Arnold Transformations as Applied to Data Encryption

Haley N. Anderson

ARNOLD TRANSFORMATIONS AS APPLIED TO DATA ENCRYPTION

by

H. NOELLE ANDERSON

(Under the Direction of Yan Wu)

ABSTRACT

As our world becomes increasingly digital, data security becomes key. Data must be encrypted such that it can be easily encrypted only by the intended recipient. Arnold Transformations are a useful tool in this because of its unpredictable periodicity. Our goal is to outline a method for choosing an Arnold Transformation that is both secure and easy to implement. We find the necessary and sufficient condition that a key matrix has periodicity. The chosen key matrix has a random structure, and it has a periodicity that is sufficiently high. We apply this method to several image and data string examples to evaluate its effectiveness.

INDEX WORDS: Arnold transformation, Euclid's algorithm, Diophantine equations, Scrambling transformation periodicity, Encryption, Decryption, Data security

2009 Mathematics Subject Classification: 15A15,41A10

ARNOLD TRANSFORMATIONS AS APPLIED TO DATA ENCRYPTION

by

H. NOELLE ANDERSON

B.S., Georgia Southern University, 2019

A Thesis Submitted to the Graduate Faculty of Georgia Southern University in Partial

Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ARNOLD TRANSFORMATIONS AS APPLIED TO DATA ENCRYPTION

by

H. NOELLE ANDERSON

Major Professor:	Yan Wu
Committee:	Martha Abell
	Yongki Lee

Electronic Version Approved:
December 2021

ACKNOWLEDGMENTS

I wish to acknowledge my advisor, Dr. Wu for guiding me through this process. He has been a mentor to me in both my undergraduate and my graduate careers at Georgia Southern, and I would not have been able to do this without him. Thank you so much for helping me on this journey.

I would also like to thank my committee members, Dr. Abell and Dr. Lee, both of whom I've had as professors and who are extraordinary teachers as well as the all of the other professors I've had at Georgia Southern.

Lastly, I'd like to thank my parents. From a young age, they instilled a love of math in me that has allowed me to come this far and have always been supportive of my goals.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

$\mathbb{R}$	Real Numbers
$\mathbb{Z}$	Integers
$detA$	Determinant of a matrix A
$M_{m\times n}(\mathbb{Z})$	m×n matrices with integer entries
T	Periodicity
i	Iteration Number
D_n	Random Matrix generated from the Diophantine Method
N	Circumference Number

CHAPTER 1

INTRODUCTION

In this thesis we discuss the use of Arnold Transformations in data encryption. In a society that is relying more and more on digital message transmission, data security is becoming an increasingly important issue. The problem then arises in how to encrypt data in a way that is both secure and efficient. If the method used is too simple, we risk our data being intercepted and decoded, but if the method is too complex, it may be too impractical to use in application. Previous results show that Arnold Transformations can be effectively used in data encryption [1−8]. Other work shows how to calculate the period of such a transformation [9−10]. Methods to get the anti-Arnold transformation have also been discussed [3−4]. Our goal is to explore the periodicity of Arnold transformations through theoretical analysis and numerical experiments so that a preferred encryption cycle is attained through a systematic construction of the key matrix.

We first identify the constraint for an Arnold Transformation that can guarantee periodicity. It is worth noting that our proof follows the results of the major proof in [1], but that we make several adjustments. We introduce a different notation system to avoid confusion between different variables. Also, at the point where [1] divides the necessary condition into two sections, we found that a supposition in the second section did not hold. Upon further investigation, this supposition was not necessary to complete the proof and, in fact, the necessary condition did not need to be split into two pieces at all. Thus in our proof, we keep our necessary condition as only one part. We then run several simulations to explore when periodicity is in an acceptable range and detail how to choose the transformation such that periodicity is ideal. In [1,3], several special Key Matrices are defined that we use in our simulations in additions to those we generate. Finally, we apply our findings to several examples including including both images and data vectors.

1.1 Introduction to Arnold Transformations

Definition 1.1.1 An Arnold Transformation is a reiterative matrix function of the following form:

$$\underset{\cdot}{x}^{i+1} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ & \dots & \dots & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \underset{\cdot}{x}^{i} (\text{modN}), \quad a_{ji}, N \in \mathbb{Z}, \underset{\cdot}{x}^{i} \in M_{m\times n}(\mathbb{Z}) \text{ for } m \in \mathbb{Z} \tag{1.1}$$

Since this is a modulo function, it is possible in some scenario that after a certain number of iterations, the output matrix is the same as the starting one. The number of iterations taken to achieve this is called the periodicity. The higher the periodicity, the harder it is to get the original matrix back and the better the encryption is. There are some transformations, however, where it is possible that the original matrix is never returned. This renders the transformation unusable for encryption. We must find, then, the conditions that a given Arnold Transformation will have periodicity for any starting matrix $\underset{\cdot}{x}$.

