Post-Quantum Cloud Security and Data Exchange using Artificial Intelligence

Napo Mosola

A Thesis Submitted to the University of KwaZulu-Natal,
School of Mathematics, Statistics and Computer Science,
College of Agriculture, Engineering and Science.

In Partial Fulfilment of the Requirement for the Degree of
Doctor of Philosophy in Computer Science.

Supervisors: Prof. Jonathan Michael Blackledge and Dr. Jean Vincent Fonou Dombeu

27 February 2023

Abstract

This thesis investigates the application of plausible modern-day cryptographic solutions for securing the cloud and exchanging confidential data. The context followed is such that the strength of an encryption algorithm is based on the difficulty to cryptanalyse it. This means the more difficult the crypto-system is to cryptanalyse, the stronger and more trusted it is. The success of cryptanalysis on a cryptographic algorithm has been a function of the computational power available at the time of performing the cryptanalysis, without consideration of future innovations, specifically, without careful consideration of Moore's law. A significant number of public-key crypto-systems can and will be compromised by a quantum computer coupled with the implementation of Shor's algorithm. This has brought a lot of focus regarding research on cryptographic solutions post quantum computing (PQC) due to the following:

- cryptographic algorithms are based on the intractability of prime number factorisation using a conventional computing power;
- a quantum computer can factorize prime numbers with relative ease.

In the past, the quantum computing paradigm was a hypothetical concept. Thus, cryptanalysis using quantum resources was a theoretical idea. This is no longer the case with the loom of quantum computers eminent. Consequently, prime number based encryption is becoming increasingly irrelevant. Low Qubit quantum computers now exist. Research and development in this area is growing. Hence the existence of the post-quantum cryptography paradigm. This paradigm is based on encryption algorithms developed and considered secure enough to withstand quantum attacks. Thus, the National Institute of Standards and Technology made a call for projects clustered under the Open Quantum Safe project (OQSP), which began in 2016. The ultimate goal of this project is development of future quantum resistant cryptographic algorithms for secure communication and data exchange. The OQSP aims to gather open source libraries which can be standalone or integrated into the public key encryption schemes to improve their security against

quantum attacks in the quest to achieve quantum resistance. The major focus is placed on quantum key exchange (QKE). It is against this background that the material presented in this thesis reports on a spectrum of algorithms that are thought to be quantum resistant based on a coherence of ideas, methods, models and software implementation, trying to meet the NIST requirements and contributing to new knowledge in the field of cryptography. The aim is to provide confidentiality guarantees on cloud-hosted data as well as secure data exchange between communicating entities, while also tackling the cumbersome key exchange and management problem. The results show that the algorithms presented in this thesis introduce new ideas in cryptography and can be tested to withstand cryptanalytic quantum attacks, while a plausible encryption key distribution and management solution is proposed.

In this context, the material presented in this thesis report on a spectrum of algorithms that are proposed to be quantum resistant based on a coherence of ideas, methods and software implementation, aimed at providing security of cloud-hosted data as well as data exchange between communicating entities. The cloud has a flexible, scalable and low cost properties. This is due to two concepts which are fundamental to cloud computing:

- virtualization;
- multi-occupancy.

These above concepts have brought infinitely many benefits which make the cloud an attractive paradigm. Among the benefits are reduced capital and maintenance costs, high processing power, enormous storage facilities etc. However, security concerns affecting confidentiality of cloud-hosted data still plague bring concerns when it comes to cloud adoption. Data confidentiality can be achieved through encryption, which is in turn implemented by cryptographic algorithms. Hence, this thesis proposes and puts into practice cryptographic algorithms to solve issues of confidentiality, specifically in the cloud.

Academic Integrity Declaration

I Napo Nathnnael Mosola, 216075642 declare that:

(i) This is my own original work;

(ii) All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged;

(iii) This thesis has not been submitted in part or whole, in support of an application for a degree or qualification to any university or any other institution of learning;

(iv) I have not allowed anyone to copy my work with the intention to pass it as his/her own.

Napo Mosola, 23 February 2023

As the candidate's supervisor, I have reviewed and approved this PhD thesis for submission.

Supervisor: Professor J.M Blackledge, 23 February 2023

Acknowledgements

I would like to thank everyone who made contributions to this study. First and foremost, my Supervisors, Professor Jonathan Blackledge and Dr Jean Fonou-Dombeu, for their assistance, guidance and advice. It was a pleasure working with you in this fascinating, yet complex field of Cryptography. Your stewardship made it very easy for me to understand and comprehend. The tough times that came with COVID-19 making my academic life difficult were very well handled by your presence whenever I called you for [virtual] meetings. Secondly, my wife, 'Mamotebang Joyce Mosola for her unwavering support and always pushing me to the limits. You believed in me and always inspired me to get the 'work' done! Spending time away from you and our kids to dedicate it all to this thesis was not easy. I love you, always. To my kids, Nathan and Natalie, a big thank you for not feeling the void I left in your lives just to make this a reality.

Dedication

I would like to dedicate this work to my late parents. My father, Daniel Mosola and my mother-in-law, Eusebia Mohapi. I know the kind of jubilation you would be having just to know that I have compiled this PhD thesis. May your souls continue to rest in peace and rise in glory.

Publications

The following publications are a selection of works that have been generated from the research undertaken by the author of this thesis.

Journal Publications

1. J M Blackledge and **N N Mosola**, *Applications of Artificial Intelligence to Cryptography*, Transactions on Machine Learning and Artificial Intelligence, Society for Science and Education, Vol. 8, No. 3, 21-60, 2020.
   https://journals.scholarpublishing.org/index.php/TMLAI/article/view/8219

2. J M Blackledge **N N Mosola**, J Al-Khatib and A Magnuski, *Cryptography using Machine Learning for Cloud Computing*, MDPI Special Issue on Cryptographic Applications in Cloud Computing and Internet of Things, 2021 (Submitted).

International Conference Publications

1. **N N Mosola**, Kopano M, Boitumelo L, Phenduka H, *Secure Rfid Digital Payment System (Digiloti),Southern Africa Telecommunication Networks and Applications Conference (SATNAC, 2021), South Africa.*

2. *J M Blackledge and* **N N Mosola**, *A Statistically Significant Test to Evaluate the Order or Disorder for a Binary String of a Finite Length, ISSC2020, IEEE UK and Ireland Signal Processing Chapter and IEEE Computational Intelligence Society (UK & Ireland), Letterkenny Institute of Technology, 11-12, June 2020.*
   https://arrow.tudublin.ie/engscheleart/311/

3. *J M Blackledge and* **N N Mosola**, *Digital Image Exchange using a No-key(s) Protocol with Phase-only Encryption, ISSC2020, IEEE UK and Ireland Signal Processing Chapter and IEEE Computational Intelligence Society (UK & Ireland), Letterkenny*

Institute of Technology, 11-12, June 2020.
https://arrow.tudublin.ie/engscheleart/312/

List of Abbreviations

AES	Advanced Encryption Standard
AI	Artificial Intelligence
ANN	Artificial Neural Network
ASCII	American Standard Code for Information Interchange
BBS	Blum Blum Shub
BER	Bit Error Rate
BIE	Binary Information Entropy
BiEn	BiEntropy
CD	Chirp Demodulation
CM	Chirp Modulation
CNN	Convolutional Neural Network
CAVP	Cryptographic Algorithms Validation Program
CPU	Central Processing Unit
DES	Data Encryption Standard
DFT	Discrete Fourier Transform
EC	Evolutionary Computing
ESF	Exponential Scaling Factor
FFT	Fast Fourier Transform
GAN	Generative Adversarial Network

GKFE	Generalised Kolmogorov-Feller Equation
GPU	Graphical Processing Unit
GUI	Graphical User Interface
IaaS	Infrastructure as a Service
IEC	Information Embedding Coefficient
IC	Integrated Circuit
IoT	Internet of Things
IF	Information Function
IFS	Iteration Function System
IIF	Infinite Impulse Response
IRF	Impulse Response Function
JB	Jargue-Bera
KDP	Key Distribution Problem
MATLAB	Matrix Laboratory
ML	Machine Learning
NKP	No Keys Protocol
NIST	National Institute of Standards
OTC	One-to-Cloud
OTP	One Time Pad
OQSP	Open Quantum Safe Project

PDE	Partial Differential Equation
PDF	Probability Density Function
POD	Phase-only Decryption
POE	Phase-only Encryption
PoS	Point of Sale
PRNG	Pseudo Random Number Generator
PSDF	Power Spectral Density Function
PEP	Retro-reflector Embedded Polymer
RIP	Regulation of Investigatory Powers
RES	Relative Entropy Signal
RGB	Red, Green, Blue
RMSE	Root Mean Square Error
ROM	Read Only Memory
RFID	Radio Frequency Identification
RSA	Rivest-Shamir-Adleman
SNR	Signal-to-Noise Ratio
TUD	Technological University Dublin
TPP	Three Pass Protocol
USB	Universal Serial Bus
UV	Ultra-Violet

WPOD Weighted Phase-only Decryption

WPOE Weighted Phase-only Encryption

XOR Exclusive OR

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The material discussed throughout this work considers the application of a 'signals and systems' approach to cryptography. This approach is used to develop data encryption algorithms which are not based on the use of prime numbers and modulo operations. Thus, the proposed algorithms have the context of post-quantum resistance. The central theme which provides an underlying coherence to this work is based on a signal and systems approach, based on the fundamental building block for a signal s given by [27] as shown in Equation (1.1):

$$s = p \otimes f + n \tag{1.1}$$

In Equation (1.1), p is the Impulse Response Function (IRF), f is the 'Information Function' (IF), and n is the 'Random Function'. The operator, $\otimes$ denotes the convolution operation (the convolution integral). This is the 'Standard Model' for a signal associated with a linear stationary system and is the underlying theme upon which the material is presented and developed.

1.1 Motivation

To bring the 'signals and systems' approach into the context of cryptology, the signal is the transmitted ciphertext, the IRF is the plaintext and the noise function is the cipher. Given Equation (1.1), none of the encryption schemes explored in the thesis depend explicitly

upon the use of prime numbers but are rather based on floating point arithmetic and processing associated with Equation (1.1), subject to a specified numerical precision. Hence, the use of Shor's algorithm, for example, coupled with a quantum computer to break a ciphertext through prime number factorization may be considered to be null and void. In this context, the cryptographic algorithms derived from this thesis may be considered to be post-quantum algorithms and potential contributors to the Open Quantum Safe project, to provide quantum resistant solutions.

The fundamental, underlying encryption model considered is based on Equation (1.1) with $p = n$ where n is the cipher. For decryption to succeed, deconvolution is then required. This is to decrypt the signal and recover the plaintext. In the context of Equation (1.1), the thesis is exclusively based on the following components:

(i) Pseudo Random Number Generators (PRNGs) with high cryptographic strength using:

- Non-linear Iteration Function Systems (IFSs);
- Evolutionary Computing to generate IFSs.

(ii) Key and data-exchange using phase-only encryption.

(iii) Order and Disorder of Binary Information Streams.

(iv) Cloud Hosted Data Confidentiality and Key-management.

In terms of component (i), most PRNGs used in conventional cryptographic systems exploit modular integer arithmetic coupled with prime numbers and/or prime number factorisation. This thesis presents the use of chaotic systems. These systems are used to produce floating point random variables. The use of chaotic systems enables the application of Evolutionary Computing concepts to develop personalised encryption algorithms [110]. This is to provide data confidentiality guarantees as data is encrypted on the client-side. In particular, a method of generating floating point PRNGs is considered using a software named *Eureqa* cloud compute [127]. Eureqa was designed and produced by the

Cornell University's Creative Machines and Artificial Intelligence (AI) laboratory. This software makes use of a genetically evolving search algorithms which will output mathematical equations that describe sets of data in their simplest form. Using a 'seeded' function, the software is initialized with natural noise sources. For example, atmospheric radio emission, radioactive decay, electronic noise etc., and an output is obtained. The output mathematical equations are non-linear functions. These functions approximate the input noise. Various tests are conducted per output function. The tests are conducted to check for potential cryptographic strength in terms of the following attributes:

(i) a positive Lyapunov exponent,

(ii) maximum entropy,

(iii) high cycle length and key diffusion characteristics.

The above approach constitutes a method of generating unlimited number of personalised and unique PRNGs. These PRNGs can be used as 1-to-1 or onto function mappings. For example, one of the applications is client-side data encryption, where data is encrypted before it is uploaded to a cloud storage by the data owner i.e. cloud user. A cloud user is essentially owning a personalised encryption algorithm rather than just a private encryption key using a 'known algorithm' that may be vulnerable to a 'known algorithm attack'. This therefore facilitates a 'one key, one session, one cipher' mapping.

The thesis then considers component (ii) in which the random noise term, n is conditioned to have a 'phase-only' property over a noise spectrum. The conditions provide a unique solution to the deconvolution problem. This is done to facilitate a decryption algorithm [98]. The deconvolution problem is associated with an IRF and the noise function, n. The latter is a source of noise, as described in component (i) above. Both of these functions have the same phase-only stochastic spectrum characteristic. Moreover, it is proven that, the deconvolution problem now has a general solution. The solution is both exact and unique as it becomes well-posed when subjected to a re-normalisation condition. This condition is related to the numerical scale, in the range [0,1], an array of floats associated with the solution.

While in other fields such as digital signal processing and communications engineering, the model considered (i.e. phase-only spectrum) in this thesis is used in a limited manner, for example, with regards to information retrieval from a noise source, its relevance and application to cryptography has a significant potential. One of its biggest advantage is that it is able to generate an encryption algorithm with the diffused plaintext completely hidden in a phase-only cipher. This is subject to the precision of the floating point arithmetic used for data processing. Consequently, the statistical property of the plaintext is fully dissipated in the cipher. This is mainly to ensure diffusion and confusion concepts. Furthermore, a computationally efficient decryption function can be generated. This is also subject to the condition that both the sender and receiver have access to an identical encryption/decryption algorithm and decryption key, such that deconvolution becomes equivalent to the decryption process. This approach provides a unique key-exchange method to be implemented using a three-pass protocol as presented in [109] for a quantum key exchange.

Component (iii) addresses a finite binary string i.e. [0,1] problem. This problem is associated with how to establish whether the binary string is an element of a non-random, well-formed or intelligible information which involves some form of repetition (periodicity). This is further analyzed to find out whether the string is a product of a truly random or a pseudo-random process. This problem is mostly applied in areas such as, although not exhaustive: artificial intelligence, cryptanalysis, image and signal processing etc., including how to extract real noise from information completely embedded in a noise source [108].

The final component of the thesis, i.e., component (iv) considers an application of the prior material presented. A novel client-side crypto-system is derived using evolutionary computing concepts. The proposed solution uses real-world random number streams to generate a fitness function. The fitness function is then used to generate encryption keys for data encryption. The stochastic properties of the input noise provides the strength of the encryption keys. Encryption keys require a cryptographically strong key management solution which is a prevalent problem in cryptography. This thesis considers Artificial

Intelligence techniques by employing a Neural Network (NN) to solve the cumbersome key management problem.

1.2 Problem Statement

A new paradigm shift known as quantum computing is inevitably upon us. According to Moore's law [113], computing power doubles every 18 months. With new and emerging disruptive technologies being adopted, there's no doubt we are headed for a quantum computing era. A new cryptographic era that has a great potential of breaking the secure communication mechanisms, implemented by classical cryptographic solutions for both symmetric (private key) and asymmetric (public key) encryption that underpins the world's digital society [206]. Today's security mechanisms are largely based on public key cryptography. This branch of cryptography is founded on the use of mathematical functions that are considered hard to break, using current computing power. This is because public key cryptography uses large prime numbers that are difficult to factorize using a conventional computer. It is the principle of factorization of these large prime numbers that makes the current computing power to struggle to break these classical crypto-systems. The current cryptographic solutions, with suitable encryption key lengths, are not susceptible to attacks as cryptanalysis is believed to take centuries to succeed. However, there's a time when quantum computers will break the entire ecosystem that underpins the field of cryptography. Quantum computers can dramatically reduce the amount of time it takes to break cryptographic algorithms that have been used for so many years. Hence, data confidentiality, integrity, availability, authentication etc., will all be compromised. There's a lot to be worried about when it comes to the potential threat of quantum attacks, and confidentiality is one of the key concerns. If a malicious threat actor could leverage on quantum computing, they may derive an entity's encryption key(s) and use them to impersonate that entity, resulting into identity theft. Hence, the call for brand new public-key Post-Quantum Cryptographic (PQC) algorithms, by the NIST [157]. It is envisaged that by the year 2030, scalable quantum computers capable

of breaking 2000-bit RSA scheme will be available. This proves to be a long-term threat to the current crypto-systems which are currently standardized by NIST [56].It is within this context that this thesis makes plausible propositions in the field of cryptography, to contribute to the Open Quantum-Safe Project, aiming to solve this problem. NIST is currently soliciting and evaluating well researched cryptographic systems needed today for the protection of the future Internet.

1.3 Objectives

The aim of this thesis is to address the envisaged security concerns brought by the invention and adoption of quantum computers. Specifically, the objectives of this research are as follows:

- Develop and implement cryptographic systems which are secure against quantum and conventional computers.
- Design and implement crytographic systems using chaos theory.
- Design, model and implement encryption algorithms based on Machine Learning.
- Design and implement a secure data exchange cryptosystem using a no-keys protocol.
- Develop and test an encryption key management solution for post-quantum cryptography (PQC).

1.4 Research questions

Based on the premise stated in Section (1.2), coupled with Section (1.3), this thesis seeks to answer the following research questions.

1.4.1 Main research question

i How can novel post-quantum cryptographic systems be developed to provide data confidentiality guarantees? To answer this main research question, the following subsidiary questions were formulated to address the problem statement and meet the set objectives.

- Can secure post-quantum cryptographic systems be developed?
- Can chaos theory be used to implement encryption systems?
- How can encryption systems using Machine Learning be designed and developed?
- How can a secure data exchange crypto-system using a no-keys protocol be designed and developed?
- How might we practically develop and implement an encryption key management solution for PQC i.e. Post-Quantum Key Distribution?

1.5 Advancement of Knowledge

Advancement of knowledge is compounded in the following items, associated with the principal components (i)-(vi) presented in Section 1.1:

- Mathematical models applied to all aspects of the concepts introduced and solutions obtained through, where appropriate, a set of theorems, proof, corollaries and remarks, in order to provide a clear and unambiguous foundation to the encryption algorithms considered.
- Conversion of a derived algorithm into a prototype MATLAB function using available 'numerical recipes' including selected functions intrinsic to MATLAB.
- Analysis of numerical performance and demonstration of the results obtained.
- Development and deployment of example source code.

1.6 Original Contributions

In each of the components (i)-(vi) presented in Section 1.1, associated cryptanalysis is considered. This is done to examine the potential weaknesses of the algorithms developed in terms of generating a successful attack. In all cases, it is shown that while a successful attack can never be ruled out, the algorithms designed have underlying properties that make them secure enough to be considered in the quantum computing era. Hence, the contributions of this thesis are:

- Computationally secure encryption algorithms, i.e. the algorithm is secure even against the maximum computational resources of an attacker.
- Unconditional security, i.e. the algorithm is secure given any computational power the attacker may have access to.
- Applciations of machine learning concepts suitable for random number generation
- Investigation of various random number generators for practical implementation in Cryptography.
- Evaluation of cryptographically secure PRNGs.

It is shown that from a concatenation of components (i)-(iv) into a single protocol, coupled with the principle of 'one-time plaintext, one-time key, one-time encryptor and one-time ciphertext', which basically translates to a one-time-pad (OTP), can provide a crypto-system that is 'quantum resistant'.

1.7 Methodology

This section highlights the methodologies used to practically implement the proposed encryption models for post-quantum cryptography. To eliminate chances of cryptanalysis success, the proposed models are generally not based on prime number factorisation as

is the case with the current public key cryptographic schemes such as RSA. This is because with the use of a quantum computer and implementation of Shor's algorithm, the modern-day crypto-systems are vulnerable to security breaches. Thus, this thesis uses Artificial Intelligence methods, such as Evolutionary Computing (EC), to build strong, random and chaotic ciphers. The use of Pseudo-Random Number Generators (PRNGs) implemented using multi-layered Artificial Neural Networks (ANNs) brings 'perfect' security when combined with the use of true random sources such as chaotic random noise. The proposed model solution makes use of a counter propagation neural network (CPNN) trained using both supervised and unsupervised machine learning concepts on MATLAB to learn patterns of an output ciphertext and the encryption key used as input to the encryption models i.e. the cipher., along with the original plaintext. This is a proposal of a post-quantum key distribution (PQKD) scheme. Once the key is learned, it can then be discarded to thwart any key attacks. This addresses the cumbersome key management problem mentioned Section (1.2) and also answers some of the questions under Section (1.4). The following steps highlight the methodology, which is further unpacked in detail in chapter 9 of this thesis as it entails the implementation of the proposed crypto-system.

- Chaotic random noise is generated from a true noise source such as Random.org [170].
- The generated random noise in then used as input into a cloud-based system called Eureqa. This system produces a series of random functions, which are an approximation of the random noise, as discussed in chapter 4.
- The resulting functions are normalised with random numbers in the range [0,1]
- The normalised functions produce a fitness function as discussed in chapters 8 and 9
- The fitness function is used to generate a random key which is used for data encryption on the client-side

- The random key patterns are recognized by the Counter Propagation Neural Network (CPNN), which is executed for 150 epochs (i.e. iterations), to produce the best validation performance while also catering for errors using a Mean Square Error (MSE) function. NB: For large data-sets (with a bit-stream of more than 2048 bits), the neural network becomes very slow and keeps the computing resources busy.

- The above steps provide guarantees for computational and unconditional security

- The proposed encryption model written on MATLAB language and presented in Appendix A and B use the derived encryption key and the plaintext binary data i.e. [0,1] as inputs.

- The encryption model (based on number theoretical functions) transforms the data using Fourier transform method, for example, as discussed in chapter 7 Section (7.3).

- The transformed data uploaded to the cloud, as shown in a case studies discussed in chapter 5 Section (5.4).

The methodology highlighted is expanded and followed for implementation throughout this thesis, specifically in chapter 9.

1.8 Thesis Structure

Each of the components (i)-(iv) presented in Section 1.1 broadly constitute a chapter in the thesis. Each chapter relies on some of the materials that have been published by the author (and co-authors) in peer reviewed journals and conferences as listed in the preliminaries section 'Publication'. Figure 1.1 provides a schematic quantification of the thesis structure in terms of the principal components associated with the crypto-system developed, the contributions the thesis has made in advancement of new knowledge in the cybersecurity arena and the chapters in which they are introduced.

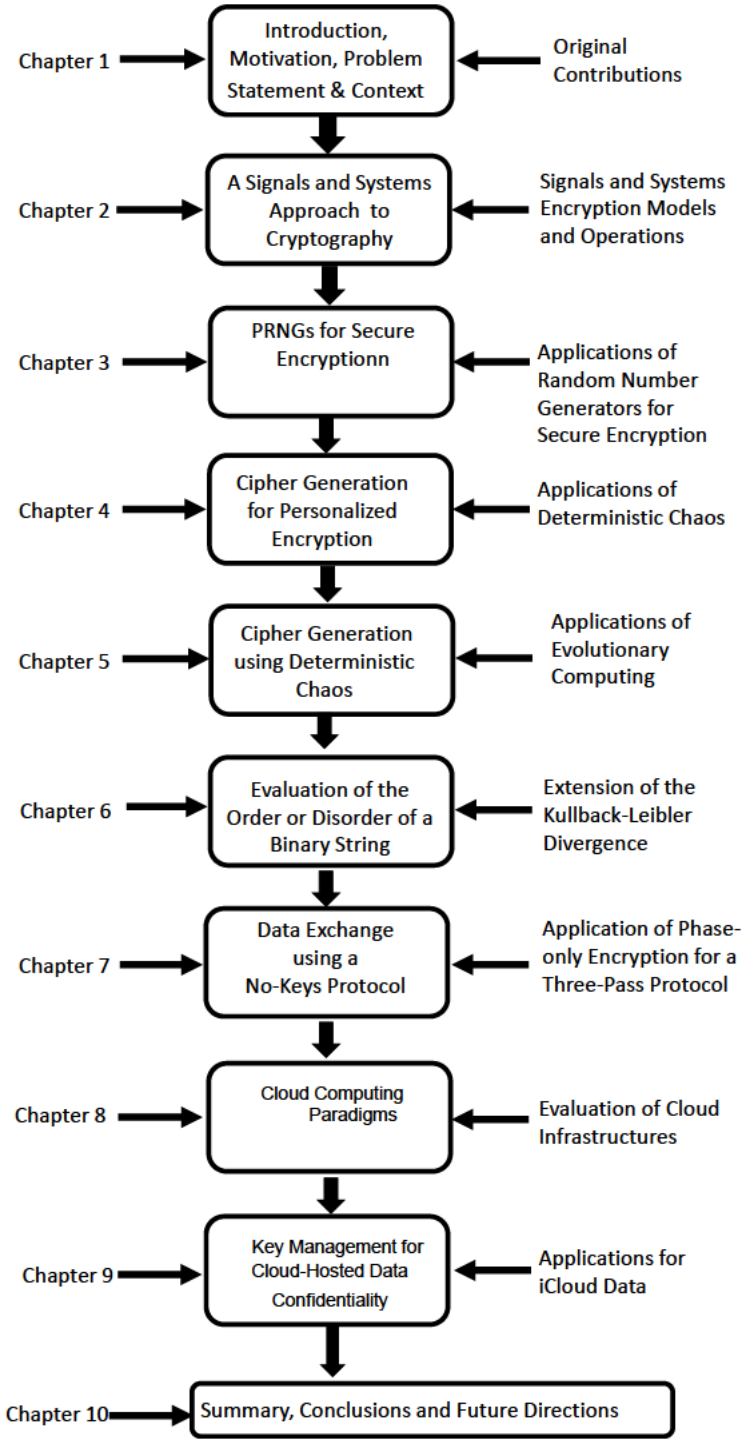

Figure 1.1: Thesis outline.

Chapter 2

Post-Quantum Cryptography: A Signals and Systems Approach

This chapter discusses the fundamental signals and systems model for data encryption in the era of quantum computers. The model presented herein forms a basis of the approach taken in this thesis. The model introduces the fundamental mathematical transformations and operators used throughout this thesis. The model derives a departure from the conventional encryption algorithms used today which heavily relies on the use of prime numbers and exponentiation. These conventional encryption systems are based on the use of prime numbers and modular integer arithmetic, whose security by and large relies on the difficulty in factorizing a product of large prime numbers. However, the eminent emergence of quantum computers renders this security element as a soon to be a legacy of the past.

2.1 Post-quantum Cryptography

The difficulty in breaking a cipher usually reflects its cryptographic strength. This aspect has traditionally been an evaluation method used to measure the strength of a cryptosystem. This measurement has always focused on the computing power that is available at the time of designing the algorithm, without careful consideration of Moore's law, which

states that computing power doubles every 18 months [113]. Public-key cryptosystems can and will be effectively compromised by quantum attacks launched through a quantum computer. As mentioned earlier, quantum computing was a hypothetical paradigm in the past but until very recently when a lot of focus was directed towards its feasibility, it has become a reality. Cryptanalysis carried out using a quantum computer against current encryption algorithms was just a theoretical idea. Quantum computers have now sprung into existence in many research laboratories, globally. A number of companies and governments have started research programs looking into a wide spectrum of quantum computing and their effects should they be available on a large scale. Consequently, research in post-quantum computing (PQC) has already begun. This research area considers secure crypto-systems which could fend off quantum attacks [223] [25] [47]. The objective is clear, it is to develop quantum-resistant cryptosystems. Thus, new developments in the crypto space have emerged [167]. When these developments began, the focus was quantum key distribution (QKD) algorithms. This is due to the threat posed by quantum computers on the current key exchange algorithms, e.g. RSA, due to their heavy reliance on modular arithmetic characterized by repeating periods and the use of large prime numbers.

Quantum computers will soon make public key encryption mechanisms redundant [157]. Although their penetration into the market is not yet clear now, they remain a threat to the future of secure communication and data security. A major concern is the potential compromise of the RSA algorithm. The RSA algorithm may be compromised through the application of Shor's algorithm [192]. Shor's algorithm was developed for efficient prime number factorisation. The algorithm's execution time is exponential in nature as the selected primes, p and q increase in size. This is due to the Shor's algorithm period finding step. However, armed with a quantum computer, a cryptanalyst can exploit the properties of a Quantum Fourier transform (QFT). This way, the algorithm can be used in cryptanalysis of the RSA algorithm. This problem can easily be equated to the quantum phase estimation problem [155][156].

Furthermore, the emergence of the quantum computing paradigm poses a potential

threat to eliminating many if not all public key encryption methods. Additionally, new methods for data encryption are being researched and will be in the market soon, especially for public key crypto-systems across all spheres of cryptography. Another important area of interest includes quantum, multiple level secret sharing schemes. These schemes are useful in encrypting quantum states carrying probability distribution of each plaintext input states. These states must have access structures for decryption to succeed [148]. A good example followed in this context is the phase-only encryption (PoE) model which deviates from the approach used by public-key crypto-systems in using prime numbers for cryptography. PoE explores floating point arithmetic to implement encryption systems. Its applications are varied but the most widely used and researched in this thesis is secret key exchange as discussed in Chapter 7. Thus, the discussions relayed in this thesis is considered as contribution to new knowledge in the field of cryptography as it proposes plausible solutions in post-quantum cryptography. The next section highlights the encryption models proposed for post-quantum cryptography.

2.2 Encryption and Decryption Models

Based on the discussions from Section (2.1), this section proposes a number of encryption models, based on floating point arithmetic. The encryption models presented are applicable in multi-dimensionsal space. Hence, development of a multi-dimensional model is considered for a square-integrable (n dimensional) function represented by a Fourier and inverse Fourier transforms for frequency analysis of a signal:

On a Lagrangian space L^2, a function f can be presented as $f(\mathbf{r}) \in L^2(\mathbb{R}^n) : \mathbb{C} \to \mathbb{C}$, 'non-unitary' Fourier transform:

$$F(\mathbf{k}) = \mathcal{F}_n[f(\mathbf{r})] \tag{2.1}$$

using an integral function on an infinite real number line, we get the following equivalence:

$$F(\mathbf{k}) \equiv \int_{-\infty}^{\infty} f(\mathbf{r}) \exp(-i\mathbf{k} \cdot \mathbf{r}) d^n\mathbf{r} \tag{2.2}$$

and the following 'non-unitary' inverse Fourier transform:

$$f(\mathbf{r}) = \mathcal{F}_n^{-1}(F(\mathbf{k})) \tag{2.3}$$

with an equivalence representation as:

$$f(\mathbf{r}) \equiv [(2\pi)^n]^{-1} \int_{-\infty}^{\infty} F(\mathbf{k}) \exp(i\mathbf{k} \cdot \mathbf{r}) d^n\mathbf{k} \tag{2.4}$$

The variable $\mathbf{k}$ is the spatial frequency vector. The dot product of the two vectors $\mathbf{k}$ and $\mathbf{r}$ is computed as in Equation (2.5)

$$\mathbf{k} \cdot \mathbf{r} = k_1 r_1 + k_1 r_1 + ... + k_i r_i \tag{2.5}$$

These two integral transforms define an (n-dimensional) Fourier transform pair which can be defined using the following notation:

$$F(\mathbf{k}) \leftrightarrow f(\mathbf{r})$$

where only the absolute values of $\mathbf{k}$ and $\mathbf{r}$ are used such that:

$r \equiv \mid \mathbf{r} \mid$ and $k \equiv \mid \mathbf{k} \mid$

With respect to the above, the multi-dimensional delta function is defined as in Equation (2.6) and this is assuming that Equation (2.2) = 1, therefore the integrand is Equation (**??**):

$$\delta^n(\mathbf{r}) = \mathcal{F}_n^{-1}[1] \equiv [(2\pi)^n]^{-1} \int_{-\infty}^{\infty} \exp(-i\mathbf{k} \cdot \mathbf{r}) d^n\mathbf{r} \tag{2.6}$$

The above equations constitute a space invariant linear system, characterized by Equation (2.7).

$$s(\mathbf{r}) = p(\mathbf{r}) \otimes f(\mathbf{r}) + n(\mathbf{r}) \equiv \int_{-\infty}^{\infty} p(\mathbf{r} - \mathbf{r}') f(\mathbf{r}') d^n\mathbf{r}' + n(\mathbf{r}) \tag{2.7}$$

where all the vectors are elements of L^2 such that: $\forall v \in L^2(\mathbb{R}^n)$ outputs are closed under all operations.

The operator $\otimes$ is used as a multi-dimensional convolution integral.

From the convolution equation, Equation (2.7), the output signal (i.e. ciphertext) is denoted by $s(\mathbf{r})$. The IRF is represented by the function $p(\mathbf{r})$. This is used to extract information embedded in a noise source and this happens when Equation (2.6) $= f(r)$ and there is no noise, i.e. $n(r) = 0$. The assumption is that the stochastic function $n(\mathbf{r})$ possesses some probability density function (PDF) characteristics. However, unlike the stochastic function $n(\mathbf{r})$, the other two functions $p(\mathbf{r})$ and $f(\mathbf{r})$ are assumed to be deterministic in nature.

Therefore, it is trivial to realise that the output signal $s(\mathbf{r})$ is a result of deterministic and stochastic function i.e. $p(\mathbf{r}) \otimes f(\mathbf{r})$ and $n(\mathbf{r})$. Hence, Equation (2.7) becomes a model describing a standard signal. This model forms the basis for subsequent encryption models discussed in this thesis.

In its simplest form, an encryption model can be represented in terms of an output signal as an additive operation on the information function $f(\mathbf{r})$ and noise function $n(\mathbf{r})$, as in Equation (2.8)

$$s(\mathbf{r}) = f(\mathbf{r}) + n(\mathbf{r}) \tag{2.8}$$

It must be noted that the functions in Equation (2.8) are not exclusively real, as they may be from other domains such as the complex domain. There is another aspect to the operands in Equation (2.8) worth noting. Under 'normal circumstance', for example, transmission of the information function $f(\mathbf{r})$, through a noise source, the noise channel cannot be controlled at all but it can be classified. Specifically, although not entirely exclusive, when considering the statistical distribution as well as its moments. Albeit this important aspect, in Cryptography, the noise function (i.e. term) is controlled by the Cryptographer. Hence, Equation (2.8) can be represented through the following terms in Equation (2.9):

$$\text{Ciphertext} = +[\text{Plaintext}, \text{Cipher}] \tag{2.9}$$

Note: $+$ is an additive function. The ciphertext term is the non-intelligible text which

has been transformed by the cipher with the use of a key. The Cipher is generated using a key dependent algorithm analogous to a one-way function. Therefore, we can have the following representation:

$$\mathcal{F}(\mathbf{k}) \rightarrow n(\mathbf{r})$$

It can now be seen that the output of the algorithm $\mathcal{F}(\mathbf{k})$ is a noise generating function, given by $n(\mathbf{r})$. The above representation now brings us to a common problem in symmetric encryption systems, given that two communicating entities Alice and Bob have an identical algorithm $\mathcal{F}(\mathbf{k})$. Two issues arise and they are listed below:

- How can Alice and Bob securely exchange the secret key ($\mathbf{k}$) to be used in the cipher generating algorithm? This is the 'Key Distribution/Exchange Problem'
- How can the algorithm $\mathcal{F}(\mathbf{k})$ generate a suitable output noise? This is 'Cryptographic Strength Problem'

The above are problems encountered in all symmetric data encryption methods. The first relates to an issue where the two communicating entities need to agree on the independent key $\mathbf{k}$ prior to exchanging any confidential data. The second problem is to ensure that the algorithm is strong enough to withstand any attacks, such that the output ciphertext is not key or algorithm dependent. These two problems are the foundation for the discussions presented in this thesis. The next section highlights how encryption is done on integer and binary data.

2.3 Data Encryption: Integer and Binary Sets

From the previous discussions and Equation (2.8), to retrieve the original information i.e. plaintext message, the function $f(\mathbf{r})$, is computed as in Equation (2.10)

$$f(\mathbf{r}) = s(\mathbf{r}) - n(\mathbf{r}) \tag{2.10}$$

Equation (2.10) removes the noise from the signal computed in Equation (2.8), for decryption to be possible. If the range for all these functions is a discrete multi-dimensional

array of floats, the result will be an encrypted/decrypted array of floats, as the range is closed under this operation. It must be noted that the accuracy in the floating point arithmetic must be given by some number of bits, for all functions. For example, let all the functions be represented by vectors $\mathbf{s}$, $\mathbf{f}$ and $\mathbf{n}$ i.e. the signal, the information and noise, respectively, with an equal number of floating point elements, for $\mathbf{r} \in \mathbb{R}^1$, the encryption and decryption process for the floating-point space is given by:

$$\mathbf{s} = \mathbf{f} + \mathbf{n}; \;\; and \;\; \mathbf{f} = \mathbf{s} - \mathbf{n}$$

The process remains the same for integers,

$$z \in \mathbb{R}^1$$

For example, a 7-bit ASCII input $\in$ would be encrypted and decrypted in the following manner:

$$\mathbf{s} = [+(\mathbf{f}, \mathbf{n})] \text{mod}(127)$$

$$\mathbf{f} = [-(\mathbf{s}, \mathbf{n})] \text{mod}(127)$$

respectively, where mod denotes the standard modulo operation, which is the remainder when *divides*.

We can now consider $\mathbf{s}$, $\mathbf{f}$ and $\mathbf{n}$ to be binary vectors, consisting of an equal number of bits. Under this condition(s), the binary space encryption and decryption processes are computed as shown below:

$$\mathbf{s}_{bin} = \oplus(\mathbf{f}_{bin}, \mathbf{n}_{bin}); \quad \mathbf{f}_{bin} = \oplus(\mathbf{s}_{bin}, \mathbf{n}_{bin})$$

The binary function, $\oplus$ is the standard XOR operator.

Based on the above, it has been shown that at the basic level, floating-point and binary encryption and decryption operations can be computed for a signal output $\mathbf{s}$ based on the information function $\mathbf{f}$ and the noise source $\mathbf{n}$. The operation of floats makes the encryption an additive function while the decryption is the additive inverse of the

operation while on binary streams the encryption and decryption function is identical to an XOR operation.

2.4 Analogue Data Encryption

The previous section looked into floating-point and binary data encryption and decryption. This section briefly discusses analogue data encryption. Although this method of encryption is beyond the scope of this thesis, its application is very important as it is used in military communications, for highly classified information is being exchanged. In comparison to the Section (2.3), assuming a floating point precision of p-bits data processor and a malicious threat actor armed with a q-bit precision processor, where the latter has a bit length less than the former, i.e. $q < p$, then the attacker may try to compute the cipher between Alice and Bob to exploit its floating-point depth. However, the threat actor will not be able to generate the decrypt directly due to the lack of floating-point precision needed to cryptanalyse a p-bit ciphertext.

Due to this failure, an attacker can potentially opt to use 'analogue' computing. This is to enable the attacker to generate analogue ciphers. These ciphers can be digitised. This is a possible scenario which may occur when key exchange is not a requirement and the two communicating entities, Alice and Bob, use personalised encryptors, using their own encryption keys, before uploading data to the cloud. In this case, there is no need for Alice and Bob to perform key exchange using a trusted third party (TTP) such as a key distribution center (KDC) or Certificate Authority (CA). This move is a good cloud security mechanism [34]. This approach has been investigated by [34], where cloud users generate their floating-point, personalised ciphers, engineered using evolutionary computing concepts for cloud-hosted data encryption. This approach is also in chapter 7 of this thesis. The next section discusses the two fundamental concepts of data encryption, diffusion and confusion. These concepts are essential in ensuring that the encryption algorithm has the required cryptographic strength, in order to solve the second problem highlighted in the previous sections.

2.5 The Cryptographic Corner-Stones: Diffusion and Confusion

There are two fundamental principles of any cryptographic scheme. These are Confusion and Diffusion. In the context of the encryption/decryption models discussed above, the target is always to maximize the levels of Confusion and Diffusion during transformation of plaintext to ciphertext. An cipher generating algorithm lacking these two principles is weak and susceptible to cryptanalysis. Below is a description of each:

- Confusion: Refers to the property of an encryption algorithm to conceal any correlation between the encryption key and ciphertext to make it as complex as possible. In other words, confusion ensures that small changes in the plaintext result in significant changes in the ciphertext [58].

- Diffusion: Ensures that the statistical relationship between the two messages i.e. ciphertext and plaintext is hidden. This ensures that the statistical distribution of the plaintext is fully dissipated in the ciphertext [30].

Confusion makes the relationship between the plaintext and the ciphertext as complex as possible, while diffusion spreads the influence of the plaintext over the entire ciphertext. The most important aspect of the second principle is to have an encryption process that will effectively dissipate the plaintext over the full extent of the ciphertext. Having described these two principles, we consider Equation (2.7) and the following encryption model given by Equation (2.11) which replaces the information function, IRF, $p(\mathbf{r})$ with the noise-generating function $n(\mathbf{r})$.

$$s(\mathbf{r}) = \overbrace{\underbrace{n(\mathbf{r}) \quad \otimes \quad f(\mathbf{r})}_{\text{Stochastic Diffusion}} + \; n(\mathbf{r})}^{\text{Stochastic Confusion}} \tag{2.11}$$

Equation (2.11) is the fundamental building block of the encryption models considered in this thesis. This model has both (stochastic) confusion and diffusion properties. The output signal $s(\mathbf{r})$ is the result of combining the noise with the transformed information

function (IRF) $f(\mathbf{r})$ by the same noise function, $n(\mathbf{r})$. The transformation is computed by the convolution operator, which is effectively an XOR for a binary domain and addition for other domains. This transformation represents a convolutional encoding function. Diffusion is therefore realised when the convolution operation $n(\mathbf{r}) \otimes f(\mathbf{r})$ is performed. Addition of a noise element, $n(\mathbf{r})$, to the convolutional encoding result is a representation of stochastic confusion. This is illustrated Equation (2.11). This makes the entire output signal a stochastic variable.

How then do we ensure diffusion and confusion? To 'maximize' stochastic diffusion, the random noise term, $n(\mathbf{r})$ is fully dissipated and statistically and uniformly distributed with a uniformly distributed Power Spectral Density Function (PSDF). To ensure 'Maximum' confusion, the noise term $n(\mathbf{r})$ must dominate the output signal $s(\mathbf{r})$. Consider Equation (2.12):

$$\mathcal{P} = \|n(\mathbf{r}) \otimes f(\mathbf{r})\|_z \tag{2.12}$$

The condition to have maximum confusion is illustrated in the following condition:

$$\|f(n) - n(\mathbf{r})\|_z >> \|\mathcal{P}\|$$

where $f(n)$ is a noise function and Equation (2.13) below,

$$\mathcal{W} = \sqrt[z]{\left(\int_{-\infty}^{\infty} [f(\mathbf{r})]^2 d^n\mathbf{r}\right)}, \; 1 \leq z \tag{2.13}$$

$$\forall r \in \mathbb{R}^n$$

where the $z-$ norm is defined as follows:

$$\|f(\mathbf{r})\|_z \equiv \mathcal{W}$$

From the above, it can be deduced using the Convolution Theorem from Equation (2.11) can be written as a multi-dimensional, incremental Fourier space, such as

in Equation (2.14)

$$S(\mathbf{k}) = *(+[F(\mathbf{k}), 1], N(\mathbf{k})) \tag{2.14}$$

where * is multiplication and + is an addition operator, presented as binary functions. With the foregoing, it has been proven that maximum confusion can be achieved through addition of the noise source to the information function. This is a key principle in ensuring that the ciphertext and the encryption key have a complex relationship which cannot be deduced by a malicious threat actor.

2.6 The Fundamental Principle of Cryptography

Given any encryption model, the fundamental principle in cryptogrpahy which ensures absolute data security against any form of attack is to use a One-Time-Pad (OTP). The OTP remains the only cryptographic model without any successful attack in modern-day cryptography. The OTP uses the following principle: *One plaintext (p), one ciphertext (c) one encryption/decryption key (k), one Encryption function (E)*, for any encryption session as in Equation (2.15):

$$C_i = E(p_i, k_i) \tag{2.15}$$

This OTP principle generates ciphertexts $_i$ that possess the following attributes:

(i) computational security,

(ii) unconditional security. i.e. secure whatever the computational power of the attacker.

These two attributes ensure that the ciphertext is secure enough even if an attacker is equipped with the highest computing power or whatever computational capabilities the attacker may have access to. Computational security refers to the level of security provided by a cryptographic system based on the assumption that certain computational problems are difficult to solve. Unconditional security, on the other hand, refers to the highest level of security provided by a cryptographic system, which is provably secure against any type of attack, regardless of the computational power of the attacker. However, achieving

unconditional security is often impractical or impossible in many real-world scenarios, so most cryptographic systems aim for computational security. It is on the premise of this fundamental principle that this thesis considers a phase-only digital encryption and decryption approach, discussed in subsequent chapters.

2.7 Steganocryptography

Using the principle discussed in Section (2.5), another field of Cryptography can be looked into. This field deals with concealing data into other media or data types to produce a 'covertext'. This field is called Steganocryptography [7]. The idea of a 'one encryption algorithm' to produce a cipher raises a question of algorithm exchange. Conventional encryption algorithms undergo public comments and scrutiny and are public knowledge as Kerckhoff's principle dictates. However, the construction of personalised cryptographic algorithms was considered by [34]. These algorithms use evolutionary computing concepts to generate algorithms that output a close approximation of the noise source, which will be an input into the noise function $n(\mathbf{r})$. The same way as key distribution occurs, the question frequently asked is how then will the algorithms be exchanged? Regarding this issue, chapter 6 of this thesis presents a plausible solution. This solution uses a three-pass protocol (TPP). Although the TPP considered is meant for quantum key exchange (QKE), the same concept can be adopted for personalised crypto-algorithm exchange. These crypto-algorithms can be used in the field of Steganocryptography as shown in subsequent chapters of this thesis. The idea is to show that personalised algorithms can be used to encrypt data and conceal it in other data types or media. The next section looks into encryption using convolutional encoders.

