Multi-Dimensional Baker Maps for Chaos Based Image Encryption

Project report submitted in partial fulfillment of the requirements for the award of the degree of

Master of Technology in Computer Science

by

Sai Charan K.

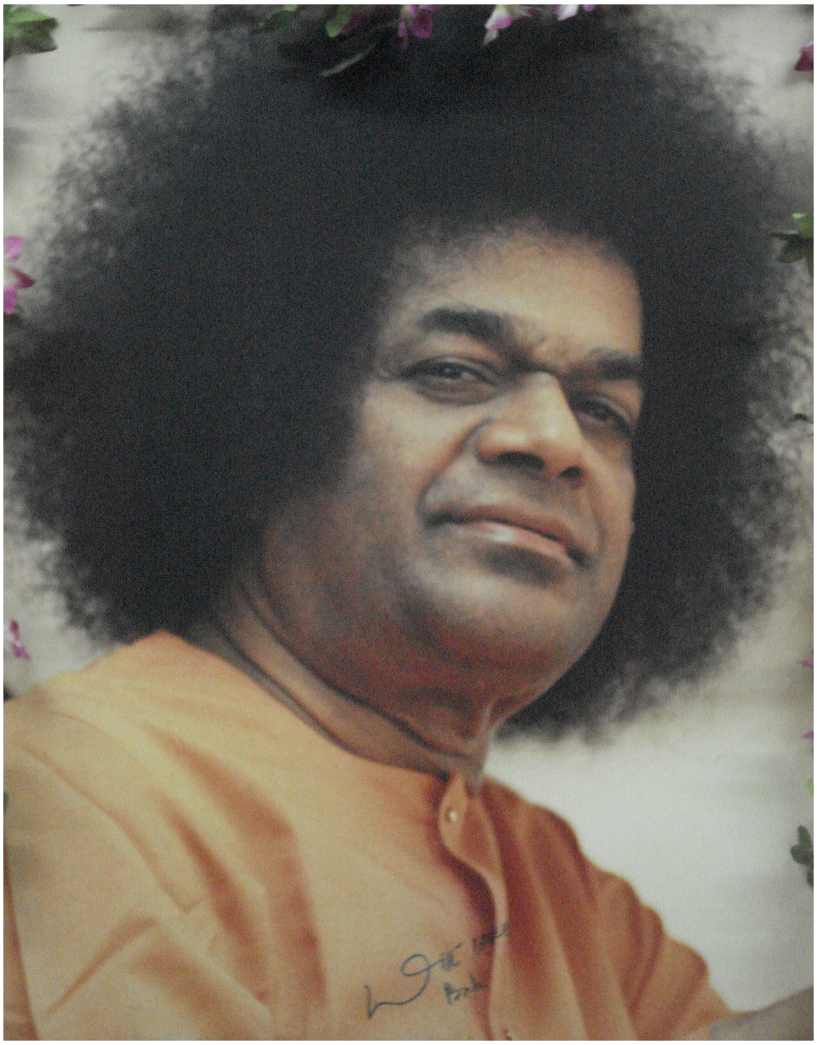
(Regd.No.: 06554)

DEPARTMENT OF MATHEMATICS & COMPUTER SCIENCE

Sri Sathya Sai University, Prashanthi Nilayam

February 2008

DEDICATION



To the most cryptic of them all...to Enigma Himself

To him, whose cryptic smile is decrypted
In ways unique to each,
Every decryption correct
Unlike in the ordinary,
Making this phenomenon
A unique cipher.

SRI SATHYA SAI UNIVERSITY

(Established under Section 3 of the UGC Act, 1956)

DEPARTMENT OF MATHEMATICS & COMPUTER SCIENCE

CERTIFICATE

This is to certify that this project report entitled **"Multi-Dimensional Baker Maps for Chaos Based Image Encryption"** being submitted by Sri. **Sai Charan K.** in partial fulfillment of the requirements for the award of the degree **Master of Technology in Computer Science** is a record of bonafide research work carried out by him under my supervision and guidance during the academic year 2007-08 in the Department of Mathematics and Computer Science, Sri Sathya Sai University, Prashanthi Nilayam campus. To the best of my knowledge, the results embodied in this project have not formed the basis of any work submitted to any other University or Institute for the award of any Diploma or Degree.

Prof. V. Chandrasekaran,
Department of Mathematics and Computer Science,
Sri Sathya Sai University, Prashanthi Nilayam.

Place: Prashanthi Nilayam
Date: 15^{th} February 2008

ACKNOWLEDGEMENTS

I would like to express my gratitude to the Vice-Chancellor Sri A.V. Gokak, the Registrar Prof. A.V. Lakshminarasimham, the Controller of examinations Prof. M. Nanjundiah and the Principal of the campus Prof. U.S. Rao for the administrative support provided with regard to the project formalities.

I express my deep sense of gratitude to my supervisor Prof. V. Chandrasekaran for his infectious enthusiasm and for being such a wonderful spring of ideas. Most of the work herein would not have been possible if not for his out-of-the box ideas and lateral, non-linear, 'chaotic' thinking. I would also like to express my thankfulness to Sri Uday Kiran, lecturer at DMACS for the wonderful course work on Algorithms and Complexity and also for timely advice regarding data-structures and implementation issues. Sri S. Balasubramanian, research scholar at DMACS was very kind in helping me out with some numerical algorithms. The critical review and invaluable comments of Sri Srikanth Khanna, research scholar at DMACS have helped shaped this thesis and the publications that were communicated to various conferences. Mr. Krishnamoorthy was absolutely forthcoming in his support with software, hardware and invaluable computing advice. Mr. Raghunath Sarma's support with lab and inspirational resources have been indispensible.

I would like to express my gratitude to the staff of DMACS for shaping me through my tenure here at the department. In particular, I would like to thank my HoD, Prof. K.S. Sridharan for his wonderful and efficient administrative and infrastructure support. I would like to thank Prof. G.V. Prabhakar Rao, Prof. C.J.M. Rao, Prof. Mrs. Tiwari, Prof. Jayaprakash, Sri. Hanumanth R. Naidu, Sri. A.S.K Prabhakar and Sri. P.C. Rao.

Special thanks to Mr. Ravi Iyer for his discussions, insights and *gyaan* on various issues and for showing interest in my work. Learning was never more interesting and deep! I cannot forget the wonderful tidbits provided by Sri Lakshmi Narayan. Thanks are due for our network administrator Sri. Renju Reghuveeran for his supportive internet facility without which

progress would have stalled. Thanks to the library and the librarian for extending their services.

My parents and brother were very supportive and understanding each time I told them that I needed to rush as I had my project to do! I have a special word of gratitude and love for Saketh, Aditya, Swagat and Amartya. Thanks to my cousin Kesh for prodding me into thinking through his deep questioning.

I must confess that the world is still a very humane place. Myriad bloggers, freelance writers, professors, companies and professionals have invested time and resources in making the internet a wonderfully rich and highly searchable database. Google has really made research more palatable and collaborative. The on-line community and public fora have been very supportive and responsive to questions. The professors of the field were very kind in mailing me their work and suggestions whenever needed. Thanks to the internet and all its open users for their kind support.

Of course there are my classmates and juniors who provided me with wonderful moral support through out the project by just enquiring about my progress. Special mention must be made of Shiva Kumar K. for lending his ear to the rant on my work. It helped me think aloud. Thanks to Sriman, KG, TV, Praveen, Rampy, Sandy, Nagi, Vishnu, Kalyan and Arun for being such a nice bunch.

Thanks to Sunil and Shyam for educating me on other areas of security like secure hashing and digital watermarking. Thanks to Andy, Sunny and Sashi for being great system admins. Thanks to Nivas, Rahul, Padhu and the rest for their spirit of enquiry and for pelting me with question and thereby adding to my KB.

PREFACE

The following is the list of papers that have been communicated to conferences.

1. "N-Dimesional Baker Maps for Chaos Based Image Encryption", submitted to the International Conference on Image Processing 2008 (ICIP '08), San Diego, USA. This paper consists of the work detailed in chapter 4.

2. "Integrated Confusion-Diffusion Mechanisms for Chaos Based Image Encryption", submitted to the IEEE Conference on Computer and Information Technology, 2008 (IEEE CIT '08), Sydney, Australia. This paper consists of the work detailed in chapter 6.

ABSTRACT

Security is an integral part of every technology and implementation today. In this IT driven society, cryptography is perhaps the most widespread form of secure communication.

Chaotic functions have certain properties which lend them directly to encryption schemes. Of special interest in this regard are sensitive dependence on initial conditions, topological transitivity and dense periodic orbits. These translate to input avalanche effect, mixing and ergodicity in cryptography.

We first implement an existing algorithm and verify the claims of the authors. We then investigate higher dimensional Baker maps for image encryption. For this, we first propose a new interpretation for the Baker map in terms of a path function S. We then apply the higher dimensional maps for image encryption and experimentally conclude that 3D Baker map suffices for encryption. That is, there is no perceptible performance gained when using higher dimensional Baker maps.

Next, in an attempt to use chaotic maps for the diffusion mechanism in the encryption scheme, we embed the diffusion process into the confusion process. For this, we first propose an alternative view of a 2D image as a 3D structure using the binary representation of the image intensity values. We extend this scheme from grayscale images to color images and show its immense value in color image encryption.

Lastly, we propose a Baker map based on random walk of the image. Here, we employ sparse decomposition of images as a method of generating the random paths. Random walk based Baker maps would be more difficult to break than traditional Baker maps because of the chaotic behavior in the walk itself.

Since statistical attacks on encryption schemes use the histograms of the encrypted image, a key requirement of the encryption process is to flatten the histogram. We introduce the use of Mean Squared Error of the normalized histogram as a measure of the flatness of the histogram. We also employ other standard performance metrics to evaluate the proposed encryption schemes.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Problem Definition 4

1.2.1 Contributions of the Thesis 4

1.3 Outline of the Thesis 5

2 A Brief Survey of Literature **6**

2.1 The Genesis 6

2.2 A Gentle Introduction to Cryptography 7

2.3 An Orderly Introduction to Chaos 9

2.4 The Marriage: Chaos Based Cryptography 10

2.4.1 Migration to discrete chaos 11

2.5 Kolmogrov Flow, Baker Map and Cat Map 12

2.5.1 The Kolmogrov Flow 12

2.5.2 The Baker Map 13

2.5.3 The Cat Map 16

3 Baker Map Based Image Encryption **18**

3.1 Baker Map Based Image Encryption 18

3.2 The Encryption Algorithm 18

3.3 Performance Metrics 21

3.4 Results and Conclusions 22

4 Generalized n-Dimensional Baker Maps **24**
4.1 Introduction 24
4.2 A Novel Interpretation 25
4.2.1 Efficient Implementation 28
4.3 Extension to Higher Dimensions 28
4.4 Paths Are Many, Goal Is One 30
4.5 A New Performance Metric 31
4.6 Results 33
4.7 Conclusions and Discussion 34

5 Embedding Diffusion in Confusion **37**
5.1 Introduction 37
5.1.1 Brief Recap 38
5.2 Motivation 38
5.3 3D Baker Map Based Image Encryption 39
5.4 Embedding Diffusion in Confusion 39
5.5 Novel Extension to Three Dimensions 41
5.6 Color Image Encryption 41
5.7 Conclusions and Discussion 44

6 Random Walk Formulation of Baker Map **45**
6.1 Introduction 45
6.2 Sparse Decomposition of Images 46
6.3 Input, Control and Output Functions 47
6.4 Results and Conclusions 51

7 Discussion and Future Work **53**
7.1 Conclusion 53
7.2 Future Work 54

List of Figures

2.1 Action of Baker map on the unit square. 15

3.1 Encryption using 3D Baker maps (key used: 1234567890123456) 22

4.1 Traversal of the 2D image. 26
4.2 Traversal of the 3D image. 27
4.3 Vertical and Spiral Walks 30
4.4 Alternate way to partition the image - partition horizontally and pile vertically. 31
4.5 Correlation between adjacent pixels (4D). The plot on the right side is of the encrypted image while that on the left is of the original image. 34
4.6 Encrypted Images and their histograms. 35
4.7 Plot of upper and lower bounds for MSE of normalized histograms. 36

5.1 Encryption using Baker maps with traditional piling algorithms (key used is 1234567890123456) 40
5.2 The cube formed by the binary representation of the pixel intensities provides an alternative view of the image in 3 dimensions. 41
5.3 Encryption using Baker maps with binary-string piling algorithm 42
5.4 Partial decryption using Baker maps with traditional piling algorithm (key used is 1234567890123456) 42

5.5 Partial decryption using binary-string piling algorithm, where each channel has been expanded to its binary representation and each channel is independently encrypted 43
5.6 Encryption using Baker maps with binary-string piling algorithm with the binary representations of each channels concatenated. 43

6.1 The unit interval is split into sub-intervals whose union is the unit interval itself. 50
6.2 Random walk assembled by the Baker Map. 50
6.3 Sparse Decomposition of Image (3 buckets are used in this example). 50
6.4 Encryption using random-walk based Baker maps with 10 intervals/buckets in the output function (key used: 1234567890123456) 51
6.5 Correlation between adjacent pixels. The plot on the right side is of the encrypted image while that on the left is of the original image. 52

List of Tables

3.1 Correlation between adjacent pixels for 3D Baker map based encryption. 23

4.1 Correlation among adjacent pixels for the images ciphered with various Baker maps. 34

6.1 Correlation between adjacent pixels using random walk based Baker map. 51

Chapter 1

Introduction

What affected me most profoundly was the realization that the sciences of cryptography and mathematics are very elegant, pure sciences.
-James Sanborn

1.1 Motivation

Your lawyer has e-mailed asking for your company's tax returns. You consider sending it by e-mail. Wait! Is it a good idea? Will it reach safely? Will it be tampered with along the way? Will it fall in the hands of your competitors? How do you know if it was your lawyer's email in the first place(!)? These are not easy questions to answer.