CHAPTER 2
PERIODICITY OF ARNOLD TRANSFORMATIONS

It turns out that the necessary and sufficient condition for a periodic Arnold Transformation is that the determinant of the key matrix and the circumference number are co-prime, which was first reported in [1]. We found ways to improve the proving argument in contrast to the proof in [1]. We expound the theoretical result in 2.1.

2.1 Theoretical Work

Consider the following transformation

$$\begin{pmatrix} x'_1 \\ x'_2 \\ \cdots \\ x'_n \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ & \cdots & \cdots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix} (\text{mod} N),\ a_{ji} \in \mathbb{Z},\ x_1 ... x_n \in \{0, 1, ..., N-1\} \quad (2.1)$$

This transformation has periodicity if $A\alpha$(mod N)$\neq A\beta$(mod N) for any two different n-dimensional vectors α and β.

Theorem 2.1. *For a given positive integer N, transformation (2.1) has periodicity if and only if det(A) and N are co-prime.*

Proof. Sufficiency. We will prove that 2.1 has periodicity when $|A|$ and N are coprime; that is, $A(\alpha$-$\beta)$(modN)$\neq$0 for any two different n-dimensional vectors α and β. For any n-dimensional vector $\underset{\cdot}{x}$=(x_1, x_2,...,x_n)^T, if $A\underset{\cdot}{x}$(modN)=0, we will prove that $\underset{\cdot}{x}$(modN)=0. Set $A\underset{\cdot}{x}$(modN)=0.

$$\begin{cases} a_{11}x_1 + ... + a_{1n}x_n = k_1N, \\ .... \\ a_{n1}x_1 + ... + a_{nn}x_n = k_nN \end{cases} \qquad \text{k}_i \in \mathbb{Z}$$

Then since A^{-1}=$\frac{adj(A)^T}{|A|}$ we have

$$\begin{cases} |A|x_1 = (A_{11}k1 + ... + A_{n1}k_n)N, \\ .... \\ |A|x_n = (A_{1n}k1 + ... + A_{nn}k_n)N \end{cases} \qquad \text{k}_i \in \mathbb{Z} \tag{2.2}$$

where A_{ij} is the algebraic complement of the element a$_{ij}$ of the matrix A.

Because $|A|$ and N are co-prime, N does not divide $|A|$, thus N must divide each x_i. Thus $\underset{\cdot}{x}(modN)$=0.

Necessity. For a given positive integer N, if transformation (2.1) has periodicity; then we show that $|A|$ and N must be co-prime. Assume that $|A|$ and N are *not* co-prime, then we show that there is an $\underset{\cdot}{x}$ $\neq$0 such that $A\underset{\cdot}{x}(modN)$=0, which implies that (2.1) does not have periodicity, giving us a contradiction. Set $A\underset{\cdot}{x}(modN)$=0. Then by (2.2) we have

$$\begin{cases} x_1 = & \frac{(A_{11}k1+...+A_{n1}k_n)N}{|A|} \\ .... & \\ x_l = & \\ x_n = & \frac{(A_{1n}k1+...+A_{nn}k_n)N}{|A|} \end{cases} \tag{2.3}$$

Let $|A|$=st and N=N_1t where t=gcd($|A|$,N)≠1 is the greatest common divisor of $|A|$ and N. Then (2.3) becomes

$$\begin{cases} x_1 = & \frac{(A_{11}k1+...+A_{n1}k_n)N_1}{s} \\ \ldots\ldots\ldots\ldots & \\ x_n = & \frac{(A_{1n}k1+...+A_{nn}k_n)N_1}{s} \end{cases}$$

Then our goal is to find k_1,...,k_n such that x_1, x_2,...,x_n are all integers, but at least one of them cannot be divided by t and thus N. Then for that x_j, $x_j(modN)$ cannot be equal to zero, and thus $x(modN)$ ≠0 and (2.1) does not have periodicity.

By number theory, we suppose the following

$|A|$=$\pm p_1^{a_1}p_2^{a_2}...p_k^{a_k}$

s=$\pm p_1^{b_1}p_2^{b_2}...p_k^{b_k}$

t=$p_1^{c_1}p_2^{c_2}...p_k^{c_k}$

(A_{11},...,A_{n1})=$\pm p_1^{d_{11}}p_2^{d_{21}}...p_k^{d_{k1}} = k_{11}A_{11} + ... + k_{n1}A_{n1}$

....