2.8 Data Encryption using Convolutional Encoding

Given a well-formed binary input sequence generated by the function $b[n] = \{0, 1\}^{\ell}$ where ℓ is a Kleene plus operation and determines the block length produced by the binary

function $f[\mathbf{n}]$. Given any binary, Impulse Response Function (IRF) $IRF[n]$, and a convolutional encoding function which yields an an output $enc^{\ell}[n]$ as in [133], a convolutional model can be created as shown in Equation (2.16).

$$enc^{\ell}[n] = \sum_{m=1}^{\infty} IRF[n-m]b^{\ell}[m] \tag{2.16}$$

From Equation (2.16), it can be deduced that this convolutional model is a description of a discrete, linear time-invariant system. Such a system is used for digital signals processing. The functions $IRF[n]$, $b[n]$ and $enc[n]$, are all coming from a Real number set of arrays i.e.,

$$\forall n \in \mathbb{R}$$

. There are various use cases of convolutional codes, such as reliable data transfer in digital media e.g., videos, mobile and satellite communications [120]. Usually, these codes are concatenated with hard-decision codes. Before the emergence of turbo codes, such implementations were efficient and compared very closely to the Shannon limit. The convolutional encoding model has one of the biggest reasons for it to be used and this is to achieve the maximum probability for decoding the signal with reasonable complexity. To achieve this, a time-invariant based decoder is used, as shown in [214], using the Viterbi algorithm. This algorithm is commonly used for cyclic error checks [149] where error corrective codes are used. For this context, the next section considers an approach which is an extension of the convolutional encoding process. This is applied on n-dimensional data fields with a consideration of applying a modification to the process that generates an inverse solution, such that deconvolution can occur. This is generally referred to as the 'deconvolution problem'.

2.9 Multi-dimensional Encoding

Consider a function $f(\mathbf{r})$ for $\mathbf{r} \in \mathbb{R}^n,\ n = 1, 2, 3, ...$ with n-dimensional Fourier and inverse Fourier transforms

$$F(\mathbf{k}) = \mathcal{F}_n[f(\mathbf{r})] \equiv \int_{-\infty}^{\infty} f(\mathbf{r}) \exp(-i\mathbf{k} \cdot \mathbf{r}) d^n\mathbf{r} \tag{2.17}$$

and

$$f(\mathbf{r}) = \mathcal{F}_n^{-1}[F(\mathbf{k})] \equiv \frac{1}{(2\pi)^n} \int_{-\infty}^{\infty} F(\mathbf{k}) \exp(i\mathbf{k} \cdot \mathbf{r}) d^n\mathbf{k} \tag{2.18}$$

respectively, $\mathbf{k}$ being the spatial frequency vector. If $g(\mathbf{r})$ is some stochastic function (a cipher) generated by a known algorithm or some other source (a 'code' or natural noise, for example), then convolutional encoding involves convolving $g(\mathbf{r})$ with $f(\mathbf{r})$ to produce an output $h(\mathbf{r})$ say, which we can write as ($\otimes$ denoting the n^{th} order convolution integral $\forall \mathbf{r} \in \mathbb{R}^n$)

$$h(\mathbf{r}) = g(\mathbf{r}) \otimes f(\mathbf{r}) \equiv \int_{-\infty}^{\infty} g(\mathbf{r} - \mathbf{r}') f(\mathbf{r}') d^n\mathbf{r}' \tag{2.19}$$

The transmission of such an output is taken to be corrupted by additive transmission noise described by the function $n(\mathbf{r})$, which introduces errors in to the recovery of $f(\mathbf{r})$ from $h(\mathbf{r})$ and thus we arrive at an equation of the form Equation (2.20)

$$h(\mathbf{r}) = g(\mathbf{r}) \otimes f(\mathbf{r}) + n(\mathbf{r}) \tag{2.20}$$

under the assumption that

$$\|n(\mathbf{r})\| << \|g(\mathbf{r}) \otimes f(\mathbf{r})\| \leq \|g(\mathbf{r})\| \times \|f(\mathbf{r})\|$$

the ratio $\|g(\mathbf{r}) \otimes f(\mathbf{r})\| / \|n(\mathbf{r})\|$ being known, in general, as the Signal-to-Noise Ratio or SNR in the fields of signal processing when $\mathbf{r} \in \mathbb{R}^1$ and image processing when $\mathbf{r} \in \mathbb{R}^2$.

2.9.1 Decoding: The Deconvolution Problem

This inverse (deconvolution) problem is as follows: Given Equation 2.20, and, with functions $h(\mathbf{r})$ and $g(\mathbf{r})$ known, obtain a solution for $f(\mathbf{r})$. In some cases, $g(\mathbf{r})$ may not be

known and the problem becomes the so-called 'blind deconvolution problem'. In general, this problem is an ill-posed problem. Consequently, this problem has a range of solutions whose purpose is usually to regularise the inverse solution. This is made in such a way that an optimum estimate of $f(\mathbf{r})$ can be obtained subject to certain conditions. Most such solutions take the form

$$\hat{f}(\mathbf{r}) = q(\mathbf{r}) \otimes h(\mathbf{r}) \tag{2.21}$$

where $\hat{f}(\mathbf{r})$ is an estimate of $f(\mathbf{r})$, $q(\mathbf{r})$ is some filter with Fourier transform $Q(\mathbf{k}) = \mathcal{F}_n[q(\mathbf{r})]$. Well-examples include the following:

The Wiener Filter

$$\hat{f}(\mathbf{r}) = \mathcal{F}_n^{-1}[Q(\mathbf{k})H(\mathbf{k})]$$

where $H(k) = \mathcal{F}_n[h(\mathbf{r})]$,

$$Q(\mathbf{k}) = \frac{G^*(\mathbf{k})}{\mid G(\mathbf{k}) \mid^2 + \mid N(\mathbf{k}) \mid^2 / \mid F(\mathbf{k}) \mid^2}, \; G(\mathbf{k}) = \mathcal{F}_n[g(\mathbf{r})] \text{ and } N(\mathbf{k}) = \mathcal{F}_n[n(\mathbf{r})]$$

under the condition that $\|\hat{f}(\mathbf{r}) - f(\mathbf{r})\|_2^2$ is a minimum with the assumption that (signal independent noise) [27]

$$n(\mathbf{r}) \odot f(\mathbf{r}) = 0 \text{ and } f(\mathbf{r}) \odot n(\mathbf{r}) = 0,$$

$\odot$ being taken to denote the n^{th} order correlation integral $\forall \mathbf{r} \in \mathbb{R}^n$, i.e.

$$h(\mathbf{r}) = g(\mathbf{r}) \odot f(\mathbf{r}) \equiv \int_{-\infty}^{\infty} g(\mathbf{r} + \mathbf{r}') f(\mathbf{r}') d^n\mathbf{r}'$$

The Maximum Entropy Filter

$$\hat{f}(\mathbf{r}) = \exp\{-1 + 2\lambda[h(\mathbf{r}) \odot g(\mathbf{r}) - g(\mathbf{r}) \otimes \hat{f}(\mathbf{r}) \odot g(\mathbf{r})]\}$$

where λ is the Lagrange multiplier and is based on the Entropy, defined as

$$E = -\int_{-\infty}^{\infty} f(\mathbf{r}) \ln f(\mathbf{r}) d^n\mathbf{r}$$

being a maximum, a 'solution' for $\hat{f}(\mathbf{r})$ that requires iteration to be applied. However, for $\lambda << 1$, we can write

$$Q(\mathbf{k}) = \frac{G^*(\mathbf{k})}{\mid G(\mathbf{k}) \mid^2 + 1/2\lambda}$$

which gives the linearised maximum entropy filter for estimate $\hat{f}(\mathbf{r})$ in the form of Equation 2.21, [27].

The Maximum a Posteriori Filter

$$Q(\mathbf{k}) = \frac{G^*(\mathbf{k})}{\mid G(\mathbf{k}) \mid^2 + \sigma_n^2/\sigma_f^2}$$

where σ_n and σ_f denote the Standard Deviations of the Gaussian Probability Density Functions $P[n(\mathbf{r})]$ and $P[f(\mathbf{r})]$ of $n(\mathbf{r})$ and $f(\mathbf{r})$, respectively, subject to the condition (Bayes rule) that, [27]

$$\frac{\partial}{\partial f} \ln P(h \mid f) + \frac{\partial}{\partial f} \ln P(f) = 0$$

Constrained Deconvolution

$$Q(\mathbf{k}) = \frac{G^*(\mathbf{k})}{\mid G(\mathbf{k}) \mid^2 + \mid P(\mathbf{k}) \mid^2 / \lambda}$$

where $P(\mathbf{k}) = \mathcal{F}_n[p(\mathbf{r})]$ is any 'constraining spectrum' such that $\|p(\mathbf{r}) \otimes f(\mathbf{r})\|_2^2$ is a minimum, [27].

2.9.2 Decoding: The Deconvolution Problem for Special Functions

For certain functions $g(\mathbf{r})$ the deconvolution problem, as posed by Equation 2.20, can be solved without resorting to a method of regularisation. The most important of these functions is the (unit amplitude) linear modulation function $g(\mathbf{r}) = \exp(i\alpha r^2),\ r \equiv \mid \mathbf{r} \mid$ for constant α. In this case, by correlating Equation 2.20 with complex conjugate $g^*(\mathbf{r})$ (i.e. applying a 'matched filter') we obtain

$$g^*(\mathbf{r}) \odot h(\mathbf{r}) = \exp(-i\alpha r^2) \odot \exp(i\alpha r^2) \otimes f(\mathbf{r}) + \exp(-i\alpha r^2) \odot n(\mathbf{r})$$

$$= \delta^n(\mathbf{r}) \otimes f(\mathbf{r}) = f(\mathbf{r})$$

under the condition that $\exp(-i\alpha r^2) \odot n(\mathbf{r}) = 0$ (i.e. the stochastic function $n(\mathbf{r})$ does not correlate with the linear modulation function - the noise is 'signal independent') where δ^n is the n-dimensional Dirac delta function, [27]. This result is the basis for an approach to embedding information in signals and images with applications that include speech and document authentication, for example. It provides the basis for an information hiding model that yields the highest degree of resilience to distortion, especially with regard to additive noise, compared to other transformation-based information hiding methods, known to date [35] [110]. In this context, the problem remains as to whether some sort of proof can be developed which proves that quadratic phase-based impulse response functions (IRFs) of this type are unique in terms of their ability to code and decode information transmitted through high noise environments. If so, then it should be noted that such quadratic phase functions are generated by non-relativistic quantum mechanical systems in relation to the short-term transient behaviour of a beam of electrons, for example, propagating through free space upon the opening of a quantum shutter, [27]. This 'time scattering' effect could potentially be used for quantum (quadratic phase) information coding.

2.10 Discussion

The nature of Equation 2.20 and the traditional (discrete) convolutional encoding/decoding processes is based on a model for an n-dimensional signal [41][40]. This model is ultimately related to the physical process that results in the generation and detection of a signal. Equation 2.20 is therefore considered to be a fundamental model for the analysis and processing of signals in general (even in the case of the linear and time or space invariant case) in a variety of applications, such as acoustics and optics and other electromagnetic information systems. With the exception of quantum cryptography, we are free to 'invent' ideas for data encryption techniques that do not necessarily need to adhere to a systems model that is based on and constrained by the laws of physics. The foundations for the

strategy taken into consideration in this thesis are described in subsequent chapters.

Chapter 3

Pseudo-Number Generators (PRNGs) for Secure Encryption

The previous chapter discussed encryption models from a signals and systems approach. Those encryption models discussed laid a foundation for the subsequent chapters. This chapter improves on the encryption models presented in chapter two by adding an element of randomness. Encryption requires randomness to be effective [188]. A random sequence consists of no patterns, hence, it is impossible to predict future values based on previous or current values [17]. To achieve the principle of confusion and diffusion discussed earlier, a statistically random sequence is required. This random sequence can be from a random noise source, for input into the noise function, $n(\mathbf{r})$ as already shown. An arbitrarily chosen noise source can yield perfect security. It is on this background that this chapter evaluates PRNGs for secure encryption. A number chosen arbitrarily from a statistically random sequence is called a random number [126]. A sequence of random numbers has an even distribution over a defined interval. A lot of domains make use of randomness. For example, they are essential in multi-variable complex systems' simulators. Random numbers are used to select samples of statistical nature from a larger finite population. The aim of this chapter is to introduce the use of random numbers in cryptography, to ensure confidentiality and integrity of crypto-systems. These two fundamental concepts

rely on the availability of random number generators. This chapter unfolds how RNGs are used to develop secure ciphers.

3.1 Random Numbers in Cryptography

The strength of a crypto-system is defined by the use of randomness in generating it. Thus, a strong cryptographic algorithm must possess some randomness properties. Random numbers are very important in cryptography [107]. Some important use cases of random numbers in cryptography are:

- Encryption key Generator: Kerckhoff's principle dictates that the security of any crypto-system must remain intact despite everything about it being public knowledge, except the 'key' [66]. An encryption key forms the basis of any cryptosystem. If it is known to the attacker, the entire cryptosystem fails and security is compromised. A strong key makes a secure cryptosystem. The strength of an encryption key is dependent on its randomness, which is based on a secure random number generator.

- One-time Pad: This is a cryptographic technique used to encrypt plaintext with a use once, randomly selected key. The key length must be at minimum, equal to the length of the plaintext. The OTP is the only unbreakable encryption algorithm to date [195]. This is guaranteed if and only if the key securely computed by a random number generator.

- Salting and Nonce: A salt is a randomly generated data padded to the plaintext before encryption. A nonce is a randomly selected number which is used only once in a cryptosystem. These two concepts rely heavily on a secure random number generator.

- Seeding: A seed initializes a pseudo random number generator. This must be randomly selected to ensure the attacker does not predict the random number sequence which will result in revealing the key.

3.1.1 Generation of Random Numbers

Two types of RNGs are:

- True Random Number Generators (TRNGs): TRNGs generate random numbers by extracting randomness from stochastic physical processes and/or quantum phenomena as opposed to using an algorithm. For example, thermal noise and radioactive decay [195].
- Pseudo-Random Number Generators (PRNGs): These generators mimic randomness through mathematical formulae and pre-computed tables.

TRNGs extract entropy from an event exhibiting some form of randomness. Using a deterministic procedure, entropy is extracted to measure expected uncertainty, which is a property of a random variable. An output produced by the extractor is a short but uniformly distributed string [190]. PRNGs use a deterministic algorithm seeded to generate a sequence of random numbers. The output of a PRNG is completely dependent on the seed used to generate it, which serves as an initialization vector (IV). The advantage of using deterministic measures is that the random sequence can be reproduced.

The benefit of the determinism is that a sequence of numbers can easily be reproduced with the same initialization vector. This makes it suitable for application in various domains such as simulations, virtual reality, software quality testing etc., [212]. If a known algorithm is used, the seed is the only parameter an attacker needs in predicting the random sequence generated. Due to virtually all algorithms having a periodic nature, the generated random sequence is bound to repeat at some point. However, modern PRNG's are developed to have a longer period for practical implementation to ensure their effectiveness and efficiency [195]. Thus, PRNGs are considered efficient as they take minimal amounts of time in generating a random sequence. They are also very easy to develop and quantify compared to TRNGs which bring difficulty in extracting and quantifying true randomness [195].

Another type of a random number sequence generator is a Cryptographically Secure Pseudo-Random Number Generator (CSPRNG). This is a variant of PRNGs suitable for

use in cryptography due to its properties. For CSPRNGs to be accepted and used in Cryptography, they must pass rigorous tests i.e. statistical randomness tests. These tests are conducted to ensure that the CSPRNGs possess the highest degree of entropy. This is a requirement for CSPRNGs to produce randon sequences that are indistinguishable from true random bit streams. All CSPRNGs are required to pass the 'next-bit test'. This test suggests that given the first k bits of a random sequence, no polynomial-time algorithm must exist which can predict the next $(k + 1^{th})$ bit with a higher than 50% success rate or higher [195].

Resilient CSPRNGs

Another attribute which CSPRNGs must possess is resiliency, under attacks. Every CSPRNG must thwart attacks and withstand state changes or compromise. Each number k in a randomly generated sequence must be independent from the next random bit in the sequence. This is to ensure that it becomes infeasible for a malicious threat actor to compute or predict the random bits in the sequence, previous or future states. A true random source is required for CSPRNGs. This source is used as an initialization vector. Thus, this thesis makes a plausible contribution in Cryptography through the use of chaotic random noise as a true random source. CSPRNGs are used mostly when a trade-off between entropy and the number of random sequences is generated. In this case, the use of CSPRNGs provides efficient, secure encryption and authentication. The wide adoption of secure random numbers in Cryptography is due to nature of their output [195] which is a long random binary sequence. This output is used to generate encryption keys, initialization vectors i.e., seeds, salting values, nonces etc., [107]. One other aspect to consider is how random bits are selected. The selection method is an important technique a CSPRNG must have. The stronger the sampling and selection techniques, the more complicated it becomes to an attacker to guess the encryption keys generated by the random sequences. For example, a 256-bit encryption key being sampled from a 10^5-bit stream would be a good selection technique as the sample sequence is long. Another issue is related to secure storage of such sequences, to avoid unauthorised access. A good

selection technique uses session sequences. These are used only once, per session.

A complex process in determining randome sequences is their evaluation as it is not a trivial task [178] [21]. To assess randomness, statistical hypothesis tests are required [200]. These are the first steps to evaluate and determine if the random number generator is good enough for cryptographic use. However, there is no definitive set of statistical tests which can absolutely proof that a certain generator is cryptographically secure as this process cannot be used as a substitute to cryptanalysis [178][21]. Based on this, some methods of random number generation have been proposed. Among these methods are Artificial Neural Networks (ANNs). This approach provides non-linear computations which simulate how neurons in the brain function. ANNs have weights. These are values between neurons. The ANN learns by adjusting these weight values. There are different designs of ANNs, varying from a simple feed-forward design to recurrent neural network topologies. The former allows a unidirectional traversal while the latter has information flowing bidirectionally i.e., to and from any neuron in the network. The attributes that make ANNs suitable for generation of random number sequences is their non-linearity and unpredictability features [126].

3.1.2 Are Neural Networks efficient for random number generation?

Various techniques are used to generate CSPRNs. Using ANNs is still a widely researched field. Researches are constantly utilizing the evolving computational power in their laboratories to come up with new, secure techniques to add to the field of Cryptography. However, the increase in computational power also benefits the attackers who aim to compromise these secure techniques, bringing the cyber world into 'crypto wars'. Encryption is used to protect sensitive information. Unauthorised access to such information could bring dire results. For example, identity theft, democratic electoral results could be compromised, company reputations could be destroyed etc. Thus, strong cryptosystems are needed to provide confidentiality guarantees against all sorts of attacks. In reality,

only TRNGs would be ideal but this is infeasible [195]. Hence, new methods which aim to improve PRNGs are required. Hence, this thesis explores the viability of neural networks in generating cryptographically secure random numbers. To evaluate the successful implementation of NNs in Cryptography, the following evaluation criteria is used:

- The efficacy of the NN: This aspect will test whether the NN meets the statistical tests required for use in Cryptography.
- The efficiency of the NN: This will evaluate where the speed of random sequences meets or surpasses modern CSPRNGs.

While the aim is to design and implement CSPRNGs with the help of Machine Learning (ML) concepts, the key objectives are to:

- Design and develop CSPRNGs using ML techniques.
- Development of an evaluation framework determining the efficacy and efficiency of the CSPRNGs.
- Compare the selected CSPRNGs to other generators.

A CSPRNG developed under ML techniques exhibits AI principles. Thus, it has capabilities of learning, either using supervised or unsupervised machine learning. Often, the efficacy and efficiency of such generators are of high standards. Chapter 9 discusses a Counter Propagation Neural Network (CPNN), developed in MATLAB and trained to learn patterns of an encryption key to serve as an encryption key store. The success of this CPNN in learning the key demonstrates the importance of AI in Cryptography.

3.2 How to create Random Number Generators

Two steps are involved in the stages/processes of creating random number generators. These are: **implementation** and **evaluation**. The implementation stage is creation of a system to generate a random sequence while the evaluation stage is a series of tests to check

the generated sequence. This section focuses on the implementation stage, considering both TRNGs and PRNGs. Other cryptographically secure implementations also discussed and ultimately the focus shifts to ML implementations, in particular, the use of ANNs as CSPRNGs.

3.2.1 True Random Number Generators

TRNGs work on physical sources. Randomness is extracted from physical phenomena to produce an output string. This output is evenly distributed over the randomly generated sequence with the highest degree of entropy [190]. A good example of an unpredictable source is the radioactive decay process. In this process, an unstable nucleus emits radiation and loses energy in the process. Based on Quantum theory, this is an unpredictable process as it is infeasible to guess the next atomic nucleus will decay [99]. Atmospheric noise has the same phenomenon. In this case, during a thunderstorm, a noise source is used to measure lightning discharges measure entropy. User dependent behavioural patterns can also measure entropy. For example, keystroke dynamics, mouse movements and clicks, operating system interrupts [190] [195]. It is complicated to extract entropy from these sources as TRNGs have been proven to be inefficient in implementing generators from these sources when compared to PRNGs. This inefficiency presents problems in some applications [99]. For example, for cryptographic systems required to generate billions of random sequence bits per second are impractical to achieve using TRNGs.

3.2.2 Pseudo-Random Number Generators

The three widely adopted type of algorithms used in PRNGs are:

i Block Ciphers,

ii Stream Ciphers, and

iii Traditional PRNGs

PRNGs are deterministic algorithms, by nature and they use a seed as an initialization vector. This is to enable generation of a random output sequence. Block ciphers process a fixed-size plaintext block to output a ciphertext block of the same size while stream ciphers encrypts each plaintext bit or character independently, using a cipher and an encryption key. These output are considered to be randomized, given an encryption and/or decryption algorithm and the key [85] [62]. One of the oldest techniques of PRNGs is the Mid-Square method [48] [23]. This is a method based on modulo arithmetic (mod). Although this method was effective, it has been replaced by other more efficient methods. Another method of PRNGs is the Linear Congruent Generator (LCG). This method uses a linear equation and a modulo operator to generate random sequences. The LCG linear equation is defined by Equation (3.1):

$$F_{n+1} = (aF_n + C)\text{mod(M)} \tag{3.1}$$

where F_0 is the seed i.e. initialization vector and F is an array of generated pseudo-random numbers. The LGC was very efficient for a long time. It has since been replaced by its quadratic variant due to its small period, which is at most m [48]. The quadratic variant of the LCG was proposed by D.H. Lehmer, Quadratic Congruential Generator (QCG), which extends the LCG, in 1949. The QCG use a quadratic equation to generate a pseudo-random sequence. However, given today's computing power, these techniques are no longer regarded as secure. Therefore, more complex techniques are now used to generate pseudo-random numbers.

Another widely used PRNG is the Mersenne Twister (MT) [5]. This generator was developed to correct some of the flaws found in other PRNGs. The MT algorithm is used in various programming languages, such as Python and C++, to generate random number sequences [207]. The MT has a period length that is chosen to be a Mersenne prime, for any integer n i.e.,

$$\forall n \in \mathbb{Z}$$

,

$$M_n = 2^n - 1$$

. The algorithm uses a linear-feedback shift register, which uses cascaded flip-flops in the process of generating random numbers. This algorithm is recursive in nature and is based on a recursive operation, the bit-wise XOR. Despite the generator having a longer period, it is not a cryptographically secure algorithm. This is because its generated outputs can be predicted given a large enough sample of previously generated outputs. Future output values can also be predicted after $2^4 \times 3 \times 13 = 624$ iterations. This is because the algorithm has an internal state of $2^4 \times 3 \times 13 = 624$, 32-bit words [5]. Also, it can be seen that the prime factorisation of the total states becomes trivial. The next section discusses CSPRNGs.

3.2.3 Cryptographically Secure Pseudo-Random Number Generators

One of the strongest attributes of CSPRNGs is that they generate random sequences that cannot be predicted by a malicious threat actor. A number of explorations have been conducted in the quest to create secure generators. CSPRNGs are classified into two categories [195]:

- Cryptographic primitive designs
- Special purpose designs

The primitive design is based on block ciphering in counter mode. Examples are the Data Encryption Standard (DES) and Advanced Encryption Standard (AES). Using a counter mode (CTR), an arbitrary counter-initialed value is encrypted and taken as input into the encryption algorithm along with the plaintext to produce a ciphertext block. One advantage of CTR mode is that there is a different counter value for each block, therefore the correlation between the plaintext and ciphertext is complex, hence, the same plaintext value can produce two or more different ciphertext values. For an n-bit input block, CTR

mode has a period of 2^n. The CTR mode can also be used with stream ciphers. An example of a CTR mode in a strewam cipher is CryptMT. This algorithm is a CSPRNG with a Mersenne Twister implementation in its internal process. Due to the MT having a high period, the CryptMT had a sufficiently high period. However, a global portfolio (i.e. eStream) assessing stream ciphers for adoption and widespread use did not accept the CryptMT to be a secure random number generator [18]. To reinforce the Mersenne Twister and improve its security, cryptographically secure hash functions can be used, such as such as SHA-1 and SHA-2 [5].

There have been different special-purpose CSPRNGs designs implemented. Special-purpose CSPRNGs have an additional entropy. However, they are not pure pseudo-random number generators as their random output sequence does not entirely depend on the initial state. This is an advantage against attackers and introduces resiliency even if the initial state of the generator is compromised. A good example is the Yarrow Generator (YG). This generator was developed in 1999 [107]. The YG was implemented on Apple operating systems i.e. iOS and OSX., devices [117]. The YG consists of the following:

- An entropy box - This is where semi-random samples are collected and accumulated.
- A random sequence generator - Generates the pseudo-random output sequences.
- A seeding mechanism - Re-initializes the key using newly selected entries from the pool.
- Controller - Determines when re-seeding should be invoked.

The generator accumulates enough entropy to get the PRNG into an unguessable state. This attribute makes it a successful generator used in the generation of secure encryption keys. It generates the encryption keys through an iteration by computing a hash of a changed entry from the entropy pool [107]. This generator implements its security by trying to thwart known cryptanalytic attacks [80]. However, an issue surrounding this generator is that it estimates the high entropy used by the generator, to achieve an unpredictable state. This process is very impractical and consumes computing resources. To

counter this problem, another generator, Fortuna, was invented. This generator discards the entropy estimation process to generate random numbers. This is achieved by ensuring the block cipher is run with counter mode (CTR) [80].

A seemingly trusted and sufficient algorithm for use in cryptography is the Blum-Blum-Shub (BBS). Proposed by [44], the BBS algorithm was regarded as a secure random number generator. The BBS algorithm is given by Equation (3.2):

$$x_{n+1} = x_n^2 \text{mod}(M),\ M = pq,\ n = 1, 2, ..., N \tag{3.2}$$

where mod is the standard modulo operator, p and q are large prime numbers and $M = p \times q$ is their product. Equation (3.2) extends the Lehmer's algorithm which speeds up the Euclid algorithm for computing GCD. Introducing the product $M = pq$ makes Equation (3.2) effective in generating pseudo-random number sequences. The security of Equation (3.2) is dependent on the intractability to factor large prime numbers. However, this security feature will soon become a legacy with the introduction of quantum computers, which will factor large primes very quickly. Hence this thesis makes an attempt to implement secure algorithms which can be used in the post-quantum era. The BBSs algorithm is further explored in Chapter 4 of this thesis, Section (4.5)

3.2.4 Artificial Neural Networks - ANNs

The use of artificial intelligence (AI) in security has brought great results. Using ANNs to generate CSPRNGs is still a fairly new approach. ANNs possess various qualities which places them as suitable candidates for generation of random number sequences. For example, some of these ANN attributes are: complexity, they are unpredictable and have non-linear activation functions [126], as an example. The Elman Neural Network was used to generate random sequences. This is a variant of the Recurrent NNs, based on Back-propagation neural networks (BPNNs). The input layer of the EPNN is capable of receiving and sending information. Hence, the network becomes recurrent. This attribute makes the ENN capable of detecting and generating variable patterns. To generate unique inputs and target random sequences, the Elman neural network uses a keyword.

The sequences are used to train the network. To generate pseudo-random numbers, an initial weight matrix was used. An advantage of this approach is that the network was efficient [212]. Only short random sequences i.e. bits were produced by the generator. Therefore, the NIST test suite was not applicable on this generator as it is applicable only to long random sequences. Another test suite, the ENT Test, was applied. The results suggested that generator is suitable for random number generation. However, the ENT test suite is not as stringent as the NIST test suite [178] [215]. Therefore, further investigations were carried out.

To extend the ENN, a Layer Recurrent NN (LRNN) was used [67]. The LRNN differs from the ENN as it consists of an iteration on every node besides the last node. However, the LRNN takes an exponential time to train compared to the ENN [67]. Therefore, a variant of the LRNN, a Binary Recurrent (BR) NN was used to generate some random sequences [103]. The BRNN contains two possible binary states: O (OFF) or 1 (ON). A concept called neural plasticity was used on the BRNN. This is a concept used for weight and output adjustments. A pseudo-random number sequence was generated using bit sampling where a bit using the state of the BRNN at a given time stamp. The chosen output bits were concatenated to produce bits in a certain range i.e. interval. Given a string of bits, this process determines the number of bits in the string. This can be used to determine the order and disorder of a binary string. Chapter 6 delves into an evaluation of the order or disorder of a binary string to detect randomness. To determine the number of bits in a string, pseudo-random data is used. NIST statistical tests revealed that the BRNN performance met industry standards for random sequence generators, with a 99.5% success rate, passing 187 out of 188 NIST statistical tests. Out of the evaluated generators, BBSs [44] and the MT [5] were the easiest to implement and were efficient in their generation.

Another pseudo-random number generator explored was the Hopfield Neural Network (HNN). The HNN utilizes a Linear Activation Function (LAF), closely compared with the Lyapunov function with a positive exponent. Chapter 4 and 10 have a discussion on a Lyapunov function with a large positive exponent to increase cryptographic strength. The

HNN was selected due to its ability to be unpredictable under certain conditions. The HNN uses a learning algorithm which minimizes an energy function. The energy function decreases over time t and the network converges when the energy function stabilizes. As long as the network does not converge, its output is always unpredictable [66].

3.2.5 Discussion

The literature reviewed in this chapter indicates the difficulty in extracting and evaluating randomness from an entropy source. It has been shown that TRNG's are inefficient [195]. Psuedo-random number generators used to be implemented using simple techniques. However, due to requirements for high entropy, modern, complex techniques are required to achieve efficiency. Furthermore, the use of PRNG's requires extension for them to be suitable candidates for cryptographic use. Cryptographically secure pseudo-random number generators go through stringent NIST statistical tests to be considered secure enough and some generators did not meet the requirements for these tests [66] [67]. Moreover, some CSPRNGs are inefficient [103]. This chapter showed that ANNs can be implemented to output secure random sequences. However, there are issues in some ANN designs and the success in securely generating random numbers depends on the design of the ANN. Some ANN designs pose a difficulty in scaling the network to produce long random sequences [212]. Despite their success in random number generation, ANNs are can be slow in comparison to other approaches [103]. It has been discovered that the most widely used ANN architecure is the Recurrent NN. Further discussions on how random numbers are generated and their implementation in Cryptography are in the subsequent chapters, 6 and 8.

Chapter 4

Cipher Generation using Deterministic Chaos

Randomness, unpredictability, complexity, and entropy are the underlying tenets of contemporary encryption. Creating a key-dependent, unexpected bijective function that transforms data using a computational resource is the basis of a cryptographic system. For any crypto-system, such as a pseudo-random number generator (PRNG), encryption method, or key exchange mechanism, a cryptanalyst or hostile threat actor has access to the time series of a dynamic system. The threat actor is aware of the PRNG function, or algorithm, because of the Kerckhoff-Shannon principle, which holds that the adversary knows the system and that the approach is iterative. However, because intermediate states are hidden and the iteration function is assumed to include a secret parameter i.e. the encryption key, the time series is not a compact subset of a trajectory. The sample has qualities that we can describe as 'complex', 'unpredictable' and 'random'.

In relation to chaotic dynamics, this chapter provides a broad review of the connections between these features. Both chaotic dynamics and the difficulties in creating pseudo-random number generators are taken into consideration (PRNGs). The chaotic systems on which PRNGs are based. Based on an examination of the complexity and entropy metrics related to chaotic systems, a complexity and information theoretic approach is

taken into consideration. Consequently, a study of pseudo-randomness that related to chapter 3 is pursued further in this thesis. This research offers a foundational framework for the numerical techniques necessary for the real-world application of chaos theory-based data encryption. In this regard, finite-state approximations to chaos or pseudo-chaos are used in cryptographic systems.

4.1 Introduction

According to Patrick Mahon's history of Hut 8, the Naval section at Bletchly Park (Station X) from 1941 to 1945, Undoubtedly, maintaining the ability to decipher Enigma ciphers was important. If there isn't a significant change in the encryption technology, it does seem to be true that if a key has been broken frequently in the past for a long period, it will likely be broken again in the future, [100].

This statement refers to the well-known Enigma crypto-system, which the German military employed from the mid-1930s until 1945, as well as the more sophisticated Lorenz encryptor, which was used beginning in the middle of 1942 for high-level communications between the German High Command in Berlin and Army Commands throughout occupied Europe. The problem with the 'method of encryption' is related to the Kerckhoff-Shannon principle, which states that a cryptographic system should be secure even if all of its components—aside from the key, are known to the general public [118]. A more succinct definition was from Claude Shannon and states that: 'The system is known by the enemy'. This chapter demonstrates how the Kerckhoff-Shannon principle can be overcome by the application of Artificial Intelligence, particularly evolutionary computing. This area of AI is the key to breaking with the Kerckhoff-Shannon principle. In order to achieve this, a background to the case is given that contextualizes the problem before considering the usage of an evolutionary computing system called *Eureqa* [127], for producing ciphers using input data streams made up of (chaotic) natural noise.

4.2 Kerckhoff-Shannon Cryptographic principle

This principle has served as a cornerstone of cryptographic research and it constantly emphasizes the importance of exchanging encryption keys in order to use certain symmetric and asymmetric algorithms that have demonstrated their ability to encrypt and decrypt data, with adequate cryptographic strength. Many cryptographically robust algorithms, as well as the keys used to drive them, have, nevertheless, been found to be cracked in actual use. At least for those that are known, the causes of this are as diverse as the encryption techniques utilized.

There are several practical benefits to adhering to the Kerckhoff-Shannon Principle, despite the fact that research into new encryption algorithms and systems is ongoing. They consist of the following:

(i) The encryption algorithm is effective. It must be reliable and cryptographically secure.;

(ii) the practices and protocols related to using an algorithm, as well as legacy code;

(iii) the cost involved in altering the algorithm (s).

But there is another problem, which we refer to as the Enigma Syndrome. This relates to the worry that an encryption algorithm is frequently created by the very authorities who want it to be used. This is because the stockpile of Enigma machines had value after 1945 in terms of gaining intelligence from governments around the world who, at the time (i.e. from the late 1940s and early 1950s), were encouraged to use it and early derivatives of it [28]. Since the late 1950s, problems of this kind have prompted the creation of new cipher bureaux' around the world, whose goal has been and has to be the creation of novel and distinctive encryption algorithms for use by the governments they serve.

Numerous new businesses have been founded after the end of the cold war in the early 1990s and the quick advancement of computing and communications technology in order to market pre-existing encryption systems or create brand-new methods of data security. This resulted in the Regulation of Investigatory Powers (RIP) Act [49] being introduced

in the UK in 2000. The RIP Act was enacted to regulate the powers of public bodies to conduct surveillance and investigations, including the interception of communications, while also taking into account technological advancements like the expansion of the Internet and the strong encryption brought about by the new generation of ciphers being developed at the time.

The RIP Act was introduced for significant and legitimate reasons, but it also brings to light a problem that defines two significant moments in the history of cryptography. Since 1945, we have been witnessing many battles, known as 'crypto wars' such as the battle between code developers versus the code controllers. Considering the years 1900 to 1945, crypto wars referred to as battle of the code developers against the code breakers ensued. Regarding the security of the data that users upload to the cloud in this context and in light of the relatively recent introduction of cloud computing, one of the main concerns of cloud users is whether or not commonly used, commercially available encryption algorithms are secure enough for this purpose. The following challenges have grown in significance in relation to the Cloud Security Alliance (CSA):

(i) the impression that many Kerckhoff-Shannon-compliant encryption techniques are weaker than the publicly known techniques;

(ii) the Enigma Syndrome.

Can we trust the code controllers? is a key concern with regard to point (ii), even if point (i) above might be demonstrated not pose any challenges. To get around this problem, one solution is to design our own, personalized ciphers, that are set up to offer end-to-end encryption. WhatsApp introduced end-to-end encryption in March 2016 post the FBI-Apple case [162]. This is the primary purpose of the information in this chapter.

Despite the technical difficulties that can arise when users create and/or utilize their own encryption methods, there is another crucial consideration that must be recognized and is reinforced as in the following quotation [6]: *Cryptology is comparable to literacy in the Middle Ages. It is an infinitely powerful, both good and bad, but fundamentally intel-*

lectual construct that, by its very nature, will reject attempts to limit its use to bureaucrats and other people who think only they are deserving of such privilege.

In this regard, we investigate how Evolutionary Computing has the potential to democratize data encryption by enabling individuals to obtain or even create their own individualized encryption algorithms as opposed to relying on a private key to drive an algorithm that is subject to public scrutiny. The following forecasts [59], serve as the context for attempting this.

(i) By 2023, there will be more IP-connected gadgets than there are people on the planet;

(ii) By 2023, there will be 3.6 networked devices per person, up from 2.4 per person in 2018;

(iii) By 2023, Up from 18.4 billion in 2018, there will be 29.3 billion networked devices;

(iv) By 2023, more than 70% of the global population will have their mobile devices connected;

(v) Global mobile subscriptions will increase from 5.1 billion in 2018 to 5.7 billion (71%) in 2023..

One of the effects of these projections is the urgent need for cryptographic solutions needed for research in the H2020 programme [102], such as the security of personal data through the use of special encryption algorithms for encrypting data before it is uploaded into the Cloud through programs like Dropbox and MS Office 365, for example.

4.3 Complexity, Randomness and Chaos

Modern encryption methods are based on theoretically quantifiable concepts like unpredictability, entropy, algorithmic complexity and chaos. Any crypto-system's design can be understood in constructing a key-dependent bijective function to produce a random

data stream which is an unpredictable bit-wise output in the eyes of a cryptanalyst. crypto-systems that include key exchange, the architecture of a PRNG-based encryption algorithm, and PRNG design protocols rely on the creation of time series or digital signals that are usually based on an iterative encryption function (i.e., the algorithm).

According to the Kerckhoff-Shannon principle, a cryptanalyst must have access to the digital signal, which is a discrete time series value (ciphertext) and the encryption algorithm, following the encryption of plaintext data using the algorithm and its transmission and/or storage (i.e. the algorithm has been made publicly available for public scrutiny). Since the intermediate stages are hidden in the signal, it is assumed that the iterated encryption function has a secret parameter, which is the encryption key.

A cryptographic strength-enhancing property is one that an encryption algorithm is created to have. On a general level, this property must be random, unpredictable and complex. These characteristics include making sure the time series are evenly distributed hence it must achieve maximum entropy with no bias towards any trajectory. It must possess a high positive Lyapunov exponent to ensure the trajectory is chaotic after a few iterations. It must also have a high cycle length, and a good diffusing properties such that different keys and any bit change produces different ciphers in which all bits of the random sequence have an equal probability of changing their state [37].

A perfect pseudo-random number generation can be used to give perfect security if the ciphertext is fully unpredictable to a cryptanalyst or attacker. That is, if all possible outcomes, such as states and sub-trajectories, are equally likely and independent of one another. This is so because there won't be any correlations in the random sequence and a uniform probability distribution exists for the state sequence. Absolute unpredictable behavior is identical to the true randomness of white noise generated by background noise. In comparison to a system that delivers perfect security and is (theoretically) flawless, the data security provided by cryptographic systems in fact is often far lower. This is somewhat due to the need to develop encryption methods that can actually be utilized in practice and primarily due to financial considerations.

In this context, dealing with pseudo concepts in which uniformly distributed noise

cannot be distinguished from pseudo-random number sequences which are computationally unpredictable using available computing resources. For the purpose of developing the theoretical frameworks for developing and evaluating encryption algorithms, a number of notions that need to be quantified are involved. These include:

(i) An algorithmic complexity measure that takes into account the length of the shortest algorithm to produce a cryptographic sequence in which, at least intuitively, the system's internal complexity creates unpredictability;

(ii) randomness embedded in an algorithm, where the random output sequence has no discernible matching patterns or redundancies, is computationally unpredictable and is identical in length to the input sequence. A diagrammatic illustration of the relationship between these concepts shown in Figure 4.1.

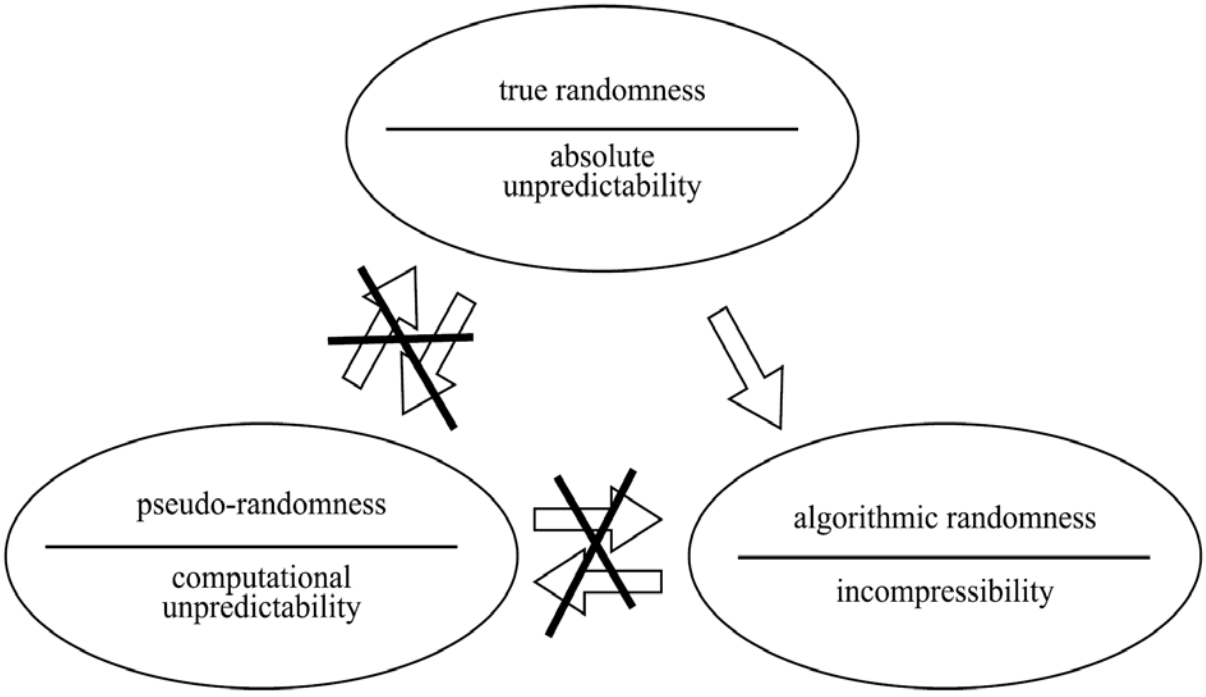

Figure 4.1: Relationship between the concepts of real randomness, algorithmic randomness and pseudo randomness [37].

From Figure 4.1, it can be noted that a purely random system is algorithmically random as well. However, there is a difference between pseudo and algorithmic randomness. Using a compact seed, a pseudo-random string is produced, but an outside observer cannot practically reconstruct the generator or predict the random sequence. In other words, the string is computationally predictable and incompressible for the attacker but readily compressible for those authorized to communicate. On the other hand, a probabilistic computer can, in the general situation, predict an algorithmic random sequence.

A string's randomness can be 'measured' using characteristics like algorithmic complexity or entropy. Entropy is considered to be a measure of the uncertainty about the precise state of any element in the random sequence and where this measure is the same for all components in the string. In ergodic systems, where statistical features of a single sequence are identical to those of all sequences, released by a PRNG, the Shannon information entropy is quantitatively directly proportional to algorithmic complexity.

The Kolmogorov-Sinai entropy, which [28] cites as the randomness indicator for chaos is a multi-resolution integration of the exponents of Lyapunov. Practically, entropy is increased if the Cipher generates output that is evenly dispersed or can be produced after being processed, a fresh output sequence that is evenly dispersed and doesn't have a lot of redundant info. In contrast with a totally predictable system, in which all states are known and has a complexity of 1, a totally unpredictable system, for instance, delta uncorrelated white noise, provides an infinite supply of information, entropy and complexity. Real cryptographic systems are halfway between these theoretical extremes. They are complex enough to be unpredictable to an intruder or attacker but not too complex to be reproducible. These are the theoretical extremes. There are two main types of stochastic fields that can be taken into account in this context, and they are covered in more detail in the following sections.

4.3.1 Natural Noise

Natural noise has infinitely many states and independent variables. It can be found in a variety of environments and sources. However, generally speaking, due to its self-organizing characteristics and correlations, such a system may not have the maximum amount of entropy. This is due to the fact that many noise sources are random self-affine strings. Most natural noise sources have well-known and essential fractal geometric features. Natural noise is used in some cryptographic applications; for instance, Intel's hardware-based PRNG extracts randomness from the computer's thermal noise. Because they cannot be replicated, such random sequences are exclusively utilized in key generation and not in encryption.

The utilization of natural noise can be seen in a significant historical case. The SIGSALLY (Green Hornet) encryptor created by AT & T Bell. The encryptor was used by President Franklin D. Roosevelt and Prime Minister Winston Churchill for 1-to-1 trans-atlantic communications from 1942 until 1946 [28]. The encryptor worked by scrambling speech signals across all frequency bands by adding noise to voice signals in order to create an analog output signal with a low signal-to-noise ratio. The source of the noise was identified as a vacuum tube by recording the output of the tube on a phonograph recording 78 disc. The output was a recording of electronic noise that was mixed with a real-world electronic noise source to conceal the voice signals.

The time registration connected with the addition and subtraction of the noise source in a two-way sense was the technology needed to put this strategy into effect. For obvious reasons, the dispersal of these noise sources or the recorded media, was rigorously regulated. However, the system represented a one-time pad and was practically hard to attack successfully as long as this 'regulation' was intact. Even in the present day, it would be challenging to break such a cipher using Bayesian techniques. For instance, a cryptanalysis based on the hypothesis that a statistical model for the probability density function (PDF) of the additive noise can be obtained and/or on the assumption that the noise is fractal, hence a model for the power spectral density function (PSDF) of the

cipher.

