A Cellular Automata Driven Image Encryption System

A Thesis

Presented to
the faculty of the School of Engineering and Applied Science
University of Virginia

in partial fulfillment
of the requirements for the degree

Master of Science

by

John A. Slovensky

August

2013

APPROVAL SHEET

The thesis
is submitted in partial fulfillment of the requirements
for the degree of
Master of Science

John Slavenski

AUTHOR

The thesis has been read and approved by the examining committee:

Gerard Learmonth PhD

Advisor

Alfredo Garcia PhD

Gabriel Robins PhD

Accepted for the School of Engineering and Applied Science:

[illegible]

Dean, School of Engineering and Applied Science

August
2013

CONTENTS

1. Abstract

Efficient pseudorandom number generation is critical for a wide variety of applications in science and engineering. Stochastic optimization, Monte Carlo simulation, and modern encryption schemes are all fundamentally dependent upon reliable sources of pseudorandom data (Vattulainen). Encryption, in particular, demands the highest quality pseudorandom numbers as the encryption power of a given symmetric cryptosystem is highly dependent upon how difficult it is for current analytical methods to find structure in pseudorandom data. Further, as analytical techniques evolve, data which was previously classified as “random” becomes unusable as its underlying structure can be at least partially detected. Thus, there is an ever-evolving search for novel forms of “randomness” generation to fuel the increasing demand for faster, stronger, and more efficient cryptosystems. The aim of this thesis is to design, implement, and study a two-dimensional cellular automaton and demonstrate its potential to serve as the basis for an image specific symmetric cryptosystem. To achieve this, a totalistic cellular automaton rule derived from digital root analysis is iteratively computed on a toroidal surface. The surface is seeded with a single initial condition and the computational feedback dynamics are studied. The motivating concept is to vary the system parameters in order to push the automaton to a critical point at which a small change in the object parameter(s) will elicit a large change in the output. Such transition points are known to produce data that is highly interdependent and thus likely to be difficult to unwind via cryptanalysis. The model is written in the programming language Netlogo in order to generate testable streams of data. The focus is on demonstrating the viability of an image encryption scheme rather than on high efficiency. In practical applications, efficiency would be gained by implementation in a more basic language or possibly even an integrated circuit designed specifically for this image encryption protocol.

2. Introduction and Background

2.1 Properties of "Randomness"

2.1.1 Stochasticity as Encrypted Determinism

Systems are typically classified as either deterministic or stochastic. In many engineering and mathematical texts, the word stochastic is used as an antonym to the word deterministic. In practice, all systems modeling is inherently deterministic as the computer architecture upon which models are built is a finite state automaton. The use of stochastic and deterministic as antonyms thus creates a fundamental misunderstanding of their true relationship as it applies to all practical computationally driven applications.

Any computational description (model) of a stochastic system necessarily uses deterministic data as an input. The model is able to capture stochastic behavior only because the nature of the input data is such that its structure cannot be detected; more succinctly, the input data is encrypted. Whereas systems theory typically presents a black and white distinction between deterministic and stochastic systems, viewing stochasticity as encrypted determinism connects them together. Although the above argument does not necessarily apply to physical systems themselves, it is perfectly relevant to computational models of physical systems.

The characterization of stochasticity as encrypted determinism also begs an important question. How does a deterministic machine accurately reproduce the behaviors and dynamics of a truly random physical system? The only logical answer to this question is that the model does not reproduce all of the behaviors of the real system, but rather only those properties that the current descriptive techniques are able to reveal. It achieves this because the analytical tools available cannot meaningfully describe, or "break," the pseudorandom data that powers the stochastic model.

If a new technique which reveals additional structure in the pseudorandom data is discovered, then the existing model would be at least somewhat obsolete. This is because when the new descriptive technique is applied directly to the analysis of a system, it will reveal insights that the now obsolete model could not capture. The investigator who is using the outdated pseudorandom number generator (PRNG) will likely assume that his/her model is capturing all possible system dynamics. This is a dangerous situation as it can lure investigators into thinking that their model is somehow universal [Scherer 314]. This logic also explains why the quality of a random number generator can significantly affect the outcome of Monte Carlo simulations [Click].

2.1.2 Randomness Is Relative

Perhaps a more elegant characterization of the concept of “randomness” is that it is a perceptual phenomenon that occurs when the observer of a data set cannot describe that data set any more succinctly than it is already stated. This implies that the observer can detect no underlying structure which could help reduce the explicit statement of said data to a smaller quantity of information. Critically, it also implies that “randomness” is really a relative concept. The mathematical lens through which the observer views the observed is a determining factor in the perception of apparent randomness. Change out the lens, and it is possible that what was once “random” can now be viewed as ordered and vice versa. This implies that if new descriptive techniques are discovered, then the definition of “random” must be altered accordingly. An important historical example of this can be found in the linear congruential random number generator (LCG). Before the discovery of spectral testing, the LCG produced bits that were apparently random. With the introduction of the spectral test in the early 1970’s, additional structure was able to be detected in LCG produced data and it was therefore no longer considered entirely “random.”

A more precise way to capture the relevant properties of deterministic “randomness” is found in the term irreducible. This term emerges from Computational Complexity Theory which argues that if the output of a given function cannot be described with less information than is required to explicitly state the output itself, then that output is irreducible [Chaitin 3]. For the purposes of computation, irreducible is therefore a preferred term to describe deterministically generated sequences which publicly known mathematical techniques cannot distinguish from “randomness.”

2.1.3 Irreducibility and Cryptography

Functions which produce irreducible output are highly valuable not only for powering stochastic models, but also for secure communications. Because computational functions are deterministic, their precise outputs can be reproduced if the initial state of the function is known. Functions with irreducible output therefore create a knowledge differential between those that have access to the initial state and those that do not. Cryptosystems leverage this knowledge differential into a cryptographic power differential by treating the initial function state as a cryptographic key. This creates a situation where a given set of data is irreducible to those who do not possess the key, but is reducible to those that do hold the key. This allows for coherent information, or plaintext, to be obscured by apparently random information. The resulting mixture is known as the ciphertext [Singh 11].

Because the key holders have the initial state of the function, they can efficiently compute the otherwise irreducible function output. Given a set of ciphertext, a key holder can compute the irreducible data and then simply subtract it from the ciphertext, thus producing the plaintext. This is possible only for the key holder to accomplish because he/she is imbued with the power to reduce the otherwise irreducible data which obscures the plaintext.

2.1.4 Irreducibility Implies Chaos

The irreducibility of deterministic computations is actually the result of a mathematical phenomenon known as chaos [Herring]. Although chaos is a widely studied phenomenon, it has no precise definition. Most texts tend toward the terms “aperiodic” and “unpredictable” to describe the output of chaotic functions. In Chaotic Dynamics of Nonlinear Systems, Rasband states: "The very use of the word 'chaos' implies some observation of a system, perhaps through measurement, and that these observations or measurements vary unpredictably. We often say observations are chaotic when there is no discernible regularity or order" [Rasband 1].

Chaos-based cryptosystems have grown in popularity and abundance over the past two decades as they have been found to provide effective confusion and diffusion of data [Patidar 327]. Chaotic systems are highly sensitive to initial conditions; a property which was discovered and published by Edward Lorenz in 1963 [Lorenz]. This sensitivity is critical in cryptographic applications as it means that approximate knowledge of a cryptographic key does not translate into approximate knowledge of the output generated by that key. Absent this sensitivity, cryptosystems would be much more vulnerable to attack as a nearly correct guess of a key would reveal some of the plaintext. Repeated guessing of the key would then give the attacker feedback about its approximate structure and the cryptosystem could be broken. The sensitivity of a function’s output to the key state is thus a necessary but insufficient condition for a secure cryptosystem.

2.2 Periodicity in Chaos

2.2.1 Finite Fields

Finite field number systems are categorized by mathematicians according to the axioms they obey. A finite field is a set of numerical values that obey axioms 1-6 below. The number of elements of a finite field is of the form p^n (also called the radix), where p must be a prime number and is called the characteristic of the field, and n is a positive integer. When n does not equal 1, there are elements in the field that are not relatively prime to p^n. This causes axiom 6 to break down as those elements which are not relatively prime to p^n (share a common factor with p^n) will not have a multiplicative inverse. The field is then classified as a commutative ring (Blyth).

1) *Closure of F* under addition and multiplication: For all a, b in F, $a + b$ and $a \cdot b$ are in F.

2) *Associativity* of addition and mult.: $a + (b + c) = (a + b) + c$ and $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.

3) *Commutativity* of addition and multiplication: $a + b = b + a$ and $a \cdot b = b \cdot a$.

4) *Distributivity* of multiplication: For all a, b and c in F, $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.