$(A_{1n}, ...,$A$_{nn})$ =$\pm p_1^{d_{1n}}p_2^{d_{2n}}...p_k^{d_{kn}} = k_{1n}A_{1n} + ... + k_{nn}A_{nn}$ (2.4)

where p_1,...,p_k are primes $a_i \in \mathbb{Z}^+$, $b_i, c_i, d_{ij} \in \mathbb{Z} \setminus \mathbb{Z}^-$, $k_{ij} \in \mathbb{Z}$ are integers, and (k$_{1j}$,...,k$_{nj}$)=1, i=1,...,n

$|A|$=st, so we can clearly see that b_i+c_i=a_i for all i=1,...,n

Since $t>$ 1, we know there exists some i such that c_i>0. We define the following

$d_{il} = min\{d_{i1}, ..., d_{in}\}$ for some 1≤ $l \leq n$

$d_j^* = min\{d_{j1}, ..., d_{jn}\}$ for all 1≤ $j \leq k, i \neq j$

Then let $q = p_1^{r_1} p_2^{r_2} ... p_k^{r_k}$, where

$$r_i = b_i - d_{il}$$

$$r_j = \begin{cases} 0 & \text{if } b_j \leq d_j^* \\ b_j - d_j^* & \text{otherwise} \end{cases} \quad j \neq i$$

Recall from (2.4) that $(A_{1l}$,...,$A_{nl})$=$k_{1l}A_{1l}$+...+$k_{nl}A_{nl}$. Set k_1=qk_{il},...,k_n=qk_{nl}. Then you can verify that x_1, x_2,...,x_n are all integers, but that x_l is not divisible by t (See the lemmas in the appendix for details) □

Thus for our purposes, we need $|A|$ and N to be co-prime. The easiest way to guarantee this is to construct A such that $|A|$ is either 1 or -1, so we demonstrate how to construct such an A.

2.1.1 An Example

We use a simple example to demonstrate the factoring scenarios in the proof. Let $A = \begin{pmatrix} 1 & 3 \\ 2 & 15 \end{pmatrix}$ and N=6. Then we have $|A|$=$3^2$$\Rightarrow$ s=3 and t=3. Thus a_1=2, b_1=1, c_1=1

Next we calculate the co-factor matrix and then determine the greatest common divisor of each column: A_{ij}=$\begin{pmatrix} 15 & 2 \\ 3 & 1 \end{pmatrix}$

Thus $(A_{11}$, $A_{21})$=(15, 3)=3 $\Rightarrow$ d_{11}=1 and $(A_{12}$, $A_{22})$=(2, 1)=1 $\Rightarrow$ d_{12}=0. Since c_1=1>0, i=1. Then since i=1, d_{il}=d_{1l}=min$\{d_{11}, d_{12}\}$=d_{12}=0. Thus l=2

Let q=$p_1^{r_1}$. Since r_1=r_i, r_1=b_1-d_{1l}=1 and q=3^1=3. Then $(A_{12}$,$A_{22})$=$k_{12}A_{12}$+$k_{22}A_{22}$ $\Rightarrow$ 1=2k_{12}+k_{22}. Let k_{12}=1, k_{22}=-1 $\Rightarrow$ k_1=qk_{12}=3, k_2=qk_{22}=-3

Then finally $x_1 = \frac{(A_{11}k_1+A_{21}k_2)N_1}{s}$=24, $x_2 = \frac{(A_{12}k_1+A_{22}k_2)N_1}{s}$=2 are both integers, but t=3$\nmid$2.

CHAPTER 3
CONSTRUCTING THE ARNOLD MATRIX

We outline two different ways to construct the matrix A. Both can be used to guarantee that $|A|$=± 1, but while the first may be simpler, the second method involves randomization and thus results in an A that is much more difficult to guess.

3.1 The Pre-Defined Matrices

Our first method is to simply use a predefined A that can be adjusted for any dimension. The following matrices all have determinant 1 for any size.

Lemma 3.1.1. *The following n x n matrix has determinant 1*

$$B_n = \begin{bmatrix} 1 & 1 & 1 & 1 & ... & 1 \\ 1 & 2 & 2 & 2 & ... & 2 \\ 1 & 2 & 3 & 3 & ... & 3 \\ 1 & 2 & 3 & 4 & ... & 4 \\ ... & ... & ... & ... & ... & \\ 1 & 2 & 3 & 4 & ... & n \end{bmatrix}$$

Proof. We do a proof by induction to show that the determinant of this $n \times n$ matrix is the same as the $(n-1) \times (n-1)$ matrix

i) Base Case, Let n=1

B$_1$=$\begin{bmatrix} 1 \end{bmatrix}$ and $|B|$ is clearly 1

ii) Induction

Let B$_n$ be the $n \times n$ matrix of the above form and and B$_{n-1}$ be the $(n-1) \times (n-1)$ matrix. From induction, $|B_{n-1}|$=1. Show that $|B_n|$=$|B_{n-1}|$

$$\mathrm{B}_n=\begin{bmatrix} 1 & 1 & 1 & 1 & ... & 1 \\ 1 & 2 & 2 & 2 & ... & 2 \\ 1 & 2 & 3 & 3 & ... & 3 \\ ... & ... & ... & ... & ... & ... \\ 1 & 2 & 3 & 4 & ... & n \end{bmatrix}$$

By the properties of determinant, subtracting row 1 from another row does not change the determinant, thus we can subtract row 1 from every row without changing the determinant

$$|B_n|=\begin{vmatrix} 1 & 1 & 1 & 1 & ... & 1 \\ 0 & 1 & 1 & 1 & ... & 1 \\ 0 & 1 & 2 & 2 & ... & 2 \\ ... & ... & ... & ... & ... & ... \\ 0 & 1 & 2 & 3 & ... & n-1 \end{vmatrix}$$