4.3.2 Natural Chaotic Noise

Low-dimensional chaotic noise has an unlimited number of states but a finite number of independent factors. Such systems, however, cannot be directly applied to digital encryption since they cannot be implemented on a finite-state machine. Only an iterative function can be used to approximate a chaotic system. This function must create a string that is repeated after a finite cycle length (or period) and finite floating point precision. All PRNGs utilized by a computer share this characteristic. The objective of applications to Cryptography is to find the best numerical implementation of a chaotic system that maximizes the cryptographic strength of the cipher while imposing the least amount of computing (algorithmic) complexity.

4.4 Pseudo Chaotic Encryption

The use of pseudo chaos in encryption design is now well recognized, accepted, and prevalent. This strategy has its roots in Claude Shannon's explicit statement of the fundamental stretch-and-fold mechanism, now connected to chaos and employed in cryptography, in the early 1950s. After that, there was a period of silence until the late 1980s, when emphasis was placed on implementing standardized, commercial symmetric and asymmetric encryption algorithms. Examples of these include the Data Encryption Standard (DES), which was later modified to the triple DES (3DES) by encrypting with a key $_1$, decrypting with a key $_2$, and then encrypting again with the initial key $_1$. This triples the length of the operational encryption key without necessarily augmenting the algorithm and the Rivest, Shamir and Adleman (RSA) algorithm. The RSA requires a Public Key Infrastructure (PKI) to be operational. It is mainly used for public key exchange and involves a trusted third party such as a Certificate Authority (CA) to sign the public keys in order for them to be trusted by all communicating entities.

Post the chaos theory popularization in the 1980s, it began to be used in Cryptography in the 1990s, when the majority of research recommended a number of ciphers focusing on the use of analog circuits for real-time applications in spread-spectrum-based military operations. For example, [28] employed fractal modulation to conceal the spread-spectrum amid background radio frequency noise. Although several chaotic maps have been proposed and multi-algorithmic techniques for data encryption on a randomized block-wise basis have been developed, the use of discrete digital chaos for data encryption has increased significantly since 2000 [37].

Chaos has a number of drawbacks when used for cryptography. This is particularly true when the cipher must be computed with high floating point accuracy and is subjected to the application of a partitioning technique to the state space in order to give a maximum entropy random sequence. This is a mandatory pre-process stage which is meant to generate redundant floating point inputs. This step significantly wastes processing time. Chaos' main advantage is its capacity to generate a wide variety of algorithms. It is feasible to achieve this with traditional random number generators, such as the Knuth M-algorithm, although chaos offers a broader variety of functions than, for, the standard mod function.

One of the problems with chaos based ciphers is that in order to produce a library of different algorithms, they have to be designed by hand often by modifying specific and well known chaotic iteration functions such as the logistics map [28] or by 'trial and error', i.e. inventing non-linear Iteration Function Systems and testing them for their properties with regard to cryptographic strength.

One of the issues with chaos-based ciphers is that to create a library of multiple algorithms, they frequently need to be manually created by altering well-known chaotic iteration functions like the logistics map [28] or by trial and error, which involves creating non-linear iteration function systems and evaluating them for their cryptographic strength.

All changed and/or created maps must pass the following tests for cryptographic strength, among others:

(i) A large, positive, Lyapunov exponent in relation to an acknowledged cryptographic algorithm, with an acceptable cryptographic strength, such as the AES [28];

(ii) the possibility of producing a cipher with a perfectly uniform discrete Probability Density Function (PDF), or more specifically, can the results of a chaotic map be divided to do so?

(iii) the distribution of the Power Spectral Density Function of the cipher is uniform, resulting in a redundant spectral attack;

(iv) The cipher's auto-correlation is a delta-type function. This indicates that no relationships exist between the length of the cipher set to be used for encryption and, consequently, the period is greater than the set upper limit, thus, preventing cyclically correlated patterns from being generated in the random output stream, when the cycle length is reached;

(v) Given the hardware, the clock cycle (CPU time) needed to compute floating point arithmetic, usually for double precision is reasonable and acceptable.

The listed points (ii), (iii), and (iv) are the most crucial. This is because an infinite and true random string has an endless and uniform power spectrum. It exhibits no statistical bias i.e. white noise. Assuming the presence of one-way, one-to-one functions, for instance, point (ii) can have probability distributions that are not uniform and are not even statistically near a uniform distribution. However, they could be computationally equivalent to a uniform distribution [112]. Therefore, it is essential to verify that the states have an equal probability. Given that the iteration function it is assumed to characterize would produce chaotic trajectories within a few iterations, a high and strictly positive Lyapunov exponent is preferred. The diffusive characteristics of a cipher, notably that the PRNG is 'Structurally Stable' are not, however, guaranteed by these tests.

There is also the problem of algorithmic complexity, which cannot be solved globally, making it impossible to simplify programs and demonstrate that their length is minimal.

This definition does not allow us to directly compare the difficulty of cryptographic algorithms or sequences. However, the theoretical applications are also crucial. In particular, the Kolmogorov complexity offers a cohesive strategy to the data compressibility problem [28].

Despite the fact that the applications of 'digital chaos' have created solutions that are economically viable, they are not scalable due to these substantial and unanswered issues. This chapter explores the use of natural noise as an evolutionary process' input to scale up a process using evolutionary computing. In this approach, the strategy under discussion looks into a way to automate and diversify the strategy in a manner similar to the Green Hornet transatlantic scrambling methodology. This is done in order to generate a potentially endless number of one-time-pads while utilizing a variety of noise sources.

4.5 Encryption using Chaos

If a safe means of key exchange is available, an encryption technique based on the Gilbert Vernam Cipher is known to be a secure way to encrypt data for one-to-one conversations [213]. This is a symmetric key encryption system. The Vernam cipher is a substitution cipher that works by creating a vector $\mathbf{x}$ out of a series of random values, $\mathbf{x} = (x_1, x_2, ..., x_N)$. This vector $\mathbf{x}$ depicts a pure noise-driven digital signal assumed to be a stochastic field. The issue is how to develop an algorithm that can be used by a computer to produce a suitable vector for data encryption using the Vernam cipher.

Assume that some input plaintext is converted from ASCII code to integers and then written in terms of a set of integer numbers, creating a plaintext vector $\mathbf{p}$. This plaintext vector normally consists of decimal integer numbers that comply to the ASCII for the natural language processing, however, any code can be used. The plaintext is typically taken to be the text associated with a natural language. In a broader sense, the plaintext vector $\mathbf{p}$ could include items that represent any input signal or image. For an 8-bit grey level image, the elements in the former scenario would normally be floating point values, whereas the elements in the later case would typically be decimal integers in the range of

0-255.

In any case, we may produce the ciphertext for a substitution cipher by by adding the two vectors together. $\mathbf{c} = \mathbf{x} + \mathbf{p}$. When the cipher is made up of N numeric (integer) values and the input plaintext data is based on natural language, using a 7-bit ASCII code, and consists of N elements, a ciphertext output can be given by Equation (4.1).

$$c_n = (p_n + x_n)\text{mod}(127),\ n = 1, 2, ..., N \tag{4.1}$$

where mod denotes the standard modulo operation. The following decryption equation is then used to recover the plaintext, as in Equation (4.2)

$$p_n = (c_n - x_n)\text{mod}(127),\ n = 1, 2, ..., N \tag{4.2}$$

Writing the integer plaintext and cipher vectors as binary strings is another approach of putting this encryption technique into practice (using ASCII text to binary representation). Using a binary space, the ciphertext is computed as follows: $\mathbf{c} = \mathbf{x} \oplus \mathbf{p}$. NB: $\oplus$ denotes the bitwise XOR operator. The decryption function which recovers the plaintext vector $\mathbf{p}$ is given by: $\mathbf{p} = \mathbf{x} \oplus \mathbf{c}$. In this case, $\mathbf{c}$, $\mathbf{x}$ and $\mathbf{p}$ denote finite binary strings of length $L > N$.

Regardless of the encryption technique used, the challenge is creating an algorithm, or multiple algorithms, to produce a cipher with characteristics compatible with strong encryption. These characteristics include guaranteeing that the vector $\mathbf{x}$ is statistically unbiased so that the cipher's histogram is evenly distributed. Also, $\mathbf{x}$ must have a uniform power spectral density function. The starting value serves as the key in most iterative cipher generation techniques. They generate pseudo-random number streams for which a number of crucial conditions must be satisfied. These conditions include the condition itemized in points (i)-(ii) listed in Section (4.4).

Numerous cipher-generating algorithms have been created that are based on the structure of PRNGs. They frequently belong to the following three classes:

(i) streams of decimal integer random numbers;

(ii) streams of floating-point random numbers;

(iii) random binary sequences.

Due to the computational efficiency of integer arithmetic, the computation of integer streams has traditionally received the most attention. The evolution of prime number-based cryptography can be attributed to the linkage of modular arithmetic with prime numbers, which has traditionally been the case. Examples include the Blum-Blum-Shub (BBS) cipher [44] is a PRNG first proposed in 1986 that is based on the Equation (4.3).

$$x_{n+1} = x_n^2 \text{mod}(M),\ M = pq,\ n = 1, 2, ..., N \tag{4.3}$$

NB: p and q are prime numbers. x_1 is the key (seed). The value of x_1 is co-prime with M. This implies that p and q are not factors of x_1 and cannot be 1 or 0. This algorithm uses a string of decimal integer pseudo-random values as input and output. Therefore it is an example of an algorithm that adheres to point (i) above.

The development of cipher producing algorithms that produce decimal numbers ran parallel to the limitations of the existing computing power, particularly the short processing time for processing floating point arrays. The creation of a new class of chaotic iterators based on non-linear iterations, which require high precision floating-point arithmetic, is now a reality through recent developments of fast floating-point (co)-processors and specialized real-time digital signal processors. It is crucial to remember that chaos-based ciphers apply to point (ii) above. Consequently, they take more CPU time to compute, by default.

The Vurhulst cipher is an illustrative example of a chaotic iterator. The cipher is given by [28]

$$x_{n+1} = 4 \times r \times x_n(1 - x_n),\ r \in (0, 1),\ n = 1, 2, ..., N \tag{4.4}$$

which, because both are quadratic iterators, has a synergy with the BBS PRNG. The beginning condition in this scenario is any floating-point value in the range (0,1), exclusive. However, given that x_n converges and then bifurcates again as r approaches 1, this iteration only offers full chaos when $r = 1$, which is prohibitive. For this reason,

Equation (4.4) is modified and can be formulated by considering the iteration [141], as in Equation (4.5).

$$x_{n+1} = 4(1+r)\left(1+\frac{1}{r}\right)^{r} x_n(1-x_n)^r,\ r \in (0,1), \tag{4.5}$$

Then, in addition to an initial value of $x_1 \in (0,1)$, a range of r values can be employed. These values convert the iterator to a two-parameter function. In this instance, the value r for a fixed value of x_1, the value x_1 for a fixed value of r, or both r and x_1 may be considered the key.

A well-known iterator was modified, in the Matthews cipher provided by Equation (4.5), to better suit its use in cryptography. A relatively straightforward physical model to analyze the dynamic behavior of a predator-prey scenario is the Vurhulst process. As a result, the Matthew's ciphers are built using a physical model that has been modified for a different use.

In Cryptography, the iterator's physical properties, which can apply to any nonlinear function, are what matter, not the physical model that goes along with it. Included are functions that can be arbitrarily created from nothing and then examined to determine whether they are appropriate for the creation of a cipher stream. In this context, we can consider a generic iterative cipher to be of the following form, as in Equation (4.6):

$$x_{n+1} = f(x_n); x \in (0,1),\ n = 1,2,...,N \tag{4.6}$$

where f is a nonlinear function that typically has a parameter or group of parameters whose numerical value (or range of values) must be determined *priori* to generate a suitable cipher $\mathbf{x}$. A database of numerous such non-linear functions might theoretically be created and then used to encrypt random data blocks or for individual communications. These functions may be pure inventions or variations of well-known chaotic maps created to study particular physical (feedback) models of chaos. To put such a strategy into practice, however, there are a number of concerns that must be understood. The next sections address these concerns.

4.5.1 One-way Functions

The application of Equation (4.6) to Cryptography assumed that the function f is a one-way function. This means x_1 (the key) cannot be derived from the knowledge of $x_n,\ n = 2, 3, ..., N$. However, there are conditions and certain functions where this is not the case. As an example, when $r = 1$, Equation (4.4) has the following analytical solution [168]:

$$x_n = \sin^2(2^n \sin^{-1}(\sqrt{x_1})),\ n = 2, 3, ..., N \tag{4.7}$$

Thus, for any $n \geq 2$, there exists an inverse for the equation to obtain the key x_1. Thus, the key can be derived from knowledge of any higher order iteration. This is obviously unacceptable when developing a cipher generating algorithm. Synergy exists between this outcome and Equation (4.3), as long as this equation has an analytical outcome given by Equation (4.8):

$$x_n = [x_1^{2^k \mathrm{mod} C(M)}] \mathrm{mod}(M) \tag{4.8}$$

NB: mod is the standard modulo operation, C is the Carmichael function [78]. Because of this, it is necessary to demonstrate that the nonlinear function is a one-way function, thus, it does not have an invertible equivalent analytical solution, in order to employ an iteration of the kind provided by Equation (4.6). It should be noted that this argument is based on the supposition that the cipher can be decoded from the ciphertext c_n provided by Equation (4.2). This implies that the encryption method is known and that some elements of the plaintext are known or can be deduced by using the right Cribb.

This highlights both the importance of developing a one-way function to construct a cipher and the relevance of changing the function itself for encrypting plaintext. Moreover, an additional layer of security can be added by translating the plaintext from one natural language to another in order to avoid using Cribb's if there is a significant difference between the two languages, such as between an Indo-European language like English and a non-Indo-European language like Chinese. With the aid of natural language interpreters

like Google Translate, this is possible. Translation can still generate uncertainty into some communications, though. Before encryption, the effect is noticeably stronger if many translations are employed.

4.5.2 Equal Probability of States

The statistical distribution of the cipher $\mathbf{x}$ is rarely uniform no matter what nonlinear function is used, altered, or created. The vector needs to be post-processed to produce such a uniform distribution. To achieve this, window all of the vector's values that follow an even distribution. However, since many computed floating-point values may need to be discarded, this wastes both the data and the processing time needed to generate it. This is a significant issue because, although the cipher itself may be computationally indistinguishable from real-world chaos, presuming the existence of one-way functions may result in uneven probability distributions. Therefore, it is essential to verify that the states have an equal probability. Implementing a partitioning technique to produce a random binary sequence as the output is one method of enforcing this criterion, as illustrated in Figure (4.2), which depicts an histogram for Equation (4.4), uniformly distributed for $x_n \in \approx [0.3, 0.7]$. A random binary sequence/string can therefore be created by generating a 0 when $x_k \in [0.3, 0.5]$ and a 1 when $x_k \in (0.5, 0.7]$ or visa-versa. In this manner, a maximum entropy bit stream cipher that encrypts data using the common XOR operation is generated. Keeping all floating point data is another partitioning method, within the range $0.3 - 0.7$, and discarding all other values, and to construct a random number stream based on this range alone, re-normalizing and quantifying the stream to integer form for application to be applied in Equation (4.2).

The nonlinear functions of the 'Maps' supplied in Table (4.1) which determines the form of the function given in Equation (4.6) are examples of such ciphers. These are examples of stream ciphers that can be created by entering a key ($x_1 \in (0, 1)$, optimizing their output ($\mathbf{r}$ value) to create a chaotic number stream. This then results in using post-processing to create a stream with a uniform distribution from this output [169].

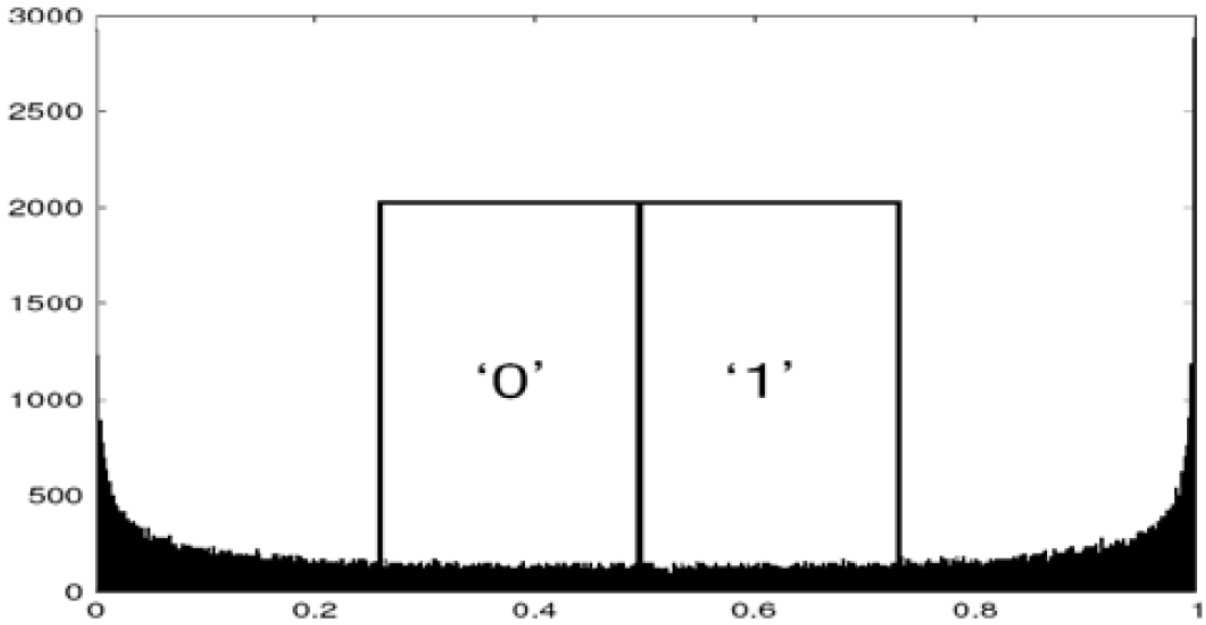

Figure 4.2: Histogram of the output for a Vurhulst process given by Equation (4.4) and the partitioning strategy used to generate a maximum entropy binary cipher[37].

Though they may display chaos, not all invented maps are suitable from a probabilistic perspective. For example, the Tangent map $f(x) = rx|1 - tan(sin(\mathrm{x}))|,\ r = 1.5$ has an extremely uneven distribution across all probability levels, making it impossible to condition it to generate a maximum entropy encryption using the partitioning method shown in Figure (4.2).

4.5.3 The Lyapunov Exponent

Equal probability states must also be guaranteed, and the nonlinear function must be described by an iteration in which the Lyapunov exponent is positive. This will ensure that the iteration is chaotic rather than convergent [24]. For a lengthy list of numbers, $x_1, x_2, ..., x_N$, Equation (4.9) provides a helpful measurement of the Lyapunov exponent.

$$\lambda(x_0) = \frac{1}{N}\sum_{n=1}^{N} \log \mid f'(x_n) \mid \tag{4.9}$$

Table 4.1: Stream Cipher Generation from Chaotic Maps

Map	**Nonlinear Operation**
Generalised Tent Map	$r(1- \mid 2x-1 \mid^{\alpha}),\ \alpha > 0$
Quadratic Feedback Map	$rx[(1-x)(1+x^2)]$
Generalised Sine Map	$\mid \sin(\pi r x^{\alpha}) \mid$
Tangent Feedback Map	$rx[1-\tan((2x)^{-1})]$
Logarithmic Feedback Map	$rx[1-x\log(x+1)]$

where $f'(x_n$ denotes the differential of the function at $x = x_n$. For chaotic behaviour it is required that $\lambda(x_0) > 1\ \forall\ x_0 \in (0,1)$. The Lyapunov exponent should be high but strictly positive since otherwise, the iteration function it is used to characterize will produce chaotic trajectories within a few iterations. Therefore, ideally, we need a nonlinear function where $\lambda(x_0) >> 1\ \forall x_0 \in (0,1)$.

The nonlinear function in this situation must also produce an iteration with a lengthy cycle duration. When it comes to the metrics mentioned, the Lyapunov exponent and cycle duration are really measured in relation to some known iteration that has been deemed to be cryptographically secure *a priori*. These problems are challenging to analyze, thus numerical tests must be conducted, including assessing the processing time, which must not be excessive.

4.5.4 Multi-Algorithmic Encryption

The following forms the foundation of multi-algorithmic encryption:

(i) The average orbit length of chaotic systems can be increased by connecting them together, which creates a single chaotic system with a wide state space and stable orbits;

(ii) For every encryption session, a distinct collection of chaotic systems may be used. An iterated function set that serves as a session key and comes with the key can be

used to do this;

(iii) to improve bit independence, the output bit can be produced after every *qth* iteration;

(iv) complex permutations of chaotic systems are possible, especially in the order in which they are used or 'switched ON' by a key.

Chapter 10 goes into more detail about how the models covered in this chapter can be used to implement chaos-based encryption. The implementation of the encryption models presented in this chapter is highlighted in the next part to demonstrate their viability.

4.6 Practical Implementation - *Crypstic*

A crypto-system called *Crypstic* was developed as a prototype to evaluate the viability of putting the encryption techniques covered in this chapter into practice. It is called *Crypstic* because it can be loaded onto an USB stick. As stated in Section (4.5.4), the system is based on numerous chaotic systems bearing extended features (i)–(iv) above [64]. The system provides a series of redundant systems by resolving issues with floating-point arithmetic. An encryption engine can really be built on any number of algorithms, each of which has been engineered to meet the necessary performance requirements (i.e. maximum entropy) by implementing proper conditioning parameters. The stream ciphers listed in Table (4.1) served as the foundation for the product's initial iteration.

In standard encryption systems, a Graphical User Interface (GUI) is typically provided with fields for entering the plaintext and exporting the ciphertext, where the user specifies the output name (with file extension). By overwriting the input file, *Crypstic* produces the ciphertext. This enables the encryption engine to be initialized/seeded using the file name. Therefore, for the decryption process to succeed, the file name must not change. The session key is initialized using the seed. The file name is changed into an ASCII code, 7-bit decimal integer stream. The stream is concatenated, and used as the seed for

a hash function whose output is a 5-tuple, (d, d, f, f, f) where d is an integer value and f is a 32-bit precision floating point number in the range (0,1).

A .dll extension is used to mask the executable file. This .dll file is contained in a folder with numerous other similar .dll files. This is because the architecture of .dll and .exe files are similar. However, this necessitates that the source code be written in a way that renders void any references to its application. This includes all references to the type of data processing involved, such as words like Encrypt and Decrypt strings that are changed to E and D, respectively, in the GUI. Therefore, the compiled file, despite being disguised as a.dll file, is forensically resistant to attacks that can be carried out with computer forensics tools like WinHEX [222]. The creation of a run-time support facility must be part of this. The goal of this strategy is to create an executable file that is forensically inactive and is obfuscated by the environment in which it is placed, therefore it is obvious that such criteria are at odds with the convention connected with the development of applications. This is based on the software engineering methodology known as forensically inert. Figure 4.3 displays the fundamental GUI linked to the execution of *Crypstic*.

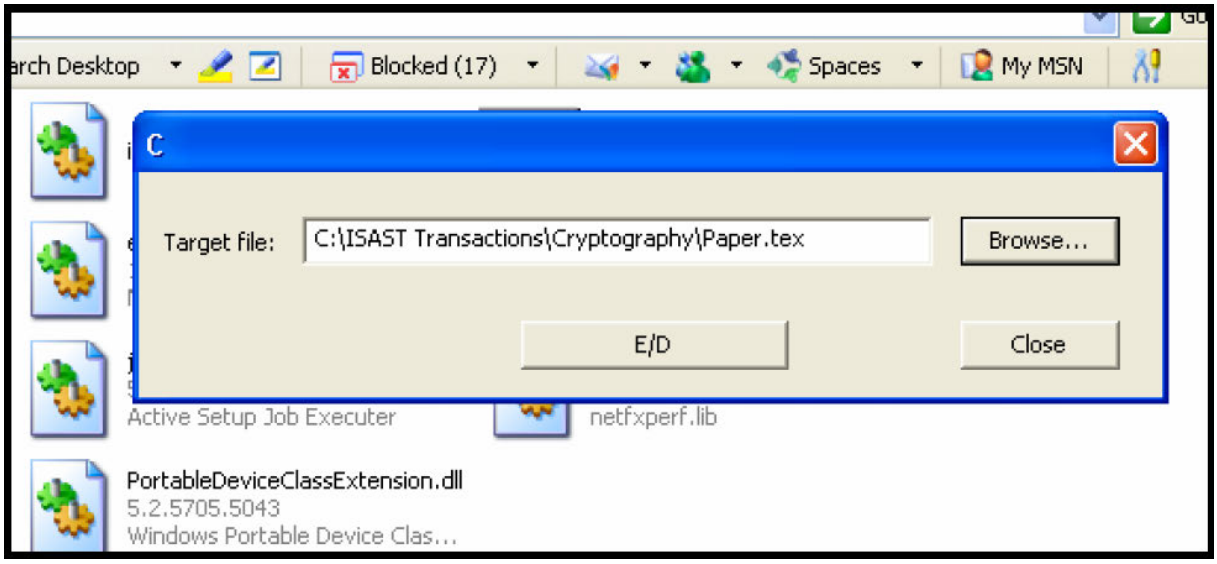

Figure 4.3: *Crypstic* App File Input GUI

Crypstic is a symmetric encryption system that relies on the user using a USB memory stick and adhering to a protocol that mimics the use of a standard set of keys, which are

generally kept on a key chain. *Crystic's* most basic application is the issuance of *Crypstic* to a single user, which includes a proprietary encryption engine using a unique set of algorithms. After applying a compression method such as zip, for example, the user can use the Crypstic to encrypt/decrypt files and/or folders on a computer before ending a session.

In this method, the user keeps their environment secure by employing a special encryption engine and a secret access channel known as the 'key'. A new pair of sticks with new encryption engines that are specific to both parties are supplied if any *Crypstic* by either side is lost. *Crypstic* can be distributed to user groups in a fashion that provides the necessary access control in addition to a two-party user system.

Current discussions on cloud computing make the assumption that users who rely on third-party hosting for their servers will not see many changes. It seems that common security measures will offer adequate security in the future. These presumptions disregard the generally known belief that cloud computing lacks security. For users who rely on third-party hosting for their servers, this perception is constantly reinforced in the user's mind by the increasingly slow and difficult anti-malware software needed and the frequent news stories about significant security breaches, frequently committed by adversarial governments. Additionally, it seems that common security measures will offer adequate security in the future.

These presumptions disregard the generally known belief that cloud computing lacks security. The user's sense of this is continuously reinforced by the need for anti-malware software, which is becoming slower and more complex, as well as by frequent news reports about significant security breaches, frequently committed by adversarial nations. To maintain their competitive position and to attempt to slow down product cycle degradation, the majority of enterprises rely on some proprietary know-how, process, design, or other commercial secret. Businesses are acutely aware of the need to avoid fixed and capital costs in order to lessen their sensitivity to volatility, particularly in the present. Although cloud computing is a capital and fixed cost-free approach, the perceived lack of security for commercially sensitive data is a significant deterrent to switching from

in-house information and communications technology solutions.

Cloud computing has expanded quickly and is predicted to do so significantly more in the future. Regarding price and cost, change management, next-generation architectures, choice, and agility etc., cloud computing brings a lot of advantages. However, users of the Cloud are mostly concerned with two things: a lack of control and, more significantly, data security. In this situation, *Crypstic* and its future derivatives take into account a method of encrypting data before it is uploaded on the Cloud. This way, every user has access to their own, personalised encryption algorithm, hence the multi-algorithmic encryption discussions in Section (4.5.4). These algorithms are based on a set of iterative function systems that output a chaotic number stream and are intended to create a cryptographically secure cipher. This is done by developing a method based on multi-algorithmic cryptography that especially takes advantage of the characteristics of pseudo-chaos or finite-state approximations to chaos. In order to comply with the Kerckhoff-Shannon principle, which states that 'the enemy knows the system', such algorithms can be assumed to be in the public domain. However, their combination can be used to secure data in a way that is particular to each user. As a result, cloud tenants have the option to upload and move data knowing that it is end-to-end encrypted in a manner that is algorithm and key dependent, thus, thwarting a known algorithm attack.

4.7 Discussion

Chaos and encryption share a basic connection. In both situations, the goal of the study is to create a dynamic system that transforms data in an unpredictable but deterministic way over time. It is conceivable to ensure cryptographic confusion and diffusion with the right entropy conscious post-processing, given chaotic systems' sensitivity to beginning conditions and their mixing characteristics. Chaos theory and cryptography, however, vary conceptually in a number of ways. The discussion of these variations follows..

(i) Chaos theory typically deals with the study of dynamical systems defined on an infinite state space, but cryptography depends on finite-state machines and all chaos

models run on a computer are approximations. This means that computers can only produce pseudo-chaos.;

(ii) Chaos theory investigates the asymptotic behavior of nonlinear systems, while cryptography deals with the effects of a finite number of iterations normally determined by the size of the input plaintext message (i.e., system behaviors when the number of iterations approaches infinity, and the Lyapunov dimension can be quantified);

(iii) Chaos theory is more interested in the interpretation of a physical model from which it has been formed than it is in complexity, as opposed to cryptography where complexity is the main issue. As a result, chaos theory does not have any analogs for the concepts of cryptography security and efficiency;

(iv) In contrast to how cryptography seeks to eradicate any structure by post-processing the output to create a maximum entropy encryption, classical chaotic systems typically include observable attractors.;

(v) Contrary to chaos in general, cryptographic systems use a variety of independent factors to create an output that is unpredictable to a viewer (i.e. attacker or cryptanalyst);

(vi) In contrast to Cryptography, where the physical model is irrelevant, chaos theory is frequently related with the mathematical model used to quantify a physically significant problem.

The importance of point (vi) in the development of chaos-based encryption engines cannot be overstated. While certain studies in this area e.g., [141], [24], [20] and [11], in this thesis, we have thought about the possibility that a designer may theoretically 'create' an endless number of systems to offer an infinite variety of multi-algorithmic encryption engines. This is a twist on the concept of well-known chaotic systems, which has not been tested, to the author's knowledge. Comparing this approach of data encryption to single algorithm based ciphers that are publicly available, a 'paradigm shift' is provided.

However, there are a few important factors that must be taken into account when creating such ciphers, and they are described below [216].

4.7.1 Structural Stability

The structural robustness of the PRNG, which is one of a cipher's diffusive properties, is not guaranteed by numerical tests. The optimal iteration must have almost the same cycle duration and Lyapunov exponent for all beginning conditions x_0. The bulk of known pseudo-chaotic systems lack this property, and there is currently no rigorous analytical method that can be used to assess it. Without resolving this issue, it is impossible to ensure that any or all encryption keys for a crypto-system based on a deterministic chaotic algorithm or combination of algorithms would always result in uncorrelated strings, or ciphertext.

4.7.2 Computational Unpredictability

Another issue is determining what aspects of a chaotic system guarantee its computational unpredictability, or how unpredictable it is. This challenge is related to the intractable issue of algorithmic complexity; there is no universal solution for streamlining programs and proving their short duration. We are unable to directly compare the complexity of cryptographic algorithms or sequences using this notion. But it's essential to consider the theoretical applications. The problem of data compressibility is especially well-solved by the Kolmogorov complexity. Subject to these important and as of yet unresolved challenges, 'digital chaos' applications can yield economically realizable products, but not necessarily products that are scalable in terms of commercial capacity.

4.7.3 Computer Generated Ciphers

With the idea that the function would only provide an approximation to the data, it would be preferable to devise a mechanism where such a function could be generated from genuine stochastic data. This is due to the fact that the issues with developing

a proper nonlinear function have already been covered previously. To do this, we use evolutionary computing, a sub-field of machine learning and a component of artificial intelligence. The next chapter shows how non-linear functions suitable for encryption can be produced automatically by a computer rather than from scratch, and this is the main theme of the chapter.

Chapter 5

Cipher Generation using Machine Learning

In the preceding chapter, it was discussed how deterministic chaos can be used to create encryption ciphers. This chapter examines recent developments in artificial intelligence-based cryptography (AI). The uses of machine learning (ML) and evolutionary computing (EC) for data encryption are explicitly taken into account. In connection with the discussion from chapter 3, a brief explanation of Artificial Neural Networks (ANNs) and the fundamentals of Deep Learning using Deep ANNs is provided. In this context, the application of EC and ANNs for creating distinctive and unclonable ciphers is the special focus.

As a practical application of the ideas covered in this chapter, an application is provided that deals with the verification of valuable documents like bank notes using a smartphone.

This involves reading (in the near field) a flexible radio frequency tag that is coupled to an integrated circuit with a non-programmable coprocessor using the antenna of a smartphone. The coprocessor stores extremely secure encrypted data created using EC that can be decrypted online, validating the document's (or another object's) authenticity via the Internet of Things and a smartphone.

5.1 Introduction

The term Artificial Intelligence (AI) refers to a variety of approaches and tools created to let computers do jobs that are typically the purview of human intelligence. AI is defined as 'the capability of a digital computer or a robot under computer control to accomplish functions typically performed by intelligent beings' [61]. The Turing Test, created by Alan Turing in 1950, serves as the foundation for the main AI test. This test determines whether a machine can behave intelligently in a way that is comparable to or uncannily similar to that of a human [210]. Applications of AI nowadays include, but are not limited to, speech recognition, finance, avionics, navigation, gaming, robots, and medical. This chapter focuses on the use of artificial neural networks and evolutionary computing to produce distinctive and unclonable non-linear ciphers using real-world noise sources. The main goal is to solve the issues raised in the previous chapter regarding manually designing nonlinear functions.

5.1.1 Machine Learning

Machine Learning is a crucial subcategory of AI. ML is frequently linked to the pattern recognition issue. In this case, it is necessary to classify a complicated data collection of potentially irregular patterns in a signal or an image, for example, into shared features and/or segments that may subsequently be classified in a predetermined manner. Statistical measurements derived from signals and/or statistical and/or geometric metrics for images are often connected to these classifications. A judgment can be made by applying a threshold to create a decision-making criterion if a cluster of such metrics within a given numerical range is sufficiently different to be associated with known features in the data..

Finding the ideal threshold to achieve this is frequently difficult because the ideal threshold value is dependent on a confidence interval, which affects how accurate the judgment made will be. The decision-making process' logic can be made softer in terms of its tolerance for the data by modifying the threshold to demarcate particular input metrics based on their known accuracy, quantity, and other prior knowledge (and associated

errors). This is the basis for putting into practice so-called 'Fuzzy Logic Systems' which serve as the building blocks for the creation and application of artificial neural networks (ANNs).

When classifying a pattern, an ANN often improves accuracy above what can be attained using traditional data categorization (based on a logical and/or fuzzy logical quantification). The foundation for this is examined in the section that follows.

5.1.2 Artificial Neural Networks

A basic artificial neural network (ANN) relies on the input of a class of metrics that are assumed to be a composite representation of the data, metrics that are produced by processing the data to form a so-called 'feature vector'. The significance of each component of this vector is then calculated (by multiplying each component of the vector by its associated weight) and the weighted components are joined together to give a single output value. The ANN produces an output in response to a decision-making process by altering the values of these weights.

This is an illustration of a data transfer mechanism that merges several inputs into a single output. By entering a similar set of weighted feature vectors or by altering the vector's input components, it can be repeated to generate a variety of outputs that are either the same or different. As a result, a 'single layer' of outputs is produced, which can be considered an input to a 'hidden layer' where the weights are calculated and modified once more. We can create a multi-layer network with n layers by repeating this procedure n times.

When transferring information (data components) from the hidden layer to the output layer, computations are assumed to include changing the numerical value of all or some of the applied weights. This is an illustration of a 'feed-forward network' in which data only moves forward, from input nodes through hidden nodes and out to output nodes. The network doesn't contain any loops or cycles. When creating any such network of input-output devices, the following four difficulties must be taken into account:

(i) What should the feature vector's size be to define the number of nodes connected to the input layer?;

(ii) What number of outputs are needed in the output layer?;

(iii) How many nodes should be present in each layer and how many hidden levels should be used?;

(iv) How can the initialization and/or subsequent adjusting of the weights be automated?

To address point (iv), it is important to create an algorithm that automates weight modification under the condition that the predicted result(s) in the output layer are known. In order to update the weights by comparing the outputs they produce with 'target' data, training data must be given.

An ANN can be trained using a variety of various but related techniques (algorithms) by changing the weights until the output for a particular input is the same as the training data (within a specified tolerance). Utilizing the back-propagation algorithm [125] for feed forward networks is the original and possibly most popular method. The Gradient Descent Method, an iterative optimization procedure for locating a local minimum of a differentiable function, is the foundation for this. The weights can be changed to produce an output that closely resembles the training data by computing the gradient of a loss function with respect to each weight (which is assumed to be a discrete vector) and repeating the process.

The approach, which is an illustration of dynamic programming, has a synergy with the use of the least squares method to solve a system of linear equations (particularly overdetermined systems) while minimizing an error function. The gradient of the loss function with respect to each weight is calculated using the chain rule, and a threshold function is then used to decide whether or not the value from one node should 'flow' further (i.e., continue on to the next node(s) in the next layer).

For the creation of AI applications, a variety of systems and AI software are available, including [50] and [104], respectively. For the development of fully engineered systems,

MathWorks MATLAB, for instance, offers specialized Toolboxes with capabilities that integrate AI into the entire workflow [138]; and machine learning with supervised and unsupervised learning [139].

Python programming is being used often to create AI systems [116]. However, the design of a network is crucial and must be customized to improve the application of AI using an ANN for a particular solution, regardless of the system, programming language, or tool-set employed. The training data's quality and amount are particularly significant. This establishes the precision of the weights, which, along with the network's architecture, serve as the keys to the network's future functionality. Since the weights are comparable to the key(s) and network architecture in cryptography, there is an obvious application where the output cipher generated by the encryption method can be employed. to encrypt plaintext data.

5.1.3 Data Processing and Deep Learning

As discussed in the preceding section, it is usual to first process the data to create a feature vector with metrics that are thought to be an accurate depiction of the data's fundamental properties, such as a digital signal. As a result, there are comparatively few nodes in the input layer when compared to the original data, or the length of the digital signal. This is crucial for making the most of the inevitably constrained processing resources that may be used to 'drive' an ANN and develop an effective decision-making process (e.g. the computational time required).

The feature vector pieces that represent good (unique and clear) properties of the data, however, can sometimes be quite challenging to characterize in a fully quantitative sense. Using newer analysis techniques and new data attributes, this issue is frequently solved. For instance, fractal geometry concepts and computing metrics like the Fractal Dimension and multi-fractal parameters can be used to quantify data texture. As a result, it is possible to quantify more perceptible aspects in the data using fractal geometry, where it is assumed that the data is self-affine [211] and defined by the somewhat elusive phrase

'texture' [43].

The size of the feature vector, which should ideally be much less than the original data from which it has been produced using a variety of different signal or image processing methods to output certain metrics, depends on what feature vector should be utilized and the elements it contains. The processing time needed to compute the feature vector in comparison to the time needed to train the ANN is an optimality issue that must be taken into account. However, in recent years, the amount of computational power made possible by hardware advancements has grown to the point where, instead of processing signals to produce a composite feature vector, this procedure is disregarded and the raw data (i.e., of the original digital signal) is utilised directly to 'feed' the input layer of an ANN. This method greatly increases the complexity of the ANN when combined with several layers made up of numerous nodes, but it eliminates the requirement to identify what signal processing procedures are necessary to build a smaller feature vector *a priori*.

This idea underlies a more contemporary approach to AI known as deep learning, which effectively abandons earlier signal processing techniques in favor of high data throughput ANNs made specifically to process raw data. In this context, Figure 5.1 shows how a traditional ANN with a single hidden layer, or a 'shallow' ANN, differs from a deep network with three hidden layers..

The deep learning approach is being driven mostly by the rise in computational power and effectiveness. Google's Inception-V3, for instance, relies on a network with 49 levels and more than 20 million links [60]. However, this is of little use unless there is enough training data to calculate the millions of weights presently needed. However, it is easier to collect the necessary training data online thanks to the rise in computing power and the Internet of Things' (IoT) exponential expansion. In this view, the development of the Big Data Society, which is increasingly pervasive, as well as the available processing performance, make deep ANNs conceivable. Companies like Facebook, Google, and the national security services, where problems like face recognition and track and trace, for example, are crucial, are the key industry drivers of the increase in computing power that is enabling a deep learning paradigm to emerge. This is made feasible by the fact that

supercomputers from ten years ago, which had thousands of processor units and were housed in vast buildings, can now be condensed into a single Graphical Processing Unit (GPU), which is needed for high resolution gaming, for example..

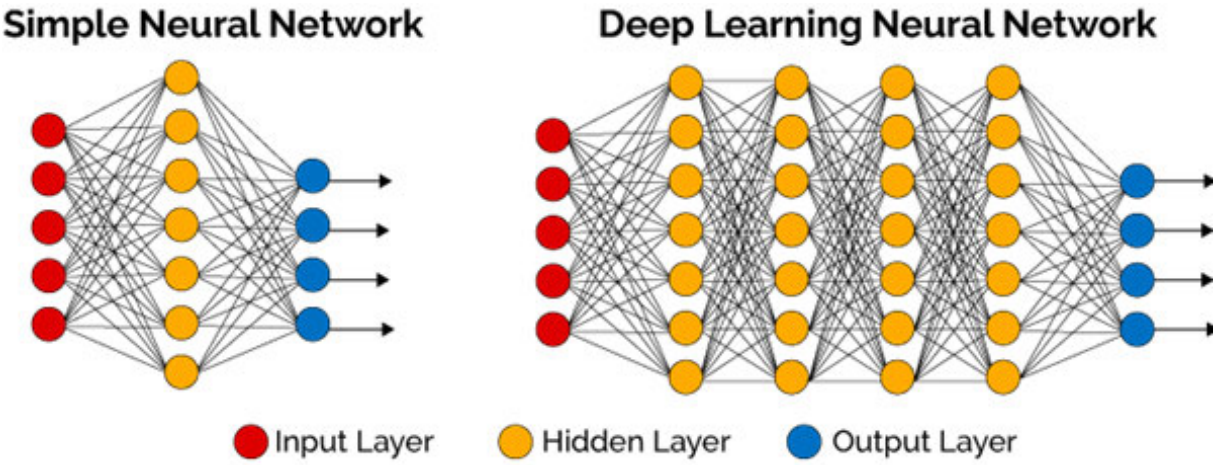

Figure 5.1: Example of a single layer 'shallow' neural network (left) that has a 5 -element feature vector with a single layer and a deep neural network consisting of 4 hidden layers, both networks consisting of a 4-node output layer [77].

Regardless of the available computing power, ANNs need real-world data to function in real-world settings. Big data access has made it possible to meet this need, however occasionally particular data may not be available or may not be present in areas of existent data. By instructing ANNs to generate the missing data using other relevant (real world) data that is already accessible, they can also be useful in this situation. As a result, AI not only benefits from the big data society, but it may also play a role in its development, expansion, and diversity (by creating even more data).

The design of an ANN, including the number of layers, nodes per layer, and homogeneity or lack thereof of the connections given, among other factors, can vary significantly when combined with the training data, just as can the specifics of the techniques used to compute the weights. This involves using various spatially invariant filters with the (discrete) convolution operation to diffuse the weights values over a layer and/or a partition through the layers to a particular depth. Early in the 1980s, Convolutional Neural

Networks (CNNs) were initially developed. They belong to a class of deep learning ANNs that have mostly been created for and used in the disciplines of computer vision and digital image processing [226]. Convolutional filters are being used in this situation to segment images into regions of resemblance and/or discontinuity (edge detection, for example), in place of several digital image processing methods that were initially created for pattern recognition [39]. This method has been used, and is still being used, to create a set of deterministic, geometric, and textural metrics that are customarily used to design a fuzzy logic decision engine, or to train 'shallow learning' ANNs with a limited number of inputs and hidden layers.

The future development of numerous technologies, programming environments, services, and systems—all of which are undergoing rapid development—requires the application of deep learning techniques to complicated pattern recognition issues. For instance, MATLAB has just released a new toolbox for applying deep learning to solve problems [140]. Beyond the scope of this thesis, deep learning's application to cryptography is not possible. However, some of the strategies examined in this thesis readily lend themselves to performance enhancements when combined with a deep learning strategy. This is especially true when it comes to replacing traditional cryptanalysis techniques, which are increasingly obsolete given some of the advances discussed in this thesis [134]. The role of ANNs in cryptography is described in the section that follows..

5.1.4 Artificial Neural Networks and Cryptography

The capacity of AI to identify patterns within large amounts of complex data is one of its key characteristics. In order to make plaintext into ciphertext as diffuse and confusing as possible, cryptography depends on this. The ciphertext, which is the encrypted data, should ideally be free of all patterns. The ciphertext must be a complicated randomized representation of the plaintext, which is typically connected with the use of a one-way function without an inverse solution. This opens up the possibility of using AI to generate ciphers, classify them based on their cryptographic strength, and analyze or perform

cryptanalysis on encrypted data.

Given that the foundation of cryptography is the creation of a random number field (the cipher), let's say we seed a network with a few initial random numbers and train it with a real source of noise, allowing the weights to be adjusted so that the network produces a reasonably accurate simulation of the target random data field. The initial random numbers and weights represent the key(s) needed to recreate the random number field that can be used to encrypt and then decrypt the data in this case, provided the network's design is known to two communicating entities, Alice and Bob. (Subject to both the initial random numbers and weights being known to Alice and Bob through application of a key exchange protocol).

The encryption/decryption algorithm is similar to the network architecture, including the derived weights. This method allows for the creation of the cipher's random number field from a small number of random elements in the input feature vector. Later in the thesis, this method is addressed and contrasted with another approach to the issue of creating an algorithm for generating ciphers. This is based on the usage of an evolutionary computing machine learning technique, which may be utilized to give a nonlinear equation that, upon iteration, can provide a random number field, with the (traditional) key as the initial condition.

Aiming to replicate a simple output subject to a complex input, AI seeks to compute a set of ideal weights through iterative procedures that control the flow of information (the amplitude of a signal at a given node). An ANN simulates a high entropy input in this way with the intention of producing a low entropy output. It is possible to reverse this process and produce a high entropy output from a low entropy input. In this manner, an ANN that has been trained to mimic natural noise can be utilized to construct ciphers. The cryptographer needs to be familiar with the ANN algorithm and the weights created during the training process in order to employ an ANN in this manner (i.e. the input of the noise sources used to generate the weights). Figure 5.2 demonstrates an example of the ANN generated output and input noise obtained from recordings of atmospheric noise [43]. It is essential to employ the right kind of ANN for this operation, and it has

been determined that a Radial Basis ANN is the most effective choice. This means that the weights are comparable to the key, and the particular ANN method is analogous to a PRNG in traditional encryption.