5) Additive and multiplicative identity: There exists an element of F, called the *additive identity* element denoted I such that for all a in F, $a + \mathrm{I} = a$. Likewise, there is an element, called the *multiplicative identity* element and denoted by 1, such that for all a in F, $a \cdot 1 = a$.

6) Additive and multiplicative inverses:
For every a in F, there exists an element $-a$ in F, such that $a + (-a) = \mathrm{I}$. Similarly, for any a in F other than I, there exists an element a^{-1} in F, such that $a \cdot a^{-1} = 1$.

2.2.2 Digital Roots

Looking at chaos in the logical context of section 3.1.2 opens the door for a very fundamental question about the nature of chaotic data. Is it possible that if the mathematical lens through which investigators view chaotic data is changed, then it will no longer appear aperiodic? Because chaos has been studied intensely for nearly 50 years, it seems unlikely that investigators would have missed any minor changes to their descriptive techniques that could reveal periodicity in chaos. Such a fundamental oversight would have to be deeply embedded in the minds of all investigators or else it would have seen exposure long ago. This is where insights from the practice of Systems Engineering can help to place such a fundamental question into a broader context.

It is firmly established in Systems Engineering practice that if the results of an analysis fail to accurately describe a system, then a scrutiny of the underlying assumptions is required. Subtle inaccuracies in the simplifying assumptions used to frame a problem can and do translate into severe engineering disasters. When applied to a description of chaotic data, this engineering wisdom would suggest that there may be a fundamental false assumption that is repeatedly escaping scrutiny. Further, if the faulty assumption is not introduced in the framing of the problem, but rather in the construction of the analytical tools used to make the assessment, then the fault will not be found.

With this in mind, let us examine a simple source of chaotic data known as the logistic map, given by: $X_{n+1} = r*X_n*(1 – X_n)$. One key feature of this map is that it exhibits the phenomenon of self-similarity at certain values of the tuning parameter r. As r approaches a critical value just below 3.57, the oscillations of the limiting values of the map begin to bifurcate in a manner which produces self-similar structures. This process happens at an increasing rate as the value of r is increased such that beyond the critical point there is only aperiodic chaos. This process occurs such that the ratio of the difference between successive bifurcation values approaches approximately 4.669; a value known as Feigenbaum's constant [Strogatz 373]. This doubling process is characterized as the "period doubling route to chaos" [Strogatz 355].

The observation of a scale invariant bifurcation structure at the chaotic transition point in the logistic map may provide a clue to the type of mathematical lens which would find periodicity in chaotic output. It suggests that the lens through which chaotic data can be more deeply understood should also exhibit scale invariant behavior. This key insight leads to a function from mathematics called the digital root function.

The digital root of any numerical value is the persistent sum of the digits. The digital root function applies to any nonzero integer value or finite decimal and it outputs a value within the set {1:9}. When the input is a positive integer, the digital root function is equivalent to modulo 9 except that it returns a nine instead of a zero when a number is divisible by nine. The examples below help clarify the process of computing the digital root of a given input.

Input value	1st Sum of Digits	2nd Sum of Digits	Digital Root
1,234,567	1+2+3+4+5+6+7=28	2+8 = 10	1
314159264	3+1+4+1+5+9+2+6+4=35	3+5 = 8	8
9,999,999	9+9+9+9+9+9+9 = 63	6+3 = 9	9

Figure 1: *Digital Root Calculation for Various Inputs*

Because the digital root function ignores the order of magnitude assigned to each digit in the input value, alteration of the digit order does not change the digital root. It also does not matter where the decimal point occurs in an input value. This means that the magnitude of the input has no bearing on the digital root value, a property known as scale invariance.

The logistic map function involves the operations of addition/subtraction and multiplication. Therefore, if we wish to apply digital root analysis to the chaotic regime of the logistic map, then we must first understand the digital root properties of addition and multiplication. It turns out that the sums and products of digital roots follow definite rules based on the digital roots of the input values. With specific regard to addition and multiplication, the sum/product of the output can be determined from knowledge of the digital roots of the inputs. For example, if we wish to use as inputs the two values 5,684,948 and 4,784,434 then we can avoid explicitly calculating their sum/product by simply performing the function on their digital roots.

Digital root calculation of 5,684,948:	Digital root calculation of 4,784,434:
5+6+8+4+9+4+8 = 44 and 4+4 = 8.	4+7+8+4+4+3+4 = 34 and 3+4 = 7.
Digital root of sum:	Digital root of product:
8+7 = 15; 1+5 = 6	8*7 = 56; 5+6 = 11; 1+1 = 2

These results are thus obtained without doing the explicit addition and multiplication calculation to find that the sum of the inputs is 10,469,382 and their product is 27,199,258,499,432. Persistent addition of these values confirms that their sum and product are indeed 6 and 2 respectively.

Because there are a finite set of digital root values, it is possible to construct sum and product tables which represent all possible nonzero input combinations. This is done in figures 2 and 3 below. The column and row headers represent the digital root values of all possible input combinations.

	1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9	1
2	3	4	5	6	7	8	9	1	2
3	4	5	6	7	8	9	1	2	3
4	5	6	7	8	9	1	2	3	4
5	6	7	8	9	1	2	3	4	5
6	7	8	9	1	2	3	4	5	6
7	8	9	1	2	3	4	5	6	7
8	9	1	2	3	4	5	6	7	8
9	1	2	3	4	5	6	7	8	9

Figure 2: Digital Roots of an addition table

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	1	3	5	7	9
3	3	6	9	3	6	9	3	6	9
4	4	8	3	7	2	6	1	5	9
5	5	1	6	2	7	3	8	4	9
6	6	3	9	6	3	9	6	3	9
7	7	5	3	1	8	6	4	2	9
8	8	7	6	5	4	3	2	1	9
9	9	9	9	9	9	9	9	9	9

Figure 3: Digital Roots of a product table.

Figure 3 is also known as a Vedic Square and it contains a variety of patterns and symmetries. Each row and column are equivalent for a given digital root value. The statement ($a_{i,j} + a_{9-i,j}$) ≡ 9 is also true for all i and j. The symbol ≡ is borrowed from modular arithmetic and is used here as shorthand for the digital root function. The pairs {$a_{i,j}$, $a_{9-i,j}$} from figure 3 will be referred to as complementary pairs as they persistently sum for 9. Explicitly, these are {1,8}, {2,7}, {3,6}, {4,5}, and {9,9}. A corollary of this observation is that if the values in a given column or row are persistently summed, the result is 9.

Finally, the order of the elements in the multiplication cycle for a given value is the reverse of that for its complementary value. The cycle for column/row 2 reads (2 4 6 8 1 3 5 7 9) and the column/row for its complement 7 reads (7 5 3 1 8 6 4 2 9). The first 8 digits of these cycles are the reverse of one another and both terminate with 9. If figure 3 were to be extended to 17 rows and columns, then the two patterns for row/column 2 and row/column 7 would be the exact reverse of one another with 9 in the center. Explicitly, these would be (2 4 6 8 1 3 5 7 9 2 4 6 8 1 3 5 7) and (7 5 3 1 8 6 4 2 9 7 5 3 1 8 6 4 2). This symmetry holds for all complementary pairs.

So long as two numbers have the same digital root, their exponentiations will result in the same numerical sequences. Figures 4 and 5 demonstrate this principle. We see that the decimal expressions of 2^n and 11^n always have equal digital roots for a given n. This is always true for values separated by 9.

N	Base 1	DR	Base 2	DR	Base 4	DR	Base 5	DR	Base7	DR	Base 8	DR
1	1	1	2	2	4	4	5	5	7	7	8	8
2	1	1	4	4	16	7	25	7	49	4	64	1
3	1	1	8	8	64	1	125	8	343	1	512	8
4	1	1	16	7	256	4	625	4	2401	7	4096	1
5	1	1	32	5	1024	7	3125	2	16807	4	32768	8
6	1	1	64	1	4096	1	15625	1	117649	1	262144	1
7	1	1	128	2	16384	4	78125	5	823543	7	2097152	8
8	1	1	256	4	65536	7	390625	7	5764801	4	16777216	1
9	1	1	512	8	262144	1	1953125	8	40353607	1	134217728	8
10	1	1	1024	7	1048576	4	9765625	4	282475249	7	1073741824	1
11	1	1	2048	5	4194304	7	48828125	2	1977326743	4	8589934592	8
12	1	1	4096	1	16777216	1	244140625	1	13841287201	1	68719476736	1
13	1	1	8192	2	67108864	4	1220703125	5	96889010407	7	5.49756E+11	8
14	1	1	16384	4	268435456	7	6103515625	7	6.78223E+11	4	4.39805E+12	1

Figure 4*: Table showing closure under integral exponentiations of the subgroup {1, 2, 4, 5, 7, 8}.*