Then we expand along the first column to get

$$|B_n|=\begin{vmatrix} 1 & 1 & 1 & 1 & ... & 1 \\ 1 & 2 & 2 & 2 & ... & 2 \\ 1 & 2 & 3 & 3 & ... & 3 \\ ... & ... & ... & ... & ... & ... \\ 1 & 2 & 3 & 4 & ... & n-1 \end{vmatrix}=|B_{n-1}|$$

Thus for any n, the determinant of this matrix is 1

$\square$

The following two matrices as defined in [1] and [2], respectively, also have determinant $\pm$ 1.

i) The Fibonacci matrix which is defined as follows

$$Q_1=[1],\ Q_2=\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix},\ Q_3=\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix},Q_4=\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix},\ ...$$

ii) The Recursive matrix which is defined as follows

$R(1,j)=1, R(i,1)=1\ i,j=1,2,...,n$

$R(i,j)=R(i-1,j-1)+R(i-1,j), i,j=2,3,...,n$

$$R_1=[1], R_2=\begin{bmatrix}1 & 1\\ 1 & 2\end{bmatrix}, R_3=\begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 2\\ 1 & 3 & 4\end{bmatrix}, R_4=\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 2 & 2 & 2\\ 1 & 3 & 4 & 4\\ 1 & 4 & 7 & 8\end{bmatrix}, \ldots$$

3.2 Diophantine Equation and Euclidean Algorithm

While the above matrices are easy to use, they can be considered predictable. To fix this, we use what we are calling the Diophantine Equation Method. Let A be any random matrix and replace any two entries with x and y. Then we can set the determinant of A equal to either plus or minus one and solve the corresponding linear Diophantine equation to find the necessary integers x and y. The problem of determining when our resulting Diophantine equation is solvable then arises.

Theorem 3.1. *The linear Diophantine equation ax+by=c has a solution if d=gcd(a,b) divides c. Also, if (x_0,y_0) is a solution to ax+by=c, then x=x_0+k$\frac{b}{d}$, y=y_0-k$\frac{a}{d}$ is also a solution for any integer k.*

Proof. We break our proof into two parts.

i) Show that $ax + by = c$ has a solution if $d = gcd(a, b)$ divides c

First we prove that ax+by=d has a solution. Let S=$\{ax + by \mid ax + by > 0\}$. Then S is clearly not empty since either a or $-a$ is in S thus let $d = minS = as + bt$ for some integers s, t. We show that $d = gcd(a, b)$

a) Show that d divides both a and b

a=dq+r for some integers q >0, $d > r$ ≥0 which directly implies $r = a - dq = a - (as + bt)q = a - asq - btq = (1 - sq)a - tqb$. Thus r is of the form $ax + by$ and r is in $S \cup \{0\}$. Then since r <d and d is the minimum of S, r=0. Thus d divides a.

It can be similarly shown that d divides b.

b) Show that there does not exist a divisor of a and b greater than d, that is if w divides a

and b then $w \leq$d.

Let w divide a and b, then $a = wu$ and $b = wv$ for some integers u, v. Then $d = as + bt = (wu)s + (wv)t = w(as + bt)$. Thus $d \mid w$ and $w \leq$d.

Thus $d = minS = gcd(a, b)$, and $ax + by = d$ has a solution, namely $x = s$ and $y = t$.

Now we prove that $ax + by = c$ has a solution. Since $d \mid c$, let $c = pd$ for an integer p. Then if x, y is a solution to $ax + by = d$, it can be easily seen that px, py is a solution to $ax + by = c$:

$a(px) + b(py) = p(ax + by) = pd = c$

ii) if x_0,y_0 is a solution to $ax + by = c$, then $x = x_0$+$k\frac{b}{d}$, $y = y_0$-$k\frac{a}{d}$ is a solution for any integer k

Let x_0,y_0 be a solution to $ax + by = c$ and $x = x_0$+k$\frac{b}{d}$, $y = y_0$-$k\frac{a}{d}$. Then $a(x_0 + k\frac{b}{d})$+$b(y_0 - k\frac{a}{d})$=ax_0+$\frac{abk}{d}$+by_0-$\frac{abk}{d}$=ax_0+by_0=c

Thus $ax + by = c$ $\square$

We write a MATLAB program that runs through all possible placements of x and y in a given matrix and returns a placement that is solvable by the above theory. We then solve the equation by Euler's algorithm and find the "best" solution. For this paper, we have chosen the "best" solution to be the one that minimizes $|x|$+$|y|$. Another way to define the "best" solution is to have x and y be close to the mean value of the row or column that they share. This method is particularly useful because of its randomness. Since the initial matrix is MATLAB generated, not user generated, it is less susceptible to brute force attack.