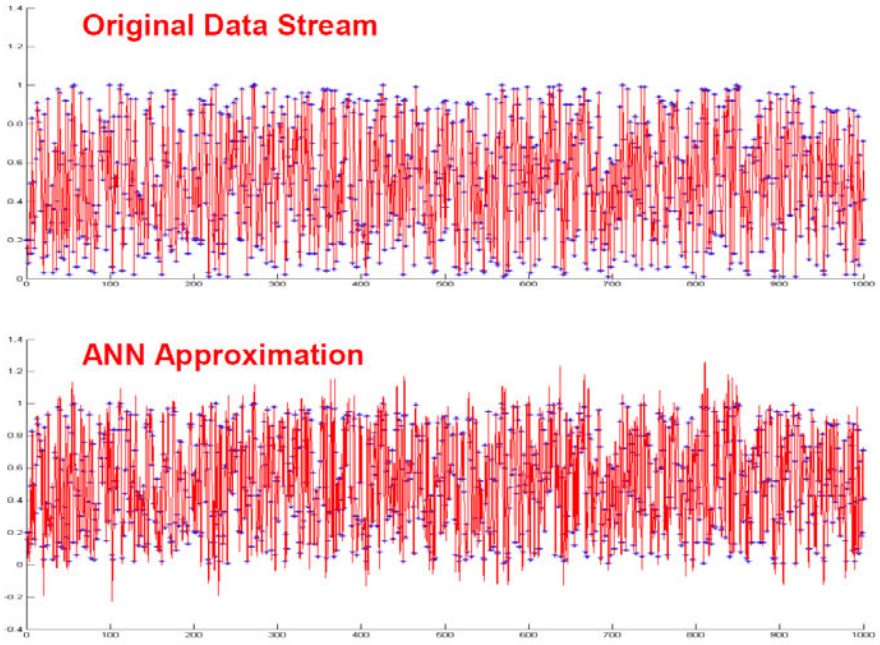

Figure 5.2: An ANN can be trained to mimic a real random number stream using the following example: both the original noise (above) and the ANN approximation (below).

In the same way that AI can be used to create ciphers, its capacity to recognize intricate patterns makes it ideal for the task of identifying patterns in ciphertexts and searching for signatures that indicate a location or point of attack where the randomized nature of the encrypted data has weaknesses. Given that a robust encryption engine shouldn't produce such flaws, AI offers an additional tool for evaluating the security of encryption techniques, including those that rely on AI to generate the encrypted data. Consequently, AI has a part to play in both cryptanalysis and data encryption. When it is necessary to identify the signature of a plaintext that is concealed within another plaintext (Steganalysis) or the signature of a ciphertext that is concealed within a plaintext, the latter situation can be extended to difficulties involving information hiding (Steganocryptanalysis). We

briefly explore the function of ANNs in cyber security in the section that follows..

5.1.5 Artificial Neural Networks and Cyber Security

A variety of strategies are used in cyber security to safeguard the production, processing, storing, and transfer of data across an open network, like the Internet. The infection of files by a virus intended to stealthily penetrate a single computer or a network of computers is a specific and well-known example of this. Computer viruses now work in a stealth mode to escape detection and have grown more sophisticated. Every day, new viruses are developed. However, the majority of these allegedly new viruses are actually variations, oftentimes rather modest variations on the themes of their predecessors.

To avoid detection, many of these viruses simply alter their appearance and signatures, but their functionality and modes of infection remain same. Assuming that future data streams and new files can be examined to identify the digital signatures of existing viruses, this offers the opportunity to train ANNs on a variety of known viruses [33][32].

5.2 Evolutionary Computing

A formulaic method to building a PRNG using iterated (nonlinear) functions does not offer the same flexibility as an ANN approach to producing ciphers, even though it can be useful in certain circumstances. After giving a general review of AI and its uses in cryptography and cyber security, this part presents a study on the use of evolutionary computing to produce original and unclonable ciphers.

A series of global optimization algorithms that are motivated by biological evolution are the focus of evolutionary computation [75]. They are essentially a class of population-based trial and error problem solvers with a meta-heuristic or stochastic optimization character [74], meaning that a starting collection of possible solutions is generated and iteratively updated. The process of stochastically eliminating less desirable solutions and making slight random alterations results in a new generation every time. Therefore, we

can conclude that evolutionary computational techniques can result in highly optimal solutions to a variety of issues.

The 1960s saw the initial introduction of evolutionary strategy concepts [172]. Later, in the 1990s, evolutionary programming was created. Although these fields developed independently, by the late 1990s, they had been consolidated as distinct manifestations [83] or 'dialects' of one technology, namely evolutionary computing (EC) [101]. A fourth stream, called Genetic Programming [76], had also arisen at about the same time, adhering to the same fundamental ideas. Since then, evolutionary computation has included more and more algorithms that draw inspiration from nature. Although EC contains many stochastic characteristics, potential solutions might start out either deterministically or stochastically. Since they are used to forecast the system's future states, genetic algorithms in EC provide methods to model biological systems that are connected to the theory of dynamical systems.

One of the first systems of its kind within the field of EC itself was a software program named 'Eureqa', which was created by the Cornell Creative Machine Laboratory (Cornell University, USA) and later made available for purchase by the general public by Nutonian Inc [128]. The fundamental idea is to create dynamic equations through genetic programming, each of which offers a progressively better 'fitness function' to describe a given dataset. The system iteratively constructs a series of non-linear functions to approximate the input signals when the dataset is complex, like a noise field [187].

Eureqa is a modeling engine based on machine learning as a result, choosing the equation that best describes a given data set through evolutionary searches. Eureqa asserts that it can 'automate the heavy work inherent in analytics and data science' [70]. The method uses sequences of mathematical building blocks based on a mix of common functions to generate equations at random and automatically develop analytical models that require almost no human input. In order to generate non-linear functions, the system might be seeded with information from actual noise sources. Then, depending on what is needed, the functions can be repeated to create a key-seeded pseudorandom number sequence.

Similar to artificial intelligence, evolutionary computing also uses continuous optimization. It belongs to the field of computational intelligence. AI attempts to discover a set of optimal weights by iterative computations that control the information flow (the amplitude of a signal at a particular node) across a network that replicates a simple output subject to a complicated input. In this way, an ANN mimics a high entropy input with the intention of producing a low entropy output. The process can be turned around to produce a high entropy output from a low entropy input, though. As soon as they are trained to do so, ANNs can be used to create ciphers by replicating real-world noise.

5.2.1 Cipher Generation using EC

The main reason for combining EC and ANNs to create stream ciphers is that the example maps provided in Table 4.1 must be derived from, modified, or simply invented under the application of the requirements necessary for them to be compatible with applications to cryptography. EC has the capacity to carry out this automatically by starting the evolutionary process with noise from the outside environment. While employing an ANN to generate ciphers can be useful in some circumstances, it does not give designers of PRNGs using iterated (nonlinear) functions as much flexibility. To accomplish this, a population-based stochastic search engine that mimics natural selection and an evolutionary algorithms approach are needed. Evolutionary algorithms have so garnered interest from a wide range of scientific and technical fields due to their capacity to develop outstanding solutions for challenging and dynamic issues within reasonable computational time.

In 2013, employing Eureqa, the use of evolutionary computation algorithms for cryptology was considered [34]. This system is used to iteratively create a nonlinear function to explain complicated input signals typically connected with experimental data of a chaotic system. Theoretically, no nonlinear function will be discovered on an evolutionary basis that properly models real random (delta uncorrelated) noise if it is introduced into the system. Therefore, entering natural noise is a technique to 'cheat' the system and 'force'

it to offer an approximation to the noise that would be useful (on an iterative basis) as a PRNG [70].

This suitability is based on a series of checks, as shown in Figure, which includes evaluating the Lyapunov Exponent, ensuring that the cipher stream is consistently dispersed, examining the Power Spectral Density Function (PSDF), and more as shown in Figure 5.3, where the ticks indicate that a given test has been passed. Comparatively, Figure 5.4 depicts an equivalent scheme for creating a stream cipher using an ANN. It is evident from comparing the two methods that an evolved function created using EC is equivalent to both the weights and the ANN design.

In either case, the National Institute of Standards (NIST), Computer Security Resource Center, offers a Cryptographic Algorithms Validation Program (CAVP) [158] which includes testing new PRNGs [159] in addition to any function being tested against the checks given in Figure 5.3 and Figure 5.4. These NIST tests are a requirement for using a cryptographic algorithm and are an international standard for evaluating the cipher's cryptographic strength.

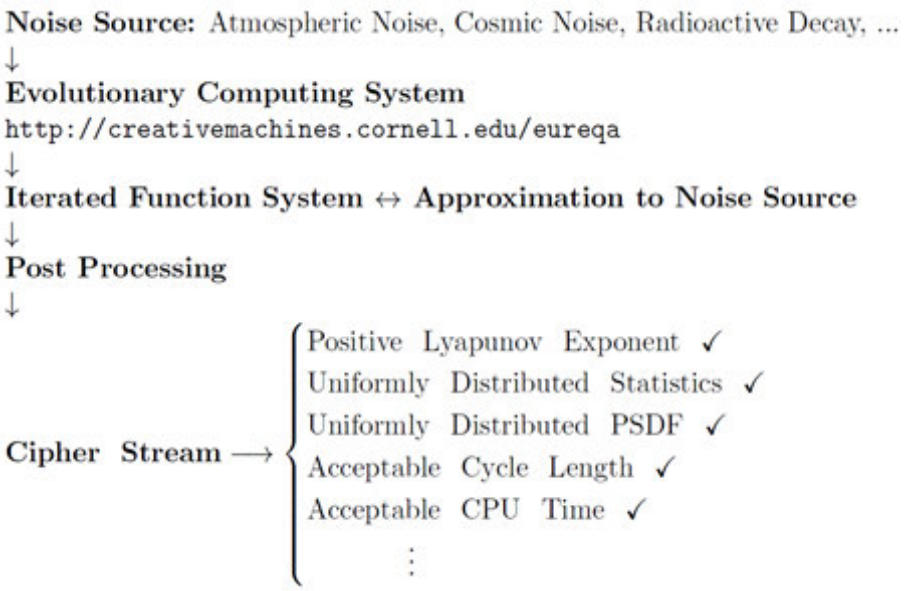

Figure 5.3: Schematic of the processes for evolving a stream cipher using EC.

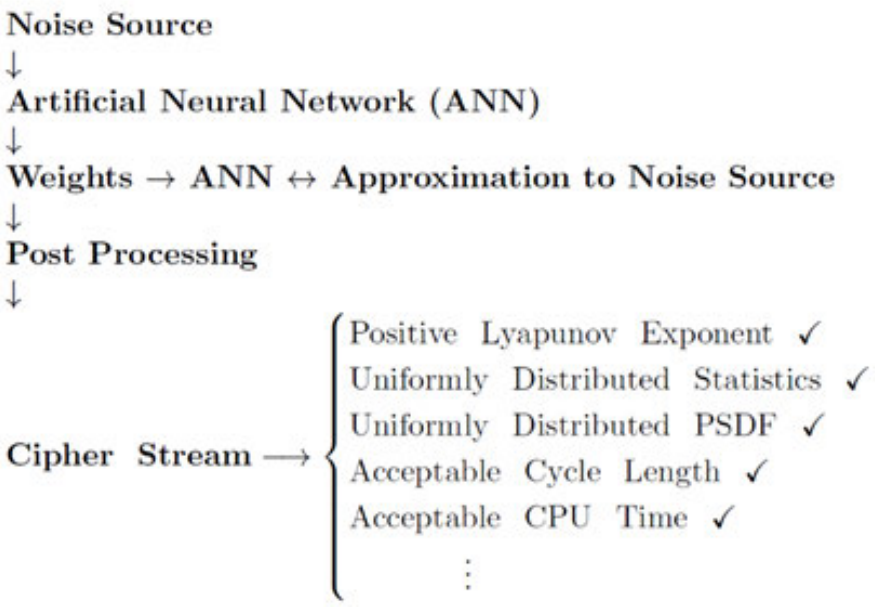

Figure 5.4: Schematic of the processes for generating a stream cipher using an ANN.

5.2.2 Real World Noise Sources

The systems whose schematics are shown in Figure 5.3 and Figure 5.4 can be input with a variety of real-world noise sources. As an open-source resource, Random.org, for instance, offers real random number streams [170]. In this instance, the random data is derived by radio transmissions from lightning that cause atmospheric noise; on average, 100 lightning strikes hit the earth every second. Another illustration is the noise produced by a reverse biased semiconductor junction. For instance, Araneus Information Systems in Finland may produce and supply an external USB interface as a means of doing this. A hardware random number generator named *Alea II* is their random number generator entropy source, sometimes known as a non-deterministic random bit generator [14].

5.2.3 Examples of Evolved Ciphers

Examples of a sizable database of such non-linear functions that can be evolved can be found in the evolved ciphers described in this section.

Table 5.1: Evolution of non-linear functions by Eureqa

Cipher	**Evolved Nonlinear Function**
Cipher 1	$\cosh^{-1}(x + x^3 \sinh^{-1}(x-1))\cosh^{-1}(x - 1 + x^3$ $\times(6 + \tan^{-1}(x \sinh^{-1}(x^3))))x^3$
Cipher 2	$0.5076 + 0.1094\sin(1.574x) + 0.1568\sin(6.107$ $+x\log(0.4619\sin(-0.04284x\sin(-0.03866x))))$
Cipher 3	$44.42 + 21.91\sin(3.362 + \cos(47.26\cos(41.49x))$ $-32.66x^2 + 20.95\cos(4.23 + 53.46\cos(\sin(3.362$ $+\cos(47.26\cos(41.49x) - 32.66x^2))^2\sin(3.362+$ $\cos(47.26\cos(41.49x)) - 32.66x^2)) + \cos(41.49x)$
Cipher 4	$\cos(1.691 + 2.216x) - 0.9704x\cos(1.807 + 2.03x)$ $-0.2623\cos(1.314x\sin(x)[\cos(\cos(1.807 + 2.03x)]$ $-0.9884x\cos(1.807 + 2.03x)) - 5.863 \times 10^5 x)$

Figure 5.4 shows generated maps given by the following non-linear function:

$$\begin{aligned} f(x) = {} & 0.753 - 0.470\sin(0.753 + \sin(\sin(\sin(-4.334 \times 10^5 \cos(x))))) \\ & - 0.600x\cos(x)\cos^{-1}(x)\sin(4.059x^2\sin(\sin(-4.331 \times 10^5\cos(x)) \\ & - 2.212 \times 10^5\cos(x))) \end{aligned} \tag{5.1}$$

Figure 5.5 comprises of the list of equations obtained after 100 iterations, the selected solution, fitness function, correlation coefficient, maximum error, and mean square error. These outputs are produced by the system. The charts in Figure 5.5 show the normalized solution fit to an input vector made up of 138 random data items (top-right) and the accuracy of the solution (Error) plotted against the complexity of the problem (lower-right) (of greater complexity but with a lower error). The chosen equation utilized 95% of the training data from Random.org

It is possible to create an infinite number of key-dependent stream ciphers by repeating the procedure and using Eureqa to create functions that approximate various random vectors (obtained using various noise sources or various noise fields from the same source).

Figure 5.5: Evolution of a nonlinear fitness function to approximate the input noise from Random.org

Eureqa is a software tool developed by Nutonomy that uses a process called symbolic regression to automatically discover mathematical models that best fit a given set of data. In symbolic regression, the goal is to find a mathematical expression (i.e., a formula) that approximates the relationship between the input variables and the output variable(s) [186]. To evolve a nonlinear fitness function that approximates input noise, Eureqa uses an iterative process that involves the following steps:

(i) Data input: The user provides a set of input and output data that represents the behavior of the system under investigation. This data can be noisy and may contain errors or inconsistencies.

(ii) Symbolic regression: Eureqa uses symbolic regression to search for a mathematical expression that approximates the relationship between the input and output data. This involves using a genetic algorithm to generate a population of candidate models, which are then evolved over multiple generations to improve their fit to the data.

(iii) Fitness evaluation: Each candidate model is evaluated based on how well it fits the input and output data. In the case of input noise, the fitness function may be designed to penalize models that are sensitive to input noise or that fail to capture the underlying patterns in the data.

(iv) Selection and evolution: The best-performing models are selected for further evolution, while poorly performing models are discarded. The selected models are then subjected to genetic operations such as mutation, crossover, and reproduction to generate a new generation of candidate models.

(v) Termination: The evolutionary process continues until a stopping criterion is met, such as a maximum number of generations, a minimum fitness threshold, or a maximum computational budget.

The end result of the Eureqa process is a mathematical model that captures the relationship between the input and output data, including any underlying patterns or noise. This model can be used to make predictions or to gain insight into the behavior of the system under investigation.

To evolve a nonlinear fitness function that approximates input noise, an iterative process that involves the following steps was modeled with Eureqa. This is done by iterating the function when $x_{n+1} = f(x_n),\ n = 0, 1, 2, ..., N$, where $x_0 \in (0, 1)$ is the key and N is the length of the Examples of maps that were randomly chosen from the list of non-linear functions generated by this procedure are shown in Table (5.2.3).

5.3 Natively Binary Chaos

The method covered in this chapter results in ciphers that use floating point iterators as their foundation. Such iterators' output can be transformed into binary streams either out of necessity (as in the segmentation approach shown in Figure 5.1) or out of a desire to encrypt in binary space (when developing real-time systems, for example). For instance, any method that offers a positive-only floating-point cipher in the latter situation,

where $\mathbf{x} \in [x_{\min}, x_{\max}]$, say, can be post-processed to generate a cipher $\mathbf{x} \in [0, 1]$ using Equation (5.2)

$$\mathbf{x} := \frac{\mathbf{x} - x_{\min}}{\|\mathbf{x} - x_{\min}\|_{\infty}} \tag{5.2}$$

Then, by using a round transformation that rounds each element of x to the nearest integer, the binary string is obtained (in this case, 0 or 1). Therefore, it is only natural to wonder if it is conceivable to create algorithms that provide natively chaotic binary outputs without the requirement for conversion from decimal integer form.

Such algorithms are more challenging to create than their floating-point equivalents, which, as will be detailed in the next chapter, can be practically infinitely large utilizing evolutionary computing. On the other hand, getting real-world binary strings is rather simple and can be done, for instance, by using [170] and [14]. This suggests that an ANN could be trained to generate random binary streams from a small number of input (binary) strings (the keys). It also highlights the significance of developing tests that can reliably assess whether a binary string is random or not, which is covered in the following section. Applying the new generation of Generative Adversarial Networks (GANs), originally considered in 2014 [95], is a way to create an algorithm that can generate natively random binary sequences. The foundation of a GAN is the use of two neural networks that compete against one another in a zero-sum game. The method learns to produce new data for a given training set, but importantly, with the same statistics as the training set. This obviously includes a binary string with a uniform distribution of 0s and 1s.

5.4 Practical Implementation

It is assumed that the output maps developed by Eureqa will pass the fundamental requirements for randomness, even if they should be frequently validated against the NIST CAVP (because the maps are essentially an approximation of real-world noise). Therefore, the distribution will by default be uniformly distributed, and a partitioning method is not required to produce a maximum entropy encryption. Any evolutionary computed cipher has the ability to iterate, allowing for the usage of the same key $x \in (0, 1)$ several times.

Another strategy is to employ EC ciphers just once, in which case repetition is not necessary. By ensuring that the cipher is structurally stable, this method solves the problem of assuring that the iterator has the same cycle and Lyapunov exponents for all beginning conditions (given that most of the known pseudo-chaotic systems do not possess this property). This, however, presupposes that a sizable database of these ciphers has been pre-created and/or can be generated instantly as needed. However, since each cipher needs a real-world noise sequence to build a map, it is possible that the original data might be encrypted instead of utilizing EC to develop a map.

This eliminates the initial requirement for EC. However, since Alice and Bob must exchange either the map or the data from which it has been evolved, exchanging a map will be preferable because it uses fewer bits to transmit over an unprotected network when using a known key exchange algorithm. Examples of such maps are those given in Table 5.2.3. A key exchange algorithm is required in both scenarios to either exchange the map (once) plus the keys used to produce various initial conditions for each plaintext, or just the map. In the latter scenario, the map is equivalent to the algorithm's key.

The technique not only provides a cipher but also a way to produce encryption keys. This has been taken into account in [150], where EC is used to create keys for encrypting cloud-bound data with the Data Encryption Standard (DES and 3-DES), the Advanced Encryption Standard (AES-128, 192, and 256), and a novel crypto-system named Cryptor, established in 2016. Key management is one of the issues that cryptography has to deal with. An ANN is used in this scenario to learn the patterns of the encryption key in order to solve the key management problem. To prevent a key attack, the key is never saved after it has been learned and is instead thrown away. The ANN then reconstructs the key in order to do the decryption.

The fact that EC generated maps may require numerous high precision floating point operations to be computed after they have been generated on a normal personal computer is a serious problem. Generally speaking, the output map gets more complex the longer EC is allowed to grow a function. But this problem is really a hardware development issue, and it offers a way to generate an infinite number of unique and unclonable ciphers

without the maps having to be 'designed by hand' like the rise of deep learning using complicated and computationally intensive ANNs.

This makes a method for encrypting data using one-time pads possible, in which an evolutionary calculated cipher (together with the key used to set the initial condition, if necessary) is used only once and then destroyed. In this situation, a well-known algorithmic attack is reduced to a thing of the past, and traditional PRNGs like those in [220] may one day be of historical importance only.

An EC system like Eureqa is designed to loop an undefined finite number of hours on a typical CPU is available in order to develop a cipher that delivers the greatest approximation to the input noise. For applications in cryptography, this might not be necessary, though, as the output after just a few iterations may offer a nonlinear function that provides the appropriate chaos, greatly lowering the amount of time needed for evolutionary computation. Multi-algorithmic cryptography, which employs such maps, can also be used to encrypt distinct blocks of data over a randomized window of plaintext by randomly selecting different maps from a data store of maps that has been generated beforehand.

Passing recognized statistical tests, such as [159], which are intended to ensure a generator's pseudo-randomness, is the foundation of practical cryptography. In most cryptography applications, pseudo-random sequences are utilized in place of truly random sequences. This is due to the fact that a symmetric crypto system can concentrate on using a key to start an iterative formula that is known by both Alice and Bob and is preferably in the public domain. In order to create an iterator that is assumed to be a rough approximation to these sequences, we have taken into consideration a technique for designing algorithms for generating pseudo random (chaotic) sequences utilizing truly random strings. This method ignores the iterator's algorithmic complexity, which is one of the key issues with using chaos to cryptography. Additionally, neither the iterator's algorithmic complexity nor structural stability are taken into account. It does, however, offer a workable solution to the issue of creating a substantial database of PRNGs for use in personalizing encryption algorithms for strictly 1-to-1 communications or '1-to-Cloud'

(encrypted) data storage [37][208][209]. This renders any public knowledge of a particular encryption algorithm impractical.

One can create a nonlinear map with automated control parameter selections by employing EC systems like Eureqa [127], seeded with noise. When an adversary has access to a sufficiently long sequence, the one-step unpredictability does not, however, ensure that the output sequence will also be unpredictable. In other words, the large number of samples may, at least theoretically, result in uncertainty. These clauses give EC the ability to create an infinite number of custom ciphers that users can employ to safeguard their cloud-hosted data.

Even though a new set of PRNGs can be created by anyone interested in doing so, evolutionary computed algorithms can be published so that the method complies with the Kerckhoff-Shannon Principle as necessary. The technique could be seen as a technical response to the 'democratization' of the cipher bureau' in this situation.

5.5 Case Study 1: Generic Applications to Cloud Data Storage

Data security has been and is a top concern when using the Cloud for data storage. This covers, for instance, safeguarding patient privacy in legal and medical settings. In order to accomplish this, EC can be utilized to create a special encryption while implementing an OTP for a 'One-to-Cloud' (OTC) application. The schematic provided in Figure 5.6, which demonstrates how a bijective link between the plaintext and the OTP is constructed using an exclusive-OR (XOR) gate to implement modulo-two encryption. It provides an illustration of the basic method. The Infrastructure as a Service (IaaS) Cloud service model receives and stores the ciphertext output from the gate.

The Key Distribution Problem (KDP), which is generally present with OTP encryption, is not present in this OTC application. A KDP occurs when a key needs to be distributed to two or more users prior to their data being sent. The Cloud client applies

encryption locally before transferring the OTP key to another site to decrypt the data once it has been downloaded from the Cloud. When utilized properly, this method offers a solution for cloud security in which the ciphertext cannot be cracked. Security is dependent on how the user manages the cipher, which is commonly stored on a Flash Memory USB Stick.

Since the key is shared by two sides, traditional bi-directional OTP communication systems have a KDP. However, because just the client encrypts/decrypts the data in this instance, there is no KDP. More significantly, the OTP is based on an encryption that can only be used by a single user. This renders side-channel and other less sophisticated assaults impossible to launch, regardless of the encryption employed. For opponents that access a Cloud server unlawfully, the encrypted data is absolutely unreadable and therefore unbreakable. It is possible to create and keep a sizable database of ciphers for this purpose at Location 1 on an air-gapped computer (disconnected from the Internet), as shown in Figure 5.6.

When necessary, a plaintext-size OTP is taken out of the OTP database and stored on a flash drive that the client carries with them to be decrypted at a different site - Location 2 as given in Figure 5.6. In both locations, it is assumed that a flash drive-type device is linked to a computer via a USB port, but it is unplugged right after after use.

5.6 Case Study 2: Radio Frequency Product Authenticity

Technologies to make high value products and documents authentic and impossible or at least difficult to counterfeit have been pioneered for many years. Current approaches to authenticating products and documents are based on attempting to make use of a smart-phone coupled with the Internet of Things (IoT). This obviates the need to acquire specialist 'readers' and because smart-phones are becoming increasingly common. The current number of smart-phone users worldwide is the order of 3.8B (48.46% of the current

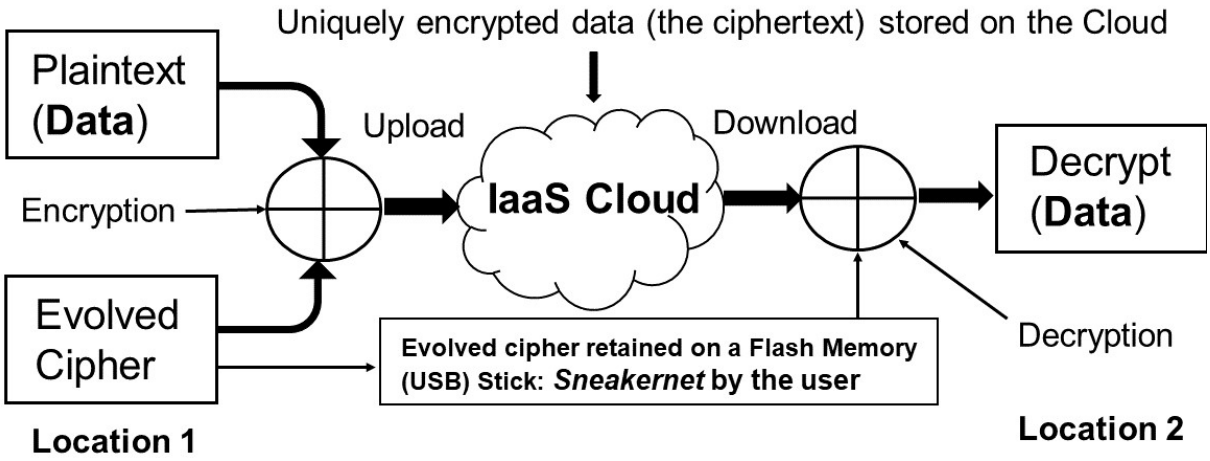

Figure 5.6: Schematic representation of data encryption (modulo-2 encryption denotes by $\oplus$) utilizing a special ciphertext obtained by EC. The user keeps the cipher on a flash memory stick to decrypt the data at the same (Location 1) or a different (Location 2) data access point where the encrypted data is downloaded from the Cloud and decoded. The encrypted data is saved on the Cloud.

world population [19]). Thus, if new security features can be introduced that can be 'read' by a smart-phone, then no added expense is required for an authentication at Point of Sale (PoS). Further, it is relatively simple to develop an App given that the majority of users can download an App from the IoT and use it, provided that they are made aware of the existence of such a facility.

In this case study, we consider the use of unclonable ciphers based on the application of EC for authenticating documents, specifically with a smart-phone. This is because smart-phones are now very common and because it is trivial to download a specialist app including those based on the methods discussed here. Consequently, there is no added expense for authentication at a PoS; it is simple to introduce new apps for upgrades and obviates the need to buy specialist readers. The case study focuses on the use of radio frequency tagging for authenticating high security documents and specifically discusses the case for an application to banknotes[1].

[1]Based on R & D undertaken by the author for De La Rue [3] in 2019-2020 in collaboration with

In order to utilise a smart-phone in this way, it is necessary to respect that there are two default facilities that are common to all smart-phones:

(i) the antenna, which primarily functions the $\sim$ 500-900 MHz band depending on the available cellular network;

(ii) digital cameras with a comparatively high resolution and a Halogen LED flash are common.

New features and strategies are currently being studied because these facilities may not always be compatible with the security measures that are now available.

5.6.1 Radio Frequency Identification

Consider using Radio Frequency Identification (RFID), a well-known technique, in which an Integrated Circuit (IC) coupled to an antenna is used. The IC is thin and flexible enough to be embedded into a document, and it has enough Read Only Memory (ROM) to store an unclonable ciphertext that is specific to the IC. The ciphertext can then be activated and read in the near field using a smartphone's antenna, which will authenticate the document online. This type of RFID requires the creation of numerous chaos-based functions, which can be generated using EC, in order to ensure that the encrypted data kept on the IC is unclonable. The ciphertext can then be activated and read in the near field using a smartphone's antenna, which will authenticate the document online. This type of RFID involves the creation of multiple chaos-based functions, which can be constructed using EC, in order to ensure that the encrypted data kept on the IC is distinct and unclonable.

FLEX NFCTM stamp antennas [82] are a new generation of ultra-thin (up to 10 μm compared to standard IC's of 75 μm or 120 μm) low cost and flexible IC's made of amorphous silicon deposited on a polyamide substrate. This extends electronics beyond the realm of conventional mono-crystalline silicon-based IC's which can be applied to

talkin' things [204]

various packaging and security documents and can be embedded in polymer-based bank notes as are foil holograms, for example [36]. They include a near field communications antenna with typical operational frequencies of 13-14 MHz, a self-resonance frequency of 100 MHz, an impedance of 50-80 Ohms and a reading distance of up to 40 mm.

5.6.2 Authentication using Unclonable Ciphers

Using physically unclonable maps, a technique to produce ciphertexts that are inextricably linked to the device, authenticity is provided. The ciphertexts are specific to a certain device, are indecipherable, difficult to suspect or eavesdrop on, and are destroyed by any interference from the device. This method can also be based on chaos theory, which allows for the expansion of the dispersion of physical parameters in electrical circuits in addition to EC. The novel aspect of the method is the use of time instability, which ensures that the ciphers and keys are unpredictable and repeated only for a particular device [94].The approach is implemented as a hardware-based, non-programmable coprocessor that is a component of a flexible IC and has a maximum entropy encrypted module with minimal physical complexity and power consumption tied to the Internet of Things (IoT).

Figure 5.7: Illustration in schematic form of how a flexible IC with a special ID number (encryption cipher/key) might be applied to a banknote maintained on a non-programmable coprocessor with a ROM.

Figure 5.7 provides a schematic of the approach for the authentication and identification of a bank note. This is based on correlating a unique ciphertext maintained of a flexible IC (embedded in the note) and the serial number of the bank note. The ciphertext is an encryption of the serial number on the bank note, for example, but other data relating to the note can be included as required. The cipher used for encryption can be an EC generated cipher which is unique to the serial number and unclonable. The ciphertext is read by the antenna of a smart-phone (in the near field), transmitted by the phone using the IoT to a secure relational data base where the encrypted data is identified. If identified successfully, the unique cipher is recovered and used to decrypt the data recovering the serial number of the banknote. This result is then transmitted back to the phone quantifying the authenticity or otherwise of the bank note while displaying the serial number as required, and, as illustrated in Figure 5.8.

In the future development and deployment of this technology, the application of EC based nonlinear functions for generating the many hundreds of millions of unique unclonable ciphers required to embed a unique ciphertext is clear. For any newly evolved function, the initial condition can be set to the serial number of the bank note, for example. Each function is 'encoded' in the flexible IC during a production run. Decryption and validation (or otherwise) is accomplished automatically by searching for the unique cyphertext held in a relational data base and its correlation with the serial number upon reception from a smart-phone over the IoT.

5.7 Case Study 3. Optical Authentication of Documents

We are used to the application of a wide range of features intended to make, for example, security documents and banknotes tough to forge and simple to identify as being the real deal. The use of watermarks, foil holograms, raised print, micro-printing, metallic threads, the production of color features under Ultra-Violet (UV) light, are well-known

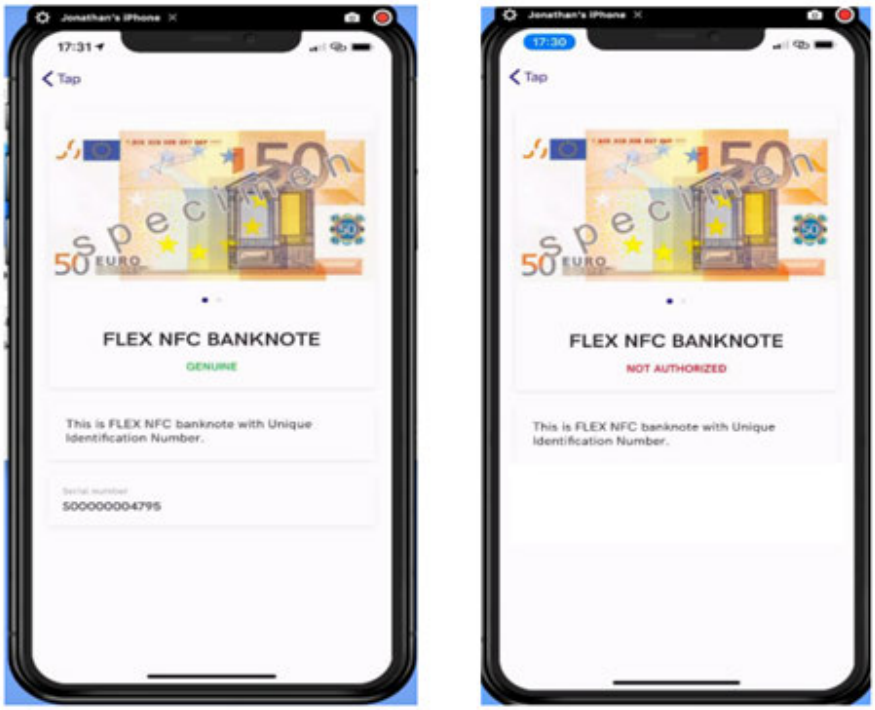

Figure 5.8: . Example of the IoT return for a banknote with an embedded flexible IC using a smart-phone App to read and transmit the ciphertext and provide the user with a decision on the authenticity of the banknote, i.e. GENUINE (left) or counterfeit and NOT AUTHORISED (right).

instances[161]. We are accustomed to Point of Sale (PoS) facilities that require the use of specialized equipment, in addition to basic visual authentication. These facilities range from the use of UV lamps to automatic counterfeit detection to examine various advanced security features on security documents and currency. This covers the detection of infrared ink, magnetic ink, metallic thread, color, size, and thickness, for example, using the Safescan program [183].

Current security document authentication methods attempt to employ a smartphone in conjunction with the Internet of Things (IoT) [81]. Due to the widespread usage of smartphones, the need to acquire specialized 'readers' is eliminatated. The growth of linked devices is now an exponential function, driven by the IoT. Due to persistently

malevolent threat actors, this interconnection necessitates increasingly strict security measures.

When compared to RFID authentication, optical authentication offers the chance to create a variety of optical ciphers and associated identifiers, but doing so involves leveraging the Internet of Things (IoT), which has greater data throughput requirements than RFID. On the other hand, basic, low-cost optical elements and complexes that are print-only and/or additive can be added. Convolutional encoding is used in this application to create ciphertext that can be printed (creating a texture code), scanned, and decrypted [42] using a unique EC generated cipher. Except for three problems—size, rotation, and cropping—the approach is reasonably resistant to ciphertext distortions. To put it another way, if the ciphertext image is printed on a piece of paper and then scanned, the resulting image must first be precisely cropped, rotated, and re-sampled to its original size. Additionally, the texture code becomes redundant data and can be binarized and printed if the plaintext image is binary. In order to identify and verify the plaintext, correlation of this binary texture code with the original encryption recovers the binary plaintext mixed with background noise but with a high Signal-to-Noise Ratio (SNR). However, even a low SNR can be used to recover the binary plaintext, although the entropy will be too low, thus, the mean square error will be high. To render the binary texture invisible in the optical spectrum, for instance, UV printing can be used. The code must then be scanned and decrypted using UV light, however this cannot be done with the capabilities of modern smartphones. As a result, a different strategy is needed, which is discussed in the next section.

5.7.1 Optical Authentication using Retro-reflective Powder

The approaches covered in the previous section can be used in practice, although they both need the realization of 'laboratory conditions'. This means that a UV or laser light source is necessary to create and retrieve the authentication patterns, which must then be processed and examined under carefully regulated settings using specialized equipment

with a minimum amount of optical recording configuration error. Therefore, it is obviously impossible to use such approaches in the field with smartphones that do not have UV or laser light sources, at least not at this time. For this reason, a strategy that can be used in the field was developed. This strategy is compatible with the optical capabilities offered by a smartphone, but uses retroreflection [219].

Despite the fact that binary texture codes can be produced without UV printing, a UV print offers a technique to conceal the code under situations of visible ambient lighting. The encryption is kept secret in the optical spectrum in this way. Another approach is to embed a random distribution of tiny retroreflectors into the surface or the (transparent) body of the document in order to introduce a cipher that is opaque in ambient light [219]. As a result, a random pattern emerges that is particular to that document. The retro-reflector complex cannot be seen visually in low-light settings, and it cannot be seen in a smartphone image that has been digitally taken. The retro-reflectors, however, will reflect the light and their spatial distribution (or rather, the combined effect of that distribution and the printed information) captured in the digital image that can be retained and uploaded to the Cloud when using a Halogen flash, which is typically available with most modern smart phone cameras.

Retro-reflectors are tools that, with the least amount of scattering, reflect light back to its source. These devices come in a variety of shapes and sizes, including corner reflectors, prism-based reflectors, and phase-conjugate mirrors, but the most popular kind uses micro retro-reflectors, which are made up of tiny glass beads or microspheres on each side. The beads can be added to paint and lacquers and range in size from 0.1 to 1 mm. When added, they make the surface of the material to which the paint or lacquer has been put sparkle. Micro-spherical retro-reflectors have been used for many years to coat items like traffic signs and painted stripes. Signs are covered with a variety of reflecting materials, some of which use layers of glass beads and others that have micro-plastic corner prisms implanted in them or micro-prisms.

For instance, Glowtec [93] produces a retro-reflective powder made of retro-reflective glass micro-spheres. The application of the micro-beads to a surface using, for instance,

clear lacquer will result in a naturally random distribution of micro-beads that are each unique. The micro-beads are 30–40 microns in size and have a refractive index of 1.93. The presence and distribution of the micro-spheres are invisible in settings of ambient lighting. A Halogen flash, however, will disclose any area of a surface over which the powder has been sprayed. An individual, optically generated texture-code may be acquired in this situation. This is a result of a combination of physical micro-bead dispersion and picture capturing resolution. The angle at which the image is captured in relation to the surface plane can be altered without affecting the texture code. This is due to the fact that retro-reflectors, regardless of the angle of incidence, reflect light back to the source subject to distortions obtained at angles of greater than the order of 30 degrees or more.

The following is an example of how this technique is typically used in PoS to validate products.:

1. On a smartphone, an image of the product surface is captured using the LED flash.

2. If the image has a 'bright flecks field', it is assumed that the product has an optical ID, which is a type of low-level authentication on its own.

3. The field is cropped to precision, and the output is submitted online, where the distribution of flecks is compared in a relational database of such images retained on the Cloud for product authentication or otherwise..

In this instance, the placement of retro-reflectors determines the cipher. In a deterministic sense, it is impossible to govern this distribution. The issue then becomes how to use the cipher on a document. Ideas for resolving this issue are covered in the following sections.

5.7.2 Retro-reflective Embedded Polymers

A retro-reflective powder that is put externally is susceptible to 'soiling' or wear and tear, on a document's surface. If the powder is incorporated into the polymers that are gradually taking the place of paper-based documents like banknotes, this problem is

eliminated. This can be achieved, for instance, by applying fluid powder and adding it to the polymer before extruding the sheets.

Figure 5.9 demonstrates the difference between taking a photo on an iPhone with and without a flash by utilizing a Retro-reflector Embedded Polymer (REP) that has been placed on top of a £20 banknote. The overlay was necessary since, up to this point, the writers were unable to work with a print security provider on a print run where the banknote information and current security features were printed onto a REP. In the case of the flash image shown in Figure 5.9, the encryption can be thought of as the amalgamation of the printed information and the retro-reflector complex. The produced image is regarded as a stochastic signature that is exclusive to the banknote's serial number or any other distinguishable security feature.

The spectrum of the cipher is guaranteed to be a broad-band spectrum by the presence of a sporadic distribution of white specks. The random spatial distribution of the flecks is caused by the retro-reflective powder being added to a polymer prior to sheet extrusion, which results in a 'natural randomness' across the polymer roll. The roll is divided into equal-sized sheets, yielding a ream of REM. Each sheet is assumed to be the same size and quality, but exhibits a 'fleck signature' when illuminated by a flash. This creates an optical cipher that is specific to each polymer sheet when combined with printed content, with the population density of the flecks being dictated by the amount of retro-reflective powder used and added to the Polymer in its liquid phase.

5.7.3 'FlashID': A New smart-phone App for Retro-Reflective Authentication

Both a YouTube advertisement and a brief technical explanation of 'FlashID' are accessible at [2] and [1], respectively. The iOS App includes the following and is set up for a banknote made of a retro-reflective polymer:

Figure 5.9: Selected region of a £20 banknote with a REP overlay in natural light (left) and in flash (right). In the latter instance, a larger version of a zoomed-out portion of the image is used to reveal the 'flecks' (superimposed onto the top right hand side of the image). The identical iPhone was used to capture both pictures. Due to the complexity of the retro-reflectors embedded in the polymer and their specific distribution to the REP overlay, the white 'flecks' in the image on the right hand side are the result of reflections from the light generated with the flash.

Mobile App for iOS

- The app searches for shapes that resemble UK banknotes and assumes that the user will find an image that closely resembles the note;
- To extract the banknote, the image is cropped and some skew is corrected using parameters that can be customized in the software;
- The 'Add to Inventory' feature in the app uploads the cropped image to the iCloud database just for testing and demonstration reasons. In actuality, this operation would be carried out throughout a print run to create a database for each note related to the print run;
- The user is presented with the decrypted plaintext image for visual examination, with the option to employ OCR to provide a more positive answer if needed.

Cloud Infrastructure

The infrastructure for the cloud has been built to be very scalable and inexpensive. It can process transactions of up to 500 per second, which should be compared to Visa, for instance, which processes 1,700 transactions per second globally, even if data throughput has not been properly tested. A very thin coating of retro-reflective powder may be used for testing with current currency. This is applied on the back of a £10 note's transparent window in the scenario illustrated in [2]. Although the presentation is based on an iPhone, the app can be used on any smartphone and/or an iPhone running iOS 13.0 or latest.

A banknote cropped using a 12MP camera is ∼30MB in size, which is a substantial amount for uploading from a mobile phone. However, the upload time can be greatly decreased by dividing the image into blocks, starting a parallel upload on each CPU core, then reassembling the image on the server. As an alternative, the app can be set up to work only in a specific area of interest. The file size is less than 1MB, for example, For a 1-2 cm^2 region. Retro-reflection applied to the entire region of interest is preferable since the robustness of the procedure depends on the area of the note that is retro-reflector sensitive.

The method for optical authentication that is taken into consideration in this thesis is based on the following two steps. Using an Internet connect smart phone at the PoS:

- **Step 1**

  - The document is captured with a flash image capturer.;

  - If a bright stochastic field is seen in the image, Step 2 will determine whether the note is genuine or not.

- **Step 2**

  - Uploaded online is the automatically cropped flash image.;

  - the serial number is entered and uploaded online;

  - The validity of the cipher and serial is determined, and the outcome is returned.

Step 2 makes the assumption that the cipher's random distribution of retro-reflectors, which, like the serial number, is specific to each document, defines the cipher.

The advantages of this approach are listed below:

- Each security document has an available statistically unique attribute;
- Using a smartphone image taken with the flash that creates a stochastic analogy of the 'red-eye' effect, it is rather simple to validate a document at a PoS;
- retro-reflection powders very cheap, e.g. 50g of powder costs approximately £25 [93]

The disadvantages include:

- a prerequisite for creating a database of ciphers;
- the need to incorporate retro-reflection powder into a polymer to ensure cipher longevity.

5.8 Conclusion

This chapter's goal was to give a concrete illustration of how AI can be used to create encryption algorithms. It has been thought about how ANNs can be trained to mimic chaos in addition to the function that evolutionary computing can play in delivering personal ciphers in the form of unclonable chaotic maps. Both instances use on real-world noise sources, including service providers like Random.org [170], for their training data. This strategy has the potential to 'democratize the cipher bureau' allowing for the relatively quick creation of customized ciphers without the use of 'hand-crafted' ciphers, many of which are now exposed to attack due to the application of Shor's algorithm. This is due to the fact that Shor's algorithm can decrypt encryption methods based on the product of big prime numbers, but this algorithm was computationally impractical before the advent of quantum computing. This is because the method under consideration belongs to the domain of post-quantum cryptography and the creation of software for quantum resistant cryptography prototyping.