You then start thinking: Can I *securely* send the email to my lawyer at all? The e-mail is transmitted over the internet which is essentially an insecure network. Can I still prevent unauthorized people from getting it. So there are chances that the e-mail could be intercepted. Is there anything I can do to keep my message/information confidential inspite of the e-mail being intercepted? What if the e-mail servers are compromised? After all, they are directly on the internet! What if the sys-admin is a bad guy? He has access to the entire system!!! Assume that the e-mail was *somehow* sent without being compromised. How will my lawyer know that the e-mail was

from me and not someone else masquerading to be me? Can he verify that the information has not been modified along the way?

Consider another example: every commercial medical information system needs to store large amounts of medical images of patients scans, x-rays etc. in their databases. There are very stringent legal requirements leading to strict security measures for such systems in order to protect the privacy of the patients. Also, it is not enough to have encryption schemes for just gray-level images, since large number of medical applications require full color processing. Tele-medicine requires experts across the globe to collaboratively diagnose ailments. For this, we need secure mechanisms for transmitting patient images over the insecure internet. Image encryption could be used for the secure transmission of images over the internet.

However, we wish to point out that security is not a modern requirement. Consider the epic Ramayana. Why did Lord Sri Rama give his ring to Hanuman before he left for Lanka? It was a security mechanism that Mother Sita could use to *authenticate* that Hanuman, who was a stranger to Mother Sita, was actually a messenger of her Lord[1]. Further, Mother Sita gave her *choodamani* to Hanuman as a proof of their meeting.

Encryption and cryptography thus play a vital role in online transmission, off-line image-archival and retrieval systems. It is therefore essential that in this information technology driven society, information security is an integral part of every technology. We endeavor to make digital transmission/archival as secure as if the message were personally delivered or stored in a physical safe storage vault. The above examples elaborately describe the typical problems and requirements that are addressed by *Security*. The buzzwords of security are [1, 2]:

- *Confidentiality*: Keeping a message secret.
- *Authentication*: Ability to verify the claimer's identity.
- *Integrity*: Ability to verify that the message has not been tampered

[1]Thanks to Sunil Kumar GMBS for this insight.

with.

- *Non-repudiation*: Ability to ensure that a party in a dispute cannot repudiate, or refute the validity of a statement or contract[3].

There are many security mechanisms available to realize these requirements including encryption, digital signatures, message digests or message integrity code (MICs), digital watermarking, steganography etc. Note that more than one of these technologies need to work synergistically in order to be effective.

Cryptography is the art and science of securing messages. It "is the practice and study of hiding information"[4]. It attempts to make sure that the message is comprehensible only to the 'legal' recipient of the message - the person for whom it is intended. The aim here is that even if an unintended person gets hold of the message, he will not be able to make sense of it with out the *key*. Much as with a *good* lock. Anyone can find their way to your home, but cannot get in without the key. In cryptography the key is some information that the sending and receiving parties agree upon. If the agreed-upon information is a secret message, we are in the domain of symmetric-key cryptography. Its counter part is public-key or asymmetric-key cryptography.

Cryptography is just one security mechanism to keep away prying eyes (or ears). There have been (and still are) many internationally accepted algorithms for encryption. For example, DES, BlowFish, PGP, RSA, IDEA, AES etc. Mathematically, cryptography has certain unique requirements: diffusion, confusion and dependence on keys.

Here is where Chaos theory comes in. Chaos theory is a branch of mathematics that deals with non-linear phenomena. These phenomena include weather, financial markets, organizational behavior, predicting epileptic seizures, fractals and other complex real-world physical phenomena. Chaotic phenomena are characterized by the fact that they are seemingly random, but have a precise mathematical formulation. Hence, given some other parameters they are repeatable/reproducible/predictable and yet apparently random. The properties required by cryptography are readily satisfied by chaotic

functions via their properties of (a)sensitive dependence on initial conditions (function parameters), (b)topological transitivity and (c)ergodicity (randomness) [5, 6, 7, 8, 9]. This makes chaos theory a good, attractive option for cryptography. Thus we see that Chaos theory has come a long way from the time a butterfly flapped its wings in the minds of Feigenbaum and Lorentz, especially in its application to cryptography and to image encryption[5].

1.2 Problem Definition

Mao *et al.*[5] propose an elegant image encryption system based on Baker maps where the 2D Baker map is extended to three dimensions, with the extension is being *more* chaotic. Further, tremendous speedup of the encryption process was achieved. Inspite of these promising results, there has been no further investigation regarding use of higher dimensional maps for efficient and fast image encryption.

In the further course of our study of chaos based cryptography, we observed that most of the image encryption algorithms use separate confusion and diffusion mechanisms. However in all these works, chaos theory plays a role only in the confusion. The diffusion mechanism does *not* use any chaotic functions. The purpose of employing chaos theory in encryption is to use its features to a greater advantage - something that has not been exploited to its fullest. We are of the opinion that chaos theory can play a vital role in all parts of the encryption scheme.

1.2.1 Contributions of the Thesis

We first set out to investigate the use of higher dimensional Baker maps for efficient, fast image encryption. We start by providing a new geometric interpretation of the Baker map. Using the geometric view, we propose a generalization of the Baker map to any arbitrary dimension. We investigate 4D and 5D Baker maps for image encryption.

Incidentally, the geometric view of Baker maps generates a multitude of

possible Baker maps. We analyzed Baker maps from a geometric *walk* point of view. Each different walk yields us a different Baker map formulation, the most general of them being a random walk based Baker map. We propose a random walk based on sparse decomposition of images. We then use this random walk based Baker map for image encryption.

In order to employ chaos theory to a larger extent, we try and combine the confusion and diffusion mechanisms using some chaotic functions. When we consider the binary representation of the intensities of the image at each pixel location, we observe that every image is inherently 3D in nature. The working of the diffusion mechanism - a substitution function of the gray-level intensities - motivated us combine confusion and diffusion by a common mechanism on the 3D view of the image, using chaos theory.

In summary, we provide a

- generalized formula for the n-D Baker map based on geometric walks and study their performance in image encryption.
- way to integrate confusion and diffusion mechanisms.
- random-walk based formulation of the Baker map for image encryption.

1.3 Outline of the Thesis

Chapter 2 reviews the current literature on the subject of chaos based image encryption. In chapter 3 our implementation of the encryption system is summarized and the differences from that implemented by Mao *et al.*[5] are discussed. Chapter 4 details the proposed n-dimensional extension, while chapter 5 explores the possibility of integrating diffusion and confusion mechanisms. Chapter 6 extends the concept of Baker maps to include random walks. We conclude the thesis with a discussion of the work and list some future work in this regard (chapter 7).

Chapter 2

A Brief Survey of Literature

If I have seen farther than other men it is because
I have stood on the shoulders of giants.
-Sir Issac Newton

If I have seen less far than other men it is because
I have stood behind giants.
-Edoardo Specchio

2.1 The Genesis

Cryptography has been around for a rather long time. The most ancient and perhaps the most simple cipher is the Ceaser cipher which is a simple cyclic shift of the english alphabet by a fixed number of alphabets. For example, "RETURN TO ROME" would become "UHWXUA WR URPH" using 3-shift. However, it is the Lord-God-Almighty of information theory - Claude E. Shannon who is widely recognized and honored as the father of modern cryptography.

Also, the seed for using chaos theory in encryption seems to have been laid more than half-a-century ago, again by Claude E. Shannon himself. He writes in his celebrated work[10]:

> *"Good mixing transformations are often formed by repeated products of two simple non-commuting operations. Hopf has shown, for example, that pastry dough can be mixed by such a sequence of operations. The dough is first rolled out into a thin slab, then folded over, then rolled, and the folded again, etc."*

His reference to *good mixing transformations* appears to be a direct allusion to modern day chaos theory.

2.2 A Gentle Introduction to Cryptography

As mentioned in the introduction, cryptography is the art and science of securing messages. Cryptography is realized through encryption and decryption procedures. Encryption is a modification of the message in such a way that its content can be reconstructed only by a legal recipient, while decryption is the reconstruction of the original message from the encrypted message (also called *cipher-text*). A cryptosystem implements the encryption and decryption mechanisms. Mathematically, a discrete valued cryptosystem is defined by a *set* of possible:

- *plaintexts* $\mathcal{P}$
- *ciphertexts* $\mathcal{C}$
- *cipherkeys* $\mathcal{K}$
- *encryption and decryption transforms*, $\mathcal{E}$ and $\mathcal{D}$

For each key $k \in \mathcal{K}$, $\exists$ an encryption function $e(k, .) \in \mathcal{E}$ and a corresponding decryption function $d(k, .) \in \mathcal{D}$, such that for each plaintext $p \in \mathcal{P}$, the condition for unique decoding, $d(k, e(k, p)) = p$ is satisfied.

Kirchoff's principle for cryptosystems assumes that the an opponent knows the structure of the encryption scheme and has access to the ciphering mechanism and claims that inspite of this, the opponent will be unable to *decipher* any encrypted message without the key.

Encryption systems are classified as block and stream ciphers. Another classification divides the encryption algorithms between symmetric and asymmetric or private and public key algorithms. Block ciphers act on a block - a fixed length of data - of plaintext. Examples of this type of ciphers are DES, IDEA, BlowFish and AES. Block ciphers work in ECB mode or CBC mode. In ECB or Electronic Code Book mode, a look-up table is used for encryption. This has the inherent weakness that identical blocks of plain text yield identical cipher texts. To over come this weakness, CBC or Cipher Block Chaining was introduced. In this mode, the previously yielded cipher block is used in the encryption of the current block of plain text, thus introducing dependencies between ciphered blocks and eliminating the problem that exists with ECB mode. Another common mode is the CFB or Cipher Feed-Back mode. In stream ciphers, a stream of pseudo-random bits, called a key-stream, are used to encrypt the plaintext. The plaintext is treated as a stream of data and this stream is operated with the pseudo-random bit stream, for example, by XOR operation. Example of stream ciphers are RC5, BSAFE etc.

Symmetric encryption algorithms use the same key for encryption and decryption while asymmetric key algorithms use a different key for encryption and decryption. Example of public-key or asymmetric key algorithms is the RSA algorithm.

There are many ways to compromise an encryption system. This field of work is called *cryptanalysis*. An attempt at cryptanalyzing the cipertext is termed an attack. The most common among them are:

- *Cipher-text only attack*: Opponent will be able to get some ciphertext and his goal is to reveal as much plaintext as possible, better still, deduce the cipher key.

- *Known-plaintext attack*: The attacker has plaintext-ciphertext pairs. The goal is to deduce the cipher key.

- *Chosen-plaintext attack*: The opponent has plaintext-ciphertext pairs *and* can choose any plaintext of his choice and obtain the corresponding ciphertext.

- *Chosen-ciphertext attack*: The attacker can choose different ciphertexts and obtain the corresponding plaintexts!

A rather straight forward type of attack is to conduct an exhaustive search of the keyspace. But, for large keyspaces, this is prohibitive unless the attacker has some additional information regarding the structure of the keyspace. Another type of attack is called differential attack where the attacker studies the effect of the encryption system on marginally differing plaintexts and use this information for breaking the encryption system. Cryptography and cryptanalysis together are referred to as *cryptology*. For further reading, the reader is urged to look at [1, 2, 11].

2.3 An Orderly Introduction to Chaos

Consider a discrete dynamical system with the general form[6]:

$$x_{k+1} = f(x_k), f : I \to I, x_0 \in I, \tag{2.1}$$

where f is a continuous map on the interval $I = [0,1]$. This system is said to be *chaotic* if the following conditions are satisfied:

1. Sensitive dependence on initial conditions:

$$\begin{aligned} &\exists \delta \geq 0 \ \forall x_0 \ni I, \epsilon \geq 0, \exists n \in N, y_0 \ni I \text{ such that} \\ &\mid x_0 - y_0 \mid \leq \epsilon \implies \mid f^n(x_0) - f^n(y_0) \mid \geq \delta. \end{aligned} \tag{2.2}$$

2. Topological Trasitivity:

$$\forall I_1, I_2 \subset I, \exists x_0 \in I_1, n \in N \ni f^n(x_0) \in I_2. \tag{2.3}$$

3. Density of periodic points in I:
   Let $\mathcal{P} = \{p \in I \ni \exists n \in N \ni f^n(p) = p\}$ be a set of periodic points of f. Then $\mathcal{P}$ is dense in I: $\bar{\mathcal{P}} = I$.

In plain english, the above definition of chaos says that (a)if any condition in the system changes at any time, we get a drastically different end system (b)every point in the space can evolve into every other point with equal probability and (c)almost (well, *almost*) every point will come back to itself after some evolution. Three words describe these conditions succinctly: (a)Expoential divergence, (b)Ergodicity and (c)Mixing.

Chaos theory is best summarized in terms of Lorentz' *Butterfly Effect* which says:

> "*...the flap of a butterfly's wings in Borneo could lead to a hurricane in Florida.*"

2.4 The Marriage: Chaos Based Cryptography

Pichler and Scharinger were the first to *use* chaotic 2D Baker map for encryption purposes[12, 13]. Soon after, Fridrich[14, 15, 16] generalized the process and provided the following general framework for chaos based encryption.

> A chaotic map is first generalized by the introduction of parameters and then discretized to a finite square lattice of points which represent pixels or some other data items.

This framework is exemplified in [17] which we summarize in detail in section 2.5.3. However, the work proposed by Fridrich was limited to a square. In [18], Baptista encrypts each character of the message as the integer number of iterations performed in the logistic equation. The intention being to "transfer the trajectory from an initial condition towards an ϵ-interval inside the logistic chaotic attractor".