N	Base 19	DR	Base 11	DR	Base 13	DR	Base 14	DR	Base16	DR	Base 17	DR
1	19	1	11	2	13	4	14	5	16	7	17	8
2	361	1	121	4	169	7	196	7	256	4	289	1
3	6859	1	1331	8	2197	1	2744	8	4096	1	4913	8
4	130321	1	14641	7	28561	4	38416	4	65536	7	83521	1
5	2476099	1	161051	5	371293	7	537824	2	1048576	4	1419857	8
6	4.7E+07	1	1771561	1	4826809	1	7529536	1	1.7E+07	1	24137569	1
7	8.9E+08	1	1.9E+07	2	6.3E+07	4	1.05E+08	5	2.7E+08	7	4.1E+08	8
8	1.7E+10	1	2.1E+08	4	8.2E+08	7	1.48E+09	7	4.3E+09	4	6.98E+09	1
9	3.2E+11	1	2.4E+09	8	1.1E+10	1	2.07E+10	8	6.9E+10	1	1.19E+11	8
10	6.1E+12	1	2.6E+10	7	1.4E+11	4	2.89E+11	4	1.1E+12	7	2.02E+12	1

Figure 5*: Table showing increasing integral exponentiations of two digit values with digital roots 1,2,4,5,7, and 8. Comparison with figure 4 reveals the scale invariant nature of digital root patterns.*

If we wish to form an insightful description of the patterns in figures 2 and 3, then it is helpful to find the most logical method of grouping the terms {1:9} so that the simplest possible rules can be ascribed to their behavior. In the parlance of number theory, the set {1:9} under the digital root function forms a finite field. More specifically, they would constitute a sub-classification called a commutative ring as discussed in section 3.2.1 [Blyth]. This is because not all of the elements of the set {1:9} are relatively prime to the modulus value of 9. Specifically, the terms in the set {3,6,9} are all divisible by 3. This observation provides some additional insight into how to group the terms {1:9} as they exhibit properties unique to the set {3,6,9}.

If the elements {1:9} are sub-categorized as {1,2,4,5,7,8} and {3,6,9} then we obtain two sets which are both closed under multiplication and thus exponentiation. Additionally, note that the digital root of any multiplication string will be a member of the set {3,6,9} if and only if the digital root of one or more of the elements being multiplied is in the set {3,6,9}. This property can be thought of as an attractor property of the set {3,6,9} under multiplication. The set {3,6,9} is also closed under addition. Finally, the set {9} is also closed under addition and multiplication, is an attractor set under multiplication, and is the additive identity element.

2.2.3 Scale Invariance in the Logistic Map

We are now equipped to detect periodicity in the chaotic regime of the logistic map. Because the discrete logistic function has two inputs and each input has 9 possible values, there are 81 unique combinations of digital root input values. These are shown in figure 6 below. Red values indicate that the computation has reached a steady state cycle. Again, all entries in figure 6 are digital root values.

r	Xn	Xn+1	Xn+2	Xn+3	Cycle
1	1	9	9		[9]
2	1	9	9		[9]
3	1	9	9		[9]
4	1	9	9		[9]
5	1	9	9		[9]
6	1	9	9		[9]
7	1	9	9		[9]
8	1	9	9		[9]
9	1	9	9		[9]

r	Xn	Xn+1	Xn+2	Xn+3	Xn+4	Cycle
1	4	6	6			[6]
2	4	3	6	3	6	[3,6]
3	4	9	9			[9]
4	4	6	6			[6]
5	4	3	6	3	6	[3,6]
6	4	9	9			[9]
7	4	6	6			[6]
8	4	3	6	3	6	[3,6]
9	4	9	9			[9]

r	Xn	Xn+1	Xn+2	Xn+3	Xn+4	Cycle
1	7	3	3			[3]
2	7	6	3	6	3	[6,3]
3	7	9	9			[9]
4	7	3	3			[3]
5	7	6	3	6	3	[6,3]
6	7	9	9			[9]
7	7	3	3			[3]
8	7	6	3	6	3	[6,3]
9	7	9	9			[9]

r	Xn	Xn+1	Xn+2	Xn+3	
1	2	7	3	3	[3]
2	2	5	5		[5]
3	2	3	9	9	[9]
4	2	1	9	9	[9]
5	2	8	8		[8]
6	2	6	9	9	[9]
7	2	4	6	6	[6]
8	2	2	2		[2]
9	2	9	9		[9]

r	Xn	Xn+1	Xn+2	Xn+3	Xn+4	
1	5	7	3	3		[3]
2	5	5				[5]
3	5	3	9	9		[9]
4	5	1	9	9		[9]
5	5	8	8			[8]
6	5	6	9	9		[9]
7	5	4	6	6		[6]
8	5	2	2			[2]
9	5	9	9			[9]

r	Xn	Xn+1	Xn+2	Xn+3	Xn+4	
1	8	7	3	3		[3]
2	8	5	5			[5]
3	8	3	9	9		[9]
4	8	1	9	9		[9]
5	8	8				[8]
6	8	6	9	9		[9]
7	8	4	6	6		[6]
8	8	2	2			[2]
9	8	9	9			[9]

r	Xn	Xn+1	Xn+2	Xn+3	
1	3	3			[3]
2	3	6	3	6	[3,6]
3	3	9	9		[9]
4	3	3			[3]
5	3	6	3	6	[3,6]
6	3	9	9		[9]
7	3	3			[3]
8	3	6	3	6	[3,6]
9	3	9	9		[9]

r	Xn	Xn+1	Xn+2	Xn+3	Xn+4	
1	6	6				[6]
2	6	3	6	3		[6,3]
3	6	9	9			[9]
4	6	6				[6]
5	6	3	6	3		[6,3]
6	6	9	9			[9]
7	6	6				[6]
8	6	3	6	3		[6,3]
9	6	9	9			[9]

r	Xn	Xn+1	Xn+2	Xn+3	Xn+4	
1	9	9				[9]
2	9	9				[9]
3	9	9				[9]
4	9	9				[9]
5	9	9				[9]
6	9	9				[9]
7	9	9				[9]
8	9	9				[9]
9	9	9				[9]

Figure 6: *Table showing all possible combinations of digital root input values of the logistic map.*

A sample calculation helps illustrate the application of figure 6. If we begin with r = 3.67 and X_n = .62 then we calculate the first iterate X_{n+1} to be .864652. From the value of X_{n+1} we then calculate the value of X_{n+2} to be .42949613234832. Persistent addition of these values shows that X_n, X_{n+1}, and X_{n+2} have digital root values of 8, 4, and 6 respectively. Using input values of $r \equiv 7$ and $X_n \equiv 8$, the table in figure 6 correctly predicts the digital root values calculated above. Note that this prediction is possible without the need for explicit computation of the outputs of the map. Note also that when r = 3.67 the map is in the chaotic regime. This demonstrates that, despite the ubiquitous characterization of chaos as "aperiodic," it is possible to find periodic structure inside the chaotic regime of the logistic map.

This digital root periodicity is present despite the value of each successive iterate changing in a chaotic manner. This implies that there is some form of dynamic numerical equilibrium that is maintaining structure in the digital root even though the overall value is wildly changing. Such equilibrium suggests highly symmetric structure beneath chaos which regulates the equilibrium. Absent a scale invariant analytical tool such as digital root analysis, this structure cannot be seen as the observer will not be able to make sense (structure) of the data as it changes scales.

2.2.4 Digital Root Symmetry

Let us now reconsider the structure of Figure 3 such that we may form a deeper understanding of the symmetry at work. Imagine that we were to cut the rows of figure 3 into horizontal strips and then overlay the terminal "9" in each cycle. If we restrict ourselves to two dimensions and organize the multiplication cycles such that complementary pairs align, then the pairs {1,4,7},{2,5,8}, and {3,6,9} spontaneously emerge. Although there are several ways to execute this process, all permutations of the result are reflections and/or rotations of the same number grid. Figure 7 shows one possible permutation of this process.

Figure 7 below is a more exact representation of the symmetries first seen in figure 3. The upper left to lower right diagonal shows the digital roots of multiples of 8. The reverse direction shows the multiples of its complementary value 1. Similarly, the upper right to lower left diagonal shows multiples of 5 and its reverse direction displays multiples of 4. The same logic applies to the vertical and horizontal directions with regard to multiples of {2,7} and {3,6} respectively. This also means that every value of 9 in the grid has a complementary set of values equidistant from it in all directions.