There is an intriguing analogy that can be drawn between the work of Station-X at Bletchley Park during the Second World War and the use of AI for cryptography in a broader context. In this instance, the challenge was to crack the Enigma cipher's key settings, which the German military used for communication. However, the language being encrypted as well as many of the expected and repetitive words, phrases, and statements were known, along with the cipher's design. Additionally, because the intercepted Morse coded traffic was scrambled, it was known that it was encrypted. The Enigma machine itself could have been changed on a regular basis, repetitive words and phases could have been eliminated, the language could have been translated to a non-Indo-European language (possible at the time but requiring multi-lingual service providers), the Morse code could have been reconfigured (quite possible at the time), the traffic could have been made to appear routine through encrypted information hiding (not technically possible at the time), and the language could have been translated to a non-Indo-European language.

This analogy's final element serves as the foundation for employing AI to create ciphers that can be updated frequently or even for every conversation to provide an algorithmic one-time pad. In the interim, there are many potential business opportunities based on setting up online service providers to distribute customized ciphers for 1-to-Cloud applications, 1-to-1 encrypted communication, and 1-to-many applications depending on the available distribution infrastructure of an organization. This is because this is not yet practical for all people (many of whom may lack the technical IT skills).

The methods proposed presuppose symmetric encryption, where the key and/or EC cipher are exchanged prior to the start of an encrypted transmission. Many key exchange techniques that take advantage of asymmetric encryption or a no keys exchange protocol using a three-way pass can be utilized for this purpose. Both approaches have drawbacks, particularly as quantum computing develops and effectively replaces legacy systems for exchanging keys and algorithms based on the RSA algorithm. Trapdoor functions, or functions with a one-way property unless a secret parameter (trapdoor) is known, are the foundation of asymmetric cryptographic systems. However, there isn't yet a counterpart of a trapdoor transformation in chaos theory. In exploring the potential solutions to

this issue, the use of EC and other techniques that fall under the umbrella of AI may be beneficial. The Case Studies taken into consideration demonstrate how AI may be utilized in conjunction with the IoT and a communications device that is widely used, namely the smartphone, to generate encrypted data for authentication. The examples shown are typical of how AI, data security, and wireless communications are being combined to safeguard the transfer of information.

Given that mono-layer Graphene absorbs about 2.3% of incident photons at any wavelength, it may be interesting to add Graphene powder to the reflecting (microspheres) powder used in Case Study 3 in addition to the optical encryption approach. Therefore, it might be possible to use a Graphene-based cipher derived from light absorption rather than, or in addition to, the reflection of the light by retro-reflective powder by replacing retro-reflective powder with Graphene powder in a polymer, for example.

Of course, there are a lot more machine learning applications for cryptography than those that have been covered in this thesis [196], [8]. It can be anticipated that present encryption techniques, protocols, and standards related to information and cyber security will drastically change in the near future when combined with complementing discoveries such advancements in DNA computing [152], deep learning, and quantum computing. Additionally, a far wider global spectrum of society will be involved in the study, development, and management of these developments.

The focus of the next chapter is not on the creation of an evolutionary computer-generated cipher, but rather on how to use the cipher to encrypt plaintext data that is important to the properties of the ciphertext and, as is covered in more detail in Chapter 6, the creation of algorithms for a no-keys exchange protocol.

Chapter 6

Evaluation of a Binary String to Detect Randomness

This chapter continues the discussion from the end of the previous chapter by discussing the topic of analyzing a finite binary string or bit stream. How to tell if a string is completely random, a somewhat random process, or something in between depends on whether it is non-random or intelligible information (for instance, one that contains some type of periodicity). Applications for this issue include cryptanalysis, quantitative finance, machine learning, and other forms of signal and image processing. All of these address the broad problem of, say, figuring out how to distinguish information that has been noise-embedded from actual noise.

After a brief introduction to the subject, the focus of the talk shifts to the use of information entropy to address the issue. This is because this basic metric measures information that is inextricably linked to a quantifiable system. A brief explanation of the concept of entropy is followed by examples of techniques that can be used to determine the binary entropy of a finite binary string, including significant variations on a well-known topic like the Bi-Entropy. When computed, a single metric of this type may have modest statistical significance and be representative of similar binary strings. Due to this, the thesis offers a solution to the issue based on the Kullback-Leibler Divergence (also known

as Relative Entropy), which provides a measurement of how one probability distribution differs from another reference probability distribution.

By repeatedly computing this metric for different references, simulated or otherwise, random finite binary strings, it is shown how the distribution of the resulting signal changes for intelligible and random binary strings of a finite size. This enables the computation of a variety of widely used statistical metrics from which the machine learning system's framework may be constructed. A few results for a few natural languages are provided to illustrate this strategy. An appendix to this thesis has a prototype MATLAB code that demonstrates the methodology described in this chapter.

6.1 Introduction

The term intelligibility' typically refers to the clarity of speech and/or writing and the degree to which it is understandable. The term is used in this thesis to indicate whether a binary string is truly random or not (i.e., a combination of both), where, in the former case, it is assumed that a binary string is, for example, a representation of natural noise. One of the keys to achieving this is to analyze binary strings in terms of their information entropy, which is discussed again in the following section.

6.2 Information and Entropy

Leo Szilard [202] developed the first, and arguably the most significant link between information and entropy as a result of his solution to the James Clerk Maxwell-inspired 'Maxwell demon' thought experiment. This thought experiment was first put forth by Maxwell in the 1860s as a result of his research on the characteristics of perfect gases. He took into account a scenario in which gas particles are free to travel inside a container and interact with one another through elastic collisions, exchanging energy and momentum in a manner consistent with their thermal surroundings [4]. The Maxwell-Boltzmann Probability Density Function $P(v)$ which incorporates this paradigm, for the velocities v

of identical gas particles with a mass m as shown in Equation (6.1)

$$p(v) = \sqrt[3]{\frac{m}{2\pi k_B T}} \exp\left(-\frac{mv^2}{2k_B T}\right) \quad (6.1)$$

where T denotes the gas' thermodynamic equilibrium temperature in °K and $k_B \simeq 1.4\times10^{-23}\text{JK}^{-1}$ (Joules per Kelvin) is the Boltzmann constant, which creates a connection between the ideal gas' temperature and its particles' average kinetic energy. The mode of this distribution provides the particle's most likely velocity, as given by Equation (6.2)

$$v_p = \sqrt{2k_B T/m} \quad (6.2)$$

To split a container into two pieces in a hypothetical experiment, a 'daemon' would operate a frictionless shutter in the middle of the container. Particles with a velocity of $v < v_p$ can enter one area of the container when the shutter is opened, while particles with a velocity $v \geq v_p$ enter into the other section, where there is excellent thermal insulation between the shutter and container. As a result, gas particles with high and low velocities are split into the container's two compartments while retaining their velocities. As a result, the equilibrium temperatures of the two components are different in both directions from those of the original container.

Traditional thermodynamic processes require a temperature gradient for work W to occur, and an irreversible processes always result in an increase in entropy S , with ΔS given by $\Delta S = W/T$ accounting for the change in entropy. This is the basis for the second law of thermodynamics, which in the thought experiment discussed above appears to be broken because the entropy of the two sections are now different, and the entropy has been reduced in one of the sections without consuming energy as a result of a rise in temperature.

In his PhD dissertation and its companion significant publication, [202], Leo Szilard showed how the paradox may be resolved by taking into account the fact that the daemon must decide whether to open and close the shutter in order to let particles of different velocities through. The knowledge of the particle's velocity is used to inform this choice.

The gathered information is used to develop a 'balance' that accounts for the decrease in physical entropy and merged in the 'Information Entropy' Maxwell had overlooked and neglected to include this crucial issue in his original thought experiment. Szilard's main contribution in this case was to take into account that the daemon must be an 'intelligent being' that can make a decision based on *a priori* knowledge of a particle's velocity.

Szilard's initial notion of information entropy served as the foundation for information theory. He showed the incremental increase of $k_B \log_2 2$ units of entropy in any measurement. Claude Shannon independently discovered this idea [191]. Additionally, Andre Kolmogorov and Yakov Sinai created a modified form [124]. In order to answer a problem in thermodynamics, Leo Szilard developed a theory that is arguably the most important symbol of the contemporary information revolution. This is because, depending on the frequency of the symbols, information entropy provides the data needed to determine the (usually) lowest number of bits needed to encode a string of symbols.

Entropy is a measure of the number of possible arrangements of a system used in statistical mechanics. It is sometimes interpreted as a measure of 'disorder' where the higher the entropy, the more disorder there is. Another way to think of this metric is as a measurement of the absence of information on the precise condition of a system. Shannon entropy, which is measured in binary digits, is the amount of information needed to accurately determine a system's state from all other potential states, or 'bits'.

More broadly, a measure of order that may be applied to any kind of structure or system is called information entropy. It calculates how many instructions are needed to establish a particular organization. There are numerous ways to gauge one's level of knowledge, but binary choices are one that is particularly helpful. Calculating the information contained in a given arrangement involves counting the number of choices we have to make among all possible arrangements in order to get at that arrangement. Given the wider variety of possible configurations, it makes natural that more information would be required to complete a particular arrangement.

6.2.1 Shannon Entropy

A digital signal $s_m,\ m = 1, 2, ..., M$ is made up of M real values. Let $p_n, n = 1, 2, ..., N$ be the (discrete) probability distribution function or histogram that a particular value s_n occurs in the signal's bin where N is the number of bins. The information necessary to indicate whether an outcome s_m is a member, a subset, or a bin, where p_n is the distribution of bins, is measured as $-\log p_n$, where p_n is given by the dimensionless quantity. The sum of all potential values n weighted by that value's p_n constitutes the mean value μ, or s_m

The total of all possible values of n weighted by p_n of that value, or

$$\mu = \sum_{n=1}^{N} np_n$$

is what we refer to as the mean value of μ. The Shannon Information Entropy (sometimes abbreviated as S) is a measure of the mean or 'expected value' in this context, of the information measure $-\log p_n$ weighted by p_n and is determined by the dimensionless quantity as in Equation (6.3)

$$\S = -\sum_{n=1}^{N} p_n \log p_n \tag{6.3}$$

The higher the entropy of a signal becomes the greater is ambiguity, and, in this context, information entropy is a measure of the unpredictability or randomness of any message contained in the signal. This is typically determined by the noise that distorts the information contained in a signal. Generally speaking, the information entropy related to the signal's information transmission tends to rise over time. This is because the signal becomes more noisy as it travels, and because of the Central Limit Theorem, the sources of this noise are diverse and likely to be Gaussian noise.

6.2.2 The Boltzmann Entropy

The entropy, which Boltzmann and Gibbs defined as a measure of energy dispersion and, in a sense, a measure of disorder in the same way that information is a measure of order,

is the physics equivalent of the entropy of information [114]. In reality, when it comes to calculating the probability of classifying items into bins, Boltzmann's entropy idea and Shannon's information concept share the same mathematical foundations. The Boltzmann Entropy in statistical mechanics is described as in Equation (6.4).

$$E = -(k_B)\sum_{n=1}^{N} p_n \ln p_n \tag{6.4}$$

Given that S and E only differ in their scaling factors, Shannon's and Boltzmann's definitions of entropy are comparable.

The probabilities p_n in the definition of the Boltzmann entropy correspond to the energy levels of a classical system (e.g. a collection of classical Newtonian particles). With the information entropy, p_n is not *a priori* assigned such precise responsibilities and the expression can be used to add some order to any physical system. As a result, information is transformed into a notion comparable to physical entropy, and any system may be explained in terms of either one or the other. Information is said to be lost when information entropy rises.

6.2.3 Renyi Entropy

The information entropy has a number of 'variations' just like many other basic notions in Mathematics and/or Physics. The Renyi Entropy is an extension of the Shannon entropy. The Renyi Entropy H_α (of order a) is given by Equation (**??**).

$$H_\alpha = (1-\alpha)^{-1}\log\sum_{n=1}^{N} p_n^\alpha,\ \alpha \geq 0,\ \alpha \neq 1 \tag{6.5}$$

where

$$\lim_{\alpha\to 1} H_\alpha = \sum_{n=1}^{N} p_n \log p_n$$

recovers the Shannon entropy. Several complementary information entropy metrics are generated from this generalization when $\alpha = 0$ (maximum), $\alpha = 2$ ('collision) and $H_\infty = -\log[\max p_n]$ (minimum).

6.2.4 Binary Information Entropy

A binary string f_n made up of L bits can only have two possible values, 0s or 1s, and these two values are mutually incompatible. The Binary Information Entropy (BIE) function indicated by H changes in this scenario, as in Equation (6.6).

$$H = -\sum_{n=1}^{2} p_n \log_2 p_n \tag{6.6}$$
$$= -p \log_2 p - (1-p) \log_2 (1-p) \text{ Bits}$$

where, if we use p to represent the likelihood that a binary string contains a 1, then $1-p$ represents the likelihood that the same string contains a 0 instead. Similarly, if p is used to represent the likelihood that 0 will appear in the string, then the likelihood of obtaining 1 is $(1-p)$. In either case, the following hold:

$$0 \log_2(0) \equiv 0 \tag{6.7}$$

and

$$H(p) = H(1-p) \tag{6.8}$$

6.3 Evaluating Order and Disorder using BIE

Our challenge is to determine, given a binary string, whether the string is a binary representation of noise or if it contains meaningful information in the sense of having some degree of determinism. This may apply to any natural language that has developed through usage, application, and repetition without conscious thought but that was binary-coded in a deliberate, intentional manner, such as the ASCII. Therefore, the goal is to develop a technique for comparing a finite binary string of arbitrary length to another string (of equal length) in terms of the relative order and/or disorder of all of its bits. It is insufficient to perform a simple binary entropy test. Its inability to distinguish between binary strings that, for instance, incorporate periodicity and are thus not random, is the main reason of this.

Table 6.1: A brief binary string association with an intuitive understanding of order and disorder [63])The perfectly ordered binary strings have a pattern and can be recognized easily. They consist of either 1s or 0s only and can be generated using a regular expression such as: Bin = [1—0]+, where + means one or more occurrences. Mostly ordered has a binary string which dominantly repeats. Somewhat disordered means the binary string has no pattern and there is no regular expression to generate the string.

Binary String	Description	Reason
11111111	Perfect order	All 1s
00000000	Perfect order	All 0s
01010110	Mostly ordered	Mostly 01s
01010101	Regular, not disordered	Repeating 01s
11001100	Regular, not disordered	Repeating 1100s
01011010	Mostly ordered	0101 then 1010
01101011	Somewhat disordered	No pattern
10110101	Somewhat disordered	No pattern

For binary strings that are completely sorted and whose elements all equal either 1 or 0, when $p = 0$ or $p = 1$, $H(p) = 0$. In contrast, when $p = 0.5$, $H(p) = 1$, indicating the greatest irregularity. Although the value of $H(p)$ looks to reflect a random binary string in the instance of a string like 01010101, which has a recurring pattern of 01, $p = 0.5$, and $H(p) = 1$, the string is actually periodic and not at all random. A measure based on entropy that can differentiate more thoroughly is needed, in Equation (6.6). This is necessary given Table (6.1), which illustrates how we could perceive the order and disorder of some 8 bit binary strings intuitively.

For the purpose of determining or measuring randomness, regularity, irregularity, order, disorder, and entropy for binary and other strings, a number of algorithms have been developed that compute various entropy-based metrics., e.g. [135] and [89]. These include techniques that use the entropy of finite strings to describe disorder and unpredictability, such as approximation entropy [165], [164], [179], [177], Sample Entropy [174] and Fuzzy

Entropy [57]. These measurements are based on algorithms that fall into the following categories.:

(i) ways for moving windows that look at the original string's sub-strings [136][53];

(ii) algorithms that produce a measurement depending on the total length of the string[123].

Applications comprise elementary randomness tests [142], cryptanalysis [52], [51] [46]. This includes creating a measurement based on a weighted average *BiEntropy* (BiEn) of the Shannon entropy for all but the last binary derivatives of the string given by Equation (6.9)[63]

$$\text{BiEn} = \frac{1}{2^{N-1}-1} \sum_{n=0}^{N-1} [-p_n \log_2 p_n - (1-p_n) \log_2 (1-p_n)] 2^n \tag{6.9}$$

which is suitable for shorter binary strings where $N \leq 32$ approximately. Based on the string's Shannon binary entropies and the first n-2 binary derivatives using a straightforward power law, this conclusion was reached. This BiEntropy metric also has a logarithmic weighted variation that can be applied to longer binary strings and is given by Equation (6.10) [63]

$$\text{BiEn} = \frac{\sum_{n=0}^{N-1} [-p_n \log_2 p_n - (1-p_n) \log_2 (1-p_n)] w_n}{\sum_{n=0}^{n-2} w_n} \tag{6.10}$$

where $w_n = \log_2(n+2)$ for $N > 32$ (approximately). The higher derivatives are given more weight by the logarithmic weighting, though different weightings can be employed depending on the application. This study is just one of several that have been done to create appropriate tests and measurements of order, disorder, unpredictability, irregularity, and entropy based on the calculation of a single metric.

Focusing on the usage of a single metric for this purpose, while desired computationally, is restricted and might be statistically inconsequential due to its self-selecting data predication. The Kullback-Leibler Divergence is applied to a stream of data that produces a statistically significant result as opposed to a single metric in the following section as a

complementary solution to the problem. This gives the groundwork for a later-discussed implementation of a machine learning approach.

6.4 Application of Kullback-Leibler Divergence

Given that intelligibility is a relative concept, one should consider a relative metric that evaluates how a binary string contrasts in some way with a string that is known to be the outcome of a genuinely random process. The information content of the binary string-related probability distributions must also be compared to another probability distribution that is utilized as a standard in a statistical analysis of this comparison.

By utilizing the Kullback-Leibler Divergence or Relative (Binary) Entropy function provided by Equation (6.11)

$$R = -\sum_{n=1}^{2} p_n \log_2(q_n) - \log_2(p_n) \tag{6.11}$$

where p_n is a binary string's histogram of the string $f_n \equiv \{0,1\}^L$. q_n is the binary histogram of some string (binary) $g_n \equiv \{0,1\}^L$, because of the finite length of both strings L. Suppose the string f_n is not random e.g. a binary string encoding of some natural language text, and g_n is a truly random string. The metric R is required to have a considerable difference in terms of its numerical value as compared to the scenario where both f_n and g_n are random binary strings. Ideally, the requirement is to establish a threshold for the value of R, below which f_n can be regarded as intelligible, and above which, f_n can be considered as genuinely random. However, this overlooks any transition and assumes the use of a binary decision-making procedure that could not be statistically significant from f_n being random to intelligible or vice versa.

Instead, we take into account a relative entropy analysis based on a reading of the statistical contrast between the scenarios when f_n is considered intelligible and g_n is a random function, and when both f_n and g_n are random strings. Thus, determining the

Relative Entropy Signal (RES) is provided by Equation (6.12).

$$R_m = -\sum_{n=1}^{2} p_{nm} \log_2 \left(\frac{q_{nm}}{p_{nm}} \right), \; m = 1, 2, ..., M \tag{6.12}$$

where q_{nm} denotes the m^{th} binary histogram of the m^{th} random bit-stream. The following cases are considered:

Case (i): p_{nm} is the m^{th} a non-random binary string's binary histogram;
Case (ii): p_{nm} is the m^{th} binary histogram of a random binary string.

The ASCII text to binary converter can be used to generate the binary version of the English text associated with the Abstract of this thesis, in the results. This converter is accessible at [203] (NB: delimiter string must be set to none). Using the MATLAB uniform distributed random number generator function *rand*[1] , which returns floating point numbers in the range [0,1], and a round transformation to produce an array of binary strings, a series of random binary arrays is created. Each array has the same length (L) and is independent of the rest in terms of the pattern of elements that is generated. Equation (6.12) applies to each array of random bits where $p_{nm} = p_n, \; \forall m$.

Figure 6.1 demonstrates some RES signal examples, given by Equation (6.12) for cases (i) and (ii) above with $M = 1000$ and the corresponding 100-bin histograms. It can be observed that:

- When one of the strings in Case (i) has a definite mean value, $M_{(\text{i})}$, a non-zero mean Gaussian-type distribution, and is comprehensible, then $R_m > 0 \forall m$.

- For Case (ii), when every string is random, the mean of $R_m > 0$ is not zero, a mean-defined distribution of the gaussian kind $M_{(\text{ii})} > M_{(\text{i})}$ is produced.

  Given random strings, $R_m > 0$ has a non-zero mean Gaussian-type distribution with a defined mean $M_{(\text{ii})} > M_{(\text{i})}$. However, the Jargue-Bera (JB) test for normalcy is

[1]The rand function in MATLAB is used to generate uniformly distributed random numbers between 0 and 1, `https://www.mathworks.com/help/matlab/ref/rand.html`.

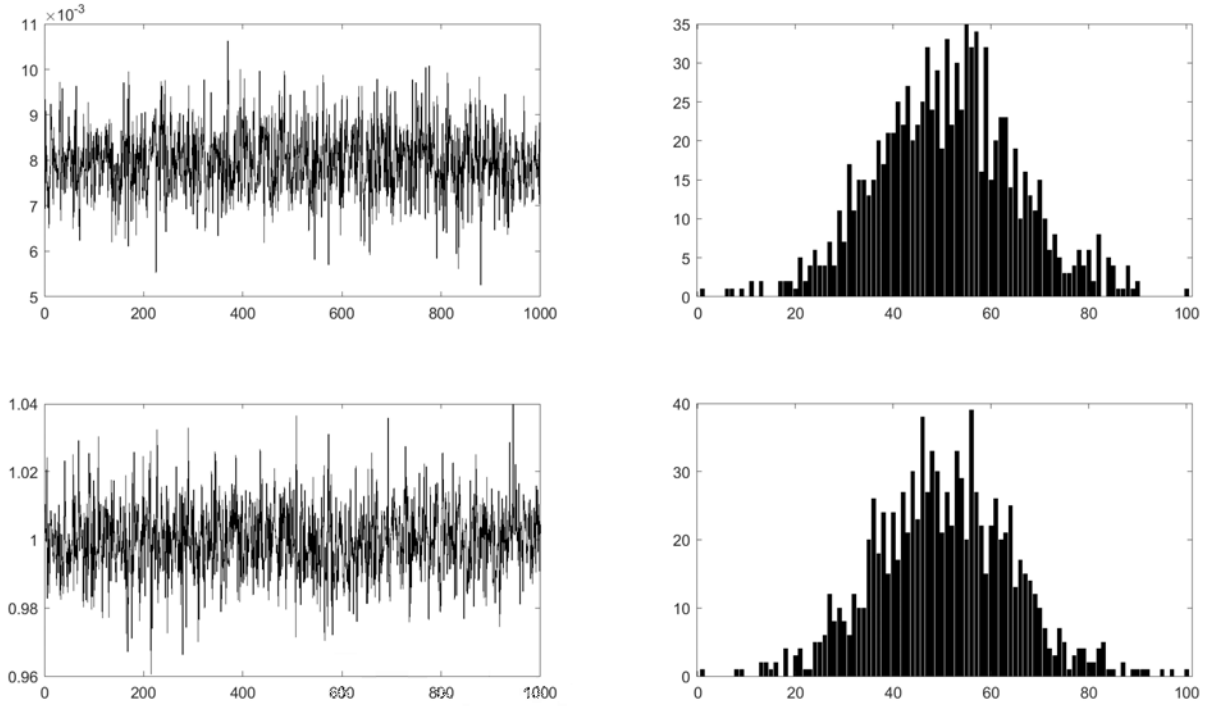

Figure 6.1: Plots of the R_m (left) given by Equation (6.12) and the corresponding 100-bin histograms (right) for Case (i) - above - and Case (ii) - below, respectively, with $M = 1000$.

used in both circumstances, and it reveals that the JB statistic is much bigger than χ^2. Therefore, the null hypothesis must be discarded, i.e. R_m, the series deviates from a normal distribution in some ways.

From Figure 6.1, the primary finding is that the statistical features of R_m for Case (i) and Case (ii) above differ. To distinguish between an understandable and random binary text, for instance, the difference in the mean of the RES for the two cases is at least one order of magnitude greater. We therefore take into account statistical measures other than the mean value.

Figure 6.2, displays the mean, standard deviation (std), median (med), and mode in logarithmic graphs for Case (i) and Case (ii) using four differernt natural languages. These are translations of the Abstract for this thesis, obtained with the online translator [97]. The output was then converted to binary strings using a text-to-binary converter

available at [203], with no delimiter setting. This study demonstrates that by computing the RES in the manner previously explained, a comprehensible binary string may be easily distinguished from a random string. The results given in Figure 6.2 demonstrate how at least four different natural languages have a major impact on the statistical properties of R_m. This is expressed on a logarithmic scale and quantified in terms of the numerical values of the metrics taken into account in Figure (6.2) .

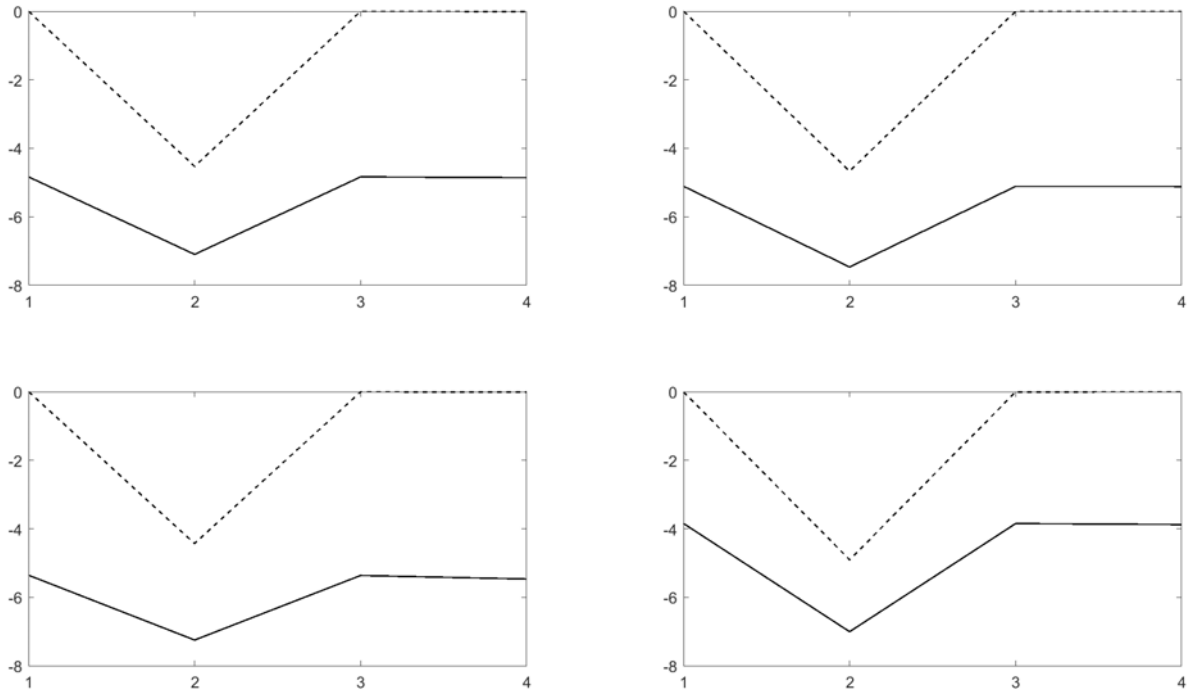

Figure 6.2: Log-linear signatures of the mean, standard deviation, the median and the mode (from left to right in each line plot - points 1, 2, 3 and 4, respectively) for Case (i) - solid line - and Case (ii) - dashed line. The top-left plot shows the result for English, the top-right plot is the result for Arabic, the lower-left plot for Chinese (traditional) and the lower-right plot for Greek.

Each time, the four metrics under consideration meet the requirements listed below:

$$\text{mean}[R_m]_{\text{(i)}} < \text{mean}[R_m]_{\text{(ii)}},\ \text{std}[R_m]_{\text{(i)}} < \text{std}[R_m]_{\text{(ii)}},$$

$$\text{med}[R_m]_{\text{(i)}} < \text{med}[R_m]_{\text{(ii)}},\ \text{mode}[R_m]_{\text{(i)}} < \text{mode}[R_m]_{\text{(ii)}}$$

where the subscripts (i) and (ii) denote the two cases considered. The standard deviation difference between the two examples is not as important as the other characteristics. However, there are differences between the two situations that are obviously statistically significant based on the mean, standard deviation, median, and mode.

A more thorough investigation in this area employing a wider range of natural languages as well as other non-random and semi-random digital signals lies outside the scope of this work. The natural languages used for this exercise have been chosen for their structural and semantic variations. In this regard, the function RBET (Relative Binary Entropy Test) in the MATLAB code provided in Appendix A was utilized to produce the four metrics shown in Figure 6.2 is offered as a proof of concept that can be expanded to work with more sources and various digital signals.

It is important to utilize somewhat lengthy binary strings $L >> 1$ and values for the length of the RES $M >> 1$ in order to produce results of this type that are statistically significant. It's vital to note that the term intelligibility is exclusively related with natural language in this test of a binary string's intelligibility. This is a constrained definition of the term that was only taken into account when creating the tests conducted. In general, a binary string that can be thought of as the outcome of a process other than a fully random process should be referred to as having intelligibility.

6.5 Machine Learning

Given that the relative entropy test, as discussed in Section 6.4 can be used to distinguish between intelligible and random binary strings, it is possible to calculate additional statistical metrics and other parameters using an analysis of the signatures shown in Figure 6.2. These may, for instance, be the statistical moments and spectral properties of R_m used to create a feature vector for a multi-class classification input into an ANN. The mean, standard deviation, median, and mode of the RES as taken into consideration in the data shown might be four components of such a feature vector.

6.6 Conclusion

The strategy discussed in this chapter offers a comparatively generic way of differentiating between random and non-random data because all data can be written as a binary string, regardless of the coding used to do so (i.e., ASCII or otherwise). The method provided in Section (6.4) is different from other approaches, as discussed in Section (6.3), in that it is based on creating a signal calculated using Equation (6.12), and characterizing the distribution of this signal.

It has been demonstrated that there is a considerable difference between the distribution of R_m provided by Equation (6.12) and random data for a small number of natural languages. This demonstrates how, at least for a natural language, the approach can distinguish between random and understandable binary streams. It has been discussed how these results can serve as the foundation for a machine learning approach using complementary statistical parameters for the signal, along with metrics related to the signal's spectrum and other transformations. Some common statistical metrics have been used to classify this effect. This analysis was done to determine whether a binary stream was truly random, comprehensible (i.e., the result of some communicative information), or both.

Chapter 7

Data Exchange using a No-Keys Protocol

After introducing the concepts of using a phase-only stochastic function for data encryption as detailed in Chapter 2 and the preceding chapters, this chapter takes a method used to develop an encryption key and, more generally, a data exchange method based on the application of a Three-Pass or 'No keys' protocol. This chapter presents the methods developed for sharing information and digital images for one-dimensional and two-dimensional arrays, respectively.

7.1 The Three-Pass Protocol

The Three-Pass Protocol (TPP)'s idea is well known, and so are the algorithms that have been created to put it into practice. The Shamir three-pass protocol is one of them [144] and the Massey-Omura method [137] is another. The protocol's guiding premise is as follows: Alice uses a private key, let's say K_A to encrypt her plaintext and send Bob the ciphertext. When Bob receives the ciphertext, he is unable to decrypt it because he lacks the knowledge of ke key K_A. Instead, Bob uses a new private key and the same procedure to encrypt the ciphertext, Bob uses the key K_B known only to Bob. Bob sends Alice the newly double-encrypted plaintext, which is only known to him. Assuming the encryption

algorithm is commutative, Alice can use it to decipher the double-encrypted ciphertext after receiving it, with the key K_A and send Bob the outcome which is a single encrypted ciphertext. The output ciphertext is then decrypted by Bob using K_B. This protocol does not require Alice and Bob to concur upon the keys K_A and K_B *a priori*. Therefore, no additional key exchange technique is needed. The following are the main prerequisites for using this protocol:

- No matter how many encryption rounds are employed, the encryption method must be commutative and be cryptographically strong enough to prevent it from being compromised using a known algorithm attack based on an intercept of any pass, especially the first and third passes, which are both single encrypted;
- The keys used must be long enough to prevent any pass from being successfully attacked exhaustively;
- For each of the three passes, if the encrypted data is intercepted, it must not be able to decipher the plaintext from the three intercepts (assumed to be complete intercepts in each case).

Any encryption system that takes advantage of this protocol must be built on algorithms that display some level of 'computational complexity', with the third of the aforementioned constraints yielding the most susceptibility. For instance, the security of the Shamir and Massey-Omura algorithms depends on how challenging it is to compute discrete logarithms in a finite field [184]. In this section, we'll look at a three-pass application that uses a phase-only encryption (commutative) technique and some related cryptanalysis.

7.2 Encryption Model for $\mathbf{r} \in \mathbb{R}^1$

The 'diffusion' and 'confusion' features that occur in the transition of an input (plaintext) to an output are a key theme in the creation of any encryption method, at least within

the context of a specific encryption model (ciphertext). The association between the key and the ciphertext must be as intricate and convoluted as possible under the situation of confusion. According to [30], diffusion is the property where the statistical distribution of the plaintext is dispersed in the distribution of the ciphertext so that the statistical signature of the plaintext is not present. The ideal procedure would produce a ciphertext that is evenly dispersed.

It is not always guaranteed that an output ciphertext with a uniform distribution will be produced when encrypting data with a known cipher (with a uniform distribution). The fact that the plaintext is distributed effectively across its whole length is what counts most, though. We take into account the following encryption model in this situation:

$$s(x) = \overbrace{\underbrace{n(x) \quad \otimes \quad f(x)}_{\text{Stochastic Diffusion}} + \; cn(x)}^{\text{Stochastic Confusion}} \tag{7.1}$$

where $f(x)$ - the information function, $n(x)$ - the noise function and the signal function $s(x)$, represent the plaintext, the cipher and the ciphertext, respectively.

The convolution integral is denoted by the operator $\otimes$ while c is a constant value. Equation (7.1) illustrates the convolution operation $n(x) \otimes f(x)$ as the process of (stochastic) diffusion, and the addition of term $cn(x)$ as the process of (stochastic) confusion. Both terms on the RHS of Equation Equation (7.1) are stochastic functions.

To maximize stochastic diffusion, the noise term $n(x)$ must be uniformly distributed. To determine maximum confusion, we need to determine the extent to which the noise function $n(x)$ is dominating the signal function $s(x)$, given the condition that $\|n(x) \otimes f(x)\| << \|cn(x)\|$; which can be determined by the value of the constant c. This makes sure that the cipher is the only factor used to determine the statistical signature of the ciphertext..

The application of Equation 7.1 along with a phase-only cipher $n(x)$ to a three-pass protocol is a relatively new approach to the problem, considering that optical Cryptography has traditionally had the exclusive use of phase-only encryption techniques[147].

7.3 Basic Algorithm

Imagine that Alice wants to exchange a single plaintext that is provided by the real function $f(x) \leftrightarrow F(k)$, the original message (plaintext) for $f(x = 0)$ due to the re-normalization considered redundant and the condition it is associated Equation (7.1). The random phase cipher is created by Alice, $\Theta_1(k)$ and similarly, Bob generates a random phase cipher $\Theta_2(k)$. By using the Fourier transform approach along with Equation (7.1), these phase functions are calculated. The algorithm(s) which generate $r(x)$, which generate these phase functions, are regarded as cryptographically strong and personal to Alice and Bob. This is through the application of an evolutionary computing approach [34], and this also applies to the keys used to initialize (i.e. seed) them. The following steps are then applied.

Step 1: For a given value of $c \gg 1$, which is Alice's secret, Alice performs and encryption on plaintext $F(k)$ to produce ciphertext $S_1(k)$ using Equation (7.2), where + is an additive operation.

$$S_1(k) = +(F(k)\exp[i\Theta_1(k)], c\exp[i\Theta_1(k)]) \tag{7.2}$$

and sends $\mathrm{Re}[s_1(x)]$ of $s_1(x) \leftrightarrow S_1(k)$ to the other party, Bob.

Step 2: After receiving the ciphertext, $S_1(k) \leftrightarrow \mathrm{Re}[s_1(x)]$, Bob encrypts $S_1(k)$ through Equation (7.3):

$$S_2(k) = S_1(k)\exp[i\Theta_2(k)] \tag{7.3}$$

Bob then sends Alice the following packet, $\mathrm{Re}[s_2(x)]$ of $s_2(x) \leftrightarrow S_2(k)$

Step 3: Alice then decrypts the encrypted packet from Bob $S_2(k) \leftrightarrow \mathrm{Re}[s_2(x)]$ as shown in Equation (7.4)

$$
\begin{aligned}
S_3(k) &= S_2(k)\exp[-i\Theta_1(k)] = S_1\exp[i\Theta_2(k)]\exp[-i\Theta_1(k)] \\
&= [F(k)+c]\exp[i\Theta_1(k)]\exp[-i\Theta_1(k)]\exp[i\Theta_2(k)] \\
&= [F(k)+c]\exp[i\Theta_2(k)]
\end{aligned} \tag{7.4}
$$

Alice then sends the following packet $\mathrm{Re}[s_3(x)]$ of $s_3(x) \leftrightarrow S_3(k)$ back to Bob.

Step 4: Bob receives and decrypts the ciphertext $S_3(k) \leftrightarrow \mathrm{Re}[s_3(x)]$ through Equation (7.5)

$$
F(k) + c = S_3(k)\exp[-i\Theta_2(k)] \tag{7.5}
$$

Thus, the plaintext message given as $\mathrm{Re}[f(x)] \mid x \mid > 0$ where $f(x) \leftrightarrow F(k)$ given that $\mathrm{Re}[f(0)]$ is undefined, or, with the re-normalization condition application, $\mathrm{Re}[f(0)] = 0$. It should be noted that the value of the constant c, used in the first pass can be randomly generated. This can be achieved once the upper limit of the constant has been computed. This is also subject to the floating point precision used in steps 1 - 4.

7.4 Cryptanalysis

7.4.1 Three-intercept Cryptanalysis

Assume that an attack is conducted to predict $f(x)$ based on knowledge of protocol provided in Section (7.3). Also, accurate records of the functions $S_1(k)$, $S_2(k)$ and $S_3(k)$ obtained by intercepting the communication associated with Steps 1-3 through computation of the Fourier transform of the results. Considering Equations (7.2) - (7.5), the following ciphers can be eliminated, Θ_1 and Θ_2 to yield Equation (7.6)

$$
F(k) + c = \frac{S_1(k)S_2^*(k)S_3(k)}{\mid S_1(k) \mid^2} = \frac{S_1(k)S_2^*(k)S_3(k)}{\mid F(k) \mid^2 + c^2},\ c \to \infty \tag{7.6}
$$

It is obvious that we must solve a cubic equation for an unknown value of $c \gg 1$ in order to obtain $F(k)$, a value which is known only to Alice. Equation (7.6) is under-determined

and, in this case, has a very large, inconsistent solutions. Thus a unique solution for $f(x)$ given knowledge of $S_1(k)$, $S_2(k)$ and $S_3(k)$ is highly unlikely.

7.4.2 Bayesian Cryptanalysis

Using Bayesian analysis, one can produce a Maximum Likelihood Estimate for $f(x)$ with the help of Equation (7.1). This necessitates the creation of a model for the statistical distribution of $n(x)$. Taking into account the scenario where $n(x)$ is described by a Gaussian PDF, the estimate for $f(x)$, $\hat{f}(x)$ is given by [27], with $G(k) \leftrightarrow g(x)$,

$$\hat{f}(x) = g^*(x) \odot s(x) \text{ where } G^*(k) = \exp[i\Theta(k)]$$

regarding an attack, phase-only encryption requires a known algorithm attack that focuses on the phase function $\Theta(k)$. This eliminates the possibility of creating a useful Bayesian attack because: I the statistical signature of the plaintext, $f(x)$, after diffusion with $n(x)$, is not available in practice; and (ii) even though the distribution of $\mathrm{Re}[n(x)]$ is known, the phase function, $\Theta(k)$, is uniformly distributed, and Bayesian estimation is not possible for uniform distributions.

7.4.3 Correlation Cryptanalysis

Think about what happens if the ciphertext from Step 1 is intercepted and compromised. Equation (7.2) can be used to create the power spectrum.

$$\begin{aligned} \mid S_1(k) \mid^2 &= \mid F(k)\exp[i\Theta_1(k)] + c\exp[i\Theta_1(k)] \mid^2 \\ &= \mid F(k) \mid^2 + c^2\left[1 + \frac{2*1}{c}\mathrm{Re}[F(k)]\right] \end{aligned}$$

For $c \gg 1$ considering the asymptotic case,

$$\mid S_1(k) \mid^2 = \mid F(k) \mid^2 + c^2,\ c \to \infty$$

or, using the theorem, where $S_1(k) \leftrightarrow s_1(x)$,

$$\begin{aligned} s_1^*(x) \odot s_1(x) &= f^*(x) \odot f(x) + c^2\delta(x) \\ &= f^*(x) \odot f(x);\ \mid x \mid > 0,\ c \to \infty \end{aligned}$$

In light of this, auto-correlating the ciphertext creates a decryption problem that is identical to the phase retrieval problem, i.e. given that $f(x) \leftrightarrow F(k) =\mid F(k) \mid \exp[i\Theta(k)]$, then if only $\mid F(k) \mid$ is known, we must calculate the phase function $\Theta(k)$ that will be used to base the Fourier inversion of $f(x)$. In infinite-dimensional spaces, the phase retrieval issue is highly ill-posed and has no uniformly stable solutions. This conclusion is true for continuous frames. In addition, the dimension affects how easily phase retrieval techniques can be used. It is well known that phase estimation algorithms have been created for the two-dimensional case to offer approximations of solutions, particularly in the context of X-ray crystallography, where the intensity of an X-ray diffraction pattern in the far-field is determined by the two-dimensional Fourier transform of the diffracting object (i.e. the crystal) [39]. The phase retrieval problem, however, is ambiguous for the one-dimensional case that is being considered. This is because it is complicated to determine the phase within the broad solution set and can only be taken into account under appropriate *a priori* assumptions or with the help of additional data. A fundamental theorem of Algebra, which asserts that any single-variable polynomial with complex coefficients has at least one complex root, has the effect of causing this. While the one-dimensional phase retrieval problem cannot be solved, the two-dimensional phase retrieval is made possible by the fact that polynomials with two variables cannot be factored. This theorem fails for polynomials with two variables. This is due to the fact that polynomials can be factored, which leads to ambiguity in which the same data can be represented by several spectral phases. Thus any attack associated with attempting to solve the one-dimensional phase retrieval problem applied to any one or all of the three passes can be assumed to fail and the application of phase-only encryption considered here will therefore remain a significant challenge for a cryptanalyst.

This assertion should, of course, be understood in light of potential future fixes for the one-dimensional phase retrieval issue. For instance, it has recently been demonstrated that if interference measurements between an unknown signal and a reference signal (unrelated to the unknown signal) are available, a signal can be retrieved from the Fourier amplitude alone [22].

The use of the Radon and inverse Radon transformations may also allow for the use of a version on the two-to-one dimensional processing equivalence principle, ($\hat{R}$ and $\hat{R}^{-1}$, respectively) first considered in [115] and compounded in the equation $\hat{P}_2 f(x,y) \equiv \hat{R}^{-1}\hat{P}_1\hat{R}f(x,y)$ where $\hat{P}_1$ denotes the one-dimensional process while $\hat{P}_2$ indicates the analogous two-dimensional process (which may or may not be directly applicable as a two-dimensional processing algorithm). In this case, a one-to-two dimensional processing equivalence principle based on the equation could be used to attempt to solve the one-dimensional phase retrieval problem. using the equation $\hat{P}_1 f(x) \equiv \hat{R}\hat{P}_2\hat{R}^{-1}f(x)$, noting that $\hat{R}^{-1}f(x)$ is symmetric, and it is an image to which the two-dimensional phase retrieval procedure denoted by $\hat{P}_2$ is then applied.

7.5 Prototype Algorithm

To implement the algorithm presented in Section 7.3 for one-dimensional array processing utilizing function *TPP(key,step,c)*, Appendix B offers MATLAB prototype software. The purpose of this function, where *TPP* stands for *Three-Pass Protocol* is to send an array of integers between two users (let's say Alice and Bob), in the form of an ASCII delimited file that represents the original plaintext. Typically, the plaintext is a key, which may, for instance, be a concatenation of the keys required to execute the functions *POE* and *POD* listed in Appendix A. The function takes the following three inputs:

- the *key* - a series of digits between 0 and 9 that Alice employed for her first and third passes and the *key* used by Bob in the second pass and as well as in the final decrypt;
- the *step* with the following input values: 1 - first pass, 2 - second pass, 3 - third pass and 4 - for the final decrypt;
- the Constant c which is required for the first pass (step =1) only, i.e. $c \gg 1$ is required to seed the initial ciphertext by Alice; NB: Bob need not be aware of this

constant's value, and it is only utilized in step 1 of the process. Thus, it can be randomly selected.

The ciphertext is written to a file called Ciphertext.txt in steps 1, 2, and 3. It is assumed that these files were transmitted through email from Alice to Bob in step 1, from Bob back to Alice in step 2, and from Alice back to Bob in step 3. The plaintext file is produced to the file, Plaintext.txt, after step=3, in which Bob decrypts the ciphertext file. In every situation, reading and writing data to and from files are done using the MATLAB functions *dmlread* and *dmlwrite*.

As an illustration, and in the context of how function operates, *TPP* given in Appendix B, take into account the key exchange provided by the array 80 97 115 115 119 111 114 100 which is the character string's ASCII decimal integer representation of the text *Password*. It is under the pretext that Alice generates a file called 'Plaintext.txt' containing the specified integers. Alice runs the function (for $opt = 1$) *TPP(1234,1,123456789)* where 1234 is Alice's key (known only to her), setting the value of c to 123456789 (also known only to Alice). Keep in mind that Alice can use any value of c up to the threshold associated with the available precision that needs to be quantified; the precision is set to 32 bits in the function *TPP*.

After receiving the file 'Ciphertext.txt' from Alice, Bob executes the function *TPP(4321,2)* with $opt = 2$, where 4321 is Bob's secret key. Bob then sends the resulting document, 'Ciphertext.txt' back to Alice. When Alice receives Bob's cipher, she executes the function *TPP(1234,3)* and sends Bob the updated ciphertext file, 'Ciphertext.txt'. When Bob receives this file, *TPP(4321,4)* is executed to retrieve the key, which is recorded in the output file 'Plaintext.txt'. If Alice and Bob, respectively, employ identical private keys for steps 1 and 3 and steps 2 and 4, the output file will contain the integer array $80, 97, 115, 115, 119, 111, 114, and 100$. Consequently, Bob receives Alice's key using a TPP.