In [19] Kocarev discusses lots of points regarding the strong relation between chaos and cryptography. This seems to have encouraged a lot of research in this area in the following years. The work by Kocarev *et al.*[7] was

another initial attempt to explore the relation between chaos and cryptography in more mathematical terms. Jakimoski[20] introduces a few block encryption ciphers and shows that they do not perform worse than standard block ciphers. Amigo *et al.*[21] address the basic definitions of discrete chaos and discusses the possible limitations of chaos based crypto systems. Alvarez and Shujun Li[8] draw from the couple of decades of research in the area of chaos based cryptography and outlines some basic requirements for building secure crypto systems based on chaos. It also serves as an excellent bibliography for this field.

Masuda *et al.*[22] use the tent map and a modified Baker map based on the shifting of the "center point" of the partition for the Baker map. Wang *et al.*'s work[23] is an initial attempt at introducing diffusion at the confusion stage by the introduction of sequential add-and-shift operations. However, it still has a separate diffusion step and does not use chaotic functions for diffusion. The aim here was to reduce the number of iterations required for a given level of security.

Shujun Li *et al.*[24] provide an excellent review of the state-of-the-art for chaos based encryption. The paper starts with the need for special algorithms for encryption of images is discussed. Some of these reasons include (a)bulk data capacity, (b)encryption Vs. compression, (c)redundancy, (d)loss of avalanche property, etc. The paper then goes on to do a comprehensive survey of chaotic encryption schemes for images and videos. They conclude with valuable lessons learnt. This work also doubles up as a comprehensive bibliography for this field. Other work in this field includes CKBA: Chaotic Key Based Algorithms and their cryptanalysis. Shujun Li's PhD. thesis[25] also has an excellent survey of the state-of-the-art in chaos based cryptography.

2.4.1 Migration to discrete chaos

Chaos has traditionally been in the analog domain. Properties like sensitivity to initial conditions, ergodicity etc., tend to degrade when the contin-

uous chaotic functions are discretized[17]. For example, Mao *et al.*[17] show that the discretized Cat map 'returns' after a certain number of iterations. Hence, the discretization procedure must take into account the loss of such properties of traditional chaotic maps (see section 2.5.1 of [25]).

2.5 Kolmogrov Flow, Baker Map and Cat Map

The Kolmogrov flow, the Baker Map and Cat map are permutation maps. While Kolmogrov Flow and Baker map have a lot of similarity[1], the work on Cat map by Mao *et al.*[17] provides a good insight into the working of the framework provided by Fridrich[14], as mentioned earlier.

2.5.1 The Kolmogrov Flow

The continuous Kolmogrov function is a permutation on the unit square (refer to [26]) for further details). The unit square is divided into a partition $\pi = (p_1, p_2, ...p_k), 0 < p_i < 1$ and $\sum_{i=1}^{k} p_i = 1$ of the unit interval I. The squeezing and stretching factors p_i, q_i are defined by $q_i = 1/p_i$. Define F_i as $F_1 = 0$ and $F_i = F_{i-1} + p_{i-1}$. F_i denotes the left-border of the vertical strip containing (x, y). Then, the continuous Kolmogrov function $T_\pi(x, y)$ is a defined by eqn.2.4.

$$T_\pi(x, y) = (q_i(x - F_i), \frac{y}{q_i} + F_i), (x, y) \in [F_i, F_i + p_i) \times [0, 1) \tag{2.4}$$

The discrete Kolmogrov flow is defined over the discrete block of size $n \times n$, defined by a partition $\delta = (n_1, n_2, n_3, ..., n_k), 0 < n_i < n$ with $\sum_{i=1}^{k} n_i = n$. The only restriction on the partition δ is that each n_i should partition the side of length n; ie. n_i divides n. We define $q_i = n/n_i$ and let $N_i = N_{i-1} + n_{i-1}$, with $N_1 = 0$. The discrete Kolmogrov map $T_{n,\delta}$ is defined by eqn.2.5.

$$T_{n,\delta}(x, y) = (q_i(x - N_i) + y \bmod q_i), (y \operatorname{div} q_i) + Ni). \tag{2.5}$$

[1]Thanks to GMBS Sunil for the enlightening discussions on the Kolmogrov flow which clarified the relations and differences between the Kolmogrov flow and the Baker map.

The analog and the discrete functions can be described as vertically compressing/squashing the vertical strips and stretching them in the horizontal direction, as is evident from the x and y coordinate terms of the formula. Scharinger[26] proves that this discretization satisfies the ergodicity, sensitive dependence on parameters and mixing properties. It also demonstrates the application of this discrete Kolmogrov map for image encryption, cryptographic message digests and digital watermarking. With this definition in mind, note how similar the Kolmogrov flow is to the Baker Map which is discussed next.

2.5.2 The Baker Map

The Baker Map is not a formula, it is a concept.

In [14], the continuous chaotic map is first discretized and then generalized by introducing parameters. This generalized, parametrized map is used in the encryption with the encryption key forming the parameters of the chaotic function. This work was extended from two to three dimensions by Yaobin Mao *et al.*[5, 17] for both the Baker[5] and the Cat maps[17]. We now summarize the main features of the Baker map.

The continuous 2D Baker map is defined by:

$$B(x,y) = \begin{cases} \left(2x, \frac{y}{2}\right), & 0 \leq x < \frac{1}{2} \\ \left(2x-1, \frac{y}{2}+\frac{1}{2}\right), & \frac{1}{2} \leq x \leq 1 \end{cases} \tag{2.6}$$

while the continuous 3D Baker map is defined in [5] as:

$$B(x,y,z) = \begin{cases} \left(2x, 2y, \frac{z}{2}\right), & 0 \leq x < \frac{1}{2}, 0 \leq y < \frac{1}{2} \\ \left(2x, 2y-1, \frac{z}{4}+\frac{1}{2}\right), & 0 \leq x < \frac{1}{2}, \frac{1}{2} \leq y \leq 1 \\ \left(2x-1, 2y, \frac{z}{4}+\frac{1}{4}\right), & \frac{1}{2} \leq x < 1, 0 \leq y < \frac{1}{2} \\ \left(2x-1, 2y-1, \frac{z}{4}+\frac{3}{4}\right), & \frac{1}{2} \leq x \leq 1, \frac{1}{2} \leq y \leq 1 \end{cases} \tag{2.7}$$

The 2D function is a mapping on the unit square. It has the effect of vertically compressing the right half of the unit square and converting it

to the lower half of the unit square by stretching it horizontally as shown in fig.2.1. This is achieved by first dividing the domain (in this case, the unit square $I \times I$) into thin strips (in this case, two strips). Eqn.2.8 is the generalization of the 2D Baker map[5], generalized by the introduction of an arbitrary number of partitions or horizontal strips.

$$B(x,y) = \left(\frac{1}{p_i}(x - F_i), p_i y + F_i\right), (x,y) \in [F_i, F_i + p_i), i = 1,2,3,... \quad (2.8)$$

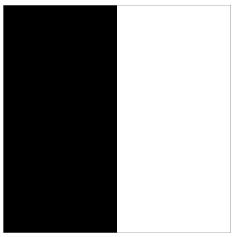
where $F_0 = 0$ and $F_i = p_1 + p_2 + \ldots + p_i, i = 1,2,3,\ldots$ with p_i being the width of the i^{th} partition/strip. We point out, however, that there is a small discrepancy in the above formula. Notice that i starts at 1. Given the interval $[F_i, F_i + p_i), i = 1,2,3,...$, we see that the above formula does *not* take into account points in the interval $[0, F_1)$. Further, if we simply reset the counter i, to start counting from 0, we find that p_0 is not defined and hence this modification does not correct this discrepancy. Instead, we rewrite the formula as in eqn.2.9.

$$B(x,y) = \left(\frac{1}{p_i}(x - F_{i-1}), p_i y + F_{i-1}\right), (x,y) \in [F_{i-1}, F_{i-1} + p_i), i = 1,2,3,... \quad (2.9)$$

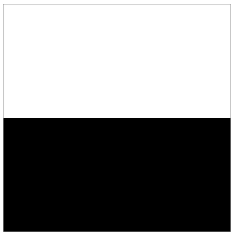
This formula was obtained by replacing all occurrences of F_i with F_{i-1}. The term $(x - F_{i-1})$ can be interpreted as the distance of the point x from the previous partition and $1/p_i$ gives the amount of stretching (since in this case p_i is less than 1). Thus the x-coordinate in eqn.2.9 gives the absolute amount by which the x-coordinate is stretched. Similar interpretation can be deduced for the y-coordinate.

The 2D Baker map vertically compresses the first strip, simultaneously stretching it horizontally and places it along the x-axis. It compresses the second strip and places it over the first compressed strip. In general, any number of strips can be used.

The action of the 3D continuous Baker map can be described as follows: the domain, (in this case, the unit cube $I \times I \times I$), is first divided into thin sub-cubes (in this formula, four strips). The 3D Baker map compresses the first sub-cube in the z-direction, simultaneously stretching it in the x- and y-



Initial configuration



After applying Baker map

Figure 2.1: Action of Baker map on the unit square.

directions, and places it on the x-y plane. It compresses the second sub-cube and places it over the first compressed cube and so on. As before, in general, any number of sub-cubes can be used, the only caveat being that we should compress to a different extent!

2.5.2.1 Discretization of the 2D & 3D Baker maps

As defined in [5], the 2D Baker map is given by:

$$B_d(r,s) = \ (\tfrac{N}{n_i}(r - N_i) + s \ mod \ \tfrac{N}{n_i}, \tfrac{n_i}{N}(s - s \ mod \ \tfrac{N}{n_i}) + N_i) \tag{2.10}$$

which can also written as

$$\begin{aligned}(r,s) &= B_d(i,j) \\ &= (\lfloor (M_{i-1} \times N + j \times m_i + i - M_{i-1}/M \rfloor), \\ &\quad (M_{i-1} \times N + j \times m_i + i - M_{i-1}) \ mod \ M)\end{aligned} \tag{2.11}$$

where we consider a square of side $M \times N$, with $M_i = m_1 + ... + m_i$, $M_0 = 0$ and $M = m_1 + ... + m_k$ for some integer k. The 3D discrete Baker map is given by:

$$S(m,n,l) = (H_{j-1} \times W + W_{i-1}) \times L + w_i \times h_j \times l + (n - H_{j-1}) \times w_i + (m - W_{i-1}). \tag{2.12}$$

The discretization is defined as follows:

$$\begin{aligned}(m',n',l') &= B_{3D}(m,n,l) \\ &= ((S \ mod \ (W \times H)) \ mod \ W, \lfloor \tfrac{S \ mod \ (W \times H)}{W} \rfloor, \lfloor \tfrac{S}{W \times H} \rfloor).\end{aligned} \tag{2.13}$$

with $W_i = w_1 + w_2 + ... + w_i, W = w_1 + w_2 + ... + w_k$ and $W_0 = 0$, and $H_j = h_1 + h_2 + ... + H_j; H = h_1 + h_2 + ... + h_t$ and $H_0 = 0$, for some integers k, t satisfying the above requirements.

Here again, we correct a discrepancy as in the case of eqn.2.8. Eqn.2.12 is rewritten as eqn.2.14:

$$\begin{aligned} S(m,n,l) = \quad & (H_{j-1} \times W + W_{i-1}) \times L \\ & + w_{i-1} \times h_{j-1} \times l \\ & + (n - H_{j-1}) \times w_{i-1} + (m - W_{i-1}). \end{aligned} \tag{2.14}$$

Note that eqn.2.14 is obtained by replacing all occurrences of w_i, h_j with w_{i-1} and h_{j-1} respectively. We now move on to a discussion of the Cat map.

2.5.3 The Cat Map

Eqn.2.15 gives the analog formula of the discrete Cat map on the unit square where x_n and y_n are the coordinates of the point under consideration.

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} mod\ 1 \tag{2.15}$$

This is generalized by the introduction of a and b as the parameters.

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & a \\ b & ab+1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} mod\ 1 \tag{2.16}$$

This parametrized map is then extended to 3D as follows:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{bmatrix} = A_z \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} mod\ 1 \tag{2.17}$$

where,

$$A_z = \begin{bmatrix} 1 & a_z & 0 \\ b_z & a_z b_z + 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.18}$$

As is evident from the formula, the number 1 at the right bottom of the matrix A_z indicates that we leave z_n unchanged, essentially performing 2D cat map in the x-y plane, leaving the z-coordinate unchanged. Similar expressions are provided for matrices A_y and A_x, where the y- and x- coordinates are left unchanged respectively[17]. These formulae are then combined to give a general 3D Cat map of the form:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{bmatrix} = A \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} mod\ 1 \tag{2.19}$$

where A is a matrix function of the parameters a_z, b_z etc. and is given by eqn.2.20:

$$\begin{bmatrix} 1 + a_x a_z b_y & a_z & a_y + a_x a_z + a_x a_y a_z b_y \\ b_z + a_x b_y + a_x a_z b_y b_z & a_z b_z + 1 & a_y a_z + a_x a_y a_z b_y b_z + a_x a_z b_z + a_x \\ a_x b_x b_y + b_y & b_x & a_x a_y b_x b_y + a_x b_x + a_y b_y + 1 \end{bmatrix} \tag{2.20}$$

We point out that Mao *et al.*[5] do not explicitly provide a method of combining A_x, A_y and A_z to arrive at the formula for matrix A. However, we have verified that a product of the these three matrices will give the matrix A. Hence we propose taking a product of the matrices A_x, A_y and A_z as a good method to obtain generalized maps, provided that the maps have a matrix notation and we are able to provide matrix formulae equivalent to A_x, A_y and A_z.

The next chapter details the description of the image encryption algorithm as described in Mao et al[5]. We have implemented this algorithm and verified the claimed results.

