of symmetric key algorithms which performs substitution. They are typically used to obscure the relationship between the key and the ciphertext.

**P-Box** In cryptography, a permutation box (or P-box) is a method of bit-shuffling used to permute or transpose bits across S-boxes inputs, retaining diffusion while transposing.

**Sub Key** In a block cipher, a rounding function is used to derive several keys from the original key that can be XORed with the input to each round of S-Boxes. In figure 5 $K_{0,1,2,3}$ are the result of the rounding function. These keys will be referred to as sub keys.
Figure 5: A sketch of a Substitution-Permutation Network with 3 rounds, encrypting a plaintext block of 16 bits into a ciphertext block of 16 bits. The S-boxes are the $S_{i,8}$, the P-boxes are the same $P$, and the round keys are the $K_{i,8}$.

5.1.1 Generation of keys

In order generate the sub keys used in the substitution-permutation network the Fredkin (B4357/02468) CA can be used. A $n \times m$ grid can be initialized where $n$ is the size of the key and $m$ is number of keys that are needed. For this algorithm the number of keys needed is 4. The $n \times m$ grid is initially filled with the repeating value of the key. The system is then run for several iterations to produce a number of pseudorandom keys. The small grid size allows a random pattern to appear in less iteration. After only 10 iterations the pattern from the original key was destroyed by the chaotic nature of the CA. To make the computation easier a $32 \times 32$ array is used to represent the $256 \times 4$ array of keys. The output after only 10 iterations can be viewed in figure 6.

5.1.2 S-Boxes

S-boxes are a key part of an effective block cipher. One of the simplest forms of a S-box is where a $n$ bit number is input into the s box and is transformed into another $n$ bit number via
Figure 6: An array of subkeys generated by the Fredkin CA after 10 iterations

A lookup. An example of a 16 bit s box can be seen in table 4. S-boxes where multiple inputs map to the same output are forbidden since such a box would make the system non reversible. 32 32 bit S-boxes were used in this block cipher encryption system.

<table>
<thead>
<tr>
<th>input</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>14</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4: A S-Box input output map
5.1.3 P-Box

P-boxes ensure that our block cipher has good diffusion. An example of how a p box functions can be seen in figure 5. For the p boxes used in our algorithm we generated 2 random P-boxes ensuring that at least one bit from each S-box was mapped to one of the 8 S-boxes in the following row.

5.1.4 Mode of Operation

In cryptography, mode of operation is the procedure of enabling the repeated and secure use of a block cipher under a single key, also referred to as chaining. A block cipher by itself allows encryption only of a single data block of the cipher’s block length. When targeting a variable-length message, the data must first be partitioned into separate cipher blocks. Typically, the last block must also be extended to match the cipher's block length using a suitable padding scheme. A mode of operation describes the process, type of chaining, of encrypting each of these blocks, and generally uses randomization based on an additional input value, often called an initialization vector, to allow doing so safely.

One of the advantages of chaining is to avoid the same input production the same ciphertext. Figure 7 shows the result of a block cipher encryption system without chaining.

![Figure 7: The image on the left shows the original image. The image on the right is encrypted using a block cipher without chaining.](image)

For this block cipher algorithm a Propagating Cipher Block Chaining (PCBC) mode of
encryption will be used. The encryption and decryption process is described in figure 8.

Figure 8: A sketch of the propagating cipher block chaining mode of operation used in this block cipher encryption system. The initialization vector is a randomly generated binary vector of length 256 bits.

5.2 Results

Testing of the effectiveness of a CA was discussed in section 3.4. In this section we will only summarize the results of using the Frekin CA as basis for the block cipher described in the previous subsections.

The results displayed in table 5 and figures 9, 14, and 15 are comparable to those generated by the first two encryption methods. However, the major difference in the results is the increase in the NPCR value. This result indicates that the block cipher method is more resilient to differential attacks then the stream cipher methods.
Figure 9: Results of encryption of 512x512 png using the block cipher method described in this section. The original image is on the left. The encrypted image is on the right. Just in case you couldn’t tell.

Table 5: Results of block cipher encryption based on the Fredkin CA. The final two tests are only valid for an encryption system since they measure how effective the system is key and input text diffusion. Therefore there are no results for the original image.
6 Conclusion

To conclude, we will expand on the joint results of all three encryptions systems, closely examine their strengths and weaknesses, and propose extensions.