The input is zero padded in Step 1 of the function *TPP*, adding a zero to the first element of the array. The re-normalization condition, which is used in step 4 of the same function when the first element of the decrypt is removed from the output, is the cause

of this. Re-normalization then becomes a natural part of the process and is unrelated to the input or output plaintext that Alice or Bob sends or receives.

The Red, Green, and Blue components of the algorithms provided in Appendix A each require three keys, hence the function *TPP* can be used to exchange these keys by utilizing a concatenated key;

$$key\_R \parallel key\_G \parallel key\_B$$

The fact that this concatenation is for an array of integers rather than a string of numbers is noted in the code. The length of the numeric strings representing each component key is limited to 10 digits, and the components of the array associated with each key are then concatenated into a string for usage in functions *POE* and *POD* (the highest limit of the restricting range for the MATLAB random number generator utilized in these functions).

This method can be expanded to include an exchange of the Spectral Embedding Constant c when the Appendix A functions *POE* and *POD* have the option to apply the WPOE selected. By utilizing a second application of the function *TPP* it is also possible to exchange the value of c independently. It should also be noted that the function "it TPP" can be used to pass any plaintext that is supplied as an ASCII decimal integer stream, as shown in the example above.

A straightforward graphical illustration of the data connected to function TPP is provided in Figure 7.1 It displays line graphs of the three passes' associated output ciphertexts and the input plaintext, which is a single sentence made up of 114 ASCII characters (steps 1-3) for $c = 1234.56789$ using the keys 1234 and 4321 for *step*=1, *step*=3 and *step*=2, *step*=4, respectively. These outcomes are connected to executing the function TPP in the following order: *TPP(1234,1,'Plaintext.txt',1234.56789), TPP(4321,2), TPP(1234,3), TPP(4321,4,'Decrypt.txt')* where Plaintext.txt is the input plaintext file (created by Alice), and Decrypt.txt is the result of the final decrypt (produced by Bob using the *step = 4*. Each Ciphertext.txt file is made up of an array of floating point values with 32 digits whose numerical scale is affected by the value of c but not directly related

to it.

It should be observed that while none of the three ciphers statistically correlate with one another, however, each cipher bears the same statistical distribution, thus, they do correlate with each other, statistically.

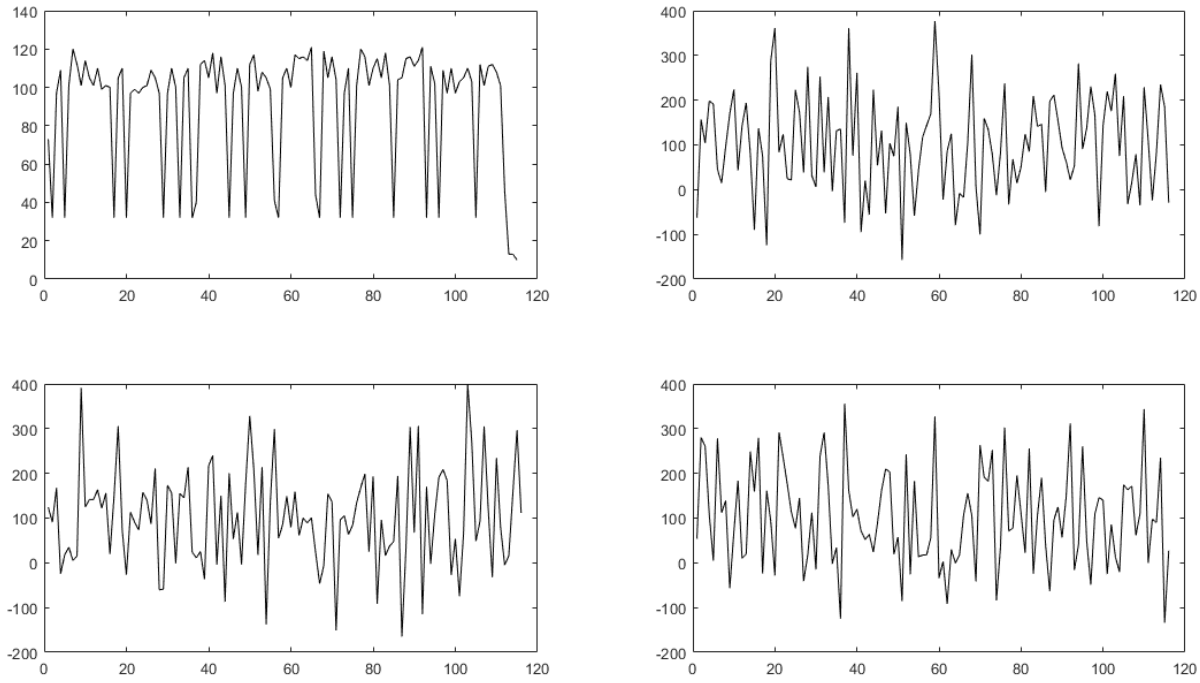

Figure 7.1: Example plots of plaintext (top-left), first pass ciphertext (bottom-left), second pass ciphertext (top-right) and third pass ciphertext (bottom-right).

The strategy under consideration is based on demonstrating the conclusion that an accurate deconvolution is possible given Equation (7.5) if the noise function $n(x)$ given in Equation (7.1) is assumed to have the same phase-only spectrum. This outcome has been applied to the three-pass protocol used for plaintext exchange without the requirement of key exchange.

According to the cryptanalysis results as shown, the arbitrary value of the Spectral Embedding Coefficient c used in the first pass — which can range from many tens of orders of magnitude depending on the available floating point precision—results in a third order polynomial equation that is under-determined, that is, an equation with no single

solution, if the ciphertext is intercepted in each pass. Because there is no known solution to the one-dimensional phase retrieval problem as a result of the Fundamental Theorem of Algebra, Bayesian and correlation assaults have been proved to be fallacious in the latter situation. In this sense, the approach is impenetrable, however future exceptions are always a possibility.

Any ACSII text data linked with a user-specified plaintext file can therefore be exchanged using the procedure described in Appendix A's function TPP. In order to encrypt multiple data types, the function can be adjusted and extended as needed. For example, the value of c used in the first pass can be automatically generated using the code:

```
c=rand(1,1)*10^(round(rand(1,1)*6))
```

This assigns a genuine random floating point value between and 0 and 10^6.

It is clear that the use of phase retrieval techniques might be used to attack the implementation of phase-only encryption for a TPP key exchange. However, the usefulness of phase retrieval techniques depends on the dimension. The phase retrieval issue has a number of well-known solutions based on iterative numerical techniques for $\mathbf{r} \in \mathbb{R}^2$. However, the phase retrieval problem for $\mathbf{r} \in \mathbb{R}^1$ is vague, making it difficult to identify the phase within the broad solution space. This problem can only be taken into consideration with certain *a priori* assumptions or with the use of extra data.

Therefore, it may be considered, at least for the time being, that an attack involving the application of a one-dimensional phase retrieval algorithm to any one of the passes is irrelevant (particularly, Steps 2 and 3 mentioned in Section 7.3). This assertion needs to be understood in light of potential future fixes for the one-dimensional phase retrieval issue. For instance, it has recently been demonstrated that, in the presence of interference measurements between an unknown signal and a reference signal (unrelated to the unknown signal), a signal can be uniquely retrieved from the Fourier amplitude alone [22]. However, the use of phase-only encryption for $\mathbf{r} \in \mathbb{R}^1$ and the key exchange solution exacerbated the problem until one-dimensional phase retrieval techniques became as easily applicable and tractable as those proposed for the two-dimensional case. For cryptanalysts, *TPP* is

difficult to break, to the author's knowledge. In this sense, it could be argued that rather than using the function to exchange only the keys used for executing the functions listed in Appendix B, plaintexts in general should be transferred using variations on the theme of function *TPP* (to include generic data I/O, for example), including digital images. A modified and additional method of exchanging a digital (full-color) image is discussed in the section that follows.

7.6 Image Exchange using a Three-Pass Protocol with Phase-only Encryption

Let's say Alice and Bob want to share an image that is provided by the actual two-dimensional function. $f(\mathbf{r}) \leftrightarrow F(\mathbf{k})$. Alice generates the random phase cipher $\Theta_1(\mathbf{k})$ and similarly, The random phase cipher is generated by Bob, $\Theta_2(\mathbf{k})$. Using the Fourier transform, these phase functions are compounded in Equation (7.6). By using an evolutionary computing technique, the algorithm(s) for generating $r(r)$ from which these phase functions are produced are assumed to be cryptographically secure and preferably unique to Alice and Bob, including the keys used to seed them [34]. The next steps are implemented.

Step 1: Alice encrypts $F(\mathbf{k})$ to produce ciphertext $C_1(\mathbf{k})$ using the equation

$$C_1(\mathbf{k}) = F(\mathbf{k}) \exp[i\Theta_1(\mathbf{k})] \tag{7.7}$$

and sends $\mathrm{Re}[c_1(\mathbf{r})]$ of $c_1(\mathbf{r}) \leftrightarrow c_1(\mathbf{k})$ to Bob.

Step 2: Upon receiving the ciphertext $C_1(\mathbf{k}) \leftrightarrow \mathrm{Re}[c_1(\mathbf{r})]$, Bob encrypts $C_1(\mathbf{k})$ using the equation

$$C_2(\mathbf{k}) = C_1(\mathbf{k}) \exp[i\Theta_2(\mathbf{k})] \tag{7.8}$$

and sends $\mathrm{Re}[c_2(\mathbf{r})]$ of $c_2(\mathbf{r}) \leftrightarrow C_2(\mathbf{k})$ back to Alice.

Step 3: Alice decrypts Bobs ciphertext $C_2(\mathbf{k}) \leftrightarrow \mathrm{Re}[c_2(\mathbf{r})]$ using the equation

$$\begin{aligned} C_3(\mathbf{k}) &= C_2(\mathbf{k}) \exp[-i\Theta_1(\mathbf{k})] \\ &= F(\mathbf{k}) \exp[i\Theta_1(\mathbf{k})] \exp[-i\Theta_1(\mathbf{k})] \exp[i\Theta_2(\mathbf{k})] \\ &= F(\mathbf{k}) \exp[i\Theta_2(\mathbf{k})] \end{aligned} \tag{7.9}$$

and sends $\mathrm{Re}[c_3(\mathbf{r})]$ of $c_3(\mathbf{r}) \leftrightarrow C_3(\mathbf{k})$ back to Bob.

Step 4: Bob decrypts the ciphertext $C_3(\mathbf{k}) \leftrightarrow \mathrm{Re}[c_3(\mathbf{r})]$ using the equation

$$F(\mathbf{k}) = C_3(\mathbf{k}) \exp[-i\Theta_2(\mathbf{k})] \tag{7.10}$$

The plaintext is then given by $\mathrm{Re}[f(\mathbf{r})]$ where $f(\mathbf{r}) \leftrightarrow F(\mathbf{k})$.

7.7 Cryptanalysis

When all three ciphers are decrypted (a three-pass intercept), as well as when only one cipher is decrypted, two attack strategies are taken into consideration. In the latter scenario, we assess the use of phase retrieval techniques to launch an attack without knowing $f(r)$ *a priori*.

7.7.1 Three-pass Interception

Assume that an attempt is made to calculate $f(r)$ using the three-pass protocol described in Section (7.6) and accurate and complete recordings of the ciphers $C1(k)$, $C2(k)$, and $C3(k)$ obtained by intercepting the transmission associated with Steps 1-3 and recording the Fourier transform of the outputs. Given Equations (7.7) - (7.9), the following ciphers can be eliminated Θ_1 and Θ_2 to obtain the Equation (7.11)

$$F(\mathbf{k}) = \frac{C_1(\mathbf{k})C_2^*(\mathbf{k})C_3(\mathbf{k})}{\mid C_2(\mathbf{k}) \mid^2}, \mid C_2(\mathbf{k}) \mid > 0 \tag{7.11}$$

However, since $\mid C_2(\mathbf{k}) \mid^2 = \mid F(\mathbf{k}) \mid^2$ we can write Equation (7.11) as $F(\mathbf{k}) \mid F(\mathbf{k}) \mid^2 = C(\mathbf{k})$ where $C_1(\mathbf{k})C_2^*(\mathbf{k})C_3(\mathbf{k})$ and it is clear that to obtain $F(\mathbf{k})$ we are required to solve

a cubic equation, a solution that is given by (obtained using [199])

$$F(\mathbf{k}) = \frac{C_r(\mathbf{k})}{\mid C(\mathbf{k}) \mid^{\frac{2}{3}}} \pm i \left(\frac{C_r(\mathbf{k}) - F_r^3(\mathbf{k})}{F_r(\mathbf{k})} \right)^{\frac{1}{2}} \tag{7.12}$$

where $C_r(\mathbf{k})$ denotes the real component of the spectrum $C(\mathbf{k})$. This solution for $F(k)$ is not unique, which causes problems in the 'plaintext solution' as the imaginary component can be either positive or negative. Moreover, both Equations (7.11) and (7.12) have the potential to incur singularities.

In regard to Equation (7.11), this occurs when the power spectrum $\mid C_2(\mathbf{k}) \mid^2$ approaches zero. Therefore, 'forcing' the spectrum $C_2(\mathbf{k})$ to have very low numerical values at the extreme limit of the floating point accuracy available to Alice and Bob, will have an impact, which will be to generate an inverse filter $1/ \mid C_2(\mathbf{k}) \mid^2$ that is singular, thereby eliminating the potential of an attack based on a three-pass intercept. To implement this effect, we introduce an Exponential Scaling Factor (ESF) denoted by α in the second pass (Step 2) and final decrypt (Step 4). This means we will modify Equation (7.8) and (7.10) to the forms $C_2(\mathbf{k}) = C_1(\mathbf{k}) \exp[i\Theta_2(\mathbf{k}) - \alpha]$ and $F(\mathbf{k}) = C_3(\mathbf{k}) \exp[-i\Theta_2(\mathbf{k}) + \alpha]$, respectively. The value of α that is applied depends on the floating point accuracy that is available. In the MATLAB code presented in Appendix B.2, we set $\alpha = 500$ (specifically in the function IF), a value that causes the numerical evaluation of Equation (7.11) to generate singularities, while maintaining a decrypt that is an identical replica of the plaintext (in a least squares sense), both the plaintext and decrypt being taken to be MATLAB generated images with .JPEG extensions.

7.7.2 Phase-Retrieval

Take into account the scenario where the ciphertext from Step 1 is intercepted, from Equation (7.7) the amplitude spectrum $\mid C_1(\mathbf{k}) \mid = \mid F(\mathbf{k}) \exp[i\Theta_1(\mathbf{k})] \mid = \mid F(\mathbf{k}) \mid$ can be constructed. This yields a decryption problem that is equivalent to the phase retrieval problem, i.e. given that $f(\mathbf{r}) \leftrightarrow F(\mathbf{k}) = \mid F(\mathbf{k}) \mid \exp[i\Theta_1(\mathbf{k})]$, then if and only if $\mid F(\mathbf{k}) \mid$ is known, we are required to estimate the phase function $\Theta_1(\mathbf{k})$ upon which $f(\mathbf{r})$ can be obtained by Fourier inversion. Reconstructing an image from its Fourier magnitude alone is a

very difficult task given that the Fourier phase is significantly more important than Fourier magnitude in the reconstruction of $f(\mathbf{r})$. The phase-retrieval problem is severely ill-posed with no uniformly stable solutions. Nevertheless, phase estimation algorithms have been developed to provide approximate solutions in two-dimensions, but the uniqueness and stability of such algorithms depends on $f(\mathbf{r})$ being composed of well differentiated features when the image is relatively sparse, i.e. a binary image. Otherwise, the problem requires *a priori* information on the function $f(\mathbf{r})$, e.g. [111] and [121]. In cryptography this is equivalent to having a Crib. Although (passive) Cribs exist in a written plaintext (in terms of expected letters, words and phrases, for example), an image plaintext does not generally have an equivalence. Application of the algorithms given in [130] [131], for example, reveal that it is possible to decrypt RGB components independently, but only as binary representations of these components when all 8-bit gray level details are lost. However, to make this form of attack altogether null and void, we can replace Equations (7.7) - (7.8) with the following: $C_1(\mathbf{k}) = F(\mathbf{k})(\beta\pi + \Theta_1)\exp[i\Theta_1(\mathbf{k})], C_2(\mathbf{k}) = C_1(\mathbf{k})(\beta\pi + \Theta_2)\exp[i\Theta_2(\mathbf{k})], C_3 = C_2(\mathbf{k})\exp[-i\Theta_1(\mathbf{k})]/(\beta\pi + \Theta_1)$ and $F(\mathbf{k}) = C_3(\mathbf{k})\exp(-i\Theta_2(\mathbf{k})]/(\beta\pi + \Theta_2)$, respectively, where, given that $\Theta_n \in [-\pi, \pi],\ n = 1, 2,\ \beta$ is a constant which, in practice, is adjusted to provide a perfect decrypt for some minimum value of $\beta > 1$.

7.8 Prototype Algorithm for $\mathbf{r} \in \mathbb{R}^2$

The MATLAB prototype software in Appendix B.2 implements the algorithm from Section 7.6. The primary function of NKP, also known as No-key(s) Protocol, is exacerbated by this. Three external functions are dependent on this function: Function POX phase-only encrypts and decrypts the data; function WTI saves the ciphertext to a file using a tagged image file format, maintaining the floating point accuracy associated with the encryption and decryption process; and function IF makes an attempt to attack the data and recover the plaintext under the assumption of a three-pass interception using Equation (7.11), which includes the Inverse Filter $\mid C_2(\mathbf{k}) \mid^{-2}$ Two users can exchange a JPEG image using the function NKP (Alice and Bob). It has two inputs and separately encrypts

and decrypts the RGB parts of a full 24-bit color image.

(i) The *key* a series of digits from 0 to 9, is used. The ASCII decimal numbers for the R, G, and B (color components) provided by 82, 71, and 66, respectively, are multiplied by this key. This is done in function POX in order to give unique encryption keys for each color component, a strategy that can be expanded upon as needed. Due to the restricting upper bound for a MATLAB random number generator with a non-negative integer seed $< 2^{32}$, this necessitates that the length of the key supplied into function NKP be no more than 7 digits. Alice makes the first and third passes using the key. Bob uses a different key on the second and final pass.

(ii) The *step* with input values 1 (first pass), 2 (second pass), 3 (third pass), and 4 has been given (for the final decrypt).

The concept behind the function NKP is the reading and writing of explicitly specified files. The standard input file is 'Image.JPEG' which can be any colour JPEG image. This image's RGB color components are extracted, transformed to double precision, and the rebuilt color image is written to the default file 'Plaintext.JPEG'. This serves as a check to make that the plaintext is entirely compatible with the decrypt within the processing environment being utilized, with the decrypt being produced to the default file 'Decrypt.JPEG' by running function NKP for *step*=4. For *step*=1, 2, 3, the ciphertext is written to (and read from, for *step*=2 and *step*=3) default file 'Ciphertext_*step*.tif'. These tif files are presumably sent from Alice to Bob (it step=1), from Bob back to Alice (it step=2), and from Alice back to Bob (it step=3) by email, for example. Bob does a final decrypt after receiving the ciphertext file 'Ciphertext_3.tif to retrieve Alice's plaintext image 'Plaintext.JPEG' which is produced as a JPEG image to 'Decrypt.JPEG'. This stage assesses the decryption's accuracy using a least squares test and makes an attempt to retrieve the plaintext under the assumption that each pass has been successfully intercepted..

Figure 7.2 shows an examples of three-pass ciphertext's and the decrypt for a full color test image based on the application of function NKP.

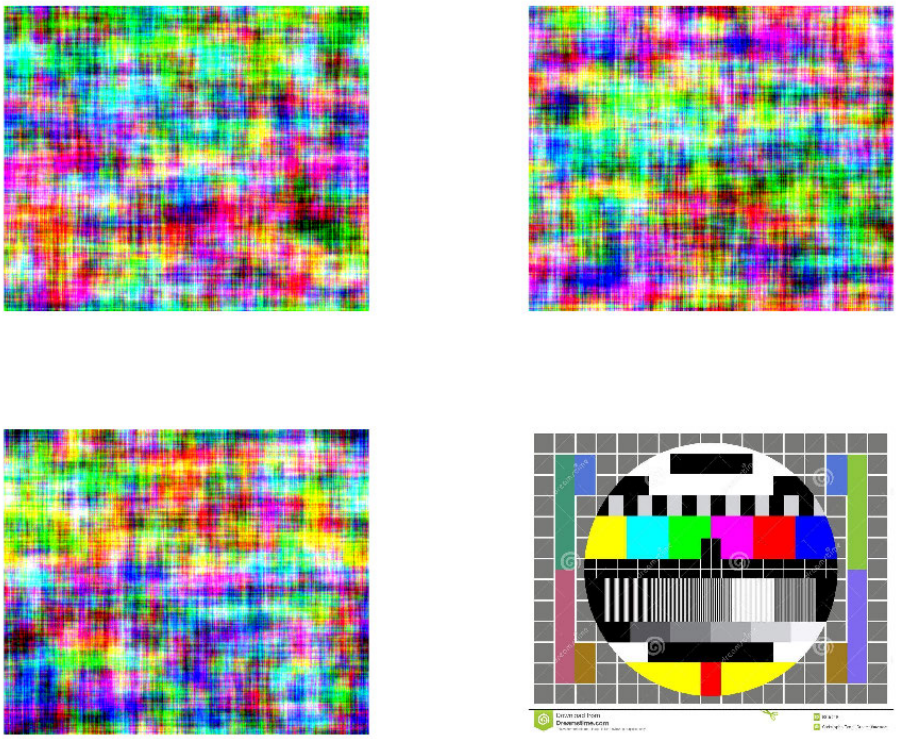

Figure 7.2: Example of the application of function NKP for a test-image showing the first pass ciphertext (top-left), the second pass ciphertext (top-right), the third pass ciphertext (lower-left) and the decrypt (lower-right).

7.9 Conclusion

A phase-only encryption mechanism provides the foundation for the information in this chapter. A deconvolution problem that is well-posed results from this method. To produce a decrypt, this problem can be solved precisely and only once. Knowing the phase-only cipher, which in turn depends on knowing the key used to compute the cipher, is therefore necessary for the solution. Based on this outcome, we have thought about a Three-pass Protocol to create a No-key(s) Protocol algorithm, allowing Alice and Bob to not disclose keys. Passing a different cipherext three times across an open network is the cost of doing this.

The algorithms can be modified to exchange other types of text and picture formats

as needed. They are currently designed to communicate plaintext and a JPEG-formatted image. Due to the singularities connected to the inverse filter function in Equation (7.11), cryptanalysis demonstrates that, if the ciphertext is intercepted on each pass, decryption is an ill-posed task. This property has been taken advantage of by adding an exponential scaling factor that 'forces' the inverse filter to become singular (within the used floating point accuracy), so preventing an attack. This prevents a Man-in-the-Middle (MITM) attack. A phase-retrieval algorithm assault can also be stopped by scaling each step with the corresponding phase function (subject to the additive constant $\beta > 1$)).

Chapter 8

The Cloud Computing Paradigm - Overview

Technology rapidly changes on a daily basis. Hence, the technology curve is rising exponentially. Among the new and emerging technologies is the cloud computing paradigm. This paradigm shift has brought about changes in the technology landscape and has certainly become a hub of most services offered through the Internet. For example, software can now be provisioned as a services (SaaS), organisations can now use leased infrastructure (IaaS) etc. These services can be accessed on a pay-per-use model. The emergence of this new paradigm is being accepted by both cloud service providers and users. This is because a lot of technologies are converging towards the cloud while other heterogeneous systems are contributors via various applications. Thus, the cloud computing paradigm is seen as a technology provider and an enabler.

The amount of data being transferred to the cloud is massively growing due to technological advancements and trends. Innovations in manufacturing new digital devices such as smart-phones, wearable devices, tablets etc. connecting to cloud services, and introducing them into the market has also catalyzed the wide adoption of the cloud. Cloud consumers generate vast amounts of digital content such as videos, pictures, documents etc., and upload them to the cloud, for processing and/or storage. This has resulted in the

cloud becoming a data lake. This mass proliferation of cloud-hosted data was predicted to increase exponentially by the year 2020 [88]. While the cloud brings a lot of advantages, its adoption requires effective, efficient and secure mechanisms aimed at providing security of cloud-hosted data.

The emergence of the Internet of Things (IOT), which is evolving to the Internet of Everything (IOE) has also increased the amount of data processed and/or stored in the cloud exponentially with many digital objects able to communicate and share signals. Today, the cloud is 'flooded' with data from the 'data lakes' as Zettabytes of data are uploaded to the cloud [151]. This widespread use of cloud services stems from a number of organisations and cloud consumers seeking shifting from on-premise deployments to cloud-hosted deployments for robust and resilient solutions [198][181]. Among others, these solutions provide benefits including scalable and flexible deployments which are cost effective and elastic in nature [15].

Despite the various benefits that are attracting cloud adoption and bringing a lot of innovative solutions, the cloud computing paradigm is plagued by security threats. These threats have made the cloud an inherently risky space, especially on data security. Due to this, some cloud consumers are hesitant to adopt the cloud due to security hindrances surrounding data confidentiality. Most organisations address these concerns through the use of policies, procedure manuals, regulatory and legislative frameworks, patch management, service level agreements (SLAs) etc., to ensure data confidentiality guarantees are met [180]. However, these governance structures are not enough to ensure data confidentiality as practical implementation of cryptographic mechanisms must be met. For example, to ensure data confidentiality, cryptographic encryption methods must be implemented. For example, major cloud service providers (CSPs) constantly release updates and upgrades of their security patches to ensure that they stay ahead of cybercriminals who are ready to exploit any vulnerabilities they can find. The Amazon Web Services (AWS) released important security patches on the 30th September 2022, to strengthen their AWS solution with a host of patches called 'Security 2' on their security center and web services [119] [].

8.1 Virtualisation - The fundamental cloud computing platform

The underlying foundation of cloud computing is virtualisation. The cloud infrastructure heavily relies on the virtual environment made up of several virtual machines (VMs). The virtualisation concept ensures that infrastructural needs are made available to cloud consumers, in a leased manner, using VMs isolated and provisioned to the tenants [145]. The threats discussed herein affect virtualised environments, even if they are not in a cloud setup. Virtualisation is achieved through multi-tenancy. This is an architecture where a single instance is deployed over multiple servers and accessible to multiple cloud tenants. This architecture makes cloud computing possible. Multiple cloud users (i.e. tenants) access cloud storage in a seamless manner, each having direct access to their own tenant, accessible using a tenant ID. This architecture makes it possible to efficiently manage shared resources residing on one physical infrastructure, logically separated by a hypervisor to support sharing of resources such as memory and processing capabilities. This concept makes it plausible for conflicting cloud users (tenants) to have their VMs co-residing on the same physical machine. For any advanced persistent threat actors (APTAs), this can be used as a vulnerability and exploited through VM sprawl and inter-VM attacks [88], [69] and [171] resulting in data leaks. If a cloud service provider has deficiencies in the configurations made on the infrastructure, data leaks are possible and can spread to other tenant VMs. Once this is exploited, data confidentiality will be compromised. These VM-level attacks are made possible by malicious VMs which target other co-residing VMs on the shared environment. To execute such attacks, APTAs 'hijack' the underlying hypervisor. Hence, this is also known as 'hyper-jack' attack. If 'virtualisation-aware' security is not implemented and a 'hyper-jack' succeeds, the APTAs can easily increase the attack surface through lateral movements to co-resident VMs [145]. To implement security on the virtual environment, virtualisation-aware security safe-guards must be implemented. Attacks on the virtual environment have a cascading effect on the entire virtual environment if lateral movement is used [6].

Due to lack of data isolation, multi-tenant infrastructure a prime cyber-target. These cyber attacks can also be launched by malicious tenants, for example, competitors. These attacks can be invisible to other tenants due to co-residency. Thus, co-residing VMs are vulnerable. Co-residency increases the risks of inter-VM attacks [201] [163]. Another security risk is tenant workload interference. If one or more tenants create an overload, this can easily and negatively affect the workload performance of other tenants. Attacks emanating in this manner are due to workload interferences from multiple tenants co-residing on the same physical infrastructure. In practice, tenants may co-reside with direct competitors, or adversaries, only separated through logical mechanisms such as access control. Any vulnerability in the deployed access control has a potential of compromising data confidentiality as resources will be accessed by unauthorised entities [160].

8.1.1 Cloud Computing Security Challenges

Section (8) highlighted an overview of the cloud computing paradigm. It outlined this paradigm shift as a new technological trend which is embraced by many across the ICT industry. The reason it becoming more appreciated is due to its seamless integration with the new and emerging technologies, especially the Internet of Things (IOT) and becoming a digital business enabler. The IoT is heavily reliant on cloud computing. It also enables cloud users to choose the services they need, on demand. It provides huge amounts of storage space. This has helped many business in migrating their data to the cloud. For example, tenants can upload their data onto the cloud and access it on demand togehter with other applications and services provisioned using a shared pool of resources [160].

The cloud computing paradigm is attractive and comes with various benefits including scalable and flexible attributes. However, tenants still rely on the cloud provider to secure their cloud-hosted data. The provider is in total control of the tenant's data once it is migrated to the cloud. This raises concerns over data confidentiality as insider attacks can be launched by disgruntled cloud service provider's employees. Therefore, tenants must be aware of threats involving delegation of powers, dependencies, data customisation

etc., by the CSPs before considering cloud services. Despite these security concerns, the amount of data flowing towards the cloud is exponentially increasing. With such increase comes high security risks with potential data leaks, either accidental or deliberate. Due to outsourced services, the cloud does not guarantee physical, logical and personal control over the data is hosts. Once the data is on the cloud, the CSPs are burdened with data security guarantees although the CSPs clearly state in their agreements with tenants that they are not accountable in the event that a security breach materialises. For example, AWS SLA states that the provider will 'strive' to keep the data secure, but there is no guarantee that cyber attack will not be completely foiled [105]. Such terms and conditions prove that even-though CSPs are in business, they are aware of the security related risks associated with cloud adoption.

This poses an intrinsic risk threat to users utilizing cloud services. Thus, as one of the risks, for example, data leakages compromise the confidentiality of cloud-hosted data [166]. When there are weak or no confidentiality guarantees, the consequences of data breaches can have a cascading effect towards the entire cloud computing model. This is because cloud service models are inter-related and interdependent. Due to these dependencies, any attack launched on any of the service layers can compromise other layers due to a ripple effect, thereby affecting confidentiality of data stored in the cloud. Thus, the following sections delve into cloud storage security risks.

8.1.2 Encryption and Key Management in the Cloud

Data security is the responsibility of a CSP. Cloud tenants may send unencrypted data to the cloud, using connections that are not secure. This is susceptible to Meet-in-The-Middle (MITM) attacks as an AT CSPs offer data encryption in the cloud; the data can be uploaded to the cloud unencrypted. This increases the threat landscape as the likelihood of insider attacks, man-in-the-middle attacks, identity theft, and other threats grows. This is such that dangerous threat actors can gain unathorised access to data storage facilities. The major problem with cloud computing is physical identification, which makes such

attacks possible. Data encryption, which is carried out via cryptographic techniques, should be maintained as a tool for enforcing data confidentiality to reduce such risks. Encryption key management is the primary difficulty in Cryptography.

In an effort to address key management difficulties, Trusted Third Parties (TTPs) are frequently introduced into as key storage facilities or key distribution centers (KDCs) [73]. Data confidentiality is still threatened by the fact that tenants' data is kept in distributed data centers and that a TTP provides and/or stores the encryption keys because TTPs are also vulnerable to attacks. For example, if a KDC is compromised, all the encryption keys managed by the KDC will be compromised. It should also be noted that the ultimate goal of cyber attackers is to reveal the encryption key(s). To enable covert surveillance on the cloud using new technologies, government agencies have proposed a range of encryption mechanisms, key recovery strategies, and TTP encryption standards. Additionally, it gives government organizations a chance to create 'back-doors' into cloud user's data. To deal with these situations, data protection methos and protocols such as the Secure Sockets Layer (SSL) or its variant Transport Layer Security (TLS) can be used to safeguard data while in transit to handle such situations and prevent compromising the secrecy of cloud-hosted data [73], [132], [84], [173].

Homomorphic encryption is a type of encryption technique that allows computation on encrypted data without the need to decrypt it first. In other words, homomorphic encryption enables the processing of data while still keeping it encrypted, which is useful for maintaining privacy and security of sensitive information [90] [91].

Homomorphic encryption is based on mathematical algorithms that enable operations on encrypted data, such as addition and multiplication, to produce encrypted results that can be decrypted to give the same output as if the operations had been performed on the original unencrypted data. This means that data can be processed without ever being revealed in its unencrypted form, providing a higher level of security and privacy [9].

Homomorphic encryption aids in thwarting risks like data leaks when in use and when the data is at rest [68]. Malicious threat actors now see the cloud as the biggest and most populated data lake in order to exploit any potential weaknesses. The Dropbox hack,

in which user credentials were exposed, is a good example of a data leak [45]. The six million credentials that were stolen from eHarmony and LinkedIn are an additional factor. Because resources are shared across many cloud consumers, cloud computing represents a complex service offering [72]. For instance, using a public cloud, one of the guest virtual machines used by a malicious user may exploit a host machine hosting numerous virtual machines. The host machines' operating virtual machines might come to the attention of an attacker. In order to co-exist with the target VMs, the attacker may thus establish a VM [160] and launch an inter-VM attack or a VM-Sprawl attack 23. The guest VM may be utilized to leak sensitive data from the original VMs [72].

8.1.3 Partial Deletions

When it comes to migrating data to the cloud, there are opportunities for hazards that can jeopardize the confidentiality of cloud-hosted data, such as partial deletions. Such circumstances need the use of efficient data sanitation techniques. When erasing data stored in the cloud, extreme caution must be used. The deletion must be carried out entirely, not just partially. When a data owner deletes the data, but some copies of it remain on the CSP side, this is known as a partial deletion. Worse even, the full record may not be erased but instead remain in memory, known as a 'shadow' record. For instance, even after a cloud customer stops using the service and the customer deletes the data, vestiges of the data may still be present on the cloud provider's servers. The residual data retained on the physical hosts in a distributed cloud system may result in accidental data breaches or intentional exploitation of these vulnerabilities because of the virtual separation and lack of hardware separation. Additionally, data forensics tools can be utilized by law enforcement authorities as well as cyber criminals to recover erased data from the CSP servers. Additionally, an insider attack can be launched by a disgruntled employee to recover improperly destroyed client data. Such incidents include the hack of the eBay online store, in which deleted client data was recovered from eBay servers [72]. Facebook does not permanently delete user's data unless the account is permanently

deleted. If a user deletes a picture from a Facebook timeline, it is partially deleted and Facebook can bring it up as a memory. This is another example of partial deletion [55].

8.1.4 Insider Attacks

For the obvious reason that they are already a part of the firm and are seen as trustworthy, insiders can develop into a serious cyber danger to businesses. Since they do not need to scale external security fences, they can pose a greater risk than outside invaders, whether they are malicious or thoughtless. It's critical to comprehend the numerous insider threat elements linked with cloud security as businesses rapidly move services and data to the cloud. Employees who are unhappy with their jobs may turn against their employer, the CSP. Given the access and authority granted to these employees, insider attacks are the most difficult to fight against. This challenge affects all cloud deployments. As resources are shared among several tenants, cloud storage must provide data confidentiality. A client's data and computing processes must be kept private from unauthorized parties in order to be considered confidential. The same personnel who are the data's custodians frequently undermine confidentiality because of the loss of physical control over cloud-hosted data. Data co-residency in public cloud installations may result in issues that could compromise the privacy of data stored in the cloud. For instance, a former employee used malicious code to remove 456 virtual machines utilized by Cisco's WebEx Teams service after gaining illegal access to the company's cloud infrastructure. As a result, for two weeks, about 16,000 WebEx users were unable to access their accounts. The ex-employee gained access to cloud infrastructure and installed his code by using his understanding of the company's security measures and exploiting their flaws. Two-factor authentication and other access management methods did not appear to be used to protect access to critical resources [106]

The history of cloud computing in this setting, where cryptography is routinely used to safeguard data confidentiality, is provided in the section that follows.

8.2 Background and definitions

There are numerous definitions of cloud computing as defined by numerous scholars. As an illustration, cloud computing is a model that offers universal, on-demand access to a shared pool of reconfigurable resources in the most practical way [143]. Networks, servers, apps, storage, and services are a few examples of these resources. Hardware, platforms, infrastructure, and software are all abstracted in this computing approach. Location independence, resource pooling, scalability, and elasticity are the five key components of a cloud computing paradigm. There are also four deployment methods (public, private, community, and hybrid clouds), and three service models (i.e. infrastructure as a service, software as a service and platform as a service) [143]. Cloud computing is a derivative of grid computing. Grid computing is characterized as an infrastructure that offers dependable, consistent, widespread, and inexpensive access to computational resources using both hardware and software[197]. Hardware virtualization is a prerequisite for cloud computing. Cloud computing is enabled by virtualisation technologies on all fronts. In its implementation, a hypervisor application is installed on a physical host machine. The real machine serves as a host for the virtual machines that the hypervisor constructs. The virtual machines can completely replicate physical machines and computers. Any program, including operating systems and applications software [71], can be run on the virtual computers. In data centers, physical hardware such as hard disks, processors, and network devices are handled. The storage and processing needs are handled by the data centers. This idea allows for the dynamic allocation of physical resources to cloud users. These resources include networks, storage, and computing power. Users of the cloud receive these resources as services.

The main distinction between grid and cloud computing is virtualisation [79]. The section that follows covers the details of various service and deployment models, and cloud computing's fundamental properties for deployment. It goes on to detail how cloud users can utilize the cloud to gain access to on-demand services, ubiquitous network connection, quick elasticity, and location freedom [153]., provided by the cloud computing architecture

as depicted in figure 8.1.

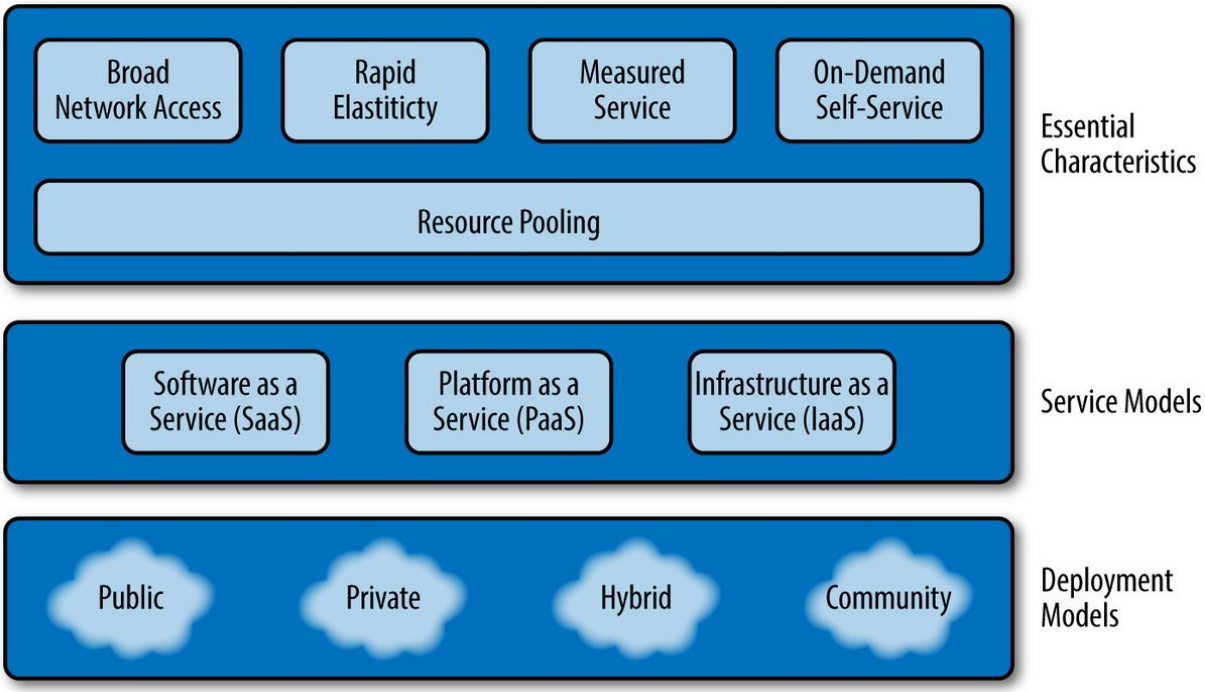

Figure 8.1: Cloud Computing Model Architecture [153].

8.2.1 Cloud Computing Services

The three different cloud computing service models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), as was previously mentioned. These service delivery models offer cloud users access to infrastructure resources, application platforms, and software as services. **IaaS** serves as the basis for these cloud services. In the services stack [153], **PaaS** is constructed on top of **IaaS** and **SaaS** is built on top of **IaaS**. The next subsections now present each of these service models.

IaaS

This is the most significant cloud computing service model made possible by the idea of virtualization, which offers networking, load balancing, and virtual machines (VMs).

The responsibility for giving cloud customers access to basic computer resources, such as processing power, storage, and networking, rests with **IaaS**. The infrastructure needed to run applications and system software on demand is provided by this service to users. Users of cloud services can manage the infrastructure, including the storage, software, running applications, and some networking components. **IaaS** is the cloud stack's most expandable service delivery paradigm. Very few application-like functionality are offered by it. As storage capacity, for instance, resources are managed, pooled, and distributed. RackSpace [153], Amazon Simple Storage Services (S3) [13], and Amazon Elastic Compute Cloud [153][105] are some examples of service providers.

PaaS

This service delivery paradigm gives developers access to a toolbox that includes an Integrated Development Environment (IDE). Using the programming languages and tools provided by the CSP, developers may easily create, deploy, and manage their application software. The intention is to provide programmers the freedom to build custom software applications and install them on top of the available platforms. It enables programming environments to access more application packages [96]. Using the CSPs' capabilities, developers can also run specialized apps that are designed for a particular platform. **PaaS** provides a service that includes administration and the full software development life cycle (SDLC) for developers. This covers the original planning, development, deployment, testing, and maintenance of software. In order to run the developed applications, processing, storage, and networking resources are not a concern for application developers, testers, deployers, or administrators. Then, CSPs are in charge of separating the workspaces and apps of customers [96].

Due to the fact that software applications are created and used on **PaaS** clouds, the **PaaS** is generally the same as any traditional computing platform. Programming languages, operating systems, databases, and web services are also part of the **PaaS** service paradigm. **PaaS** is available, for instance, on the Microsoft Azure platform and Google App Engine [54]. Through an application interface, these platforms let developers

run and deploy services. Additionally, **PaaS** facilitates collaboration so that several people can work on the same project at once, boosting the efficiency of teams.

SaaS

CSPs are required to ensure that the software they offer is updated before deploying cloud services. **SaaS** can be combined with other service models to create new service offerings that can be supplied through adaptable networks [221]. One example is that cloud tenants can use the offered software applications on a pay-per-use model rather than purchasing and installing software on premises or their workstations. This service delivery uses Microsoft Office 365, Salesforce, and Google Docs [16][185][146]. CSPs provide software applications as a service by utilizing **SaaS**. This service model is entirely in charge of giving cloud customers the ability to use CSP software applications (like email and Microsoft Office) that are available as web services on a cloud infrastructure. Through a program interface, these software programs are made remotely available from any client device, via a program interface without being installed on the client device, like a web browser. Because **SaaS** allows for scalability and transfers management from users to CSPs, tenants who use this service do not have any influence or management of the underlying cloud architecture [143]. The CSP, which can be a cloud provider or a TTP organization, is in charge of managing the entire service, from the application level all the way down to the most fundamental infrastructure level. As part of the SLA, the CSP is also responsible for developing application-tier security policies. The CSP is required by the SLA to provide security guarantees to customers using **SaaS**. Before utilizing such services, clients must confirm that the SLA satisfies their requirements for data confidentiality. The **SaaS** model also saves cloud consumers money since they no longer need to invest in the infrastructure that houses the applications. On-demand access to **SaaS** is provided via the Internet. As a result, this service is both low-cost, mobile and location independent. Google Apps, Facebook, and Salesforce [225] are few companies that offer **SaaS**. Cloud computing deployment models are briefly described in Section (8.2.2).

8.2.2 Cloud Computing Deployment Models

Public, private, hybrid, and community clouds are the four deployment models that make up the cloud computing model. as shown in figure 8.1. These are discussed in the sub-sections below.

Public clouds

On the cloud providers' environment, there are public clouds. The general public community is welcome to utilize this deployment model. This deployment architecture illustrates the fundamental pay-per-use notion made available by the cloud computing paradigm. By utilizing the shared pool of resources and services available to them, other organizations can access the cloud services provided by other organizations through SLAs using this deployment paradigm. Therefore, this deployment does not need to be on-site. This feature of public clouds consequently lowers the expense of setting up and maintaining the infrastructure. For instance, the New York Times archive is made up of Terabytes (TB) of S3 storage and hundreds of Amazon EC2 instances. For millions of pieces of digital content, this storage acts as a digital library. All of this costs only a small portion of what conventional archiving methods do [221]. Concurrency control is another benefit of public clouds [160]. CSPs offer a variety of cloud services to numerous unrelated individuals, businesses, or other cloud suppliers who use the services concurrently and independently. Because it is the CSP's responsibility to offer sufficient security guarantees and maintain the protection of data hosted on a public cloud deployment, cloud consumers do not need to worry about maintenance or security settings. It is crucial to remember that in a public cloud environment, since the infrastructure is controlled by the CSP, all cloud customers share the same configured infrastructure.

Private Clouds

Internal clouds, commonly referred to as private clouds, are perfect for exclusive uses like computation and storage. A network that is privately controlled by a single organization

hosts the deployment model. Because the infrastructure is typically hosted on the organization's premises, this deployment offers the highest level of control over performance and dependability. Private clouds can, however, also be hosted off-site. The security configurations can be handled and controlled inside the boundaries of the organizations that own the cloud when using this form of cloud deployment. Clients from outside the organization's security perimeters may be given remote access. To maintain some kind of security, access must be watched over by a boundary controller, such as an intrusion prevention system. Figure 8.2 depicts an on-premises private cloud deployment model.