8	5	2	8	5	2	8	5	2	8	5	2	8	5	2	8	5
1	7	4	1	7	4	1	7	4	1	7	4	1	7	4	1	7
3	9	6	3	9	6	3	9	6	3	9	6	3	9	6	3	9
5	2	8	5	2	8	5	2	8	5	2	8	5	2	8	5	2
7	4	1	7	4	1	7	4	1	7	4	1	7	4	1	7	4
9	6	3	9	6	3	9	6	3	9	6	3	9	6	3	9	6
2	8	5	2	8	5	2	8	5	2	8	5	2	8	5	2	8
4	1	7	4	1	7	4	1	7	4	1	7	4	1	7	4	1
6	3	9	6	3	9	6	3	9	6	3	9	6	3	9	6	3
8	5	2	8	5	2	8	5	2	8	5	2	8	5	2	8	5
1	7	4	1	7	4	1	7	4	1	7	4	1	7	4	1	7
3	9	6	3	9	6	3	9	6	3	9	6	3	9	6	3	9
5	2	8	5	2	8	5	2	8	5	2	8	5	2	8	5	2
7	4	1	7	4	1	7	4	1	7	4	1	7	4	1	7	4
9	6	3	9	6	3	9	6	3	9	6	3	9	6	3	9	6
2	8	5	2	8	5	2	8	5	2	8	5	2	8	5	2	8
4	1	7	4	1	7	4	1	7	4	1	7	4	1	7	4	1

Figure 7: *Alternate layout of the multiplication cycles from figure 3 in which the element 9 is shared. The sets {1,7,4}, {3,9,6}, and {5,2,8} emerge in horizontal rows. This arrangement represents a more condensed representation of the symmetries found in figure 3.*

Not only does figure 7 preserve perfect numerical symmetry about each value of 9, but it also exhibits scale invariance. That is, when the digital root of each 2x2 square is calculated, the result is either a reflection or rotation of figure 7. Imagine pulling back the perspective from figure 7 until each individual square is composed of a 2x2 block. If the digital root of each 2x2 block is taken, then permutation of figure 7 results. This permutation captures all of the same symmetry properties as figure 7. This is equivalent to viewing the grid from a higher perspective.

The entire grid in figure 7 can be generated from repetition of the central colored 3x3 square in which the values {1:9} all occur exactly once. This means that permutations of the grid are only really permutations of the values in the central 3x3 square. Although it is too laborious to present here, it can be shown by exhaustive enumeration that permutations of figure 7 only exhibit perfect symmetry when the elements {1:9} are grouped into the row/column sets {1,4,7} {2,5,8} and {3,6,9}.

As a consequence of these properties, the digital root of any 3x3 set of nine squares is actually equal to 9. Equivalently, the digital root of the perimeter of any 3x3 set of squares is equal to the complement of the central value. This observation suggests that figure 7 answers the question "How does one populate a Cartesian grid with digital roots such that the digital root of each 3x3 square is equal to nine?" This is equivalent to the question "How does one populate a Cartesian grid with digital roots such that the digital root of the 8 nearest neighbors of each cell is equal to its complement?"

Because of the exact symmetries found in figure 7, and the extreme care taken in its deduction, it is unlikely that such a fixed state would arise by brute force computation. This would be advantageous if one were trying to produce irreducible data as it would provide a situation in which the system generating such data would be extremely unlikely to fall into a fixed state; a necessary but not sufficient condition for a PRNG. This is the motivating concept behind the cellular automata driven image encryption system that follows.

3. Cellular Automaton Encryption System

3.1 Computational Dynamics

Let us assume that figure 7 and its reflections/rotations are the only non-trivial combination of cell assignments that will answer the question posed above. This would imply that an update rule which calculates the digital roots of the 8 nearest neighboring values on a Cartesian grid would be very desirable to use as a computational rule as it would be extremely unlikely to fall into a single fixed state. Its dynamics would therefore be both interesting and possibly useful for the generation of irreducible data.

To test this hypothesis, a cellular automaton was created in which each cell becomes the digital root of its 8 nearest neighbors on each update. By starting with an initial fixed state in which every cell has value 9 and then perturbing that state with a single initial condition, the nature of the update rule can be visualized. Because the "background" of the grid is the additive identity, all changes in the state of the automaton are due entirely to the effect of successive updates on the single initial condition.

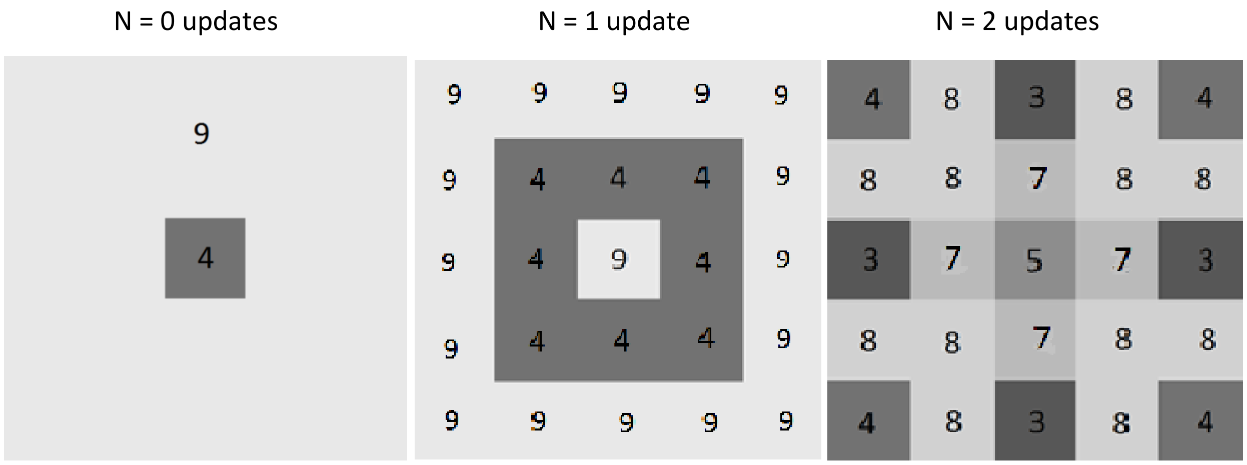
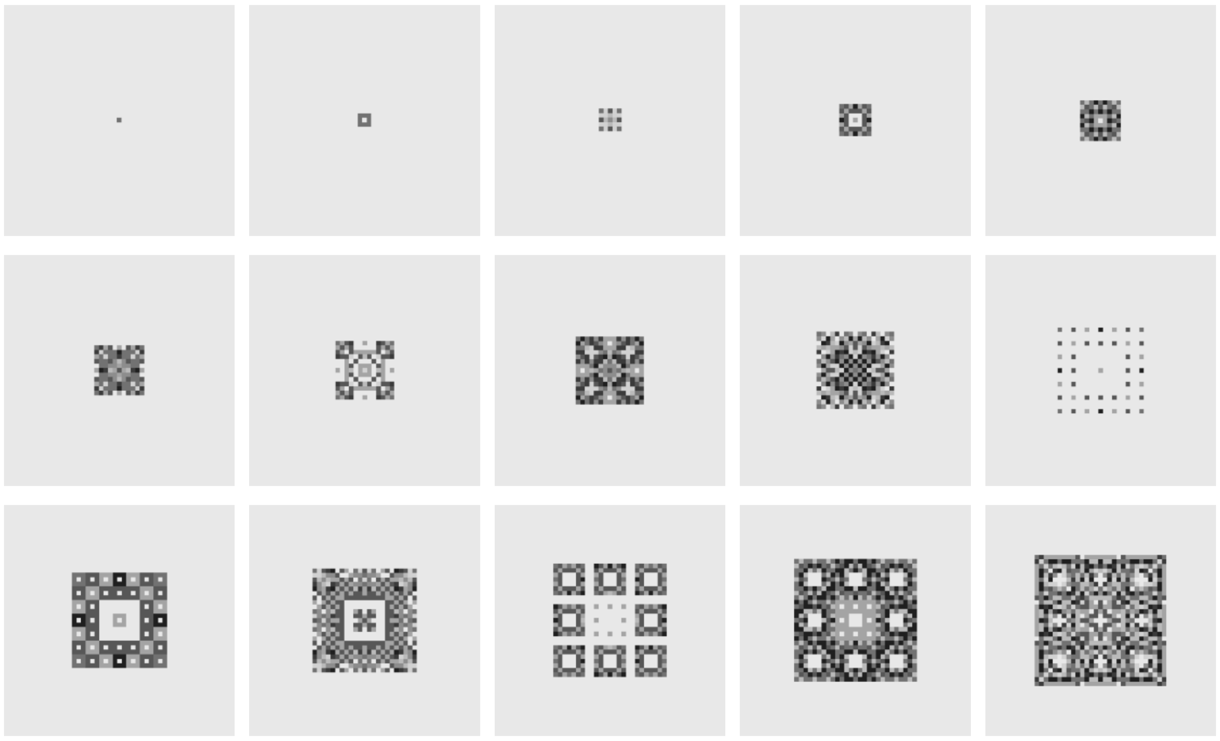


Figure 8*: The first three iterations of a digital root based Cellular Automaton on a 5x5 grid. The initial background color represents the value 9 and the initial condition in the center is the value 4.*

To better visualize the progressive nature of this update rule, the first 25 iterations of initial condition 4 are presented in figure 9 below.