3.2.1 An Example

Let A=$\begin{bmatrix} x & y & 3 \\ 3 & 2 & 1 \\ 1 & 1 & 4 \end{bmatrix}$. Setting $|A|$=1 gives us the Diophantine equation $7x - lly = -2$. Since 7 and 11 are co-prime, this is clearly solvable by the above theorem. We now apply the Euclidean Algorithm. First we solve $7x - lly = 1 \rightarrow 7x + ll(-y) = 1 \rightarrow 7x + ll\ \hat{y}$ =1. Then our final solution will be (-2x,2$\hat{y}$)

The Euclidean algorithm uses a backwards substitution recursive formula. Assume a<b and $gcd(a,b)$=1. Let $r_{-2} = b, r_{-1} = a$, $r_0 = q_2r_1 + r_2$, ..., $r_i = q_{i+2}r_{i+1} + r_{1+2}$,..., $r_{n-1} = q_{n-1}r_n + r_{n+1}$ and $r_{n+1} = d$. In our MATLAB, program, we run a loop to replace each $r_0, ..., r_{n+1}$. We do this by establishing a vector that maps our equation in terms of $b, a, r_0 ... r_{n+1}$ and their respective multiples. Then we run a search function and replace the first r_n, and we find and re-simplify our equation. Once the find function finds no more r_n, we are finished.

For our example, $r_{-2} = 11\ r_{-1} = 7\ r_0 = 4\ r_1 = 3\ r_2 = 1$ and $q_0 = q_1 = q_2 = 1$.Then

1=$r_{n+1} = r_2 = r_0 - q_2r_1 = (r_{-2} - q_0r_{-1}) - q_2(r_{-1} - q_1r_0)$

$= r_{-2} - (q_0 + q_2)r_{-1} + q_1q_2r_0 = r_{-2} - (q_0 + q_2)r_{-1} + q_1q_2(r_{-2} - q_0r_{-1})$

$= (1 + q_1q_2)r_2 - (q_0 + q_2 + q_0q_1q_2)r_{-1} = (1 + q_1q_2)b - (q_0 + q_2 + q_0q_1q_2)a$

$= (1 + 1)(11) - (1 + 1 + 1)7 = (2)(11) + (-3)(7)$

Thus our solution to $7x + 11y = 1$ is (-3,2) and our solution to $7x - 11y = -2$ is (6,4). Substituting this back into the original matrix gives $A = \begin{bmatrix} 6 & 4 & 3 \\ 3 & 2 & 1 \\ 1 & 1 & 4 \end{bmatrix}$ and $|A|$=1.

CHAPTER 4

NUMERICAL TESTING

In order to better understand how to optimize the periodicity, we test how it varies as the variables N, n and A change, respectively. We maintain that the determinant of A is one.

4.1 VARYING CIRCUMFERENCE N

Using several 3x3 and 4x4 matrices, we can see how increasing N changes the period. For this table, N is between 2 and 257.

Matrix	Highest Periodicity	Corresponding N	Lowest Periodicity	Corresponding N
B_3	123,721	255	6	13
Q_3	114,576	230	7	2
R_3	103,152	238	7	2
D_3	113,799	254	4	3
B_4	108,549	246	7	2
Q_4	22,279,040	223	15	2
R_4	14,452,480	254	7	2
D_4	6,475,140	218	5	2

Table 4.1: Numerical Testing on Period Over N

We see that the highest periodicity values typically come from high N values and the lowest values typically come from lower N values. We can conclude that N and the periodicity have a loosely positive correlation. Since we set A so that $|A|$=1, we can always adjust N to be sufficiently large.

4.2 Varying Key Matrix Size n

We use the testing matrices found in [1,3] and a randomly generated Diophantine matrix to test how increasing a matrix's size changes the periodicity. Here we let N range from 2 to 30 and note the maximum period found in that range.

B_n			Q_n		
n	Highest Periodicity	Corresponding N	n	Highest Periodicity	Corresponding N
3	2,821	30	3	1,736	30
4	1953	30	4	14,480	19
5	959,171	26	5	5,226	29
6	2,613,660	19	6	7,797,131	26

R_n			D_n		
n	Highest Periodicity	Corresponding N	n	Highest Periodicity	Corresponding N
3	1,281	26	3	280	29
4	25,260	29	4	12,166	23
5	488,255	22	5	959,171	26
6	472,626	30	6	21,243,690	29

Table 4.2: Numerical Testing on Varying Key Matrix Size

Table 4.2 shows evidence that increasing n possibly extends the period.

4.3 Varying Key Matrix

Now we look at several key matrices of the same size, with a fixed N=257 to see if there is a clear winner for encrypting the input vector. In the table below, the periodicity listed applies to any dense input vector.

3X3 Matrices		4x4 Matrices	
Matrix	Periodicity	Matrix	Periodicity
B_3	**66,307**	B_4	66,307
Q_3	**66,307**	Q_4	**16,974,592**
R_3	65,692	R_4	66,048
D_3	66,048	D_4	3,408,180

Table 4.3: Numerical Testing on Varying Key Matrix

The best performing matrix varies across different matrix sizes, thus we cannot conclude that any given matrix is always superior over the other. Thus, we need several valid encryption matrices so that we can test which is the most useful for any given encryption. This makes our Diophantine method particularly useful, since it can be used to generate numerous matrices quickly.