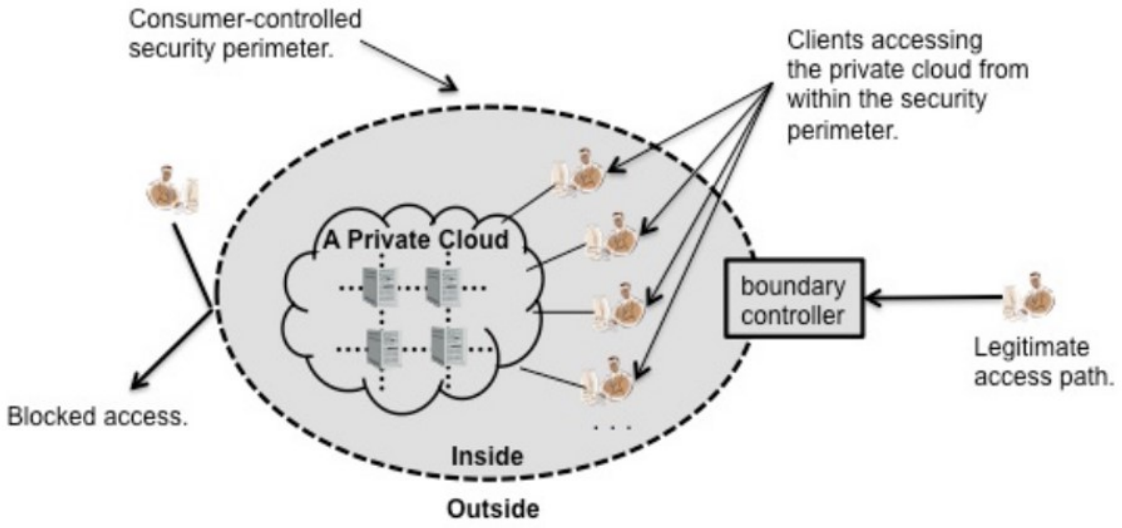

Figure 8.2: Private Cloud Computing Model[86].

Community Clouds

Community cloud models are set up for usage by a community of cloud users who share similar characteristics including policies, security needs, and compliance. Organizations have the option to share infrastructure resources under this arrangement. The deployment can either be on-premises or off-site as hosting options are available for the infrastructure. This cloud computing paradigm helps organizations overcome common issues like pricing,

technological complexity, security worries, and a lack of services tailored to particular industries. Organizations have realized that by collaborating to address shared obstacles and seize shared opportunities, better business outcomes may be provided for their cloud customers. Figure 8.3 is a scenario for establishing community clouds.

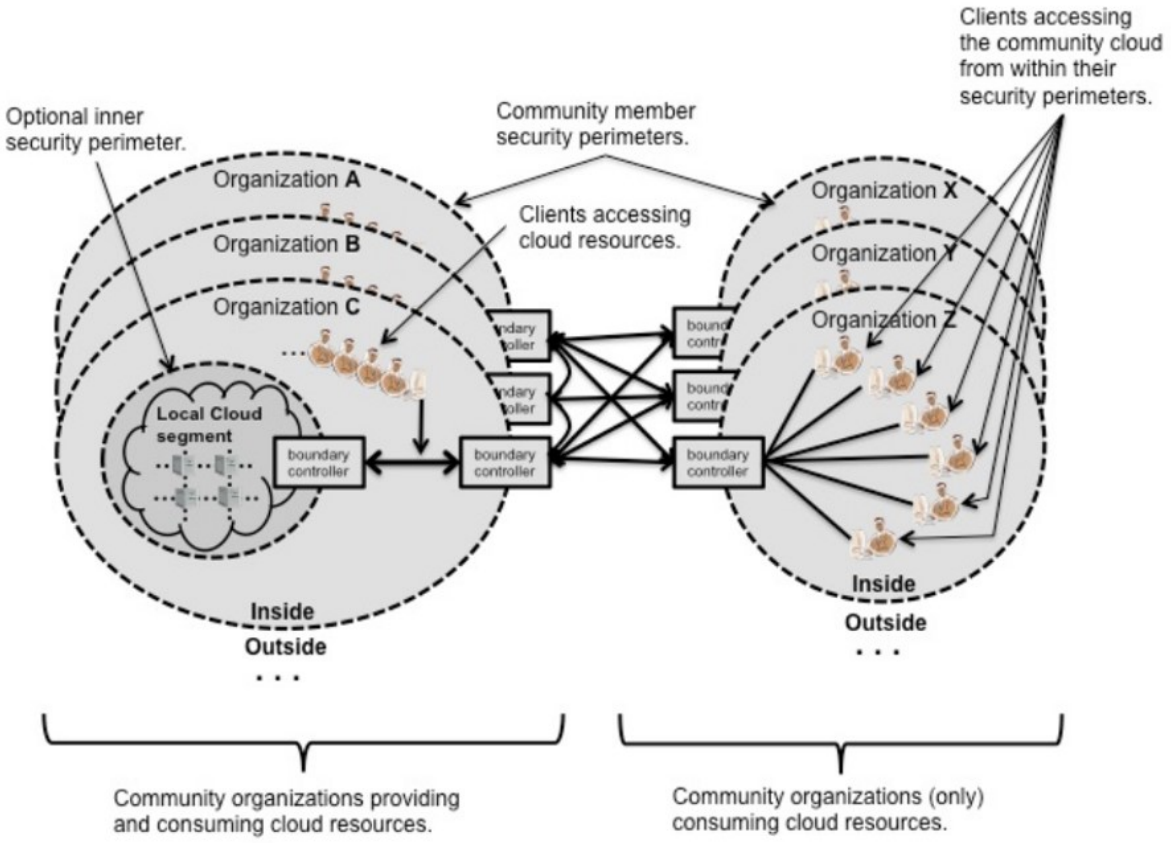

Figure 8.3: Community Cloud Computing Model[86].

From Figure 8.3, organisations on the left provide services to the organisations on the right. A perimeter wall divides the internal networks from the external networks to ensure security, using secure zones for network segregation. This is accomplished by employing logical separation mechanisms to segment the network. The boundary controller of each organization facilitates communication between the service-providing organizations and the customers. This boundary control may consist of Intrusion Prevention Systems (IPSs) or firewalls, on which connections, internal or remote such as Virtual Private Networks

(VPNs), terminate on.

Cloud-hosted data must be protected using a variety of technologies, particularly encryption and key management strategies. According to the literature reviewed in this chapter and others in this thesis, encryption is the most used data protection technique. Cloud-specific attacks, however, are still on the rise. To provide robust Cryptography and confidentiality guarantees in this situation, there are still gaps to fill in the key management and encryption space. The confidentiality of an encrypted message depends on the key's secrecy in terms of encryption. This is due to the fact that sharing the encryption key with other organizations or third parties violates the data confidentiality. For key management and revocations, the majority of the currently available and reviewed literature uses third-party Key Distribution Centers (KDCs) or Certificate Authorities (CAs). This idea requires a high level of trust as security is a web of trust.

However, regarding trust, none of the entities in the cloud can be fully trusted. As such, 'zero trust' should be implemented. Zero Trust is a cybersecurity framework that assumes that all resources, systems, and data in a network are potentially vulnerable and untrusted, regardless of whether they are located within or outside the network perimeter. Zero Trust principles require verification and authentication of every access request, whether from inside or outside the network, before granting access to resources [176][194].

In a Zero Trust model, access control is based on the principle of "least privilege," which means that users are granted the minimum level of access necessary to complete their tasks. Access is also continuously monitored and logged for auditing purposes. The Zero Trust model assumes that threats can come from both inside and outside the network, and therefore, continuous monitoring and analysis of all traffic and activities are essential to detect and mitigate potential threats [205].

'Zero-trust' must be used because of how the cloud ecosystem operates. Therefore, the shortcomings in key management and encryption must be fixed in order to maintain the privacy of data stored in the cloud. In order to fill the gaps found, this thesis suggests a model crypto-system. This model crypto-system is a lightweight, end-to-end, client-

side encryption system. This system seeks to reduce resource utilization (i.e. processing power and memory) as in the era of the IOT, devices are becoming smaller in size while processing power is doubled. Therefore, it is advantageous for deployment on hardware with modest memory requirements, such as mobile devices. It can, however, be deployed on any infrastructure.

8.3 Discussion

The fundamentals of cloud computing, including its service delivery models, deployment methodologies, and key features, were covered in this chapter. According to the discussions entailed herein, a cloud computing environment raises security issues. In light of this, it may be said that the cloud faces numerous security issues, including those relating to data security. The widespread adoption of cloud services is hampered by these issues. For instance, there are no guarantees for data confidentiality, data breaches, cyberattacks, etc. Symmetric and asymmetric techniques are applied to maintain data secrecy. Asymmetric systems are frequently used for key exchange protocols while symmetric schemes are frequently used for data encryption. This necessitates a hybrid of the two techniques. To prevent man-in-the-middle attacks, key management and exchange must be done securely. Even in the post-quantum computing era. Hence this thesis offers tenable answers to these problems. The proposed model, its implementation, and the methods utilized to actually employ the encryption models mentioned in this thesis are all described in the next chapter.

Chapter 9

Implementation and Methodology for the proposed Crypto-system

The preceding chapter covered the history of cloud computing. In an effort to resolve some of the security issues raised, it also discussed the difficulties with cloud computing security and the usage of cryptography. Before introducing the suggested model, this chapter first outlines the functional system requirements discovered as a result of the gaps discovered in the preceding chapter. The proposed model, which serves as the basis for the proposed crypto-system described, has the following important functional requirements:

(i) light-weight solution: The client-side cryptographic solution must be capable of encrypting data and deployable on devices with low specifications;

(ii) securely manage encryption keys - The system can securely and successfully manage encryption keys;

(iii) quantum data sharing- securely share data with other cloud users;

(iv) self-destructing capability where the encryption key contains a time-stamp for expiration.

9.1 System Architecture

Figure 9.1 depicts the proposed model of the crypto-system. The crypto-system is symmetric. It is built with machine learning ideas, specifically evolutionary computing. The proposed architecture is designed and implemented with each of the functional needs in mind. The model explains how encryption keys are managed, including key generation, storage, revocation and destruction, as well as the process of encrypting client-side data ready to be uploaded to the cloud. To initiate the process, real random noise is first gathered from a noise source. The noise is then transmitted via a secure channel and encrypted using Transport Layer Security (TLS) into the Eureqa cloud-based system. A non-linear fitness function that is the best approximation of the input noise is produced by the Eureqa system using the input noise [154]. Then, a symmetric key is created using the fitness function. This key is used to encrypt plaintext (i.e. cloud-bound data). The random and chaotic characteristics of the input noise can be seen in the key. As a result, unlike what has been recommended for some time, the strength of the encryption key generated also depends on its randomness and chaotic characteristics. This brings new innovation into the field of Cryptography as the length of the key has been cited as the only solution to bringing cryptographic strength.

9.1.1 Key Generation

The fitness function is normalized using random floating-point integers in the [0,1] range in order to provide random encryption keys. The results are a collection of random output sequences. The outputs are chosen at random, then they are transformed into a binary stream. The Data Encryption Standard's key length, 56 bits, corresponds to the minimum length of the binary stream (DES). This was decided upon based on detailed comparisons with existing crypto-systems and the length of a DES key. The encryption algorithm to be employed, such as DES, 3-DES, AES, and 'CloudCrypt', the crypto-system described in this chapter, determines the key length. A 56-bit encryption key is used by both DES and CloudCrypt. However, CloudCrypt has a variable key length feature. This means

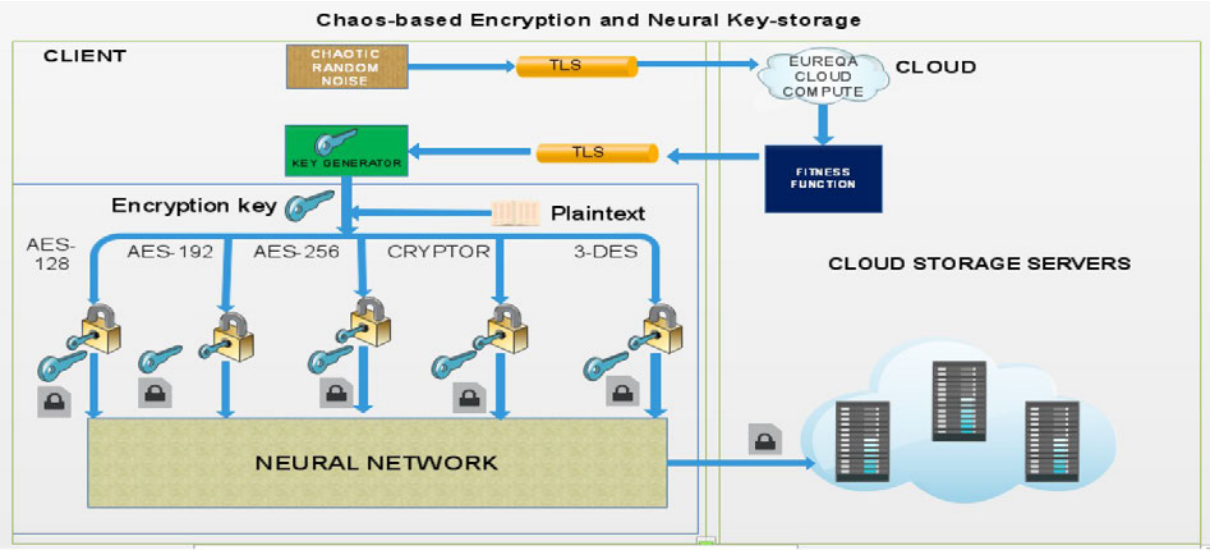

Figure 9.1: Proposed System Architecture.

that the crypto-system can use variable key sizes instead of just one key size. i.e., keys with different lengths in contrast to DES, which only uses 56-bit keys. This is consistent with using static algorithms with evolving key streams, a feature that lets users choose the key length. On the basis of this assumption, it is thought that the proposed system will provide strong encryption keys that depend not only on the key length but also on the key's chaotic randomness features. Section (9.1.2) provides a step-by-step breakdown of the encryption algorithm.

9.1.2 Encryption Algorithm

(i) Generate genuine random noise.

(ii) Input into the Eureqa system chaotic random noise to get a fitness function.

(iii) Create a vector of randomly generated floating-point values in the following range [0,1].

(iv) Put random floats in the fitness function to normalize it.

(v) To create random encryption keys, convert random outputs to binary:

- choose an encryption key at random;

- based on the intended encryption technique, determine the length of the encryption key.

(vi) To create ciphertext, encrypt plaintext data.

(vii) Parse the ciphertext into the Neural Network.

(viii) Give the neural network time to learn the ciphertext and key patterns.

(ix) Let the Neural Network output the ciphertext alone.

(x) Upload the ciphertext to the cloud.

(xi) Discard/destruct the encryption key.

The procedure described above includes a key creation phase that applies to all encryption techniques. Because CloudCrypt's key lengths are flexible and evolve over time, the cryptographic system resembles a one-time pad. This guarantees that cryptanalysis is rendered impossible while also addressing the issue of key distribution because the procedure can be quite time-consuming. The code below can be easily written in MATLAB to demonstrate the encryption technique.

9.1.3 Encryption code

```
function Encryptor = Cipher(data, L)
%Function to encrypt data of length L-bits.
%
%Generate the noise and fitness function using Eureqa.
Noise = Eureqa_ComputeNoise();
fitness_function = Eureqa_Compute_Function(Noise);
%Generate an array of floats, normalizing between [0,1]
```

```
floats [] = fitness_function.normalize(0,1);
%randomly pick a key
for i=0:N-1
    enc_key = random(KeySpace[n]));
if enc_key.length() < data.length()
enc_key = enc_key + random(KeySpace[n+1]);
elseif enc_key.length > data.length()
enc_key = trim(enc_key,data.length());
end
%encrypt the data with the key
cipher = encrypt(data,enc_key);
%return the ciphertext
return cipher;
```

The plaintext data is first transformed into a binary string before encryption begins. The encryption algorithm receives the string as input. The encryptor compares the lengths of the generated binary key string and the generated binary plaintext string since they must both be equal. A new key is generated and attached to the existing key if the plaintext binary stream exceeds the length of the key stream. If the plaintext binary stream is shorter, the key stream is trimmed and the surplus bits are discarded. Otherwise, this key is picked at random.

The ciphertext is generated using a bit-wise exclusive OR (XOR) operation when the two binary streams are identical in length. The encryption key and the ciphertext are inputs into a Neural Network (NN). The neural network is trained to find and learn the encryption key using the input vectors X and Y. This is done to make sure that the CPNN 'learns' the key after it is generated and used to encrypt the plaintext. Then, to prevent further tampering, the key can be destroyed. Section (9.1.4) goes over the NN's structure.

9.1.4 The Neural Network (CPNN)

The CPNN combines elements from many neural networks. It is made up of a variety of structures collectively referred to as a competitive network. Three layers make up a CPNN, a type of neural network: the input, the Kohonen (or hidden) layer, and the Grossberg (or output) layer. A CPNN has two significant features. These two types of learning are supervised and unsupervised. While the Kohonen layer employs unsupervised learning, the Grossberg layer uses supervised learning. The weights of the input vectors are automatically modified based on the learning strategy that is being utilized [87]. The ciphertext and encryption key are sent to the CPNN as a collection of binary vector pairs named $(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)$. The following equation is assumed to hold given two vectors $\mathbf{X}$ and $\mathbf{Y}$ and a continuous function f that connects the two vectors:

$$f(x) = y \tag{9.1}$$

Any value of x within the range will teach the CPNN how to approximate the mappings. An array of training vectors serve as the range's description. If $f^{-1}(y)$ (i.e. the inverse of f exists), the CPNN can additionally learn the inverse mapping to determine the value of x, such that the Equation (9.2) holds:

$$x = f^{-1}(y) \tag{9.2}$$

This makes it an effective tool because pattern matching may be carried out using a backward or reverse sweep. The network's mathematical model functions as a perceptron. Given a specific input value, the perceptron calculates an output value. For error detection and correction, a criteria function is employed. The output error is likewise minimized using the criterion function. It is characterized as a departure from the desired result [154] and is given Equation (9.3)

$$E_k = d - y \tag{9.3}$$

where d is the desired output, y is the actual output from the CPNN, and E is the error of the k^{th} output neuron.

Equation (9.3) reduces the output error. The reduction of error is achieved through a process known as weight adaptation or weight updating. After a training input is presented to the network, the output layer generates a response based on the weights of the connections between the competitive layer and the output layer. The difference between the desired output and the actual output of the network is used to compute an error signal, which is then used to adjust the weights of the connections between the competitive layer and the output layer [122].

The weights of the connections from the winner neuron in the competitive layer to the output layer are updated in the direction of the error gradient, while the weights of the connections from the non-winning neurons are updated in the opposite direction. This weight update rule enables the network to reduce the error by adjusting the weights in a way that allows the network to better discriminate between different input patterns [65].

The CPNN reduces error by adjusting the weights of the connections between the competitive layer and the output layer based on the error signal, thereby improving the network's ability to accurately classify or predict outputs for a given input pattern. The network responds to the training data with high precision when the error value is low. Finding the lowest error value necessitates adjustment of the weights. Adjusting the weights affects the network's overall final error value, identifying the winner node as the one with the lowest error value. Contrary to other neural networks like back-propagation networks, the CPNN prevents erroneous values from spreading to earlier neurons in the hidden layers [92]. This makes the CPNN more effective and offers it a competitive edge over its rivals. The following are the main advantages of CPNNs [92][87]:

(i) The hybrid nature of CPNNs allows them to combine the benefits of various neural networks;

(ii) Both supervised and unsupervised learning strategies are employed in CPNNs;

(iii) they are efficient as they have a high learning rate;

(iv) They are a great option for quick prototyping.

Section (9.1.5) discusses an application to key learning and management.

9.1.5 Key learning

The CPNN is where the main learning happens. The resulting ciphertext and the key are inputted into the CPNN once data encryption is complete. The learning process is aided by the employment of supervised and unsupervised learning systems. The CPNN performs two crucial operations on binary data [87], [92].However, as a pre-requisite to the processing in the CPNN, the binary inputs (i.e., the ciphertext and the key) are transformed to decimal equivalents for computational reasons while determining weights. Figure 9.2 provides an example of key learning/training implemented using Matlab.

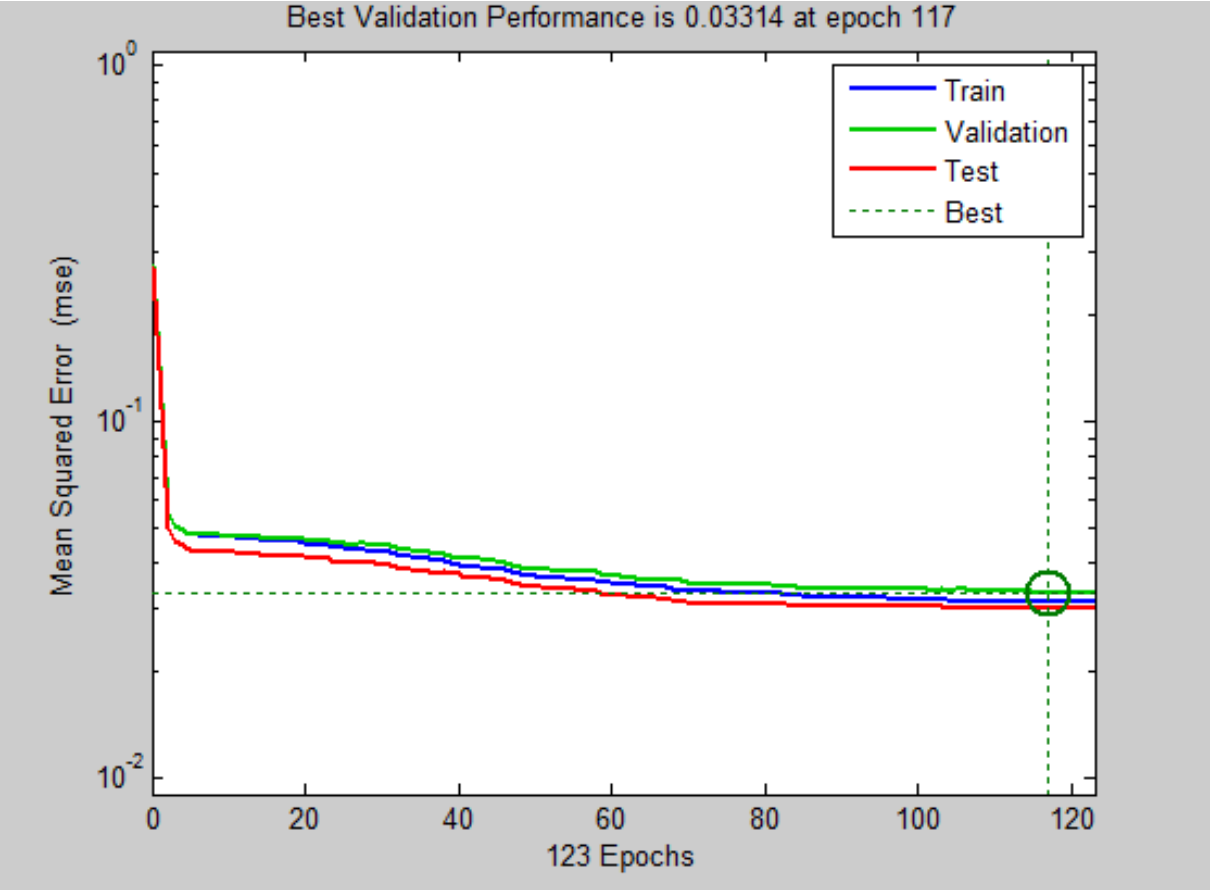

Figure 9.2: Encryption Key learning.

Using the unsupervised learning scheme, the training data is represented by the vector X, Y is the target i.e., output vector and W contains weight. To train the network, the following process was followed:

(i) Compute the Euclid distance of the input and weight vectors. This is done in the hidden layer using Equation (9.1.5)

(ii) The node with the shortest is the output- this is the winner;

(iii) The weights of each node connected to the winner node in the hidden layer should be calculated, using Equation (9.5) and Equation (9.6)

X - Y $= \sqrt{(x_i - w_i).(x_j - w_j)}$

(9.4)

$$w_{ij} = w_{ij} + \alpha(x_i - w_{ij}) \tag{9.5}$$

$$v_{kj} = v_{kj} + \beta(y_k - v_{kj}) \tag{9.6}$$

where w is the weight from the input unit, v is the weight from the target unit, $i = [1, n]$, $j = [1, m]$ and α and β are elements of $\mathbb{Z}$.

This method is also applicable to supervised learning, with the exception that the weights are determined by a cluster layer unit and that the winner node's link to the output from the winner is set to 1 while all others are set to 0.

9.2 OpenNebula Cloud

The OpenNebula 6.0 software was used to put up a cloud infrastructure. This version's hypervisor is a Kernel-based Virtual Machine (KVM). Techniques for virtualization employ the KVM. This is done in order to evaluate CloudCrypt's functionality and applicability on a real cloud architecture. OpenNebula makes use of CloudCrypt as an external plugin. This makes it possible to use CloudCrypt as a cloud-based encryption service..

In order to construct OpenNebula, a client-server architecture approach was used. The platform allows for the creation of a server. Clients can also be servers. One server and many client computers can connect to the server in the cloud, which was designed in a way that allows for this. A cluster is a collection of data centers with similar requirements, and one cluster with the name UKZN-cluster was developed. Three data centers with a combined 128 GB of memory were built within this cluster.

Different file types were stored in each data center. One, for instance, was utilized to store pictures. To store system files such operating system images, another was employed. A host machine is a real server that may operate virtual machines, and they are used to connect to the server.

The UKZN cluster now has three host computers. The hosts have varying requirements. For instance, Host1 and Host2 possessed the following features, respectively: Host1: Intel Core i7-2600 processor, 7.18GB of memory, and eight cores. Host 2: Intel Pentium 1.80GHz processor, 2GB of memory.

VMs were created with varying specifications on a host computer. These VMs were leased to tenants who then used them to store data on the OpenNebula cloud. Figures 9.3 and 9.4 depict the host machines and the VMs created on the hosts.

The next section discusses experimental results.

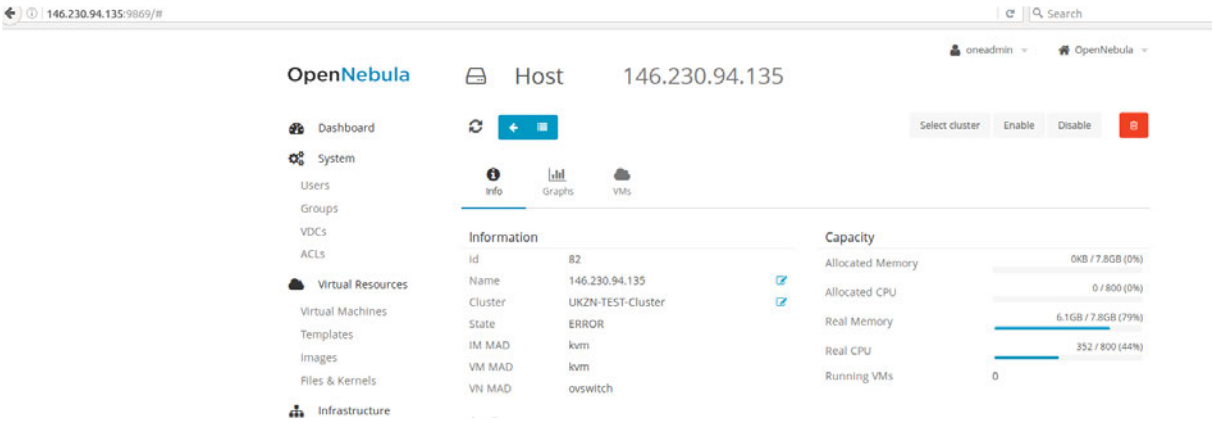

Figure 9.3: Host Machines managed on OpenNebula cloud.

9.3 Experimental results

9.3.1 Approximation of the random input noise

On the system, several experiments were run. These experiments were conducted to assess the system's resource consumption, efficiency, and utilization (i.e., memory and CPU usage). With the help of the cloud-based Eureqa technology, the initial tests were conducted to produce the random noise. For a minimum of 24 hours, the system ran on numerous computers to generate a variety of non-linear fitness functions that most closely matched the input random noise. The Eureqa system received noise inputs from a variety of sources and Figure 9.5 depicts the various fitness functions that were produced, given different noise inputs [154], [128].

9.3.2 Key generator

From figure 9.5, On the left, the best fitness function, which resembles an approximation of the input random noise is depicted. The plot in the middle and the plot of the fitness function that results are used to illustrate the input random noise, respectively. Then, symmetric encryption keys are generated using the fitness function.

The strength of the encryption keys, which is based on the distribution of the random

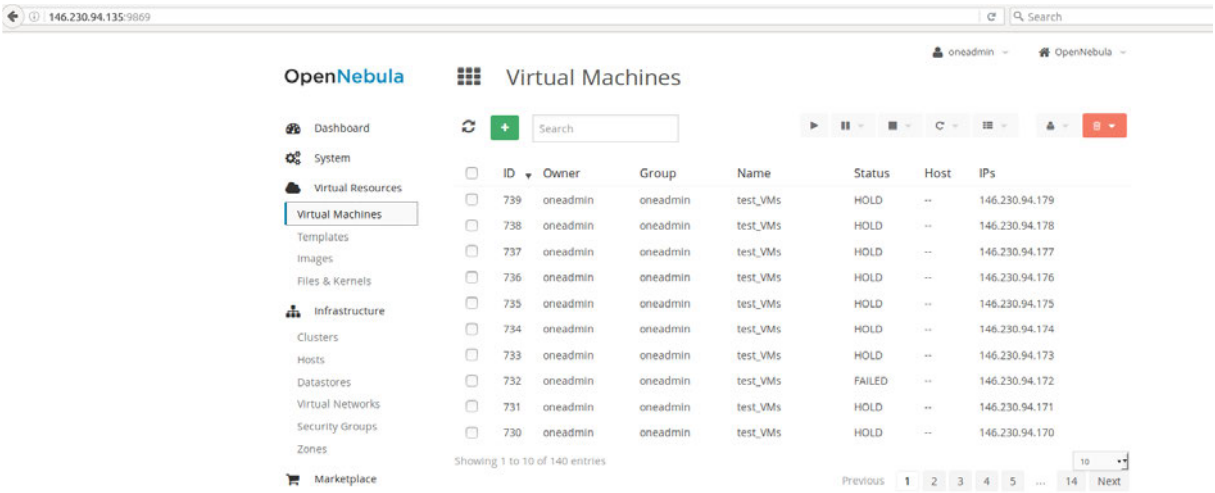

Figure 9.4: Virtual Machines.

input noise, is then utilized to encrypt cloud-bound data. The first input noise resulted into the following fitness function in Equation (9.3.2):

$$f(x) = 128.007 - 14.065\cos(17.043*x) + \sin(128.694*x) + (1.215\cos(17.043*x))! - 38.033\sin(5.274 + 0.001*x^2 + \sin(128.694*x) - 0.226*x) \quad (9.7)$$

The fitness function above is normalised with a random sequence of floats in the range [0,1]. Thereafter, it becomes an input into the key generator to obtain an encryption key.

Figure 9.6 shows the results of randomly generating floating point numbers in the range. An initial random 56-bit key is constructed using the generated random floating point numbers and fitness function. Because the plaintext length was more than the encryption key's beginning length, four executions were necessary (i.e. 56-bits). There were four generations of keys as a result.

The keys are then iteratively concatenated. The key generating algorithm terminates

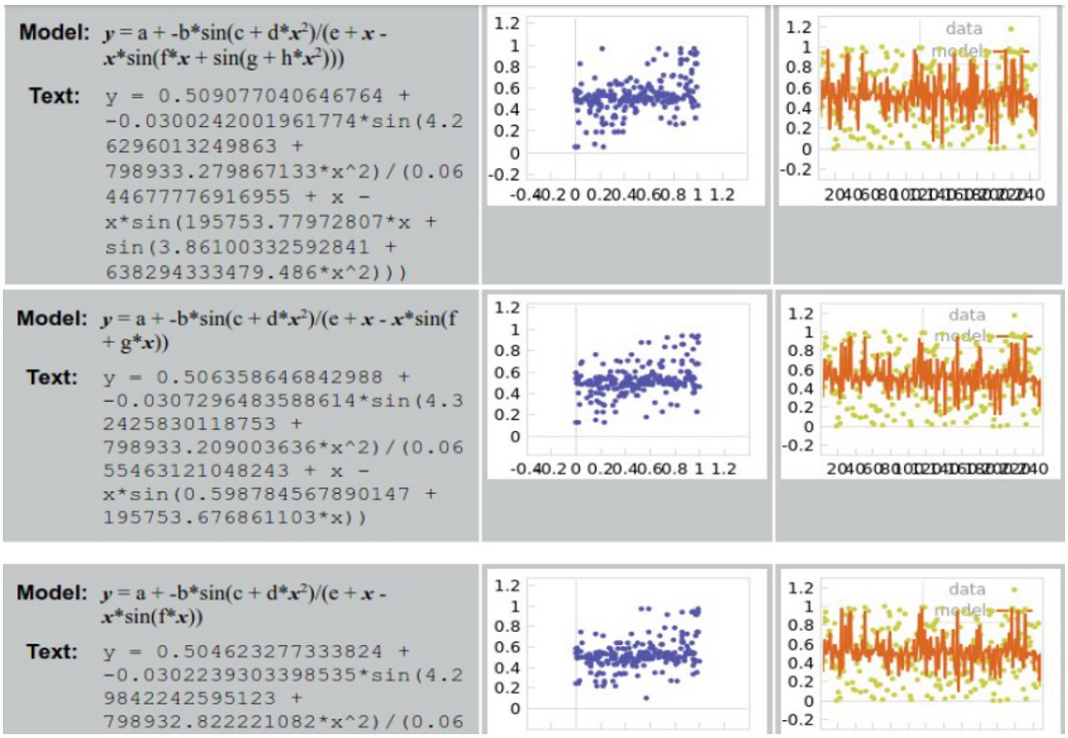

Figure 9.5: Fitness functions generation.

if the length of the key is more than or equal to the length of the plaintext. If the length of the final key exceeds the length of the plaintext, the surplus bits are discarded and the key bits are trimmed before the final key is used for encryption. Figure 9.6 shows a set of four keys resulting from the four floating point normalisations. The first float execution results in key gen 1, the second results in key gen 2 and so on.

Figure 9.7 depicts the four key generators from the four float runs.

9.4 Data Encryption

We tracked each cryptosystem across ten executions. The outcomes' average was calculated and noted. as given in Table 9.1 which compares CloudCrypt with some well known encryption systems in the field of Cryptography.

DES, 3-DES, and several AES variations are the encryption techniques that were used

Table 9.1: Performance results of Crypto-Systems.

Crypto-systems [En/De]cryption performance						
Variable	CloudCrypt	DES	3DES	AES(128)	AES(192)	AES(256)
Key length(bit)	≥ 56	56	192	128	192	256
Document Size(kilobyte)	200	200	200	200	200	200
Encryption runtime (ms)	126.1	12.4	13.4	87.9	164.5	149.6
Decryption runtime (ms)	128.3	10	14	67.7	143.7	130.4
Memory (MB)	8	4	4	6.3	11.1	15.3
CPU (Percent)	1.01	1.23	1.32	1.00	1.00	1.00

for comparisons. Due to their symmetry and similarity to CloudCrypt, these systems were chosen. It shows how long (in milliseconds) it took to encrypt and decrypt a 200-kilobyte document. Additionally, it displays each cryptosystem's CPU and memory use (in percentages). The performance tab of the task manager was used to observe this.

The file took 126 milliseconds to encrypt using CloudCrypt. The same document was DES encrypted in 12 milliseconds. The file was 3-DES encrypted in 13 milliseconds. The same file was encrypted with AES-128, AES-192, and AES-256 in 87, 164, and 149 milliseconds, respectively. Only DES, 3-DES, and AES-128 encrypted the document faster than CloudCrypt in this regard. The former takes the shortest time to encrypt the a 20MB file.

Although it should be noted that these results were attained after 10 consecutive iterations, CloudCrypt surpassed AES (192 and 256). Consequently, CloudCrypt outperformed the other two AES variants on average (i.e. AES-192 and AES-256). Additionally, it should be mentioned that unlike the previous crypto-systems, CloudCrypt uses stream ciphers instead of block ciphers. The results were impacted by this factor since stream ciphers are slower.

9.4.1 Encryption and Decryption times

With regard to Table 9.1, the results were plotted in Figure 9.8

Figure 9.8 also depicts a plot of the outcomes of the decryption as well. Each crypto-system generated unique ciphertext files of varying sizes. DES generated a 120-byte ciphertext file from a 167-byte input plaintext file, while CloudCrypt generated a 162-byte ciphertext. The output of AES (AES-128, AES-192, and AES-256) was 150, 146, and 162 bytes, respectively.

The size of the ciphertext file varied for each of the 10 runs. The ciphertext size was approximated and averaged. The ciphertext's decryption by CloudCrypt took 128 milliseconds. The ciphertext file was decoded by DES in 10 milliseconds and by 3-DES in 14 milliseconds. It took 67.7 milliseconds for AES-128. AES-192 decrypted the ciphertext in 143 milliseconds while AES-256 did it in 130 milliseconds. Once more, DES, 3-DES, and AES-128 reported the shortest decryption times, respectively. AES-192 and AES-256 were outperformed by CloudCrypt in terms of decryption speed. Because each byte is transformed to the American Standard Code for Information Exchange, CloudCrypt generates bigger ciphertext files (ASCII), hence the high encryption and decryption times. All characters in plaintext, including white spaces, are therefore transformed and encrypted. The amount of resources used in terms of CPU and memory use for all crypto-systems are covered, in the next section.

9.4.2 Resource Consumption - CPU

The Java system monitoring library (also known as java sysmon) was used to track CPU consumption throughout the encryption process, starting when the crypto-systems began the encryption and decoding procedures. Utilizing real, operating system-independent processes, this library is used to gauge system performance.

Figure 9.9 shows the CPU consumption for each crypto-system. It was observed that CloudCrypt and AES share a similar CPU utilization. As seen in the graph, DES and 3-DES encryption used the CPU the most.

9.4.3 Resource Consumption - Memory

After 10 runs, a memory consumption diagram for each cryptosystem was recorded, with readings rounded to the nearest MB. The records for each crypto-system were obtained using the Java system monitoring library.

Figure 9.10 demonstrates how 8MB of the system's memory was taken up by CloudCrypt, which also used up RAM. At 4MB, DES and 3-DES were tied. The memory requirements for AES-128, AES-192, and AES-256 were 6MB, 11MB, and 15MB, respectively, with CloudCrypt outperforming each of these encryption algorithms. This is a good outcome because one of the specifications for the system was to create a lightweight crypto-system. As a result, CloudCrypt may be integrated into hardware with modest CPU and memory requirements, such as mobile phones.

9.5 Conclusion

CPU, memory, key size, encryption, and decryption times are just a few of the different metrics that are shown in the analysis and results. According to these findings, AES and CloudCrypt outperformed the others in terms of CPU consumption. However, in terms of memory consumption, encryption, and decryption times, AES-192 and AES-256 are the worst. AES-256 and AES-192 have slightly different performance characteristics on CPUs, but the difference is small and may depend on the specific implementation and hardware used.

Since AES-256 requires more rounds of transformation, it took slightly longer to encrypt or decrypt data compared to AES-192. However, modern CPUs have hardware acceleration for AES instructions, which means that the actual performance difference between the two algorithms may be negligible. Overall, the performance difference between AES-256 and AES-192 on CPUs is generally small, and the choice of algorithm may depend on other factors such as security requirements, implementation constraints, and available hardware acceleration.

In addition, the memory requirements of the algorithms may affect their performance

on different systems. If a system has limited memory, AES-256 may cause more frequent cache misses and incur additional overhead in managing memory, which could slow down its performance compared to AES-192. However, if the system has sufficient memory, the difference in performance between the two algorithms may not be significant.

The shortest key used by CloudCrypt is 56 bits long. Key sizes vary. Multiple executions (10 runs) were carried out because of this. An average performance across several characteristics was calculated from the 10 experiments. In terms of CPU and memory use, CloudCrypt performed similarly to AES. In this regard, CloudCrypt substantially resembled AES, one of the most well-known crypto-systems. According to the results, CloudCrypt uses modest to large amounts of memory. Therefore, if used in industry, it is a better crypto-system. It is deployable on hardware with modest CPU and memory requirements. To provide data confidentiality on the cloud and for the IoT, this chapter makes proposed a client-side encryption and key management scheme.

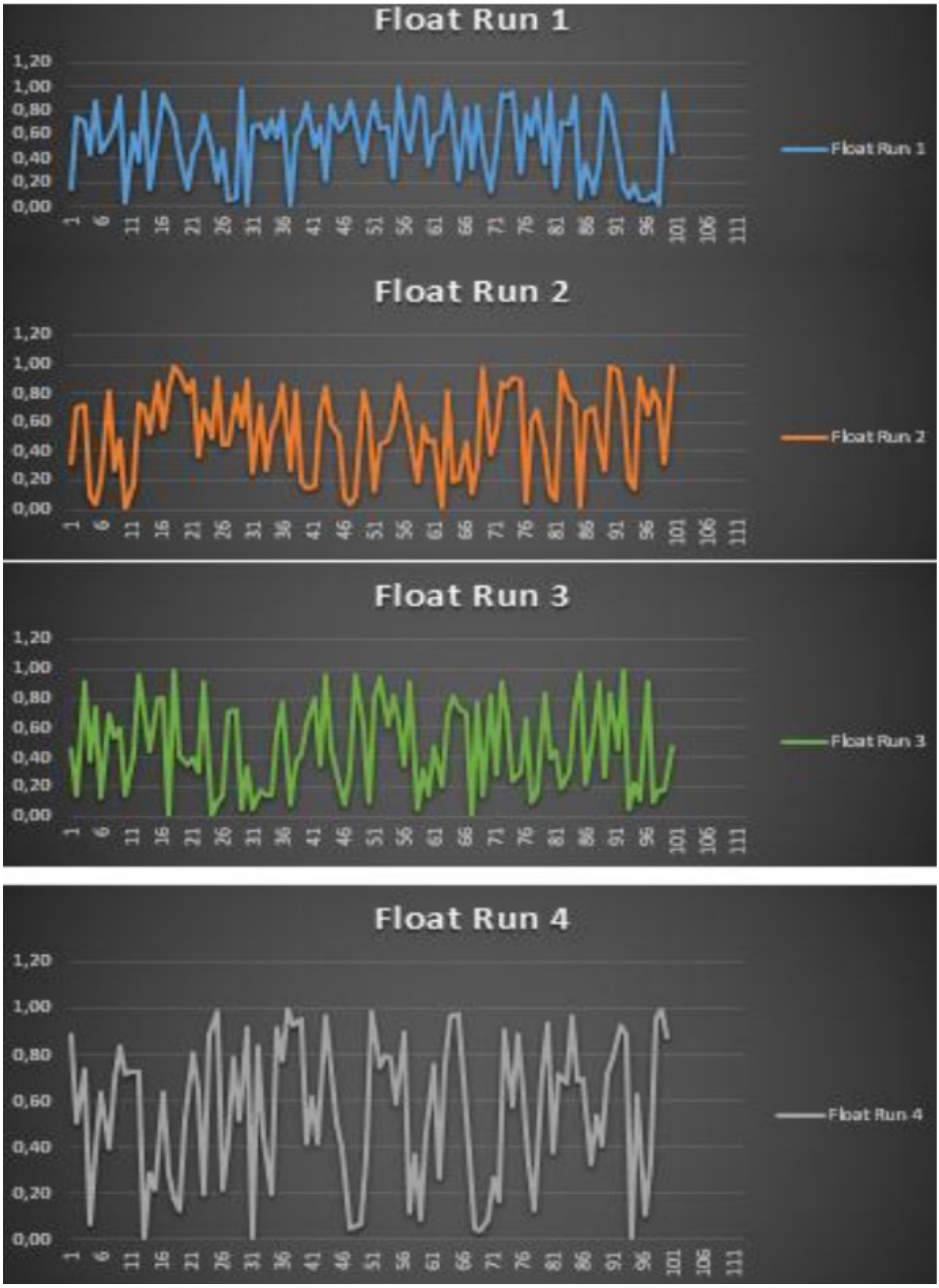

Figure 9.6: Generating floating point numbers.

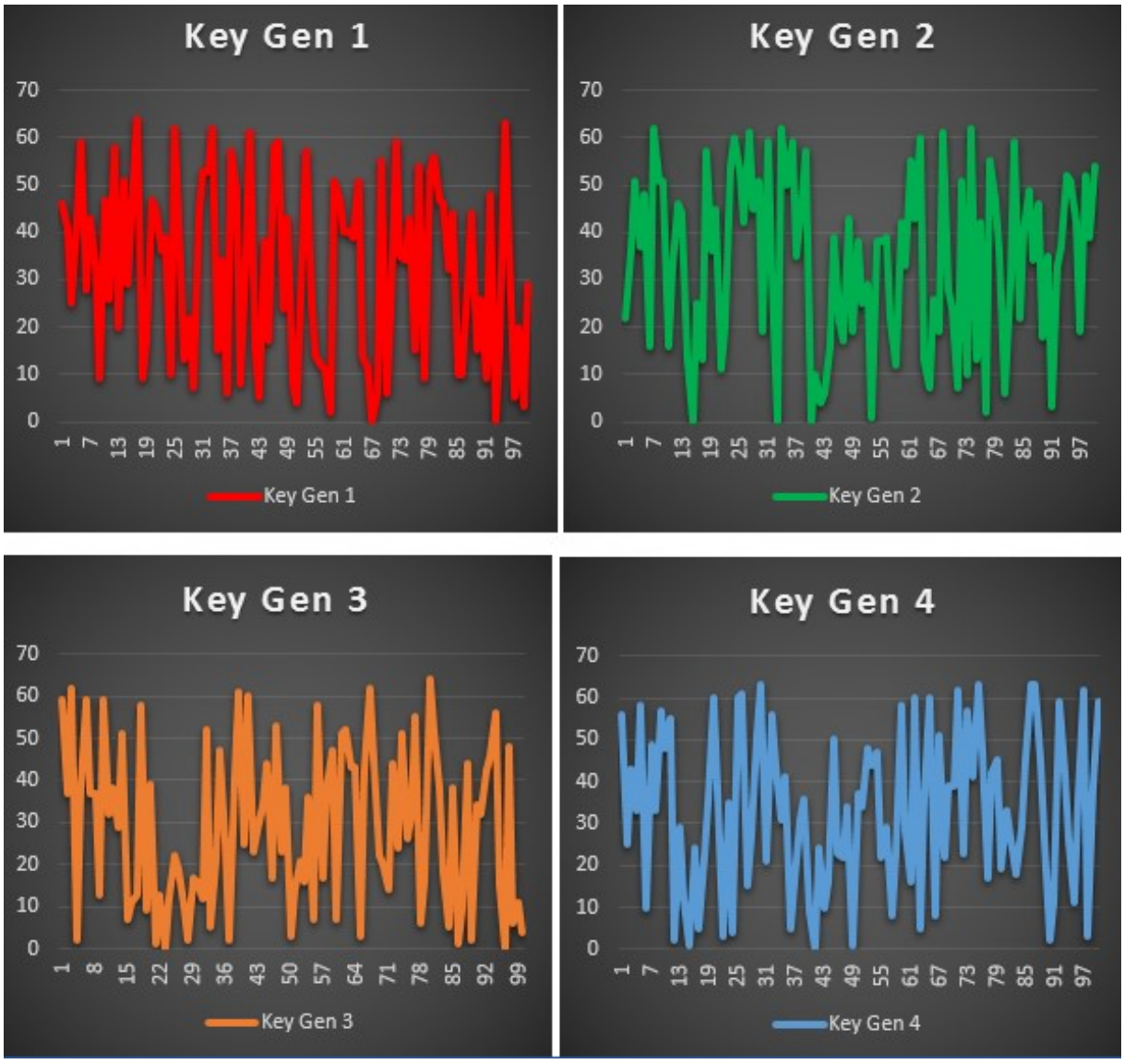

Figure 9.7: Generation of keys.