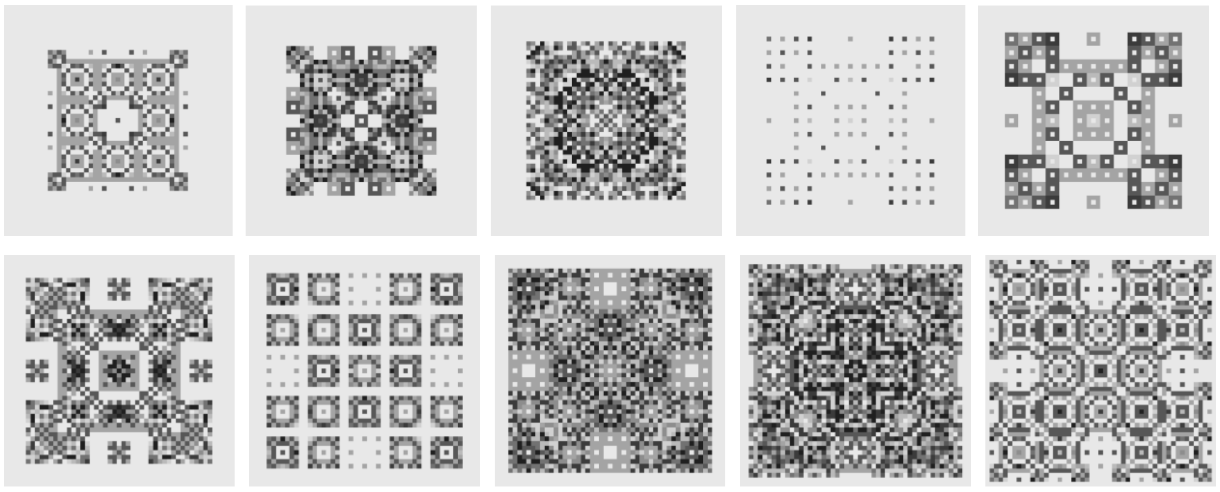
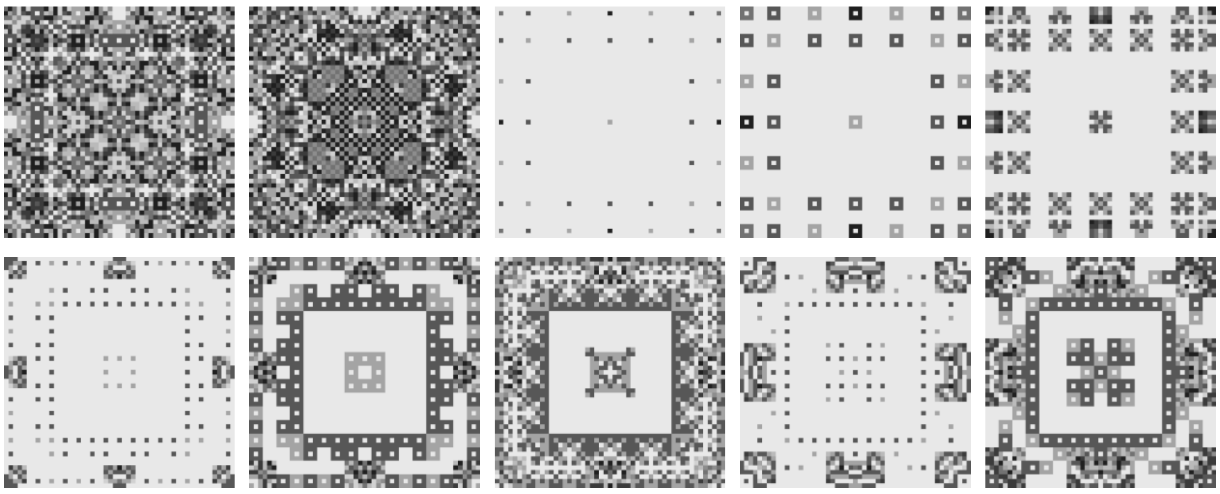


Figure 9: *The first 25 cellular automata updates listed from left to right. The initial condition used is 4 and the grid is a 51x51 square.*

If we were to continue computing the final image in figure 9, then we would reach the boundary of the grid. To allow the computation to continue, we "wrap" the edges of the grid such that the neighbors of each edge cell are the cells on the opposite edge. This is equivalent to wrapping the grid around a torus. Further computation of the rule then allows the automaton to feedback on itself as the information being emitted from the initial condition collides with itself upon reaching the edge. The next 25 computations in which this feedback occurs are given in figure 10 below.



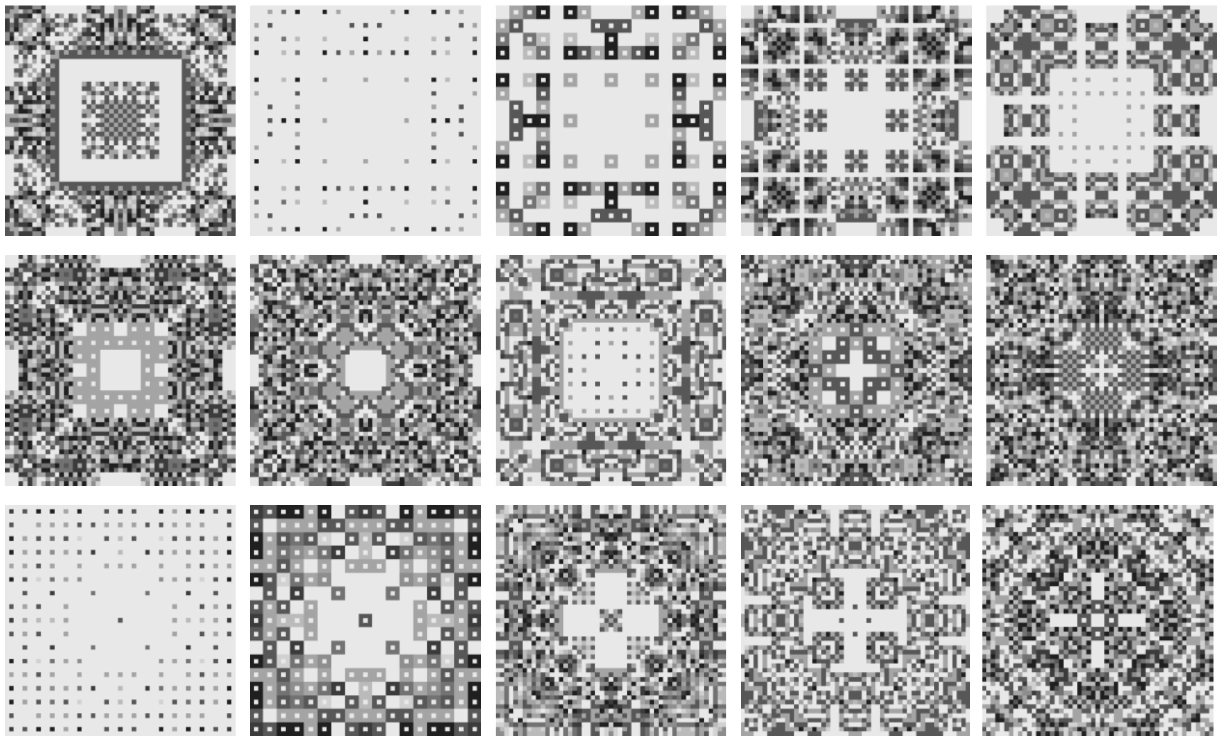


Figure 10: *Continuation of figure 9 showing iterates 25-49 of the update rule. The initial condition used is 4 and the grid is a 51x51 square.*

3.2 Self-Similarity of the Cellular Automata

Deeper examination of the rule shows that it also produces self-similarity; the structures shown below highlight 3 instances of self-similar structure at 3 different scale resolutions.