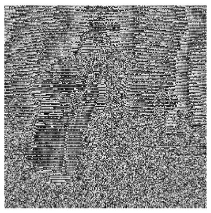
CHAPTER 5
APPLICATIONS

We now show how our encryption method performs in applications. We divide this method into two majors parts: image encryption and data string encryption.

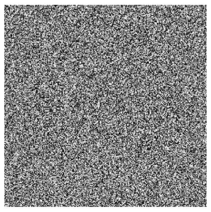
5.1 Image Encryption

We test the method's efficiency on several images including both those built into MATLAB and those imported from the web. We saw earlier that the larger A is, the higher the periodicity. In order to keep periodicity low enough that MATLAB can run quickly, we dissect our image into smaller parts and then apply the algorithm to each piece separately. For example, if our image is 50 x 50 pixels, we may instead divide into one hundred 5 x 5 images and encrypt those individually or in parallel. Then once each section is encrypted, we simply recombine the pieces. In applications, we found that when the image size exceeds 5x5, the process requires an impractically high computation time, so for our uses we limit our size there.

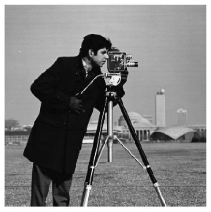
a)



b)



c)



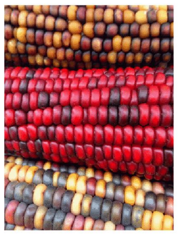
d)

Figure 5.1: Cameraman Image at Various Iterations of the Transformation

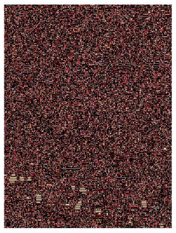
a) original image b) after one iteration

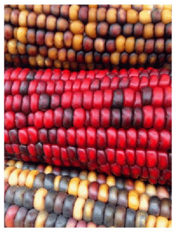
c) halfway through iterations d) final returned image

In the above, we see the MATLAB "Cameraman" image at several stages of the process. After only one iteration, we can still pick out a vague outline of the cameraman, however after several hundred iterations, the image is completely unrecognizable until the final iteration when it is returned. We can also apply this process to a color image. In this case, it suffices to only scramble the indexed value and leave the color map as is.



a)





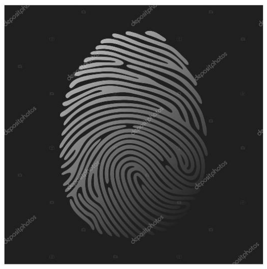
b) c))

Figure 5.2: Corn Image at Various Iterations of the Transformation

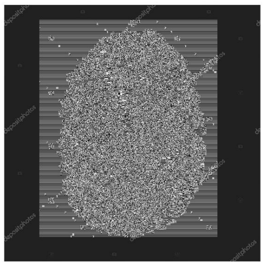
a) original image

b) halfway through iterations c) final returned image

It is also possible to only scramble part of an image. In this case, since we have already sectioned out the image, we simply only apply the algorithm to those pieces we want to scramble. This method is particularly useful for very large images, where scrambling the entire image can be very tedious. In this following fingerprint image, for example, even removing a small border section from the process can reduce computation time significantly.



a)



b)

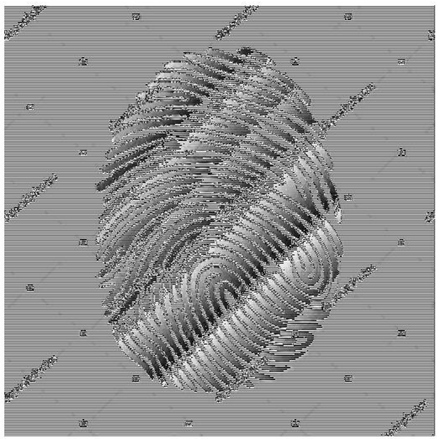
Figure 5.3: Partial Fingerprint Scramble at Iteration 500

a) original image b)after 500 iterations

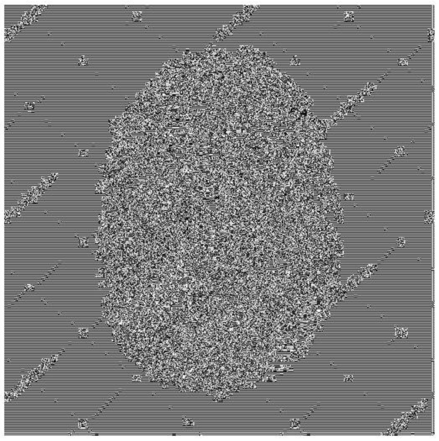
There is still one more scenario we would like to test. In the case of image encryption, our N is fixed at 257, a prime. According to our earlier proof then, it is no longer strictly necessary that $|A|$=1. Let A$_n$= $\begin{bmatrix} 2 & 2 & 2 & ... & 2 \\ 2 & 4 & 4 & ... & 4 \\ 2 & 4 & 6 & ... & 6 \\ ... & ... & ... & ... & ... \\ 2 & 4 & 6 & ... & 2n \end{bmatrix}$.