Chapter 3

Baker Map Based Image Encryption

I hear and I forget, I see and I remember, I do and I understand.
-Chinese Proverb

3.1 Baker Map Based Image Encryption

This chapter presents the detailed implementation of the chaos based image encryption algorithm that we implemented. The procedure that we present here was implemented and tested and is based on the work of Mao *et al.*[5, 6]. In turn, their work is based on the general framework provided by Fridrich[14] (see chapter 2).

3.2 The Encryption Algorithm

The algorithm is a five-step process as described below:

Step 0: Preprocessing. The Baker map is first discretized as described in section 2.5.2.1.

Step 1: Key Generation. A string of length 16 characters (128 bits) is chosen. The 128 bits are then split into six groups. The first four groups

contain 24 bits each and the last two groups contain 16 bits each. These six groups are mapped into six numbers: three integers (k_4, k_5, k_6) and three floating point numbers (k_1, k_2, k_3) in the range $(0, 1)$. While Mao *et al.* do not provide a method for "mapping" the bits into these numbers, we have used the following method. For the first four groups, three characters at a time are converted to an integer. For the last two groups, two characters at a time are converted into integers. To get the floating point numbers in the range $(0, 1)$, we merely divide by 1000 (since the integer is at most "three decimal places long", division by 1000 guarantees that we always get a number between 0 and 1).

Step 2: Pile up the Image to 3 dimensions. If W and H are the width and length of the image, then we set $T = W \times H$ and then factor out the prime factors and list them as $p_1, p_2, p_3, ..., p_n, 1$. An explanation is in order regarding the presence of the number 1 in the list of primes. This is best explained by an example: Consider an image of 512 pixels width/height. Only the number 2 appears in the list of prime factors of 512. If we did not include the number 1 in the list, any number of permutations on the list of 2s (there will be 18 of them) would leave the list unchanged for all practical reasons. But the introduction of the number 1 could cause a new permuatation to be formed. Back to our original discussion: clearly, $W \times H = T = p_1 \times p_2 \times p_x \times .... \times p_n \times 1$. The list of prime numbers is then permuted and the permuted list is used to create three groups of prime numbers (three groups because we want to pile up to three dimensions). In each group, the product of the prime numbers in that group is taken to get three numbers L', W', H'. Clearly, $W \times H = T = L' \times W' \times H'$. These are the dimensions of the cube to which we will "pile-up" the (2D) image.

NOTE: For the permutation process, we will need a seed and a number to determine the number of rounds to permute. For this, we will use the keys k_5 and k_6.

Step 3: Perform the 3D Baker Map. For this, we first need to partition the side of the cubes so that we get thin sub-cubes on which to work the

Baker map. To this end, we permute the logistic function using the key k_1 as the seed. Each output of the logistic function is converted into an integer. Further details are not furnished in [5]. The following is our implementation of this process. We start with a flag variable initialized to zero. As each integer is output, we compare the difference between the length of the side we are partitioning and the flag variable. If the integer is less than that difference, we add it to our list and increment the flag variable by that amount. If not, we simply discard it. This process goes on till we have got the required partition. Just to hasten the process, we used the following optimization: whenever the difference goes less that 10, we terminate the process and add that difference to our list. This process is repeated with the another key k_2 and another side of the cube. Note that it is enough to partition just two sides of the cube as this is enough to get sub-cubes (see fig.4.2).

Step 4: Diffusion. We first set $L_i = k_3$ and $S = k_4$. L_i is used as the initial condition/seed for the logistic function which is iterated until we get a value in the interval (0.2,0.8). We note that the point 0.5 is a 'trap'. The function will go on forever outputting 0.5. For this, we perturb the input to the next iteration. The diffusion is then carried out using the formula in eqn.3.1

$$C(k) = \phi(k) \oplus \{[I(k) + \phi(k)] \bmod N\} \oplus C(k-1) \tag{3.1}$$

where we use $I(0) = S$. For the inverse diffusion (to be used in decryption), we use the formula in eqn.3.2.

$$I(k) = \{\phi(k) \oplus C(k) \oplus C(k-1) + N - \phi(k)\} \bmod N \tag{3.2}$$

While Moa *et al.*[5] do not provide a proof of 3.2 being the inverse of 3.1, we propose the following self-explanatory proof using the properties of the mod

operator:

$$\begin{aligned} & C(k) = \phi(k) \oplus \{[I(k) + \phi(k)] \bmod N\} \oplus C(k-1) \\ \implies \quad & C(k) \oplus \phi(k) \oplus C(k-1) = [I(k) + \phi(k)] = I(k) \bmod N + \phi(k) \\ \implies \quad & [C(k) \oplus \phi(k) \oplus C(k-1) - \phi(k)] = I(k) \bmod N \\ \implies \quad & [C(k) \oplus \phi(k) \oplus C(k-1) - \phi(k) \pm rN] \bmod N = I(k) \\ \implies \quad & [C(k) \oplus \phi(k) \oplus C(k-1) - \phi(k) + N] \bmod N = I(k) \end{aligned} \tag{3.3}$$

Step 5: Transform back to 2D. The image is converted back to 2D from the 3D cube.

3.3 Performance Metrics

To make infeasible statistical attacks, we need to minimize the amount of statistical information available to the attacker. One way to measure this type of information is to plot the histograms of the plain and ciphered images. The histogram does not yield much information to the attacker if the histogram is relatively flat over the entire graylevel set. Yet another way to measure how much statistical information is available is to measure the correlation between adjacent pixels in the plain and ciphered images.

Further, the concept behind differential attack was introduced earlier. In order for this type of attack to be made infeasible, we need to ensure that a small change in input causes a significant change in the output. This is sometimes referred to as the *avalanche effect*. For this, two metrics have been proposed: NPCR or Number of Pixels Change Rate and UACI or Unified Average Changing Intensity, which are defined as follows.

Given two input images that differ in only one pixel and whose ciphered images are C_1, C_2,

NPCR is defined as a binary array $D(i,j)$ as $D(i,j) = 1$ if $C_1(i,j) - C_2(i,j) = 0$ and $D(i,j) = 0$ otherwise. In essence, D is a measure of the percentage of the number of locations that are different in the ciphered im-

ages. Thus NPCR is defined by:

$$NPCR = \frac{\sum_{i,j} D(i,j)}{W \times H} \times 100\% \tag{3.4}$$

Clearly, larger the NPCR, better the encryption procedure.

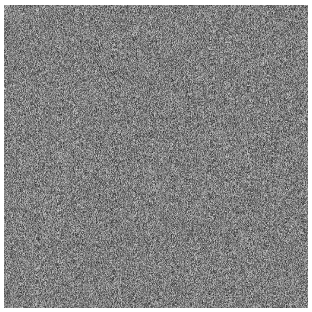
UACI is defined as

$$UACI = \frac{1}{W \times H}\left[\sum_{i,j} \frac{C_1(i,j) - C_2(i,j)}{255}\right] \times 100\% \tag{3.5}$$

Again, larger the UACI value, better the encryption scheme.

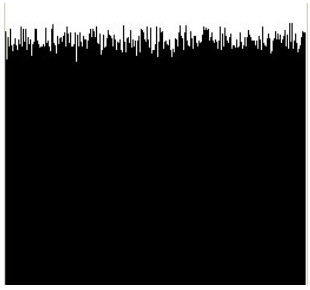
3.4 Results and Conclusions

Original Image



3D Baker map Encrypted

Histogram of Original Image



Histogram of encrypted image

Figure 3.1: Encryption using 3D Baker maps (key used: 1234567890123456)

Fig.3.1 shows the results of the image encryption process. Notice that the encryption process flattens the histogram thus making the statistical

and known-ciphertext attack infeasible. Thus, 3D Baker map based image encryption is a viable, feasible option for encryption schemes.

The following table indicates that the encryption procedure decorrelates pixels that are adjacent horizontally, vertically and diagonally.

	Plain Image	**Ciphered-Image**
Horizontal	≈ 1	0.0175
Vertical		0.0020
Diagonal		-0.3499

Table 3.1: Correlation between adjacent pixels for 3D Baker map based encryption.

The platform of choice was a PC with an Intel Pentium 4, 2.4 GHz processor with 1 GB RAM running Microsoft Windows XP Professional with SP2. We used the OpenCV[27] library for handling images and related functions. For builds, we used the GNU MinGW 5.1.3 and MSYS 1.0.10.

In the later chapters, we introduce further enhancements through simplifications. For example, we provide an alternate treatment of the Baker maps and using this treatment, we can extent the Baker map to any arbitrary dimension.

Chapter 4

Generalized n-Dimensional Baker Maps

Things should be made as simple as possible, but no simpler.
-Albert Einstein

Bloch: Space is the field of linear operators.
Heisenberg: Nonsense, space is blue and birds fly through it.
-Felix Bloch,
(in Heisenberg and the early days of quantum mechanics*).*

4.1 Introduction

Fridrich[14] first proposed a general framework for using chaos theory for image encryption. He generalized an existing chaotic function by the introduction of some parameters. The paper on Cat map [17] is a very lucid implementation of this framework. This generalized function is then used for encryption, with the generalization parameters acting as the 'key' of the crypto system. Since then, the 2D and 3D Baker maps have been extensively used in chaos based image encryption. In this paper, a novel way of interpreting the discrete Baker map based on the path taken while traversing the image is presented. This led us to formulate a new path

function S. Using this new perspective, we propose a generic approach for the construction of n-dimensional Baker maps. To assess the encryption quality, we employ an additional measure called "the degree of flatness" defined by MSE of the normalized histogram of encrypted image. Experimental results indicate that baker maps of dimension greater than 3 do not seem to improve any of the quality measures and 3D Baker map is therefore sufficient for chaos based image encryption.

4.2 A Novel Interpretation

We now provide a new interpretation for the formula represented by eqn.2.11. which can be rewritten as:

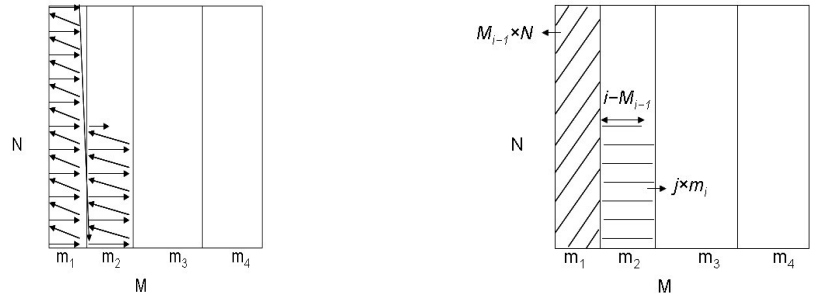
$$(r, s) = B_d(i, j) = (S/M, S \ mod \ M) \tag{4.1}$$

where $S = (M_{i-1} \times N + j \times m_i + i - M_{i-1})$.

Consider the following pseudo code for implementation of this formula, with N, k defined as above, and the array mi representing m_i:

```
for(i=1;i<k+1;i++)
   for(j=mi[i-1];j<mi[i];j++)
      for(k=0;k<N;k++)
            calculate S;
            perform Bd(i,j);
```

S can now be interpreted as follows. These loops traverse the image one vertical rectangle at a time. It first traverses the bottom row of the first column, then the second row from the bottom of the (same) first column and so on until the first column is exhausted. Next, it moves on to the bottom most row of the second column and proceeds successively to the rows above it. Fig.4.1 pictorially depicts the general case. Given i, the first term in the expression for S, $M_{i-1} \times N$, gives us the area/number of pixels under the columns that have been completely traversed so far. The second term, $j \times m_i$,



The 2D traversal path. Interpreting the S formula.

Figure 4.1: Traversal of the 2D image.

gives the area/number of pixels that have been traversed in the column that is currently being traversed. The last term, $i - M_{i-1}$, gives the number of pixels that have been traversed in the row of the column currently being traversed. Thus, the quantity S can be thought of as the number of pixels traversed so far, given the current path of traversal. Note that the formula is a one-to-one map from (x, y) to $[0, (M \times N) - 1]$. Our interpretation is consistent with this property: when we move one step in the loop, we have moved to the next pixel in the image. Thus, we also add 1 to the quantity S. When the pixel under consideration is the first pixel of a column, only the first term contributes to S; when the pixel under consideration is the first pixel of the row (in any given column), the second term alone contributes. Otherwise most of the time, third term contributes. Note that the usage of 'first pixel' is related to the path being taken in the traversal. The 'effect' of the Baker maps is to cause 'kneading' of the image (and hence the name, Baker map). In fact, the Baker map is a systematic algorithm to achieve 'kneading' of baker's dough!