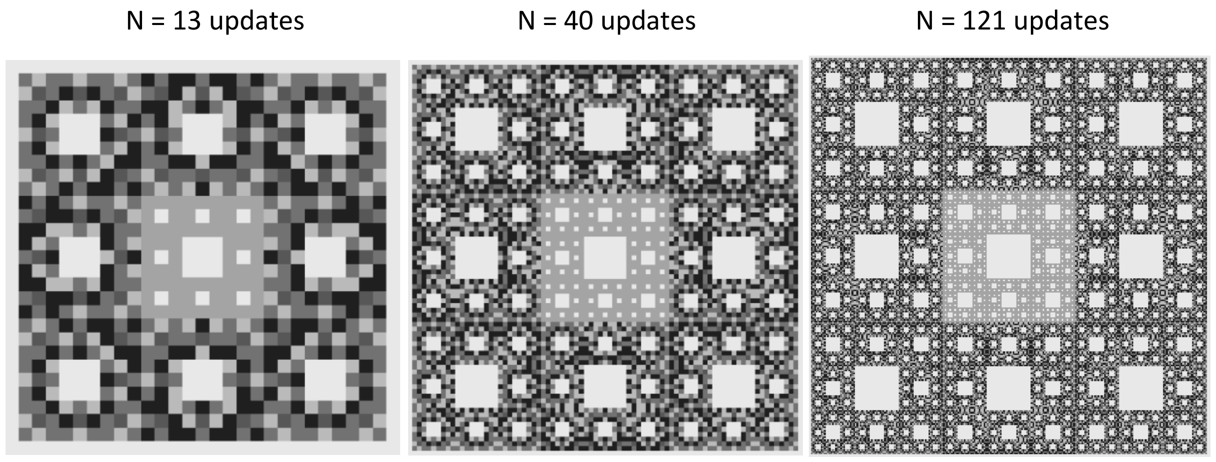


Figure 11: *Three different sized square grids showing updates of 13, 40 and 121 of the update rule.*

There is clear self-similarity demonstrated as the set of 9 nested squares increases in resolution. The grid sizes are 27x27, 81x81, and 243x243 respectively.

3.3 Critical Transitions

Let us now consider the effect of changes in the dimensions of the grid on the repetitive cycle length exhibited by the computational rule. Figure 10 shows the results of computations conducted by searching for repetitive states in the update rule while varying the dimensions of an m x n torus.

Square Tori			**Rectangular Tori**		
m	**N**	**Cycle Length**	**m**	**n**	**Cycle Length**
9	9	54	6	12	18
10	10	24	7	11	43046718
11	11	726	8	10	24
12	12	18			
13	13	78			
14	14	78			
15	15	72	9	15	54
16	16	240	11	13	43046718
17	17	19680	10	14	2184

Figure 12: *State cycle lengths of update rule when computed on various size tori.*

Although the state cycle for a 12x12 square torus is only 18 iterations long, the state cycle for an 11x13 rectangular torus leaps to 43,046,718. Although this is only a change in the dimension of a 12x12 torus by about 8%, the state cycle changes by approximately 239,148,400%. A similar result occurs when moving from a 9x9 square torus to a 7x11 rectangular torus. These results suggest that the torus geometry may be "tuned" to find critical points which do not behave in an easily predictable manner.

When a small change in the object parameters of a system results in a large change in the output of a system, it is known as a critical transition. At a critical transition point, a system cannot be understood from an understanding of its parts [Scheffer 461]. This could be advantageous if we wish to produce apparently random output as the output of the system will not be understandable by a simple analysis of its parts.

3.4 Symmetric Key Image Encryption:

Because images are inherently 2 dimensional, they pair naturally with two dimensional cellular automata. The cellular automaton rule presented so far can be expanded to include larger radix values and thus can be used to encrypt images. This can be accomplished simply by taking an initial hidden state of the automaton (the key) and computing sufficiently far so as to produce apparently random data. The resulting matrix of values, or pad, can then be added to the image via modulo addition. If the content of the pad is irreducible, then it will effectively obscure the image. This process is demonstrated visually in the figure below.

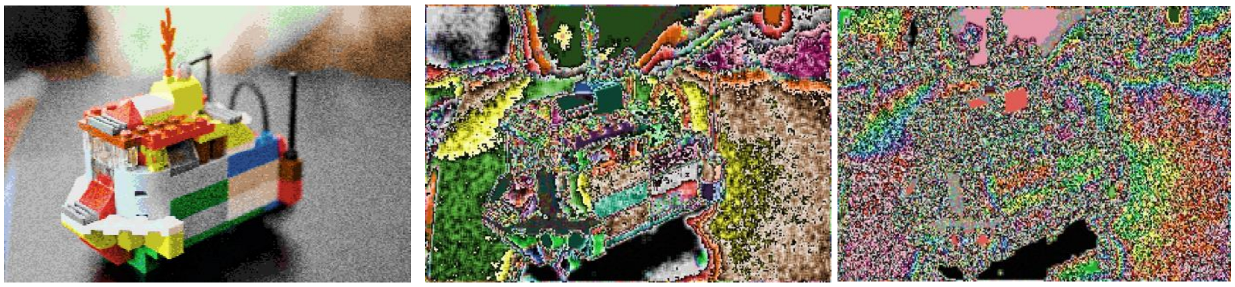


Figure 13: The original key state and the key after one and two updates respectively. After two updates, the original image is almost unrecognizable. An image is used for the initial key state to show that the initial state is not composed of “random” data.

The result of this process is that the cellular automaton algorithm produces a stream of randomized grids known as pads. The pads can be added via modulo addition to plaintext images in order to achieve encryption. To decrypt the image, the pad is simply subtracted out to recover the plaintext image. This type of cipher is known as a stream cipher and the protocol is a symmetric encryption scheme(reference).

Successive computations of the initial key state result in a grid resembling white noise. Because this static is generated deterministically, knowledge of the original key state can be used to compute the grid. This is not an easy accomplishment for anyone but the holder of the key.

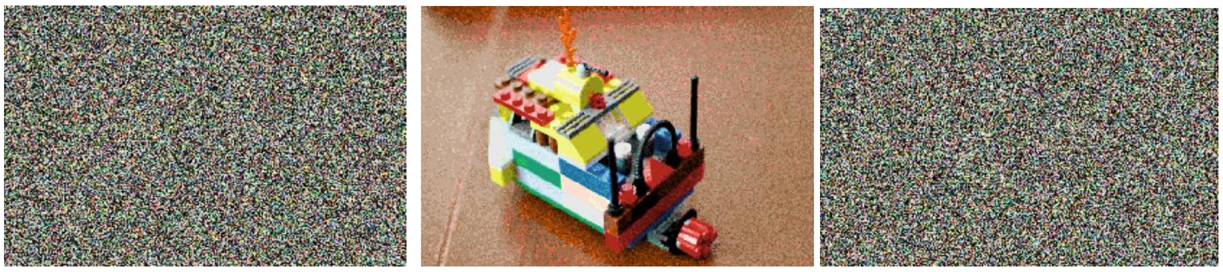


Figure 14: *The original key after 100 iterations, plaintext, and ciphertext respectively. The ciphertext is generated by modulo addition of the pad and plaintext. This process can be reversed if the pad is known.*

4. Conclusions

The observation of periodic structure within discrete chaos should elicit serious and sincere attention. Although it has no agreed upon definition, descriptions of chaos in texts and literature universally refer to chaos as being aperiodic and unpredictable. Yet, as demonstrated in this thesis, digital root analysis shows that there is both periodicity and predictability of discrete chaotic data. Because chaos is embedded in such disparate disciplines as physics, chemistry, biology, geology, finance, optimization, and encryption, any change in the understanding of its structure will necessarily affect those areas of inquiry. As such, the work presented here deserves a deeper and more comprehensive investigation so as to put these observations into a broader context.

Finally, the assembly of scale invariant digital root patterns into scale invariant 3D geometry may represent a framework for understanding the equilibrium dynamics that seem to lurk beneath discrete chaos. Specifically, this geometry may present a latticework upon which simple computational rules can be applied so as to produce apparently complex output. Because this geometry is scale invariant, such computations may prove valuable for modeling scale invariant behavior in natural processes.

5. Future Work

5.1 Image Encryption Improvement

The image encryption protocol presented in this thesis is not intended to be a finalized system. In order to achieve good security, the output of the cellular automaton algorithm described here must be repeatedly tested and vetted to ensure it is not vulnerable to systematic attack. One way to achieve a far greater level of data randomization without additional computational duty is to introduce a shift operation upon the edges of the torus. This involves assigning neighbors at the edge of the torus to cells which are not directly opposite the edge bound cells. Topologically, this is equivalent to twisting the torus. The degree of offset on each axis should also be relatively prime to the corresponding geometric dimension of each axis. This will ensure that the periodic cycle of the CA output is as long as possible. The exact values of these parameters should also be included as part of the system's cryptographic key.

Another important improvement upon this cryptosystem is to reduce the size of the key as much as possible. In an ideal system, the number of potential key states should be exactly equivalent to the number of independent key attempts needed by an attacker to break the system. This means that the size of the key is theoretically minimal; thus minimizing the quantity of information that must be kept secret.

5.2 Three Dimensional Digital Root Symmetry

Because there are only nine possible digital root values for all quantities, we may envision that their relationships could be captured in a very simple 3D geometry. If we begin by listing the sharpest symmetry constraints observed in digital root analysis, we may assert that the following 3 constraints must hold.