Then it can be easily shown that $|A|_n$=2^n and for any A_n, A_n and 257 are co-prime. Below we see A_n successfully applied to the same fingerprint image from before.

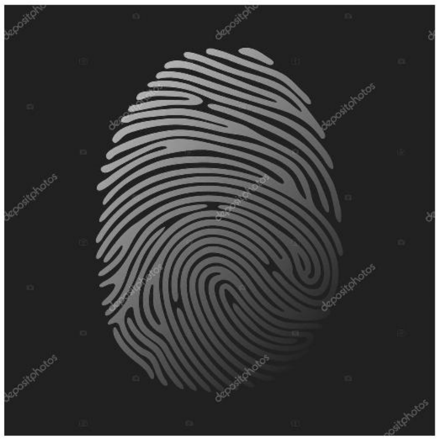
a)



b)



c)



d)

Figure 5.4: Fingerprint Image Scrambling when $|A|\neq 1$

a) original image b) after one iteration

c) halfway through iterations d) final returned image

5.2 Data String Encryption

It is often necessary in areas such as online banking to encrypt someone's personal information when they are stored on the server. In this case, we only need to convert that information into a neat matrix and then apply the encryption. We do this by making each piece of information a column in the new matrix. We use the MATLAB function str2double to map our string inputs to numeric values. We can guarantee that each column is the same length by simply filling in any empty space with zeros. We test the algorithm as applied to the following data using key matrix B_3.

Name: 'Name'

SSN: '123456789'

Bank Number: '12341234'

$$\begin{bmatrix} N & 1 & 1 \\ A & 2 & 2 \\ M & 3 & 3 \\ E & 4 & 4 \\ 0 & 5 & 1 \\ 0 & 6 & 2 \\ 0 & 7 & 3 \\ 0 & 8 & 4 \\ 0 & 9 & 0 \end{bmatrix} = \begin{bmatrix} 78 & 49 & 49 \\ 97 & 50 & 50 \\ 109 & 51 & 51 \\ 101 & 52 & 52 \\ 0 & 53 & 49 \\ 0 & 54 & 50 \\ 0 & 55 & 51 \\ 0 & 56 & 52 \\ 0 & 57 & 0 \end{bmatrix}$$

a)

$$\begin{bmatrix} 142 & 106 & 106 \\ 76 & 241 & 241 \\ 89 & 138 & 138 \\ 128 & 215 & 185 \\ 226 & 199 & 124 \\ 146 & 62 & 153 \\ 0 & 67 & 94 \\ 0 & 157 & 199 \\ 0 & 243 & 157 \end{bmatrix} \qquad \begin{bmatrix} 78 & 49 & 49 \\ 97 & 50 & 50 \\ 109 & 51 & 51 \\ 101 & 52 & 52 \\ 0 & 53 & 49 \\ 0 & 54 & 50 \\ 0 & 55 & 51 \\ 0 & 56 & 52 \\ 0 & 57 & 0 \end{bmatrix}$$

b) c)

Figure 5.5: Encryption of a Data String

a)original data b) halfway through iterations c) final returned data

CHAPTER 6

CONCLUSIONS

The objective of this paper is to study the principle of the Arnold transformations and to develop a method of constructing alternative Arnold key matrices to encrypt data. We established the conditions under which the periodicity of such a transformation exists and detailed how to construct a transformation such that those conditions were met. We then explored the relationship between the transformation's periodicity and the key matrix, matrix size, and the circumference number, which is then used to control the periodicity of a key matrix.

We apply Arnold transformations to both images and data string encryption. In cases where the periodicity of our initial transformation was too high or too low, the adaptability of our method allowed for easy adjustment. We thus consider our method to be successful in the practical situation of data encryption and maintain that our results contribute to the enrichment of the Arnold transformation in data encryption.

REFERENCES

[1] Qi, Dongxu Zou, Jiancheng Han, Xiaoyou. (2012). A new class of scrambling transformation and its application in the image information covering. Science in China, Series E: Technological Sciences. 43. 304-312. 10.1007/BF02916835.

[2] Zhang, Min-Rui Shao, G.-C Yi, Ke. (2005). T-matrix and its applications in image processing. Electronics Letters. 40. 1583 - 1584. 10.1049/el:20046517.

[3] Wu, Lingling Zhang, Jianwei Weitao, Deng He, Dongyan. (2009). Arnold Transformation Algorithm and Anti-Arnold Transformation Algorithm. 10.1109/ICISE.2009.347.

[4] Tao, KONG Dan, ZHANG. (2004). A New Anti Arnold Transformation Algorithm. Journal of Software. 15.

[5] Ding, W. Yan, W.-Q Qi, D.-X. (2001). Digital image scrambling technology based on Arnold transformation. 13. 338-341.