At this stage, we wish to point out that the formula in eqn.2.12 provided by Mao *et al.* in [5], is not one-to-one and therefore did not lend itself to decryption. For encryption to be reversible, i.e., to make decryption possible, we need the Baker map to be one-to-one. Since S is a mapping from (x, y, z) to $[0, (L \times W \times H) - 1]$, and the Baker map is written in terms of S, S should

also be one-to-one function. Hence, taking cue from the new interpretation of S for the 2D Baker map, we propose this following alternative formula for $S(x, y, z)$ for the 3D case:

$$\begin{aligned} S_{new} &= H_{j-1} \times W \times L + W_{i-1} \times h_i \times L \\ &+ w_i \times h_i \times z + w_i \times (y - H_{j-1}) + x - W_{i-1} \end{aligned} \tag{4.2}$$

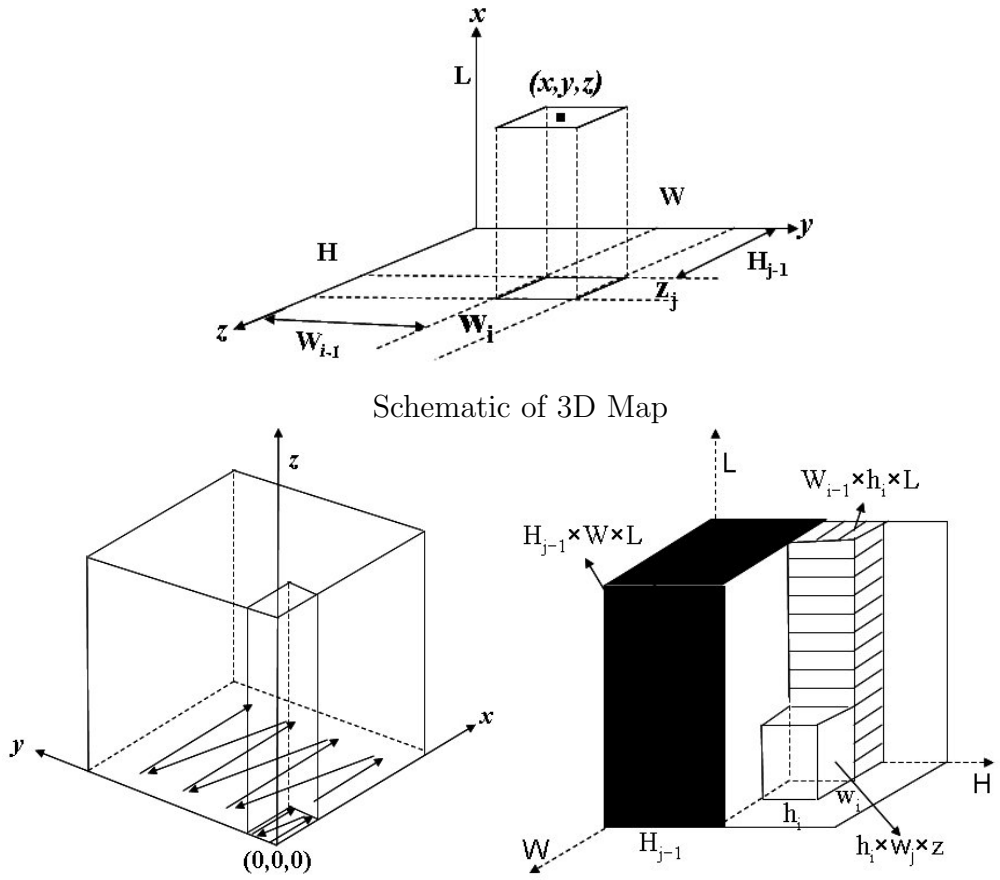


Figure 4.2: Traversal of the 3D image.

Fig.4.2 provides a pictorial interpretation of this formula, which is similar to the 2D case. The following pseudo code implements the formula for the discrete 3D case with array wi representing w_i, the array hj representing h_j. H_j and W_i are as shown in fig.4.2. Here, these loops traverse the 3D cube, one

sub-cube at a time. First, the first row of the first sub-cube is traversed. Once a rectangle has been completed, the next rectangle above it is traversed, and so on until we have completely walked through the entire sub-cube. Then, the next sub-cube is traversed bottom-up in a similar fashion and so on.

```
for(i=1;i<k+1;i++)
   for(j=1;j<t+1;j++)
      for(z=wi[i-1];z<wi[i];z++)
          for(y=hj[j];y<hj[i];y++)
             for(x=0;x<L;x++)
                    calculate S;
                    perform B3d(i,j);
```

4.2.1 Efficient Implementation

Under this new interpretation, note that in the pseudo code, the calculation of the path function S merely increments the value of S by 1. Thus, we can forgo $M \times N$ function calls and replace the function call to S by S++. This holds true for both 2D and 3D versions of the Baker map and in general, for any dimension as will be evident from the following discussion.

4.3 Extension to Higher Dimensions

From the results of Mao *et al.*[5] and our own implementation, it is clear that the 3D Baker map performs better than the 2D map. This motivates us to extend the Baker map to higher dimensions and investigate its performance. We note that if we have a = b mod c, then b = c * k + a, where k = b/c. Let us consider the following modification of eqn.4.1:

$$(r, s) = B_d(i, j) = (S \ mod \ M, S/M) \tag{4.3}$$

Notice that the quantity S has been proportioned into two parts using the modulo and the / operators. Since we are operating S on M, if we 'read'

$\lfloor S/M \rfloor$ as the integral number of times M can be removed from S (which is indeed the definition of division), we can interpret the quantity $\lfloor S/M \rfloor$ as the number of rows of size M that can be cut-out from S, or the x-coordinate of the Baker transformation of S. If we read quantity S mod M as the number of pixels left over after division by M (which is indeed the definition of modulo), we can think of this as the number of pixels that remain after removing $\lfloor S/M \rfloor$ rows of pixels of length M. Hence, we can interpret S mod M as the y-coordinate of the Baker transformation of S. A similar interpretation also holds true for the 3D Baker map given by eqn.2.7. We observe that the formula in eqn.4.1 given by Mao *et al.*[5], is equivalent to eqn.4.3, except that the encrypted image has its dimensions flipped i.e., an M-by-N image is encrypted into an N-by-M image. We now propose the following formulae for 4D and 5D Baker maps.

$$\begin{aligned} B_{4d} = \; & ((((S\%vol)\%(area))\%L), \\ & (((S\%vol)\%(area))/L), \\ & ((S\%vol)/(area)), \\ & (S/vol)) \end{aligned} \tag{4.4}$$

$$\begin{aligned} B_{5d} = \; & ((((S\%hvol)\%vol)\%(area)\%L), \\ & (((S\%hvol)\%vol)\%(area)/L), \\ & (((S\%hvol)\%vol)/(area)), \\ & ((S\%hvol)/vol), \\ & (S/hvol)) \end{aligned} \tag{4.5}$$

where % denotes the modulo operator, $area$ is $L \times W$, vol is the volume ($L \times W \times H$) and $hvol$ is the hyper volume ($L \times W \times H \times F$). The use of notations like volume and hyper volume is in agreement with the interpretation we provided at the beginning of this section. In the 2D case, we divided by lengths alone. In the 3D case, we divide by areas and lengths. In 4D we have volumes, areas and lengths while 5D also includes hyper volume.

4.4 Paths Are Many, Goal Is One

What is presented in section 4.2 is but one of the many possible paths that can be taken while traversing the image. Note that as long as we calculate the path function S based on the location of the pixel, every pixel will get mapped to exactly the same location because S uniquely maps every location and no matter what path is taken for traversing the image, the Baker map we obtain will be the same. But once we adopt the new, efficient S++ implementation of the path function (see section 4.2.1), we could use many different paths for traversing the image and get a different Baker map for each implementation. For example, the path described in section 4.2 traverses each strip horizontally first and moves upwards. Another obvious way to traverse the strip will be to traverse the strip vertically and move right as each column of pixels is completed (see fig. 4.3). Still another way is to traverse the strip in a spiral. Spiral paths open many more possibilities of traversal: traverse inside out, outside in, travel left, travel right, start traversal from top-left, top-right, bottom-left, bottom-right and so on. Thus we see that imagination alone limits the way we can define a Baker map! Chapter 6 discusses the ultimate walk based Baker Maps: Random Walk based Baker Map. Another related idea is the partitioning into strips/cubes.

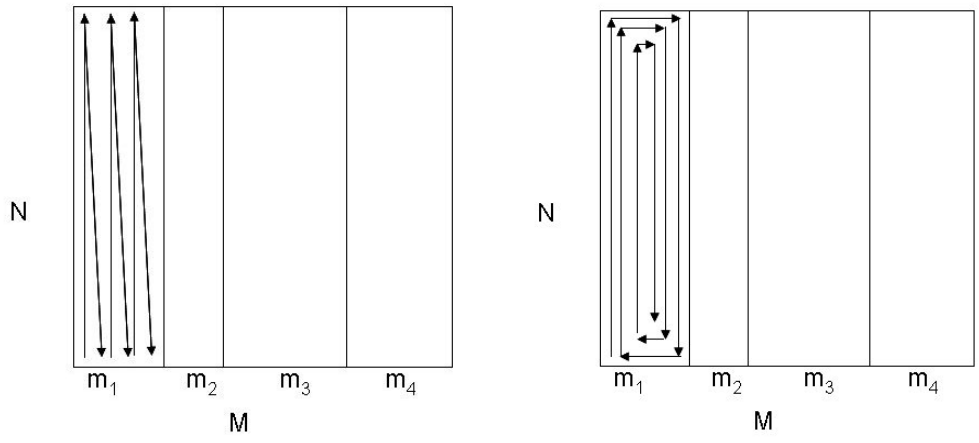


Figure 4.3: Vertical and Spiral Walks

In section 4.2, we partitioned the images vertically. Alternatively, we could partition the image horizontally and pile up the "baked" slices vertically adjacent to each other. For 3D Baker Maps, the sub-cubes are depicted in 4.2 are vertical. Again, we could have the cubes along the x-axis or along the y-axis. Each of these again give us different Baker maps - but the concept behind the Baker map is the same: compress and stretch simultaneously, in effect simulating the baker's "kneading" action over many iterations.

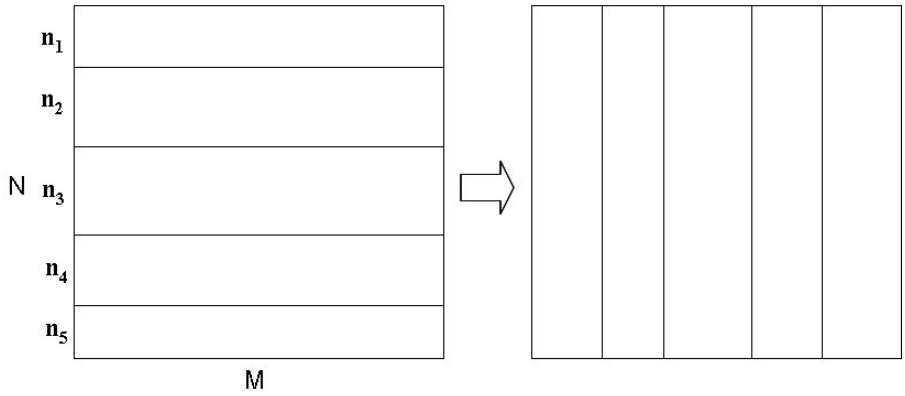


Figure 4.4: Alternate way to partition the image - partition horizontally and pile vertically.

4.5 A New Performance Metric: Mean Square Error of Normalized Histogram

As mentioned earlier, we wish to flatten the histogram of the ciphered image in order to thwart or make infeasible statistical and known ciphertext attacks. To measure the flatness of the histogram, we employ the MSE of the normalized histogram of the image as the metric. Usually, the MSE is defined by:

$$MSE = \frac{1}{K}\sum_{i=1}^{K}(x_i - \overline{x})^2 \tag{4.6}$$

where K is the number of bins in the histogram and $\overline{x}$ is the mean of these K histogram points. In our case, we want to take the MSE of the histogram of the image, the MSE taken w.r.t. the *ideal* mean. Meaning, if our image is a graylevel image, there are exactly 256 bins in the histogram. So, if the image size is 512×512, the *ideal* flat histogram has exactly $(512 \times 512)/256 = 1024$ pixels distributed into each bucket. Thus, the *ideal* mean for this image is 1024. Hence we will use $\overline{x} = 1024$ while computing the MSE for a 512×512 image.

To verify that the histogram of the image is indeed uniformly flat in all parts of the image, we will measure the MSE at varying window sizes. To start with, we could use a 64×64 window and grow the window size by a fixed amount till we cover the entire image. For each window size, we calculate the MSE of the block starting at the top left corner. We next move the window to the next non-overlapping block of the same size in the horizontal direction. Once we complete traversing horizontally, we move down and traverse horizontally again. This traversal is much like running a mask on an image, except that we choose non-overlapping blocks for calculating the MSE. But, this presents us with the following new problem: Since the window size is varying, the mean $\overline{x}$ will keep increasing with window size. Thus, the MSE will increase with increasing window size, leading to the misleading interpretation of loss in flatness of the histogram with increasing window size!

To overcome this problem we note that we will need to standardize the histograms across window sizes. The simplest way to standardize a histogram is to normalize it. Hence, we will normalize the histogram before we take the MSE. Another crucial point to note here is that once the histogram is normalized, in the MSE formula we will have to use the normalized mean which is $\overline{x} = 1$.

In our case, we start with a window size of 125 and progressively increased the window size in steps of 50. At each window size, the MSE values chosen are the min and max of the MSE calculated over the entire image. For a

comparative study, we plot the max and min values vs. increasing window sizes. This helps us to monitor local variations with increasing window size.

We now describe the normalizing factor we will use. To standardize the histogram, we want to make the number of pixels in each bin appropriately scaled (up or down as required). We propose to divide the number of pixels in each bin by the ideal mean; i.e., we choose the normalizing factor to $N/256$, where N is the total number of pixels in the current image/window. Consider the extreme case where all the pixels are in the same bin. Then, for that bin containing all the pixels, the term $x/(N/256)$ will evaluate to 256 since $x = N$. Thus the difference term in the MSE formula for this bin will be 255^2 i.e.,$(256-1)^2$. For all other bins, it will be $(0-1)^2$. Hence the MSE will be $(1/256)(255^2+255) = (1/256)(255+1)255$ which is exactly equal to 255, which is one less than the number of bins.