6.1 Overall Results

One of the most valuable results is that all three chaotic cellular automata encryption systems had excellent confusion. All of the systems exhibited high entropy, high key diffusion, and good bit frequency as well as passing the Matlab runs test. Despite this common characteristic the stream ciphers failed to exhibit diffusion while the block cipher demonstrated excellent diffusion due to the S and P boxes.

The encryptions systems were further separated by their computational requirements. The 1D rule 30 stream cipher proved unrealistic due to its exorbitant memory and computational costs when dealing a data stream of larger than a few thousand bits. The 2D Fredkin stream cipher fared better by limiting the memory usage to the size of the data stream being encrypted. This limited data size kept the computational requirements for each iteration constant leading to a significant improvement over the 1D rule 30 encryption system. However, the best performance was the block cipher. Combining multiple permutations together with a small size CA grid composed of the sub keys significantly reduced the computational requirements.

While the block cipher was able to outperform the stream ciphers under normal conditions, it lacks the parallelism that is available in the stream ciphers. Table 6 shows how calculating CAs using a Nvidia CUDA program can reduce computation time by several fold. It would also be possible to further increase performance with a hardware implementation such as the one suggested in [6].

<table>
<thead>
<tr>
<th>CA</th>
<th>Grid Size</th>
<th>Iterations</th>
<th>Python</th>
<th>C</th>
<th>CUDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 30</td>
<td>128 to 8320 bits</td>
<td>4096</td>
<td>323s</td>
<td>0.205s</td>
<td>0.046s</td>
</tr>
<tr>
<td>2D Fredkin</td>
<td>512 x 512 bits</td>
<td>1000</td>
<td>954s</td>
<td>2.156s</td>
<td>0.049s</td>
</tr>
</tbody>
</table>

**Table 6**: Results of running rule 30 and Fredkin CA using python, C, and CUDA. Pythons slow performance was due to the use of lists instead of python arrays. Additionally, 0.035s of the CUDA code was spent sending data to and from the GPU. For larger iterations or grid sizes CUDA would increase its lead over c and python.
6.2 Strengths, Weaknesses, and Extensions

As mentioned above all of the encryption systems performed extremely well when basic crypt-analysis tests were performed. However, to ensure that the output of the encryption system produces an excellent pseudorandom ciphertext more advanced randomness tests could be run. In [3] the diehard and ent tests were used to test the randomness of the life like CAs. These randomness tests could be used on the output of our encryption systems to ensure that the output is pseudorandom.

One weakness with using CAs for cryptography is the danger of weak keys. In cryptography, a weak key is a key, which, used with a specific cipher, makes the cipher behave in some undesirable way. Weak keys usually represent a very small fraction of the overall keyspace, which usually means that, if one generates a random key to encrypt a message, weak keys are very unlikely to give rise to a security problem. Nevertheless, it is considered desirable for a cipher to have no weak keys. A cipher with no weak keys is said to have a flat, or linear, key space. Testing for weak keys is a difficult task, however, there is one week key for all of the encryption systems in this paper. A key of all 0s would produce a grid of all zeros in the stream ciphers and all of the sub keys in the block cipher would be comprised of only zeros. To avoid this problem a key of all zeros could be forbidden for use with these encryption systems.

One weakness of the block cipher encryption system is the possibility of weak S boxes. For example the DES encryption system was originally designed with weakens S boxes. However, the NSA mysteriously changed the S boxes when the government was reviewing the DES system. The 8 S-boxes of DES were the subject of intense study for many years out of a concern that a backdoor might have been planted in the cipher. The S-box design criteria were eventually published [2] after the public rediscovery of differential cryptanalysis, showing that they had been carefully tuned to increase resistance against this specific attack. Other research had already indicated that even small modifications to S-boxes could significantly weaken encryption systems.
7 Appendix

Figure 10: This histogram shows the distribution of rgba values in the original test image in figure 3
Figure 11: This histogram shows the distribution of rgba values in the encryption image in figure 3
Figure 12: This histogram shows the distribution of rgba values in the original test image in figure 4
Figure 13: This histogram shows the distribution of rgba values in the original test image in figure 4.
Figure 14: This histogram shows the distribution of rgba values in the original test image in figure 9
Figure 15: This histogram shows the distribution of rgba values in the original test image in figure 9
References