1. 9 must be in the geometric center.
2. Complementary pair elements must be collinear with, and equidistant from, 9.
3. The organization of the 9 elements must adhere to subsets R1 {1,4,7}, R2 {2,5,8}, and R3 {3,6,9}.

These three constraints can all be understood by examination of figure 7. Clearly, 9 must be in the center as this allows the multiples of complementary values to all be symmetric about 9. This also implies that complementary values must be collinear. The emergence of groups R1, R2, and R3 in the horizontal rows demonstrates that the digital roots "want" to self-organize in this way.

Initially, these constraints seem to imply the structure of a cube. All of the elements {1:8} are assigned a corner and {9} is in the center. This model initially seems to fit all of the constraints but in fact it is impossible to assign numbers to the corners such that the systems R1 and R2 are not directly connected. If there were a configuration in which the elements of groups R1 and R2 could be further segregated, it would be superior to the cube model as it would more fully satisfy the third constraint. This can be accomplished by connecting each corner of the cube to the three corners which are diagonally opposite each face. The result is a regular polyhedral compound known as a stellated octahedron as seen in figure 15. This arrangement allows each element of the sets R1 and R2 to remain indirectly connected and thus more fully satisfies constraint 3.

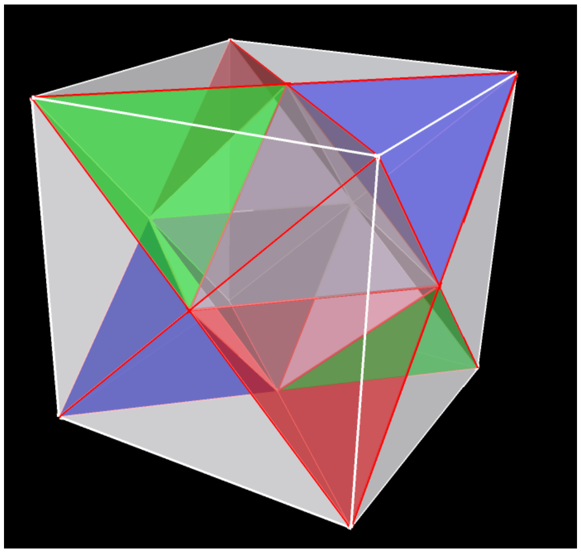


Figure 15: *A stellated octahedron is generated when each corner of a cube is connected to its three diagonally opposite corners.*

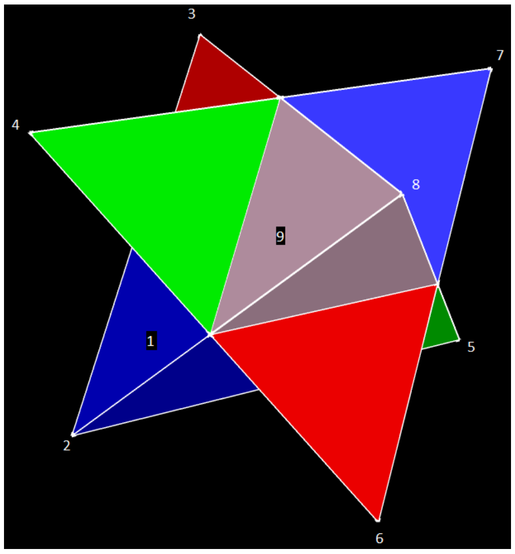


Figure 16: A stellated octahedron with vertices numbered according to the three symmetry constraints. This is the 3D analog of figure 7.

Amazingly, the stellated octahedron displays three dimensional scale invariant properties when grown according to the behavior of digital roots. That is, when 8 stellated octahedrons are arranged at the vertices of a stellated octahedron, the resulting geometry depicts a larger stellated octahedron. The vectors of the larger stellated octahedron are exactly twice the length of the corresponding vectors in the constituent stellated octahedrons. More specifically, it produces a polyhedral compound of 64 tetrahedrons which forms a stellated octahedron at its outer edges.

Even more amazingly, this resultant polyhedral compound also exhibits a second scale invariant geometry in the form of a central cuboctahedron. That is, the eight stellated octahedrons all have one tetrahedron pointing toward a common center, thus producing a cuboctahedron. Because the 8 tetrahedrons which form this central cuboctahedron are but the “tips” of larger tetrahedrons, there is also a larger cuboctahedron formed at the perimeter of the 64 tetrahedron network. The length of the vectors in the outer cuboctahedron is exactly twice the vector length in the inner cuboctahedron. A wireframe depiction showing both the inner and outer cuboctahedrons appears in figure 20.

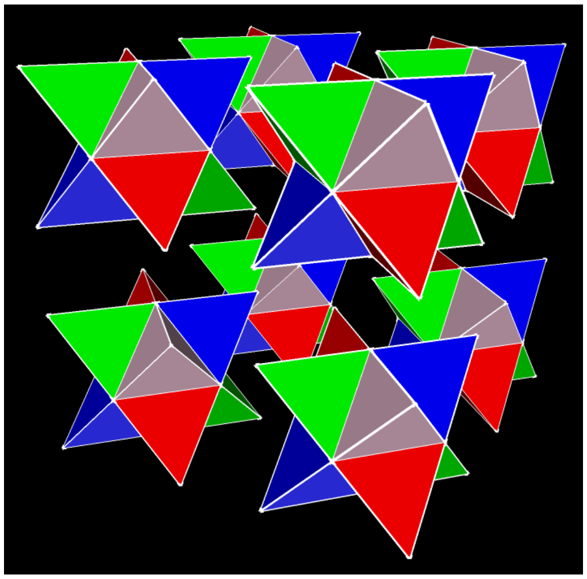


Figure 17: Depiction of 8 stellated octahedrons coming together to form a larger structure. This is an exploded view of figure 18.

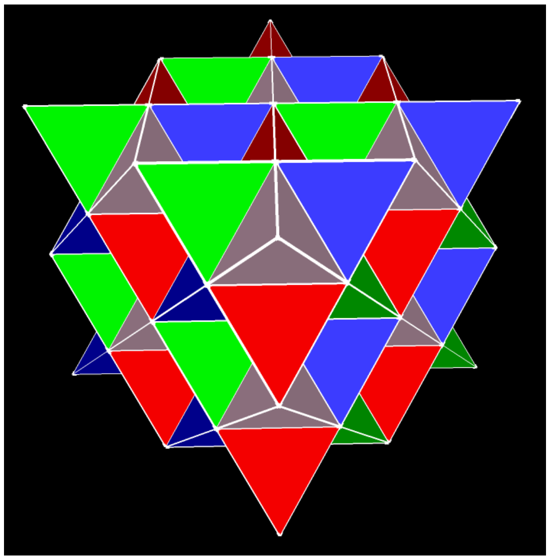


Figure 18: Axial View of the geometry generated when 8 stellated octahedrons are connected to a common center.

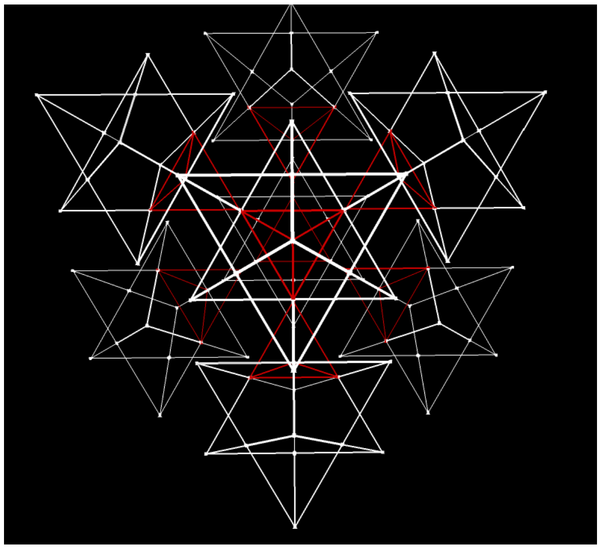


Figure 19: Exploded wireframe of the geometry in figure 18. The 8 interior tetrahedrons pointing to a common center are highlighted.