Normalizing the histogram before taking the mean is mathematically equal to dividing the difference between the sample and the mean by the normalizing factor and then squaring that term (normalized histogram$_i = \frac{x_i}{N/256}$):

$$\begin{aligned} MSE &= \frac{1}{256}\sum_{i=1}^{256}\left(\frac{x_i}{(N/256)}-1\right)^2 \\ &= \frac{1}{256}\sum_{i=1}^{256}\left(\frac{x_i-(N/256)}{(N/256)}\right)^2 \\ &= \frac{1}{256}\sum_{i=1}^{256}\left(\frac{x_i-\overline{x}}{(N/256)}\right)^2 \end{aligned} \tag{4.7}$$

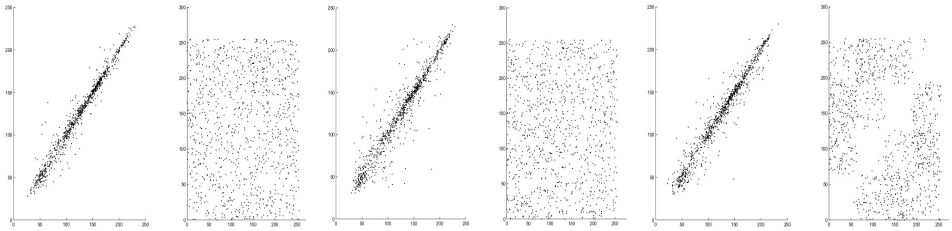
4.6 Results

We implemented the 2D, 3D, 4D and 5D Baker maps as given in eqns. 2.10, 2.13, 4.4, 4.5. The encryption algorithm was tested on the 512x512 Lena image. One round of encryption using 3D Baker map is 'visually' better than one round using the 2D (see fig.4.6). More quantitatively, about 5 rounds the 2D Baker map are required on an average to even 'visually' compete

with 3D map. The correlation between adjacent pixels is a measure of the mixing achieved by the map (see fig.4.5 and table 4.1). We consider a random sample of 1000 pixel locations that are horizontally, vertically and diagonally adjacent and calculate the correlation between the pixels at those locations before and after encryption. We observe that the encryption process actually decorrelates the adjacent pixels. The quantitative results are tabulated in the table 4.1 for just one round of encryption.

	Horizontal	**Diagonal**	**Vertical**
2D	0.0013	0.0190	-0.2650
3D	0.0175	0.0020	-0.3499
4D	0.0138	0.0130	-0.2606
5D	0.0087	0.0076	-0.2440

Table 4.1: Correlation among adjacent pixels for the images ciphered with various Baker maps.



Horizontal Diagonal Vertical

Figure 4.5: Correlation between adjacent pixels (4D). The plot on the right side is of the encrypted image while that on the left is of the original image.

4.7 Conclusions and Discussion

We see from fig.4.5, that the encryption process is indeed de-correlating the adjacent pixels. Further, from the plots of the histograms in fig.4.6, we

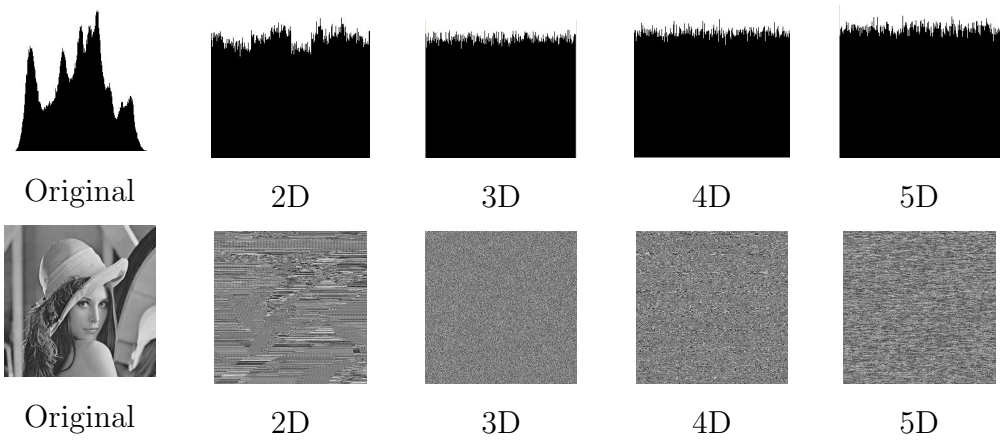


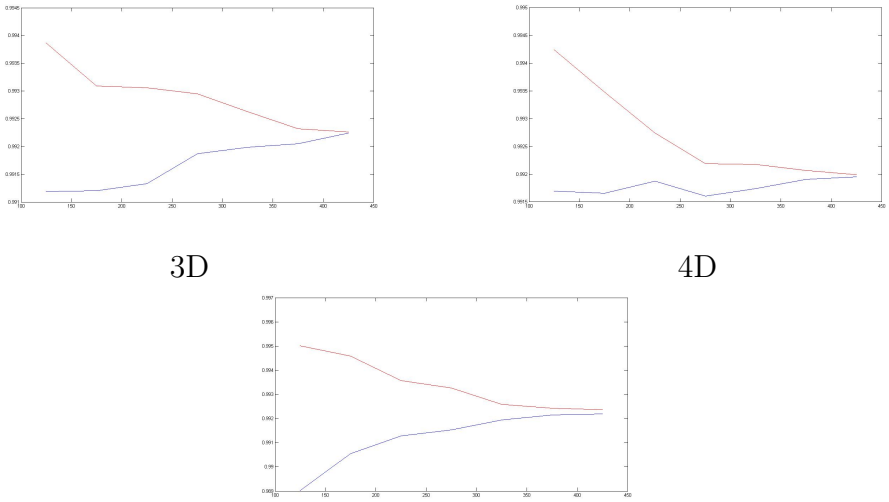
Figure 4.6: Encrypted Images and their histograms.

see that relative to the 3D map, the 4D map has not enhanced the flattening of the histogram of the encrypted image. To measure the flattening of the histogram of the encrypted image, we use the MSE deviation of the normalized histogram of the encrypted image from the ideal flat histogram with the following normalization factor: $(L \times W)/256$ where $L \times W$ is the number of pixels in the image and 256 represents the 256 bins of the gray-level histogram. The difference between the MSEs of 3D and 4D histograms is of 10^{-10} order of magnitude. Fig. 4.7 is a plot of the MSE vs. window size.

The two plots in each graph (fig.4.7) are the upper and lower bounds of the MSE of normalized histograms for various Baker maps. We see that while there is a larger difference between the min and max of the MSE for smaller window sizes while for larger window sizes, the difference between the min and max is close to zero. i.e., the histogram is "more flat" globally.

In summary, we see that while we can easily extend the Baker map to higher dimensions, the 3D Baker map is adequate for efficient, fast image encryption purposes. We wish to point out that as we move to higher dimensions, the number of pixels available to each dimension reduces drastically. This is because the total number of pixels available remain constant and the number of pixels distributed to the various higher dimensions differ.

We remark that the applicability of higher dimensional maps might be in



3D 4D

5D

Figure 4.7: Plot of upper and lower bounds for MSE of normalized histograms.

encrypting larger images. Since larger images imply larger number of pixels available for each dimension and hence, the higher dimensional Baker maps would 'behave' more chaotically. Thus we note that even though for small images 3D Baker maps would suffice, for large images, higher dimensional Baker maps would have an added advantage. However, this is subject to experimental analysis.

Chapter 5

Embedding Diffusion in Confusion

Not all diffusions are created equal.
-Thomas F. Denove

5.1 Introduction

Most block-cipher image encryption schemes based on Chaos theory have independent modules for confusion and diffusion processes. So far no attempt has been made to integrate these mechanisms to make the encryption process efficient. In this paper, we extend 2D images to 3D by using grayscale image intensities in 8-bit binary form and then applying the 3D Baker map based confusion algorithm. Thus, the diffusion process is accomplished by a permutation of binary bits in the third dimension eliminating the need for a separate diffusion process. The proposed method is also extended to color images by using the 24-bit color information. Color image encryption is usually performed by encrypting each channel independently and then combining to get the encrypted image. We demonstrate that with this simplistic approach, decrypting even a single channel would reasonably reveal the information contained in the image. In our approach, this drawback is

eliminated by the introduction of dependence on the data contained in all the channels highlighting the inherent superiority of the proposed algorithm for color image security.

5.1.1 Brief Recap

Cryptography has certain unique mathematical requirements: diffusion, confusion and dependence on keys. These properties are readily satisfied by chaotic functions by their sensitive dependence on initial conditions (function parameters), topological transitivity and ergodicity (randomness) [8, 6, 7, 5]. This makes chaos theory a good, attractive option for cryptography. Therefore, chaos theory has been successfully applied to cryptography for about a decade now. Josef Scharinger[28] introduced chaos theory to encryption. Thereafter, Fridrich[14] proposed a general framework for using discrete chaotic maps for image encryption. This framework has been used time and again for image encryption, for example the works by Mao *et al.*[5, 17]. In this framework, the analog chaotic map is first discretized. Next, it is generalized by the introduction of some parameters. This map is then extended to three dimensions. The parameters of the map serve the purpose of the 'key' for the encryption system. This discrete generalized parametrized map is used in the encryption algorithm. As an example, refer to section 2.5.3 for the work on Cat map proposed by Mao *et al.*[17].

5.2 Motivation

Traditional crypto-systems are often viewed as substitution-permutation (SP) networks[26], modelled by eqn.5.1 in terms of confusion(C) and diffusion(D) functions[29].

$$Y = [D(C(X, K_1), K_2)]^n \tag{5.1}$$

where X is the message to be encrypted, K_1, K_2 are the keys for confusion and diffusion respectively. In our case, confusion is achieved by the use of

the Baker Map, which is essentially a permutation map while diffusion is achieved by substitution. We first examine the process of diffusion in more detail. Mao at al.[5, 17] use the following function for diffusion:

$$C(k) = \phi(k) \oplus \{[I(k) + \phi(k)] \ mod \ N\} \oplus C(k-1) \tag{5.2}$$

The pixel intensity $I(k)$ is replaced by a new pixel value $C(k)$, the ciphered, or more precisely, the diffused value. In essence, eqn.5.2 simply changes the intensity value apart from permuting the image. We note that this diffusion mechanism does *not* use chaos theory.

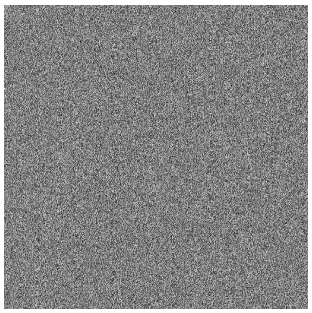
5.3 3D Baker Map Based Image Encryption

As is evident from the framework described in sect.5.1.1, the image should be re-arranged into three dimensions to be able to use the 3D maps for encryption. We first present results based on traditional methods to 'pile up images to higher dimensions'. One of the standard methods to re-arrange the map is to first find three integers L', W', H' such that $L' \times W' \times H' = M \times N$, where M and N are the dimensions of the image. Then, the pixels of the image are distributed into the three dimensions such that the above equality is satisfied. Let us call the product $M \times N$ as P. The product P is factorized into its prime factors $p_1, p_2, p_3, ..., p_n$ for some n. This list of prime factors is then divided into three parts which are then multiplied independently to get L', W' and H'. The process of dividing into three parts uses the key provided by the user (see [5]). Fig.5.1 shows the result of encrypting an image using 1234567890123456 as the key.

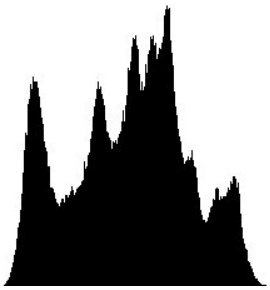
5.4 Embedding Diffusion in Confusion

In our analysis, we view substitution of intensity values as a permutation of the bits that make up the intensity value. For example, 145 in binary is 10010001 and 146 in binary is 10010010, a mere permutation. Thus a

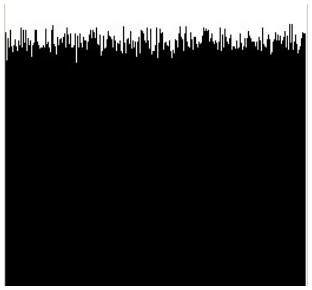
Original Image



3D Baker map Encrypted



Histogram of Original Image



Histogram of encrypted image

Figure 5.1: Encryption using Baker maps with traditional piling algorithms (key used is 1234567890123456)

substitution of 146 for 145 is a mere permutation of the bits in its binary representation. This simple observation gives us a novel approach to extend the image to three dimensions in which we integrate confusion and diffusion at the decimal representation by using only permutations on the binary representations. We emphasize that this permutation at the binary representation level is equivalent to integrating confusion and diffusion at the decimal level. Thus, by integrating confusion and diffusion using only permutations based on chaotic functions, we attempt to bring the properties of chaoticity into the diffusion mechanism as well.

5.5 Novel Extension to Three Dimensions

We propose the following simple treatment of an image: The image is viewed as a cube that is composed of bits arranged in three dimensional space. At each pixel location of this cube, the z-axis consists of the bits of the 8-bit binary representation of the intensities of the pixels at that location. For example, if the pixel value at location (10,10) of the image is 145, then its binary equivalent is 10010001. Thus, we consider the image to be a cube composed of binary strings where the LSB (least-significant-bit) is at the top of the cube while the MSB (most-significant-bit) is at the base of the cube (see fig.5.2). We now see that *every image is inherently three dimensional in nature*. Fig.5.3 shows the result of using this novel approach to encrypt an image. We see that reasonable flattening of the histogram is achieved.