[6] Min, Li Ting, Liang Yu-jie, He. (2013). Arnold Transform Based Image Scrambling Method. 10.2991/icmt-13.2013.160.

[7] Zou, Weigang Li, Wei Cai, Zhaoquan. (2019). High-dimensional Arnold inverse transformation for multiple images scrambling. International Journal of Computational Science and Engineering. 20. 362. 10.1504/IJCSE.2019.103941.

[8] Zou, Weigang Huang, Jiangyan Zhou, Caiying. (2010). Digital Image Scrambling Technology Based on Two Dimension Fibonacci Transformation and Its Periodicity. 415-418. 10.1109/ISISE.2010.98.

[9] Li, Bing Xu, Jia-wei. (2005). Period of Arnold transformation and its application in image scrambling. Journal of Central South University of Technology. 12. 278-282. 10.1007/s11771-005-0414-1.

[10] W. Ji-zhi, W. Ying-long and W. Mei-qin, “Periodicity and Application for a kind of n-dimensional Arnold-type Transformation,” 2007 IEEE Intelligence and Security Informatics, 2007, pp. 375-375, doi: 10.1109/ISI.2007.379511.

APPENDIX

LEMMAS

Lemma .0.1. $\frac{(A_{11}k1+...+A_{n1}k_n)}{s}$,..., $\frac{(A_{1n}k1+...+A_{nn}k_n)}{s}$ *as defined in the Theorem 2.1 are all integers*

Proof. Let E_γ= $\frac{(A_{1\gamma}k1+...+A_{n\gamma}k_n)}{s}$. Show that E_γ is an integer.
E_γ= $\frac{(A_{1\gamma}k1+...+A_{n\gamma}k_n)}{s}$=$\frac{q(A_{1\gamma}k1+...+A_{n\gamma}k_n)(a_{1\gamma}k_{1\gamma}+...+a_{n\gamma}k_{n\gamma})}{s}$ for some a$_{1\gamma}$, ..., $a_{n\gamma}\in\mathbb{Z}$
=$\frac{q(A_{1\gamma}k1+...+A_{n\gamma}k_n)}{s}Z_\gamma$ where Z_γ=$a_{1\gamma}k_{1\gamma}$+...+$a_{n\gamma}k_{n\gamma}\in\mathbb{Z}$
We examine $\frac{q(A_{1\gamma}k1+...+A_{n\gamma}k_n)}{s}$ =$\frac{(p_1^{r_1}...p_k^{r_k})(p_1^{d_1\gamma}...p_k^{d_k\gamma})}{p_1^{b_1}...p_k^{b_k}}$ =$\frac{(p_1^{r_1+d_1\gamma}...p_k^{r_k+d_k\gamma})}{p_1^{b_1}...p_k^{b_k}}$
For any p_m we have $\frac{p_m^{r_m+d_m\gamma}}{p_m^{b_m}}$=$p_m^{r_m+d_m\gamma-b_m}$
Case 1:r_m=0
Then $d_{m\gamma}\geq b_m$ by our definition of r_j. Then r_m+$d_m\gamma$-b_m $\geq$0 and $p_m^{r_m+d_m\gamma-b_m}\in\mathbb{Z}$
Case 2: r_m=b_m-d_m^*
Then $p_m^{r_m+d_m\gamma-b_m}$=$p_m^{d_m\gamma-d_m^*}\in\mathbb{Z}$

Thus in either case, $p_m^{r_m+d_m\gamma-b_m}\in\mathbb{Z}$ which implies that $\frac{(p_1^{r_1+d_1\gamma}...p_k^{r_k+d_k\gamma})}{p_1^{b_1}...p_k^{b_k}}$=
$\frac{q(A_{1\gamma}k1+...+A_{n\gamma}k_n)}{s}\in\mathbb{Z}$ and finally, E_γ =$\frac{q(A_{1\gamma}k1+...+A_{n\gamma}k_n)}{s}Z_\gamma\in\mathbb{Z}$ □

Lemma .0.2. E_l=$\frac{(A_{1l}k1+...+A_{nl}k_n)}{s}$ *as defined in the Theorem 2.1 is not divisible by t*

Proof. Recall that $t = p_1^{c_1}...p_i^{c_i}...p_k^{c_k}$. Since c_i >0, if t *does* divide E_l, then E_l must have at least one power of p_i.
E_l=$\frac{(A_{1l}k1+...+A_{nl}k_n)}{s}$=$\frac{q(A_{1l}k_{1l}+...+A_{nl}k_{nl})}{s}$=$\frac{(p_1^{r_1}...p_k^{r_k})(p_1^{d_1l}...p_k^{d_kl})}{p_1^{b_1}...p_k^{b_k}}$=$p_1^{r_1+d_{1l}-b_1}$...$p_k^{r_k+d_{kl}-b_k}$
At p_i we have $p_i^{r_i+d_{il}-b_i}$=$p_i^{b_i-d_{il}+d_{il}-b_i}$=$p_i^0$
Thus there are no powers of p_i in E_l, thus t cannot divide E_l □