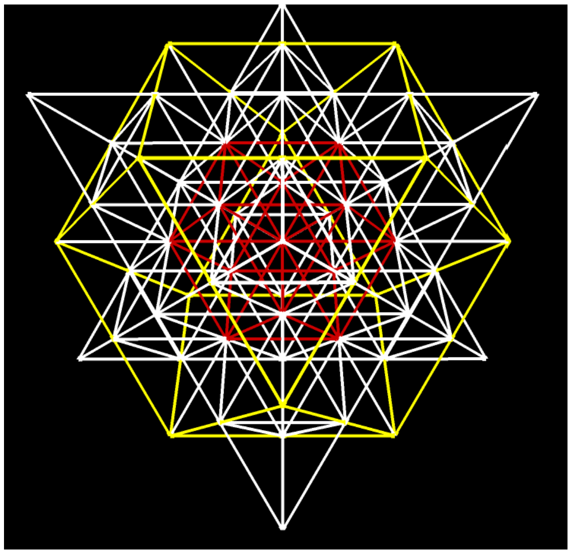


Figure 20: Wireframe depiction of the geometry in figure 18. This view reveals the innermost cuboctahedron in red and the outermost cuboctahedron in yellow.

The periodicity of digital root values in the chaotic regime of the logistic map indicates the presence of a dynamic numerical equilibrium. When the map is iterated, the output values change in an unpredictable way, yet the sum of the digits change in such a way as to only add or remove multiples of 9 (over the course of an entire periodic cycle in the digital root). This suggests the presence of an underlying rule structure which, if identified, could lend far greater insight into the underlying dynamics of the map. Further, the scale invariant nature of the digital root function implies that the underlying dynamics it reveals must also be scale invariant.

6. Appendix:

6.1 Netlogo Code

Netlogo code for modular arithmetic cellular automata:

```
Globals [Match lock_1 lock_2 lock_3 lock_4 lock_5 lock_6 lock_7 lock_8
  lock_1px lock_2px lock_3px lock_4px lock_5px lock_6px lock_7px lock_8px
  lock_1py lock_2py lock_3py lock_4py lock_5py lock_6py lock_7py lock_8py
  Key_1 Key_2 Key_3 Key_4 Key_5 Key_6 Key_7 Key_8]

Patches-own [NeighborSum DRNeighborSum Key_Pcolor]

To Setup
  __clear-all-and-reset-ticks
  Set Match (0)
  resize-world 0 (World_Xmax - 1) 0 (World_Ymax - 1)
  set-patch-size (Grid_Size)
  ask patches [set pcolor Mod_Val]
  ask patch (World_Xmax / 2 - 1) (World_Ymax / 2 - 1) [set pcolor Initial_Condition_1]
  if IC2_Switch = true [ask patch IC2_xcor IC2_ycor [set pcolor Initial_Condition_2]]
  if Rand_Switch = true [repeat Randomize [ask patch random-pxcor random-pycor [set pcolor Random Mod_Val]]]
  display

  set lock_1px random-pxcor
  set lock_2px random-pxcor
  set lock_3px random-pxcor
  set lock_4px random-pxcor
  set lock_5px random-pxcor
  set lock_6px random-pxcor
  set lock_7px random-pxcor
  set lock_8px random-pxcor

  set lock_1py random-pycor
  set lock_2py random-pycor
  set lock_3py random-pycor
  set lock_4py random-pycor
  set lock_5py random-pycor
  set lock_6py random-pycor
  set lock_7py random-pycor
  set lock_8py random-pycor
```

```
End

To Invert                              ;returns the additive inverse of all patches
  set pcolor Mod_Val - pcolor
  if pcolor = 0 [set pcolor Mod_Val]
End

To ResetMatch
  set match 0
End

To SetupMatch
  set match 0
  ask patch lock_1px lock_1py [set lock_1 pcolor]  ;sets lock values
  ask patch lock_2px lock_2py [set lock_2 pcolor]
  ask patch lock_3px lock_3py [set lock_3 pcolor]
  ask patch lock_4px lock_4py [set lock_4 pcolor]
  ask patch lock_5px lock_5py [set lock_5 pcolor]
  ask patch lock_6px lock_6py [set lock_6 pcolor]
  ask patch lock_7px lock_7py [set lock_7 pcolor]
  ask patch lock_8px lock_8py [set lock_8 pcolor]
End

To ColorUpdate
  set NeighborSum (sum [pcolor] of neighbors)  ;;show sum of neighboring cells and Digital Root of that sum.
  set DRNeighborSum (NeighborSum mod Mod_Val)
  if DRNeighborSum = 0 [set DRNeighborSum Mod_Val]

  set pcolor (DRNeighborSum)
  ;;set pcolor (pcolor mod Mod_Val)
  ;;if pcolor = 0 [set pcolor Mod_Val]
End

To Search
  ColorUpdate
  ask patch lock_1px lock_1py [set Key_1 pcolor]  ;sets all 8 lock patches with random coordinates
  ask patch lock_2px lock_2py [set Key_2 pcolor]  ;there is no guarantee of uniqueness of patches but
  ask patch lock_3px lock_3py [set Key_3 pcolor]  ;the sample size is small engough to make duplication
  ask patch lock_4px lock_4py [set Key_4 pcolor]  ;highly unlikely
  ask patch lock_5px lock_5py [set Key_5 pcolor]
  ask patch lock_6px lock_6py [set Key_6 pcolor]
  ask patch lock_7px lock_7py [set Key_7 pcolor]
  ask patch lock_8px lock_8py [set Key_8 pcolor]

  Ifelse (lock_1 = Key_1 and lock_2 = Key_2 and lock_3 = Key_3 and lock_4 = Key_4  ;If the key fits the lock, match =
1
    and lock_5 = Key_5 and lock_6 = Key_6 and lock_7 = Key_7 and lock_8 = Key_8)   ;This stops the search button
code
  [set Match (1)] [set Match (0)]
end

;*******Below is the code for the buttons involved in the encryption protocol process*******
;**********Key Import and Save**************
```

```
To Import_Key_Seed
  import-pcolors "C:\\Users\\John\\Pictures\\Encryption\\Key.jpg";
  ask patches [set pcolor (pcolor mod Mod_Val)]
  ask patches [set pcolor (floor pcolor)]
End

To Store_Key
ask patches [set Key_Pcolor pcolor]
End

;**********Plaintext Import, Encrypt, Save*******
To Import_Plaintext
  import-pcolors "C:\\Users\\John\\Pictures\\Encryption\\Plaintext.jpg"
  ask patches [set pcolor (pcolor mod Mod_Val)]
  ask patches [set pcolor (floor pcolor)]
End

To Encrypt_Plaintext
  ask patches [set pcolor ((pcolor + Key_Pcolor) mod Mod_Val)]
End

To Save_Cyphertext
  export-view "C:\\Users\\John\\Pictures\\Encryption\\Cyphertext.png"
End

To Decrypt_Cyphertext
  ask patches [set pcolor ((pcolor + Mod_Val - Key_Pcolor) mod Mod_Val)]
End

To Import_Cyphertext
  import-pcolors "C:\\Users\\John\\Pictures\\Encryption\\Cyphertext.png"
  ask patches [set pcolor (pcolor mod Mod_Val)]
  ask patches [set pcolor (floor pcolor)]
End
```

6.2 References

Baranger, M. (2010). Chaos, complexity, and entropy; a physics talk for non-physicists. *Department of Physics; Massachusetts Institute of Technology*.

Blyth, T.S.; Robertson, E. F. (1985), *Groups, rings and fields: Algebra through practice*, Cambridge University Press

Chaitin, G. (1987). *Algorithmic information theory*. Cambridge, MA: Cambridge University Press.

Click, T.H. *Quality of random number generators significantly affects results of Monte Carlo simulations for organic and biological systems* J Comput Chem. 2011 February; 32(3): 513–524.

Delfs, Hans & Knebl, Helmut (2007). "Symmetric-key encryption". *Introduction to cryptography: principles and applications*. Springer

Herring, C. (1995). Communications of the Army Corp of Engineers. Vol. 38 Issue 1 Jan. 1995 121-122

Kankaala, K. (1995). A comparitive study of some pseudorandom number generators. *Comput.Phys.Commun., 86*, 210-226.

Lafe, O. (1996). Data compression and encryption using cellular automata transforms. *Intelligence and Systems, 1996., IEEE International Joint Symposia on*, 234-241.

Lorenz, Edward N. (March 1963). "Deterministic Nonperiodic Flow". *Journal of the Atmospheric Sciences* **20** (2): 130–141

Machicao, M. (2011). Chaotic encryption method based on life-like cellular automata. *Expert Systems with Applications*, *39*(16), 1-8.

Patidar Vinoid , (2009). A Novel Pseudo Random bit Generator Based on Chaotic Standard Map. *Electronic Journal of Theoretical Physics*. 20, pp.327

Scheffer, M, (2009). Early-Warning Signals for Critical Transitions. *Nature*. 461 (), pp.53-55

Scherer, W. (2007). *How to do systems analysis*. Hoboken, NJ: John Wiley & Sons.

Shannon, C. (1949). Communication theory of secrecy systems.

Singh, S. (1999). *The code book*. New York : Anchor Books.

Wolfram, S. (2002). *New kind of science*. Champaign, IL: Wolfram Media Place.