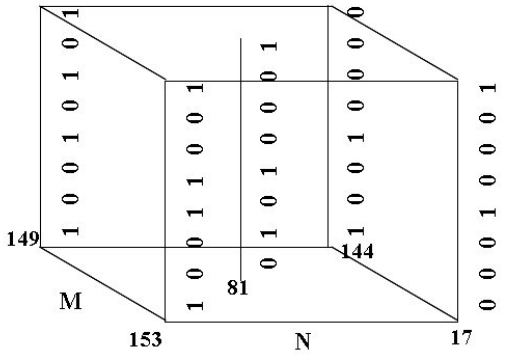
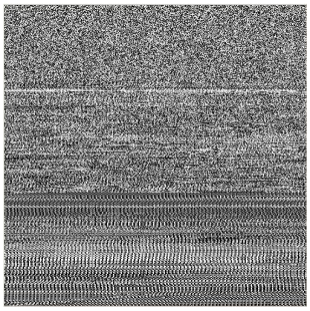


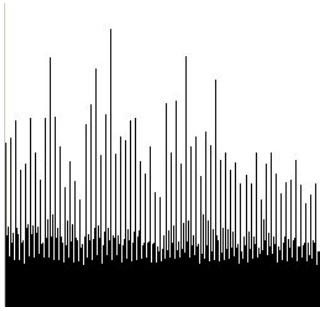
Figure 5.2: The cube formed by the binary representation of the pixel intensities provides an alternative view of the image in 3 dimensions.

5.6 Color Image Encryption

A naive approach to color image encryption is to independently encrypt the three channels (preferably using different keys). We observe that it is enough to decrypt just one channel of the color image thus encrypted to be able to see the 'necessary content'. Fig.5.4 (a) shows that encryption of a



3D Baker map Encrypted



Histogram of Encrypted Image

Figure 5.3: Encryption using Baker maps with binary-string piling algorithm

single channel is not enough - to the human eye, the image is almost indistinguishable from the original. (b) shows that decrypting just one channel reveals enough information to a human attacker of the system.

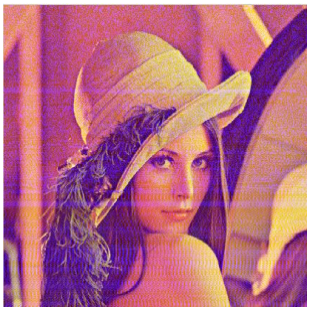
(a)Only R channel encrypted

(b)Only B channel decrypted

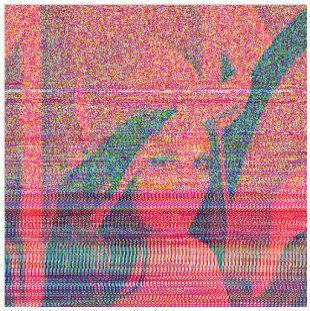
Figure 5.4: Partial decryption using Baker maps with traditional piling algorithm (key used is 1234567890123456)

Alternatively, we could expand the intensity in each channel to 8-bits and use our new approach to encrypt each channel independently. But, this also has the same drawback, ie. decryption of just *one* channel is enough to give away the necessary visual information (see fig.5.5).

We now use the binary-string approach for color image encryption. We consider each pixel location to be consisting of a triplet (i_1, i_2, i_3), comprising of the intensities for each of the three channels. We expand the intensity values into their binary representation and concatenate the binary strings to



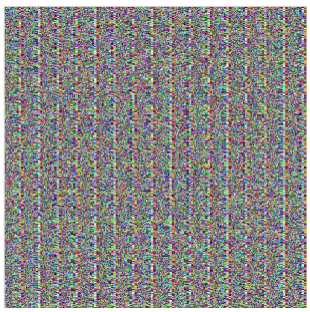
(a)Only R channel encrypted



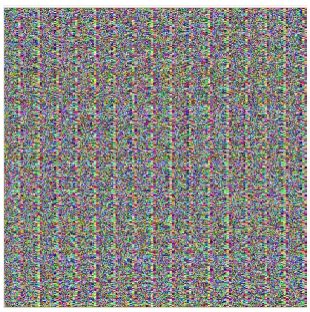
(b)Only B channel decrypted

Figure 5.5: Partial decryption using binary-string piling algorithm, where each channel has been expanded to its binary representation and each channel is independently encrypted

form a 24-bit string at each pixel location. The cube formed by these strings is then subjected to the Baker map encryption algorithm - sans the piling. For partial-decryption, we wish to decrypt just one channel. That is, we treat the 24-bit binary-string based encrypted image as if it were composed of three channels of 8-bits each and use the 8-bit binary-string based decryption. The result of subjecting this cube of 24-bit binary strings is given in fig.5.6. We see that decrypting just one channel does *not* reveal any visual information.



Encrypted



Partially Decrypted

Figure 5.6: Encryption using Baker maps with binary-string piling algorithm with the binary representations of each channels concatenated.

5.7 Conclusions and Discussion

In this chapter we propose an alternative treatment of an image which helps us to achieve simultaneous confusion and diffusion. This integration allows us to incorporate chaoticity into the diffusion mechanism as well - a feature lacking in the diffusion formulae proposed in the literature. Our results show that a reasonable flattening of the histogram is achieved, as is evident in fig.5.3. However, we wish to point out that this algorithm is inherently slower than the traditional encryption. In traditional encryption schemes, the number of permutations is at least equal to the number of pixels in the image, while in our approach, the number of permutations per round is at least equal to 8 times the number of pixels (and hence 24 times for 3 channel color images) because the binary representation of the intensity has 8 bits. This process can be speeded up by the use of look-up tables. The look-up table can be generated once and then reused for multiple rounds of the map.

Further, in the traditional Baker map based encryption schemes, the piling up algorithm uses part of the key. In this approach, there is no occasion to use the key for piling since there is no piling involved! The superior performance of our algorithm is in the case of color image encryption, where we treat the image as a cube of height 24 (bits), where 8 bits are contributed to by each channel of the color image.

Chapter 6

Random Walk Formulation of Baker Maps based on Sparse Decomposition of Images

Random numbers should not be generated with a method chosen at random.
-Donald E. Knuth

Any one who considers arithmetical methods of producing random numbers is, of course, in a state of sin.
-John Von Nuemann

6.1 Introduction

We mentioned in section 4.4 that we would discuss random walk based Baker maps. In this chapter, we explain how we designed the random walk for the Baker map and show its application in image encryption.

A random walk, as the name suggests, is a walk that is unpredictable. This unpredictability is what random walks share with chaos theory. But, we have some unique restriction on the random walk to be able to use it for the Baker map. We need the Random walk to be a *Hamiltonian Cycle*. A hamiltonian cycle is a simple closed path that contains all the vertices

of the graph. We can model an image as a graph with each pixel location being a vertex of the graph. In our case, we require that every pixel location be covered. Moreover, any given location should be covered exactly once (in graph theoretic jargon, this second condition translates to a *simple cycle*)[30].

6.2 Sparse Decomposition of Images

For generating the random walk, we use the sparse matrix decomposition first introduced by Hoffman *et al.*[31]. Hoffman *et al.* apply different "criteria" to every pixel and then peel them off into sets - the sparse images. The outline of the process we employ for sparse decomposition of images for generating a random walk is as follows. The image is first decomposed into a set of sparse images. Next, each of the sparse images is traversed in some manner. This gives us a "random" walk that covers every single point of the image.

Chandrasekaran *et al.*[32] modified Hoffman's scheme to allow a pixel to participate in the selection process more than once. It can be considered as sparse image decomposition with replacement - a pixel can be present in more than one sparse image. For this, a time varying quantity called the $freq$ variable was introduced. In effect, the $freq$ variable creates a family of $control()$ functions - one $control()$ function for each value of $freq$. The new formulation of the system is the following definition:

$$input() = f(x, y, z) \tag{6.1}$$

$$control() = g(input(), freq) \tag{6.2}$$

$$output() = h(control(), freq) \tag{6.3}$$

where z is the intensity value at the location (x, y). A typical example of an input function is the Euclidean distance: $\sqrt{x^2 + y^2 + z^2}$. This quantity is used as a parameter for a set of control functions which are essentially like gating functions. To get the sparse images, we choose one control function at a time and traverse the entire image using the output function as the criterion

to determine if the current pixel can belong to the sparse image corresponding to the current control function. The output function is a threshold function which determines if the gate (control function) has to open or close. If the gate opens, then the pixel under consideration is added to the sparse image corresponding to that control function. This way, a set of sparse images can be obtained.

However for our purposes, we need to restrict a pixel to exactly one sparse image (in order that our Baker map remains a one-to-one function). To achieve this, we modify the approach of Chandrasekaran *et al.* by introducing a participation function $\mathcal{P}(x, y)$, which is simply a different interpretation of the $freq$ variable. We multiply the $control()$ function with the participation function $\mathcal{P}(x, y)$ to get $control_{new}() = \mathcal{P} \cdot g(input())$. If we implement $\mathcal{P}(x, y)$ as a binary matrix which has been initialized to 1s, we can set $\mathcal{P}(x, y)$ to zero as soon as a pixel has been output to any sparse image. Thus, in future calculations of that pixel, we will get zero as output from the control function which we will consider as a closed gate, thus preventing that pixel from participating in more than one sparse image.

We can implement the above setup in an efficient manner so as to do away with the participation function. We will traverse the image just once during which we will test every pixel location with each of the control functions until the pixel is allowed into a sparse image after which we do not test that pixel with any remaining control functions. We simply move on to the next pixel. Thus, (a) we need not use the participation function as it is implicitly in force and (b) we traverse the image just once - a definite improvement in performance!

6.3 Input, Control and Output Functions

We now discuss the input, control and output functions that we have used in our implementation. Our aim in selecting functions is to use chaotic functions wherever possible so as to utilize their properties (eg. sensitive

dependence on initial conditions, mixing and ergodicity) thereby enhancing the performance of the encryption system.

As the input function, we have used the Euclidean distance function. However, for our implementation, we do not use the z coordinate in the Euclidean function. If we did use the z coordinate (the pixel intensity values), the system will not be reversible and hence we cannot implement the decryption process. We will use the following Euclidean function: $\sqrt{x^2 + y^2}$. For implementation purposes, we note that if our coordinate system starts at (0,0), then the Euclidean function evaluates to 0 and hence cannot be used as an input to the logistic function (which we use as our control function; see the next paragraph for details) because the logistic function is defined on (0,1). Thus, we perturb the Euclidean function whenever it is zero. We point out that this perturbation is a good place to introduce the user's key. Hence we make the perturbation a function of the user's key. Another issue is that the output of the Euclidean function could be larger than 1, while the logistic function is defined only on the open real interval (0,1). Hence, we convert the output of the Euclidean function to a real number using the simplistic approach of dividing it by the root of the sum of the squares of the dimensions of the image ($\sqrt{M^2 + N^2}$, for an $M \times N$ image). The rationale: the largest value that the Euclidean distance can take is $\sqrt{(M-1)^2 + (N-1)^2}, 0 \leq x \leq M-1, 0 \leq y \leq N-1$. Hence, division by that quantity guarantees that the resulting value is always less than one. We will refer to the value output by this modified Euclidean function as $eDist$ in the following discussion.

For the control function, we utilize the one dimensional logistic map. An explanation is in order as to why we chose the logistic function. We note that the logistic function is chaotic. Hence, by the definition provided in section 2.3, we observe that it satisfies the properties of topological transitivity. In effect, this translates to the following: every point in I has equal probability of being mapped to every other point. This can be interpreted as uniform distribution by the logistic function. Thus, the logistic function is a good

choice so as to get evenly sparse images. That is, the pixels of the original image are uniformly/evenly distributed across all the sparse images. To start with, we keep track of the value previously output by the logistic function (say *prev*), using part of the key as the initial seed value. Before each call to the logistic function, we compute the following value: $prev = prev \times eDist$ which is provided as input to the logistic function. But because of this multiplication, however, we must assert that $prev \neq 0$ and $prev \neq 1$ before we use *prev* as input to the logistic function the reason being that on digital computers, the product of positive non-zero reals can become zero (and ofcourse it can become one) because of the finite precision. For example, $3.95253e^{-323} \times 0.0615273 = 0$ on the Intel 32 bit platform. Hence, we must guard against this so as to prevent erratic logistic function outputs.

We emphasize that it is important to introduce the user key in the above mentioned cases. If not, the sequence of outputs becomes more predictable, thereby defeating the efforts of the encryption process. For example, in the case where the Euclidean distance becomes zero if the perturbation is by a constant quantity, the output is predictable once that quantity is known. In practical scenarios, we assume that the attacker has access to the encryption algorithm (or will eventually have access to it). Thus, it is not unreasonable to assume that it is a bad idea to use a constant for perturbation.

As the output function, we devise the following function: We divide the range of the logistic function, namely, $I = (0, 1)$ into a number of equal-length sub-intervals whose union is I (see fig. 6.1). Each interval has a sparse image (or bucket) associated with it. The number of intervals we actually divide I into is a function of the user-key (or it could be a constant, with a corresponding reduction in security). The output of the logistic function is tested against the upper and lower bounds of each of these sub-intervals. A pixel will then belong to a sparse image corresponding to an interval if the value output by the logistic function falls in the range of that interval. Once the sparse images/buckets have been constructed, we generate the required random walk by simply looking at each of the pixels in every bucket in the

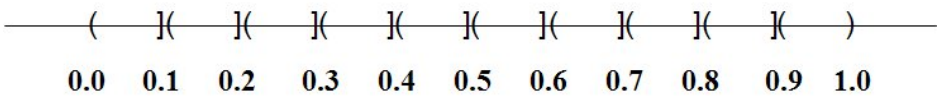


Figure 6.1: The unit interval is split into sub-intervals whose union is the unit interval itself.

order in which they were put into that bucket. Fig. 6.2 shows the action of the Baker map in this regard. We observed during our experiments that larger the number of intervals, better the encryption performance.

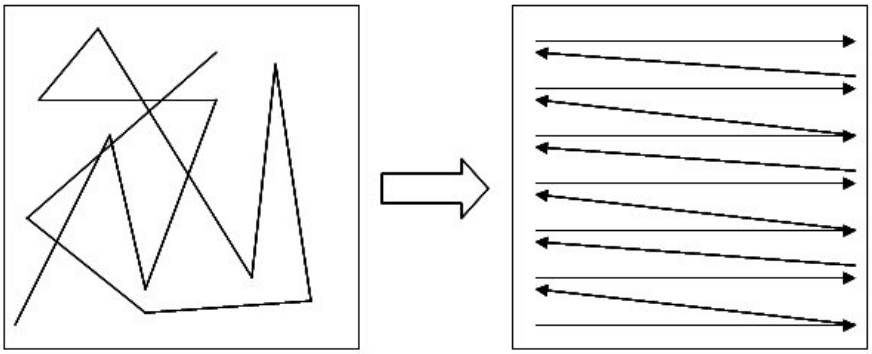
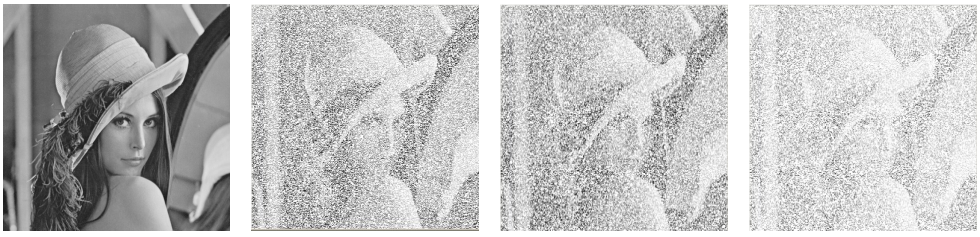


Figure 6.2: Random walk assembled by the Baker Map.

Fig. 6.3 shows the 3 sparsely decomposed images using 3 buckets for decomposition.



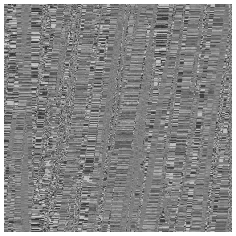
Original Image Sparse image 1 Sparse image 2 Sparse image 3

Figure 6.3: Sparse Decomposition of Image (3 buckets are used in this example).

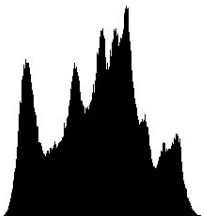
6.4 Results and Conclusions

Fig.6.4 shows the results of the image encryption process. Notice that the encryption process flattens the histogram thus making the statistical and known-ciphertext attack difficult.

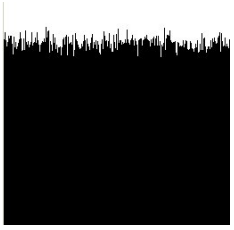
Original Image



Using random-walk based Baker map



Histogram of Original Image



Histogram of encrypted image

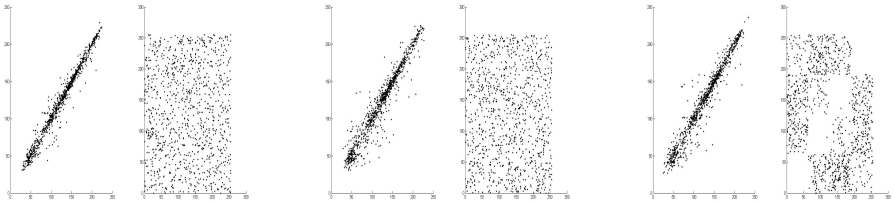
Figure 6.4: Encryption using random-walk based Baker maps with 10 intervals/buckets in the output function (key used: 1234567890123456)

Table 6.1 and fig. 6.5 indicate that the encryption procedure decorrelates pixels that are adjacent horizontally, vertically and diagonally.

	Plain Image	**Ciphered-Image**
Horizontal	≈ 1	0.0054
Vertical		-0.2410
Diagonal		0.0076

Table 6.1: Correlation between adjacent pixels using random walk based Baker map.

Hence the encryption process using the random walk based Baker maps also perform well for image encryption. They have the added advantage that



Horizontal Diagonal Vertical

Figure 6.5: Correlation between adjacent pixels. The plot on the right side is of the encrypted image while that on the left is of the original image.

the path taken while traversing the image is based on the user key and hence the system will be difficult to compromise.

Chapter 7

Discussion and Future Work

...nothing is safe that does not show how it can bear discussion and publicity.
-Lord Acton

7.1 Conclusion

In this report, we have presented the implementation of the work by Mao *et al.* in [5], and verified their claims. We have modified the Baker map to be treated as a function of S, which we call the path function. With this modification, we obtained a way of generalizing the Baker Map to any arbitrary dimension. We used this new extension of the Baker map in image encryption and show that the 3D Baker map is optimal for image encryption. We then moved on to propose a novel way to view any 2D image as a 3D entity. Using this view, we embed the diffusion mechanism into the confusion process using the 3D Baker map and show its usefulness in application of image encryption to color images. Lastly, we present a random-walk based Baker map and use it in image encryption. We point out that the random-walk based baker maps are harder to decrypt because of the key based random walk used in this map.

7.2 Future Work

In spite of all this, we note that a lot of work can still be done in this area. We now summarize some of the directions in which we propose to proceed.

1. Mathematical analysis of the chaoticity of the higher dimensional Baker maps is yet to be explored. For example, we can calculate the Lyapunov exponent of the higher dimensional Baker maps and see if there is any improvement in the chaoticity of the map.

2. We wish to explore the application of fractional chaos in the encryption scheme. For this, we first need to get the fractional counterparts of the Baker map and verify that the properties of chaos are retained.

3. We have incorporated the diffusion mechanism into confusion. However, efforts must be put into exploring the possibility of using chaotic substitutions for the diffusion mechanism. The main criterion for this exploration should be the speed and efficiency with which the diffusion can be achieved using the chaotic maps.

4. In natural sciences, lots of diffusion phenomena are modeled by differential equations. We wish to explore the applicability of these models in the diffusion process of encryption schemes. Further, we can utilize fractional calculus to create fractional diffusion equations and investigate the applicability of these equations for diffusion process for encryption.

5. We need to investigate the relation between this approach and the Fiestel networks[1] of traditional cryptography. We believe that we could learn some security techniques from the traditional Fiestel networks.

6. Our method of encryption in ideally suited for implementation in embedded systems where memory is at a premium and the processor is slow. In such cases, we must resort to calculating the permutation each time. This calls for a trade off between speed of the cipher and strength

of the cipher. However, for rich platform applications, we could use a look-up table based implementation of the Baker Map. A look-up table based implementation on a rich platform would actually speed up the performance because look-up tables calculate the permutation just once and the same permutation can be used over and again for different rounds. But this limits the strength of the cipher in the following way: a fixed permutation implies that we cannot propagate or chain the cipher through cipher rounds. This could reduce the security of the cipher. In a non look-up table based implementation, we can use the chaining process to propagate the chaining across rounds of the cipher too. Hence there is a trade off between security and performance of the cipher in the two approaches.

7. We need to verify quantitatively, the randomness or chaoticity of the paths generated by the sparse decomposition of images method.

8. We could explore alternative ways to generate random/chaotic paths for Baker maps. But we have presented the germ of the idea which could be explored further.

9. We have introduced the idea of embedding diffusion mechanism into confusion and demonstrated it for the 3D maps. We could however, use it for any dimension. For example, if we wish to use a 5D map, we could first extend the image to four dimensions using the approach presented by Mao *et al.*[5]. For extending to the fifth dimension, we could use the binary representation of the pixel intensities as described in section 5.5. Working on this 5D representation of the image is equivalent to embedding diffusion into confusion using the 5D Baker map! (see section 5.4 for details).

10. We need to integrate all the work done in this thesis and investigate the performance of the hybrid encryption scheme.

A chinese proverb goes *"I hear and I forget, I see and I remember, I do and I understand."* It is only in implementing the algorithms that we get these insights. And as we stated in the beginning of this chapter, and like any good encryption algorithm, we are open to discussion and scrutiny - for in public scrutiny lies learning.

"The End of Education is Character"

-Sri Sathya Sai

Bibliography

[1] Charlie Kaufmann, Radia Perlman, and Mike Speciner. *Network Security: Private Communication in a Public World*. Prentice Hall, second edition, 2003.

[2] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, second edition, 1996.

[3] Non-repudiation. Wikipedia, accessed, 30th Jan 2008. `http://en.wikipedia.org/wiki/Non-repudiation`.

[4] Cryptography. Wikipedia, accessed, 30th Jan 2008. `http://en.wikipedia.org/wiki/Crypography`.

[5] Yaobin Mao, Shiguo Lian, and Guarong Chen. A novel fast image encryption scheme based on 3d chaotic baker maps. *International Journal of Bifurcation and Chaos*, 14(10):3616–3624, 2004.

[6] Bayro Corrochano, editor. *Chaos Based Image Encryption, in Handbook of Geometric Computing*. Springer-Verlag, NY, 2004.

[7] Ljupčo Kocarev, Goce Jakimoski, Toni Stojanovski, and Ulrich Parlitz. From chaotic maps to encryption schemes. *IEEE International Symposium on Circuits and Systems*, 4(3), 1998.

[8] Gonzalo Alvarez and Shujun Li. Some basic cryptographic requirements for chaos-based cryptosystems. *International Journal of Bifurcation and Chaos*, 16(8):2129–2151, 2006.

Notes: This paper provides excellent recommendations for chaos based image encryption. It draws on the last couple of decades of research in this field.

[9] Steven H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Perseus Books Group, first edition, 2001.

[10] Claude E. Shannon. Communications theory of secrecy systems. Appeared in a confidential report of Bell Labs, 1946. Available in electronic form from Jiejun Kong.

[11] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. Electronic version available at: `www.cacr.math.uwaterloo.ca/hac`.

[12] F. Pichler and J. Scharinger. Efficient image encryption based on chaotic maps. *Johannes Kepler University, Linz, Austria.*, 1996.

[13] F. Picher and J. Scharinger. Ciphering by bernoulli shifts in finite abelian groups. In *Contributions to General Algebra. Proc. of the Linz-Conference*, 1994.

[14] Jiri Fridrich. Secure image ciphering based on chaos. In *Final Report for AFRL*, Rome, New York, USA, 1997.

[15] Jiri Fridrich. Image encryption based on chaotic maps. *System, Man and Cybernetics*, 1997.

[16] Jiri Fridrich. Symmetric ciphers based on two-dimensional chaotic maps. *International Journal of Bifurcation and Chaos*, 8(6):1259–1284, 1998.

[17] Yaobin Mao et al. A symmetric image encryption scheme based on 3d chaotic cat maps. *Chaos, Solitons & Fractals*, 21:749–761, 2004.

[18] M. S. Baptista. Cryptogaphy with chaos. *Physics Letters A*, 240:50–54, 1998.

[19] Ljupčo Kocarev. Chaos-based cryptography: A brief overview. *IEEE Circuits and Systems Magazine*, 1:6–21, 2001.

[20] Goce Jakimoski and Ljupčo Kocarev. Chaos and cryptography: Block encryption ciphers based on chaotic maps. *IEEE Transactions on Ciruits and Systems - I*, 48(2):163–169, 2001.

[21] J. M. Amigo, Ljupčo Kocarev, and J. Szczepanski. Theory and practice of chaotic cryptography. *Physics Letters A*, 366:211–216, 2007.

[22] Naoki Masuda and Kazuyuki Aihara. Cryptosystems with discretized chaos. *IEEE Transactions on Circuits and Sytems - I*, 49(1):28–40, 2002.

[23] Kwok-Wo Wang, Bernie Sin-Hung Kwok, and Wing-Shing Law. A fast image encryption scheme based on chaotic standard map. arXiv:cs/0609158v1 [cs.CR], 2006.

[24] Shujun Li, Guanrong Chen, and Xuan Zheng. *Chaos-Based Encryption for Digital Images and Videos*, pages 133–167. CRC Press, 2004.
Notes: This is yet another comprehensive survey on image encryption systems. Starting with a motivation on the need and requirements of image encryption systems, it goes on to provide comparative results from various algorithms. It concludes with lessons learnt from the exercise.

[25] Shujun Li. *Analyses and New Designs of Digital Chaotic Ciphers*. PhD thesis, Xi'an Jiaotong University, 2003.
Notes: This is a great piece of work! Contains a comprehensive survey of existing work. Nice classification of existing ciphers is provided.

[26] Josef Scharinger. Analysis and application of chaotic permutation systems. *Cybernetics and Systems*, 1:57–62, 2002.

[27] OpenCV. `www.intel.com/technology/computing/opencv/`.

[28] Josef Scharinger. Fast encryption of image data using chaotic kolmogorov flows. *Journal of Electronic Imaging*, 7(2):318–325, 1998.

[29] Shiguo Lian, Jinsheg Sun, and Zhiquan Wang. Security analysis of a chaos-based image encryption algorithm. *Phisica A*, 2005.

[30] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. Prentice Hall of India, second edition, 2001.
Notes: By far the best book on Algorithms. Very comprehensive and deep.

[31] Richard Hoffman and Anil K. Jain. Sparse decomposition for exploratory pattern analysis. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, PAMI-9(4):551–560, 1987.

[32] V. Chandrasekaran and Shi-Qiang LIU. Robust face image retrieval by fuzzy gated neruronal architechture. Technical Report 96/15, Department of Computer Science, University of Melbourne, Australia 3052.

[33] Ljupčo Kocarev and G. Jakimovski. Chaos and cryptography: From chaotic maps to encryption schemes. *IEEE Transactions on Circuits and System - I*, 48:163–169, 2001.

[34] Stanley Lippman and Josée Lajoie. *C++ Primer*. Pearson Education, third edition, 2003.

[35] HyperBook.net. `http://www.hyperbook.net`.

[36] MathWorld. `http://mathworld.wolfram.com`.

[37] ChaosBook.org. `http://ChaosBook.org`.

[38] BrainyQuote. `www.brainyquote.com`.

[39] `www.sfu.ca/~vkyrylov/JavaApplets/Cryptography/`.