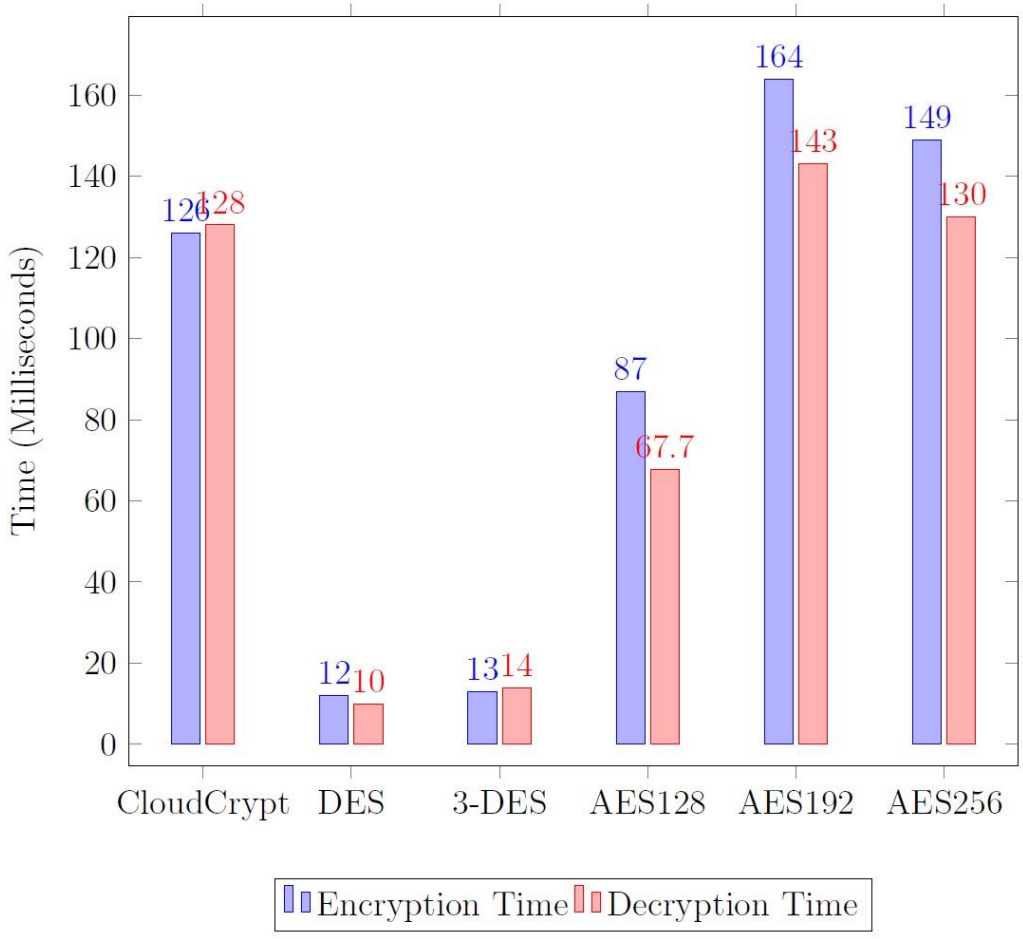

Figure 9.8: Encryption vs Decryption.

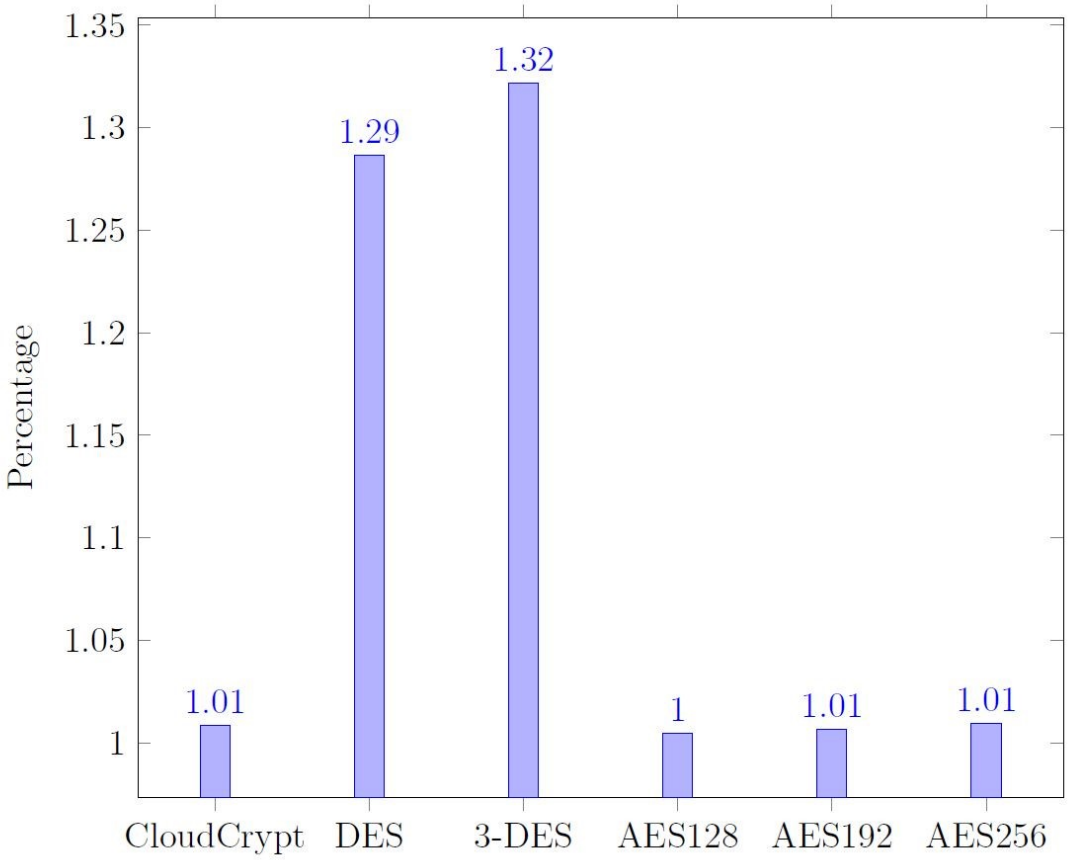

Figure 9.9: CPU usage per crypto-system.

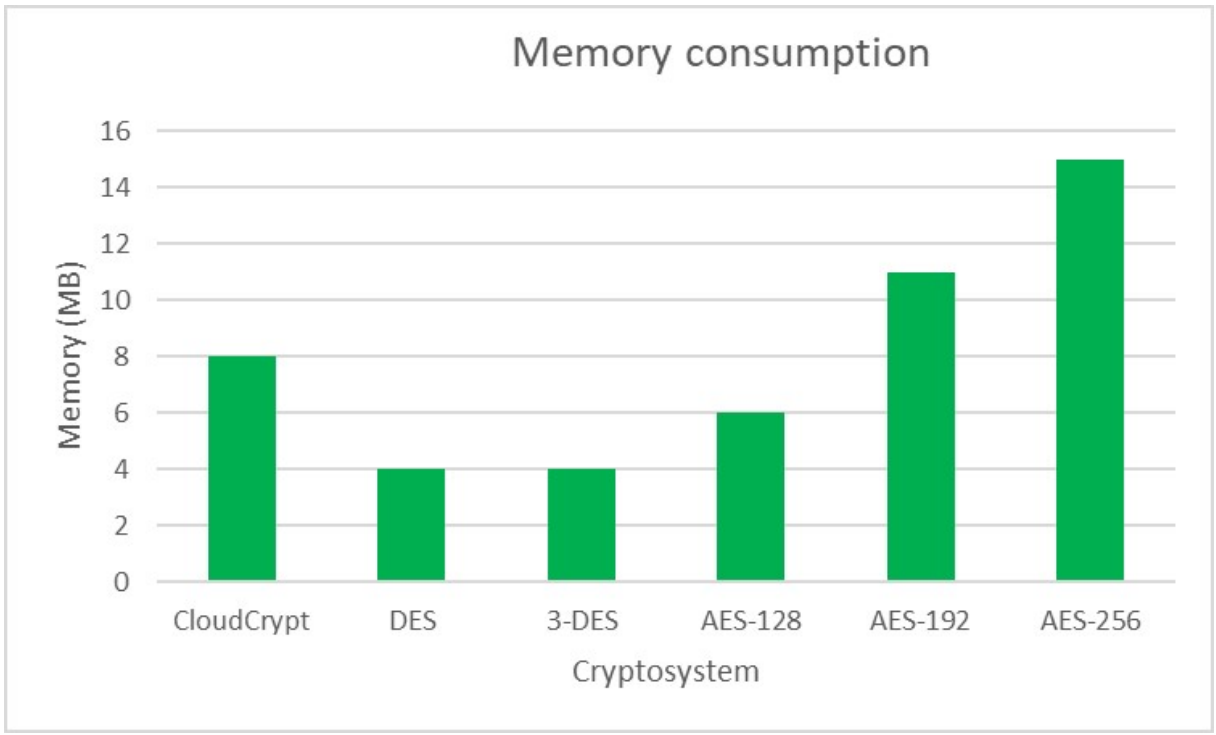

Figure 9.10: Memory usage per crypto-system.

Chapter 10

Discussion, Conclusions and Future Research

10.1 Chaotic Cryptography

Chaos and Cryptography share a basic connection. In both of these fields, the system under investigation is a dynamic one. This system transforms data in an apparently unpredictable yet deterministic way through an iterative, nonlinear transformation. It is conceivable to ensure cryptographic confusion and diffusion with the right entropy conscious post-processing, given chaotic systems' sensitivity to beginning conditions and their mixing characteristics. Chaos theory and cryptography, however, differ conceptually in a number of ways. These are listed below:

(i) While cryptography relies on a finite-state machine and all chaos models implemented on a computer are approximations, i.e., digital computers can only generate pseudo-chaos, chaos theory frequently focuses on the study of dynamic systems described on an infinite state space;

(ii) While cryptography focuses on the impact a number of iterations typically based on the size of the plaintext, chaos theory studies the asymptotic behavior of a nonlinear

system (i.e., the behavior of the system as the number of iterations approach infinity when the Lyapunov dimension can be quantified);

(iii) In contrast to cryptography, where complexity is the primary concern, chaos theory is more interested in the interpretation of a physical model from which it has been derived. As a result, the concepts of cryptographic security and efficiency have no analogs in chaos theory;

(iv) In contrast to the fact that most classical chaotic systems contain recognizable attractors, cryptography tries to eradicate any structure by post-processing the ciphertext to generate a maximum entropy;

(v) Cryptographic systems, in contrast to chaotic ones, use a mix of independent variables to produce an output that is unpredictable to a cryptanalyst;

(vi) In contrast to cryptography, where the physical model is irrelevant, chaos theory is frequently related with the mathematical model used to quantify a physically significant problem.

Regarding the design of chaos-based encryption engines, point (vi) is particularly crucial. This thesis considered the idea that, in principle, various systems can be 'invented' by the designer in order to provide a limitless range of multi-algorithmic encryptors, in contrast to previous publications in the field that have considered variations on a theme of established chaotic systems. The method of data encryption covered in this thesis constitutes a 'paradigm shift' in comparison to publicly available single algorithm based ciphers.

10.2 Cloud Security

The use of the iCloud (the cloud storage and cloud computing service) to store and exchange data has seen a significant expansion since Amazon debuted its Simple Storage Service (S3) in March 2006 and the Elastic Compute Cloud (EC2) in August of the same

year. Due to its various benefits in terms of sale and cost, change management, next-generation architectures, choice, and agility, it is therefore likely to persist in the future. However, lack of control and, more importantly, data security are two of the main issues that users of the Cloud worry about.

The information in Chapters 4 and 5 discusses a method of encrypting data prior to it being 'placed' on the Cloud where each user has access to their own encryption algorithm (client-side encryption), an algorithm that is based on a set of Iterative Function Systems (IFSs) that outputs a chaotic number stream and is intended to create a cryptographically secure cipher.

A method based on the idea of multi-algorithmic cryptography that takes advantage of the characteristics of pseudo-chaos was created in Chapter 4 after a study of cryptographic systems utilizing finite-state approximations to chaos or 'pseudo-chaos'. The combination of these algorithms can be used to encrypt data in a fashion that is specific to each user, even while they can be assumed to be in the public domain to comply with the Kerckhoff-Shannon principle. This makes it possible for Cloud customers to upload and transfer data while being aware that their data is encrypted in a way that is algorithm and key dependent, thwarting a known algorithm attack. In Chapter 4, a pioneering instance of applying this strategy commercially was described.

Current discussions on cloud computing make the assumption that users who rely on third-party hosting for their servers won't see much changes. Additionally, it seems that common security measures will offer adequate security in the future. These presumptions disregarded the commonly accepted belief that the Cloud remains unsafe. The user's sense of this is continuously reinforced by the need for anti-malware software, which is becoming slower and more complex, as well as by frequent news reports about significant security breaches, frequently committed by adversarial state funded cyber attackers.

Most businesses rely on some proprietary know-how, process, design or other commercial secret to preserve their competitive position and to try and delay product cycle decay. Business, especially now, is very conscious of the need to avoid fixed and capital costs to reduce their vulnerability to volatility. Cloud Computing, as a capital and fixed cost free

approach, is an obvious solution but perceived lack of security for commercially sensitive data is a major barrier to conversion from in-house information and communications technology.

To maintain their competitive position and to attempt to slow down product cycle degradation, the majority of enterprises rely on some proprietary know-how, process, design, or other commercial secret. Businesses are acutely aware of the need to avoid fixed and capital costs in order to lessen their sensitivity to volatility, particularly in the present. Although cloud computing is a clear solution in terms of capital and fixed costs, the perceived lack of security for economically critical data is a significant impediment to switching from internal information and communications technology.

10.2.1 The Role of Encryption

For commercial users, traditional encryption has a number of shortcomings as a method of data security. They consist of the following:

(i) Decision-makers lack a thorough understanding of encryption and how to compare the relative merits of various methods;

(ii) Governments and commercial organizations rely on industry certification standards and regulatory requirements that stratify encryption strength by key length while evaluating the underlying algorithms' resistance to common assaults. This popular strategy is not exclusive to cryptography and is used in many different sectors;

(iii) The way in which certification is applied causes, as an unintended consequence, systemic risks to be inherent in any approved system;

(iv) Certification is both expensive and slow creating a high barrier to entry for innovative encryption systems and making commercially available systems lag years behind the technologies available to hackers.

(v) The process of commercializing innovative ideas may need significant cash investment due to state legislation regarding the selling of encryption technology.;

(vi) It is evident that governments regard the certification process as a way to comprehend, manage, and restrict the strength of encryption to suit their security requirements for monitoring. Sadly, this strategy is inherently flawed since it falsely assumes that adversarial agencies do not possess comparable or superior ability to circumvent encryption.

10.2.2 Cloud-hosted Data Encryption

How will people make advantage of the cloud? Commercial users must process, store, communicate with new data, and output from stored data in order to function practically. The majority of the time, a Website, Database, and secure communications are required. Additionally, malware needs to be protected against. When malware infiltrates a data stream while being encrypted, it is impossible to detect, which is where encryption has problems. Furthermore, despite assertions to the contrary, a database cannot be encrypted before being effectively used. Data must be readable in order to be used.

Therefore, the problem is how to process data in the cloud securely. By encrypting the communication route using transport layer security, installing anti-malware on servers and end-points, implementing physical protection, and isolating a specific user's servers, the issue with server hosting is resolved. Consequently, the key is to ensure data security by employing encryption for all communication channels and when data needs to be kept, as well as to physically and electronically protect the environment where live data processing occurs.

The biggest risk associated with employing conventional encryption in the Cloud is that it only uses key management to keep confidential data apart from the rest of the Cloud, which poses systemic concerns. In other words, a serious flaw in the encryption engine might cause the entire structure to collapse. This problem serves as the focal point for the work presented in Chapters 4 and 5, which introduced a method of encrypting data in which all systemic risk can be minimized by swapping the problem of key management with the management of meta-encryption-engines using multiple chaos-theoretic encryp-

tion algorithms with Multi-algorithms, where the encryption algorithms can be created using Evolutionary Computing or computed by hand (as mentioned in Chapter 4). On a USB memory 'key' the meta-encryption engines can be mounted and executed, such as on USB tokens for authorisation of users and encryption. No matter where the engines are mounted, their control and processing environment must take place within a cluster of physically and electronically secure hosting locations.

Each meta-engine is unique to a single user when using an encryptor such as Cloud-Crypt to secure data on the cloud. Thus, each user must be properly vetted and authorized to have a meta-encryption engine that may be submitted to a user upon request. To achieve secure communication to and from the exchange without the requirement for the parties to disclose their meta-encryption engines, each meta-encryption engine device offers a secure entry point and the Cloud, which functions like an exchange server using a handshake protocol. The system can behave like a 'one-time pad' since each engine is seeded for each file, therefore this is 'one-message, one-algorithm, one-key' solution.

10.2.3 Cloud Computing and Encryption using Chaos

Cloud computing is expected to overtake all other security-related topics due to its attributes. A guidance document from the Cloud Security Alliance (CSA) dealing with critical areas of focus in Cloud Computing version 2.1 gives an overview of the problems related to cloud security. The following problems are relevant to the use of chaos in encryption:

Cloud computing will always exist. For instance, it reduces the 'time to market' for business and allows for real-time updates without the need to purchase the infrastructure. It is cost-effective as it provides switching the business model from capital investment and depreciation costs as servers get old and will need to be decommissioned, to a predictable operating cost model which is instantaneously expandable to accommodate unforeseen spikes or declines in traffic volumes. Examples of early users of the Cloud include the New York Times, which desired to electronically archive '70 years' worth of content dig-

itally for information retrieval (IR). This was a 'once-off' project expenditure that was accomplished using the Cloud in less than 24 hours, with no residual IT infrastructure. The Cloud can be used by start-ups to provide them with full IT capabilities with a very thin IT budget and low upfront investments and the flexibility to adapt to changing conditions and quickly scale up if successful. Without the need to alter the data or the way that applications are processed, the Cloud offers low revenue cost 'Customer Relationship Management' services. Trust, privacy loss, regulatory violations, data replication and loss of integrity and coherence, application sprawl, and dependencies are just a few of the problems with cloud computing. This is a summary of the 'Pros and Cons' of this paradigm shift.

Among these 'pros and cons', security has been rated a major problem. It is imperative to think of the Cloud as a hostile territory. Consequently, user-centric security likely the solution. It is against this context that chaos-based encryption is seen as a solution. The approach considered in this thesis as laid out in Chapters 4, 5 and 7, is based on each tenant having a personalized encryption engine. This engine enables both protection and control over cloud-hosted data, based on the following:

$$\text{Digital device} + \text{Internet connection} + \text{Personalised Chaos} - \text{based Encryptor} = \text{Cloud Security}$$

10.3 Future Directions for Encryption using Chaos

Users can develop an infinite variety of different cryptographically secure encryption engines by applying chaos to the creation of ciphers via evolutionary computing. The commercial answer is to create a website where users can pay for the creation of a special encryption engine that can be downloaded and used to encrypt users' data prior to "lq storage" on the Cloud after receiving a remote payment. To do this, a sizable database of encryption engines must be compiled. Once built, a user-by-user generated series of these algorithms can be produced. Depending on the nation in which the company is registered will determine the operational circumstances under which this strategy can be

implemented on a commercial basis. In the UK, for instance, commercial activities must abide with the Regulation of Investigatory Powers (RIP) Act 2000 [49], which necessitates the establishment of an infrastructure and is consequently capital and overhead-intensive.

A super-set of various random number generators that are utilized in common encryption algorithms can be thought of as Chaos. Although employing chaos for cryptography has several drawbacks, it is nonetheless a fascinating use of nonlinear dynamics. The main benefit of chaos is the potential for a wide variety of algorithms. It is feasible to achieve this with traditional random number generators, such as Knuth's M-algorithm, but chaos offers a wider variety of functions (other than the mod function, for example). However, this strategy still has some significant theoretical and computational issues, including the following:

10.3.1 Stable Pseudo-Chaotic Systems

A crypto-system that has (nearly) the same cycle length and Lyaponov exponent for all initial conditions is what we call a structurally stable crypto-system. There is currently no rigorous analytical method for determining this trait, and the majority of known pseudo-chaotic systems lack it. It is impossible to ensure that a crypto-system based on a deterministic chaotic algorithm or combination of algorithms would always create uncorrelated number streams for any and all keys, making this a crucial concern. Certain conditions are required to ensure unpredictability.

10.3.2 Unpredictability Conditions for Chaotic Systems

What characteristics of a chaotic system ensure its unpredictability in terms of computation? There is currently no theoretically tenable way to assess a chaotic system in terms of the necessary/sufficient conditions and attributes that will unquestionably ensure the system's unpredictability to acceptable cryptographic standards. The current strategy relies more on trial and error and does not make use of an algorithm proving tool. Regarding this problem, formal software engineering methodologies may be useful.

10.3.3 Chaos in Native Binary Systems

Although there are theoretically an infinite number of chaos-based algorithms that may be created, they currently rely on floating point arithmetic and need high precision Floating Point arithmetic to generate a reasonable amount of cycles (deterministic chaotic algorithms have relatively low cycle lengths which is another disadvantage). Given that the number streams these floating point systems generate are typically transformed into bit streams in any case, they take a lot of time. Therefore, designing algorithms that emit bit streams directly would be highly advantageous. It appears that no theoretical investigation of this natively binary chaos has been made to date.

10.3.4 Asymmetric Chaos-Based Cryptography

Trapdoor functions, or functions with a one-way property unless a secret parameter (trapdoor) is known, are the foundation of asymmetric systems. The RSA algorithm, which uses the trapdoor-designing characteristics of prime numbers, is one of the best-known examples of this aspect. To the author's knowledge, chaos theory does not have an equivalent to a trapdoor transformation. Future prospects have an excellent chance to fill this gap.

10.3.5 Phase-only Cryptography

It is possible to apply the theoretical underpinnings of phase-only cryptographic mechanisms, which were covered in Chapter 5. It highlighted an n-dimensional data encryption without losing generality. The two-dimensional example and the encryption of full-color images have received the majority of attention in the application and encryption algorithm development as discussed in Chapter 5. It should be noted that any color utilized in the algorithms listed in Appendix A can be swapped out for a different grey level or a binary input picture (plaintext). In either case, the algorithms provided assume that the encryption/decryption keys are known apriori and in this respect, they reflect a normal symmetric encryption scheme that necessitates the use of a key-exchange technique, hence

the Three-Pass-Protocol proposed in chapter 6 to consider the use of phase-only crypto.

Bibliography

[1] Flashid promotional, 2021. [ONLINE] Available at https://www.youtube.com/watch?v=epwky4d6V7c&feature=youtu.be.

[2] Flashid technical demonstration, 2021. [ONLINE] Available at https://www.youtube.com/watch?v=_hrgIpB7vE4&feature=youtu.be.

[3] Delarue authentication, 2022. [ONLINE] Available at https://www.delarue.com/authentication/overview.

[4] laws of thermodynamics — definitions facts 2022, 2022. [Online] Available at https://www.britannica.com/science/laws-of-thermodynamics.

[5] A. Jagannatam and D. Tran. Mersenne Twister A Pseudo-Random Number Generator, 2008. [Online] Available at https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.85.2732&rep=rep1&type=pdf.

[6] Aaqif Afzaal Abbasi, Almas Abbasi, Shahaboddin Shamshirband, Anthony Theodore Chronopoulos, Valerio Persico, and Antonio Pescapè. Software-defined cloud computing: A systematic review on latest trends and developments. *IEEE Access*, 7:93294–93314, 2019.

[7] AR Al-Rawi. *Digital Rights Management using Steganocryptography*. PhD thesis, PhD Thesis, Dublin Institute of Technology, 2013.

[8] Mohammed M Alani. Applications of machine learning in cryptography: a survey. In *Proceedings of the 3rd International Conference on cryptography, security and privacy*, pages 23–27, 2019.

[9] Ayman Alharbi, Haneen Zamzami, and Eman Samkri. Survey on homomorphic encryption and address of new trend. *International Journal of Advanced Computer Science and Applications*, 11(7), 2020.

[10] The Cloud Security Alliance. The cloud security alliance, 2021, 2021. [ONLINE] Available at https://cloudsecurityalliance.org/.

[11] E Alvarez, A Fernández, P Garcıa, J Jiménez, and A Marcano. New approach to chaotic encryption. *Physics Letters A*, 263(4-6):373–375, 1999.

[12] DI George Amalarethinam and J Madhupriya. An assured encryption strategy for private data protection in hybrid cloud. *ICT Systems and Sustainability: Proceedings of ICT4SD 2021, Volume 1*, 321:271, 2022.

[13] Amazon. Amazon. amazon simple storage service, 2009. [ONLINE] Accessed on 6-Aug-2022 http://aws.amazon.com/s3/.

[14] Araneus. Araneus information systems, 2020. [ONLINE] Available at https://www.araneus.fi.

[15] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[16] Ramirose Attebury, Julie George, Cindy Judd, and Brad Marcum. Google docs: a review. *Against the Grain*, 20(2):9, 2008.

[17] Rahul Awati. random numbers, 2022. [Online] Available at https://www.techtarget.com/whatis/definition/random-numbers.

[18] S Babbage, C Canniere, A Canteaut, C Cid, H Gilbert, T Johansson, M Parker, B Preneel, V Rijmen, and M Robshaw. The estream portfolio, estream, ecrypt stream cipher project, 2008.

[19] bankmycell.com. Bank my cell, 2022. [ONLINE] Available at https://www.bankmycell.com/blog/how-many-phones-are-in-the-world.

[20] MS Baptista. Cryptography with chaos. *Physics letters A*, 240(1-2):50–54, 1998.

[21] L Bassham, A Rukhin, J Soto, J Nechvatal, M Smid, S Leigh, M Levenso, M Vangel, N Heckert, and D Banks. A statistical test suite for random and pseudorandom number generators for cryptographic applications, 2010-09-16 2010. [ONLINE] Available at `https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762`.

[22] Robert Beinert. One-dimensional phase retrieval with additional interference intensity measurements. *Results in Mathematics*, 72(1):1–24, 2017.

[23] K Bhattacharjee, K Maity, and S. Das. A search for good pseudo-random number generators : Survey and empirical studies, 2018.

[24] Mark E Bianco and Dana A Reed. Encryption system based on chaos theory, September 10 1991. US Patent 5,048,086.

[25] Mojtaba Bisheh-Niasar, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. High-speed ntt-based polynomial multiplication accelerator for crystals-kyber post-quantum cryptography. *Cryptology ePrint Archive*, 2021.

[26] Mathias Björkqvist, Christian Cachin, Robert Haas, Xiao-Yu Hu, Anil Kurmus, René Pawlitzek, and Marko Vukolić. Design and implementation of a key-lifecycle management system. In *International Conference on Financial Cryptography and Data Security*, pages 160–174. Springer, 2010.

[27] J. Blackledge. *Digital Signal Processing: Mathematical and Computational Methods, Software Development and Applications*. Elsevier: Woodhead Publishing Series in Electronic and Optical Materials, 2006. eBook ISBN: 13-978-1904275268 [Online] Available at `https://arrow.dit.ie/engschelebk/4`.

[28] JM Blackledge. Cryptography using steganography: New algorithms and applications. *Lecture Notes, Centre for Advanced Studies, Warsaw University of Technology, Warsaw*, 2012.

[29] JM Blackledge. Cryptography using steganography: New algorithms and applications. *Lecture Notes, Centre for Advanced Studies, Warsaw University of Technology, Warsaw*, 2012.

[30] JM Blackledge. Cryptography using steganography: New algorithms and applications. *Lecture Notes, Centre for Advanced Studies, Warsaw University of Technology, Warsaw*, 2012.

[31] Jonathan Blackledge. Information hiding using stochastic diffusion for the covert transmission of encrypted images. In *IET Irish Signals and Systems Conference (ISSC 2010)*, pages 12–17. IET, 2010.

[32] Jonathan Blackledge, Omotayo Asiru, and Moses Dlamini. Application of artificial intelligence for detecting computing derived viruses. In *16th European Conference on Cyber Warfare and Security (ECCWS 2017), University College Dublin, Dublin June*, pages 29–30, 2017.

[33] Jonathan Blackledge, Sergei Bezobrazov, and Paul Tobin. Cryptography using artificial intelligence. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2015.

[34] Jonathan Blackledge, Sergei Bezobrazov, Paul Tobin, and Francesco Zamora. Cryptography using evolutionary computing. In *24th IET Irish Signals and Systems Conference (ISSC 2013)*, pages 1–8. IET, 2013.

[35] Jonathan Blackledge and Oleksandr Iakovenko. On the application of two-dimensional chirplets for resilient digital image watermarking. 2013.

[36] Jonathan Blackledge and Napo Mosola. Applications of artificial intelligence to cryptography. *Transactions on Machine Learning and Artificial Intelligence*, 8(3):21–60, Jun. 2020.

[37] Jonathan Blackledge and Nikolai Ptitsyn. On the applications of deterministic chaos for encrypting data on the cloud. In *Third International Conference on the Evolving Internet IARIA Luxembourg*, pages 78–87, 2011.

[38] Jonathan Blackledge, Paul Tobin, J Myeza, and CM Adolfo. Information hiding with data diffusion using convolutional encoding for super-encryption. 2017.

[39] Jonathan M Blackledge. *Digital image processing: mathematical and computational methods*. Elsevier, 2005.

[40] Jonathan M Blackledge. *Digital image processing: mathematical and computational methods*. Elsevier, 2005.

[41] Jonathan M Blackledge. *Quantitative coherent imaging: theory, methods and some applications*, volume 11. Elsevier, 2012.

[42] Jonathan M Blackledge and Eugene Coyle. Authentication of biometric features using texture coding for id cards. In *2010 Fifth International Conference on Internet Monitoring and Protection*, pages 74–83. IEEE, 2010.

[43] Jonathan M Blackledge, AK Evans, and Martin J Turner. *Fractal Geometry: Mathematical Methods, Algorithms, Applications*. Elsevier, 2002.

[44] Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on computing*, 15(2):364–383, 1986.

[45] C. Brook. Linkedin hacked and 6.5 million passwords stolen - opusfidelis, 2012. [ONLINE] Accessed on 6-Aug-2022 Available at `https://opusfidelis.com/insights/linkedin_hacked_and_6-5_million_passwords_stolen/`.

[46] Christian S Bruwer et al. *Correlation attacks on stream ciphers using convolutional codes*. PhD thesis, University of Pretoria, 2007.

[47] Johannes A Buchmann, Denis Butin, Florian Göpfert, and Albrecht Petzoldt. Post-quantum cryptography: state of the art. *The new codebreakers*, pages 88–108, 2016.

[48] Easttom C. An overview of pseudo random number generators, 2017.

[49] Rico Calleja. Rip act 2000—uk: The regulation of investigatory powers act 2000. *Computer Law & Security Review*, 16(6):400–401, 2000.

[50] Capterra. Artificial intelligence software, 2020. [ONLINE] Availabile at `https://www.capterra.com/artificial-intelligence-software`.

[51] JM Carroll and Y Sun. The binary derivative test for the appearance of randomness and its use as a noise filter tech report no. 221 university of western ontario. *Dept. of Computer Science*, 1998.

[52] John M Carroll. The binary derivative test: noise filter, crypto aid, and random-number seed selector. *Simulation*, 53(3):129–135, 1989.

[53] Gregory J Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM (JACM)*, 13(4):547–569, 1966.

[54] David Chappell et al. Introducing the windows azure platform. *David Chappell & Associates White Paper*, 2010.

[55] Aaron Charles. Does facebook fully delete your records?, 2022. [ONLINE] Available at `https://smallbusiness.chron.com/facebook-fully-delete-records-67021.html`.

[56] Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on Post-Quantum Cryptography*. National Institute of Standards and Technology, 2016.

[57] Weiting Chen, Jun Zhuang, Wangxin Yu, and Zhizhong Wang. Measuring complexity using fuzzyen, apen, and sampen. *Medical engineering & physics*, 31(1):61–68, 2009.

[58] Paar Christof. *Confusion and Diffusion - Understanding Cryptography, A Textbook for Students and Practitioners*. Springer, 2010.

[59] CISCO. Cisco annual internet report (2018-2023) white paper, 2021. [ONLINE] Available at https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[60] Google Cloud. Ai and machine learning products, advanced guide to inception v3 on cloud tpu, 2020. [ONLINE] Availabile at https://cloud.google.com/tpu/docs/inception-v3-advanced.

[61] B. J. Copeland. Artificial intelligence. encyclopedia britannica, 2020. [ONLINE] Availabile at https://www.britannica.com/technology/artificial-intelligence.

[62] Luca Crocetti, Stefano Di Matteo, Pietro Nannipieri, Luca Fanucci, and Sergio Saponara. Design and test of an integrated random number generator with all-digital entropy source. *Entropy*, 24(2), 2022.

[63] Grenville J Croll. Bientropy-the approximate entropy of a finite binary string. *arXiv preprint arXiv:1305.0954*, 2013.

[64] Crypstic. Multi algorithmic encryption on a usb key: Novel multi-algoritmic encryption technology that encrypts/decrypts data on storage devices, 2021. [ONLINE] Available at https://secure.ipnexus.com/en/techs/42.

[65] Dipankar Dasgupta, Zahid Akhtar, and Sajib Sen. Machine learning in cybersecurity: a comprehensive survey. *The Journal of Defense Modeling and Simulation*, 19(1):57–106, 2022.

[66] H. Delfs and H. Knebl. *Introduction to Cryptography.* Springer Berlin, Heidelberg, Vol.2, 2008. [Online] Available at https://doi.org/10.1007/978-3-662-47974-2.

[67] Veena Desai, Ravindra Patil, and Dandina Rao. Using layer recurrent neural network to generate pseudo random number sequences. *International Journal of Computer Science Issues*, 9(2):324–334, 2012.

[68] Tim Dierks and Eric Rescorla. The transport layer security (tls) protocol version 1.2. Technical report, 2008.

[69] MT Dlamini, Jan HP Eloff, and Mariki M Eloff. Cbac4c: conflict based allocation control for cloud. In *The 9th International Conference for Internet Technology and Secured Transactions (ICITST-2014)*, pages 447–448. IEEE, 2014.

[70] Renáta Dubčáková. Eureqa: software review: Genet program evolvable, 2011. [ONLINE] Available at `https://doi.org/10.1007/s10710-010-9124-z`.

[71] Naone E. Technology overview, conjuring clouds, 2009. [ONLINE] Accessed on 6-Aug-2022 `https://www.technologyreview.com/2009/06/23/212416/conjuring-clouds/`.

[72] Ebay. Ebay online shopping, 2012. [ONLINE] Accessed on 6-Aug-2022 `https://www.ebay.com/`.

[73] Eci.com. Public and private clouds explained, 2022. [Online] Available at `https://www.eci.com/solutions/managed-services/cloud-solutions/managed-cloud/.Accessedon6-Aug-2022`.

[74] R Ehrenberg. Software scientist: With a little data, eureqa generates fundamental laws of nature, 2012.

[75] Agoston E Eiben and Marc Schoenauer. Evolutionary computing. *Information Processing Letters*, 82(1):1–6, 2002.

[76] Agoston E Eiben and James E Smith. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[77] Global Engage. Deep learning in digital pathology, 2020. [ONLINE] Availabile at http://www.global-engage.com/life-science/deep-learning-in-digital-pathology.

[78] Paul Erdos, Carl Pomerance, and Eric Schmutz. Carmichael's lambda function. *Acta Arith*, 58(4):363–385, 1991.

[79] Bob Evans. Cloud wars: Microsoft cloud to top 100 billion: 7 reasons behind remarkable achievement, 2022. [ONLINE] Availabile at https://accelerationeconomy.com/cloud-wars/microsoft-cloud-to-top-100-billion-7-reasons-behind-remarkable-achievement/.

[80] N Ferguson, B Schneier, and T Kohno. *Cryptography engineering: design principles and practical applications*. John Wiley & Sons, 2011.

[81] Céio Márcio Soares Ferreira, Charles Tim Batista Garrocho, Ricardo Augusto Rabelo Oliveira, Jorge Sá Silva, and Carlos Frederico Marcelo da Cunha Cavalcanti. Iot registration and authentication in smart city applications with blockchain. *Sensors*, 21(4):1323, 2021.

[82] NFC Flex. Nfc flex stamp antenna, 2020. [ONLINE] Available at https://productfinder.pulseeng.com/products/datasheets/L152.pdf.

[83] David B. Fogel. *Artificial Intelligence through Simulated Evolution*, pages 227–296. 1998.

[84] Alan Freier, Philip Karlton, and Paul Kocher. The secure sockets layer (ssl) protocol version 3.0. Technical report, 2011.

[85] A Gaeini, A Mirghadri, G Jandaghi, and B Keshavarzi. Comparing some pseudo-random number generators and cryptography algorithms using a general evaluation pattern. *International Journal of Information Technology and Computer Science*, 18:25–31, 2016.

[86] Gowtham Gajala. Cloud computing: A state of art of the cloud. *International Journal of Computer Trends and Technology*, 4(1):35–38, 2013.

[87] Gowtham Gajala. Cloud computing: A state of art of the cloud. *International Journal of Computer Trends and Technology*, 4(1):35–38, 2013.

[88] John Gantz and David Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future*, 2007(2012):1–16, 2012.

[89] Yun Gao, Ioannis Kontoyiannis, and Elie Bienenstock. Estimating the entropy of binary time series: Methodology, some theory and a simulation study. *Entropy*, 10(2):11–14, 2008.

[90] Craig Gentry. A fully homomorphic encryption scheme - phd thesis, 2009. [ONLINE] Available at `https://crypto.stanford.edu/craig/craig-thesis.pdf`.

[91] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[92] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*, pages 201–210. PMLR, 2016.

[93] Glowtec. Glowtec, reflective powder, 2021. [ONLINE] Available at `https://glowtec.co.uk/Reflective-powder/`.

[94] Krzysztof Gołofit and Piotr Z Wieczorek. Chaos-based physical unclonable functions. *Applied Sciences*, 9(5):991, 2019.

[95] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley,

Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[96] Google. Google app engine. development, 2009. [ONLINE] Accessed on 6-Aug-2022 `https://cloud.google.com/appengine`.

[97] Google.com. Google translate. [Online] Available at `https://translate.google.co.uk/`.

[98] J M Blackledge W Govere and D Sibanda. Phase-only digital encryption. *IAENG International Journal of Applied Mathematics*, 49:212–228, 2019. Available at `https://arrow.dit.ie/engscheleart2/193/`.

[99] M Haahr, W Scott, and D Eddelbuettel. random.org: Introduction to randomness and random numbers, 1999.

[100] O Hoare. Enigma: Code breaking and the second world war - the true story through contemporary documents, 2002.

[101] John H Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[102] Horizon. The horizon 2020 programme, 2021. [ONLINE] Available at `https://ec.europa.eu/programmes/horizon2020`.

[103] James M Hughes. Pseudo-random number generation using binary recurrent neural networks, 2007. [ONLINE] Available at `https://www.cs.umd.edu/~gasarch/TOPICS/cryptoml/crackprng.pdf`.

[104] AI System Inc. Artificial intelligence software, 2020. [ONLINE] Availabile at `https://aisystems.ai/`.

[105] Amazon Inc. Amazon elastic compute cloud (amazon ec2), 2022. [ONLINE] Accessed on 6-Aug-2022.

[106] ISC2. Insider threats can turn your cloud security into a storm - why are insiders a risk?, 2022. [ONLINE] Available at https://www.isc2.org/articles/insider-threats-can-turn-your-cloud-security-into-a-storm#:~:text=Why%20are%20insiders%20a%20risk,behavior%20across%20their%20cloud%20footprints.%E2%80%9D.

[107] J. Kelsey, B. Schneier, and N. Ferguson. Yarrow-160: Notes on the design and analysis of the yarrow cryptographic pseudo-random number generator. In *Springer - International Conference on Selected Areas in Cryptography*, pages 13–33. Springer, 1999. [Online] Available at https://link.springer.com/content/pdf/10.1007/3-540-46513-8_2.pdf.

[108] J M Blackledge and N Mosola. A statistically significant test to evaluate the order or disorder for a binary string of a finite length. In *ISSC2020, IEEE UK and Ireland Signal Processing Chapter and IEEE Computational Intelligence Society (UK & Ireland)*, pages 11–12, 2020. [Online] Available at https://arrow.tudublin.ie/engscheleart/311/.

[109] J M Blackledge, P Tobin, W Govere and D. Sibanda. Phase-only digital encryption using a three-pass protocol. In *ISSC2019, IEEE UK and Ireland Signal Processing Chapter, Maynooth University*, pages 17–18, 2019. [Online] Available at https://arrow.dit.ie/engscheleart2/193/.

[110] J M Blackledge, S Bezobrazov, P Tobin, and F Zamora. Cryptography using evolutionary computing. In *24th IET Irish Signals and Systems Conference (ISSC 2013)*, pages 1–8, 2013. [Online] Available at https://arrow.dit.ie/engscheleart/197/.

[111] Kishore Jaganathan, Yonina C Eldar, and Babak Hassibi. Phase retrieval: An overview of recent developments. *Optical Compressive Imaging*, pages 279–312, 2016.

[112] Goce Jakimoski and Ljupco Kocarev. Chaos and cryptography: block encryption ciphers based on chaotic maps. *Ieee transactions on circuits and systems i: fundamental theory and applications*, 48(2):163–169, 2001.

[113] Luke James. Moore's law in 2022: What's the status quo?, 2022. [ONLINE] Available at `https://www.power-and-beyond.com/moores-law-in-2022-whats-the-status-quo-a-dc63a87e669b554d4d33d2a5ba73692a/`.

[114] Abdo Abou Jaoude. The paradigm of complex probability and ludwig boltzmann's entropy. *Systems Science & Control Engineering*, 6(1):108–149, 2018.

[115] Blackledge Jonathan. Digital image processing in radon-space and the inversion of limited fourier data. *Optik*, 73(2):74–82, 1986.

[116] Prateek Joshi. *Artificial intelligence with python.* Packt Publishing Ltd, 2017. [ONLINE] Available at `https://www.amazon.com/Artificial-Intelligence-Python-ComprehensiveIntelligent/dp/178646439X`.

[117] Ansgar Kellner, Micha Horlboge, Konrad Rieck, and Christian Wressnegger. False sense of security: A study on the effectivity of jailbreak detection in banking apps. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 1–14. IEEE, 2019.

[118] A. Kerckhoff. La cryptographie militaire. *Journal des Sciences Militaires*, pages 161–191, 1883.

[119] Ali Khajeh-Hosseini, Ian Sommerville, and Ilango Sriram. Research challenges for enterprise cloud computing. *arXiv preprint arXiv:1001.3257*, 2010.

[120] Rajashri Khanai, GH Kulkarni, and DA Torse. Performance analysis of conventional crypto-coding. *International Journal of Latest Trends in Computing*, 2(1):193–197, 2011.

[121] Michael V Klibanov. On the recovery of a 2-d function from the modulus of its fourier transform. *Journal of mathematical analysis and applications*, 323(2):818–843, 2006.

[122] Morten Kolbæk, Zheng-Hua Tan, and Jesper Jensen. Speech intelligibility potential of general and specialized deep neural network based speech enhancement systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(1):153–167, 2016.

[123] Andrei N Kolmogorov. Three approaches to the quantitative definition ofinformation'. *Problems of information transmission*, 1(1):1–7, 1965.

[124] Andrei Nikolaevitch Kolmogorov. Entropy per unit time as a metric invariant of automorphisms. In *Dokl. Akad. Nauk SSSR*, volume 124, pages 754–755, 1959.

[125] S Kostadinov. Understanding the back-propagation algorithm, towards data science, 2019. [ONLINE] Availabile at `https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd`.

[126] K.Tirdad. *Developing pseudo random number generator based on neural networks and neuro-fuzzy systems - PhD thesis*. Ryerson University, 2010. [Online] Available at `https://rshare.library.ryerson.ca/articles/thesis/Developing_pseudo_random_number_generator_based_on_neural_networks_and_neurofuzzy_systems/14644767`.

[127] Cornell Creative Machine Lab. A software tool for detecting equations and hidden mathematical relationships in your data, 2013. [ONLINE] Available at `http://creativemachines.cornell.edu/eureqa`.

[128] Nutonian Lab. The ai-powered modeling engine, 2020. [ONLINE] Availabile at `www.nutonian.com`.

[129] Nutonian Lab. The ai-powered modeling engine, 2020. [ONLINE] Availabile at `https://www.datarobot.com/nutonian/`.

[130] T Latychevskaia. Examples of iterative phase retrieval algorithms, 2020. [Online] Available at https://uk.mathworks.com/matlabcentral/fileexchange/68646-examples-of-iterative-phase-retrieval-algorithms.

[131] Tatiana Latychevskaia. Iterative phase retrieval for digital holography: tutorial. *JOSA A*, 36(12):D31–D40, 2019.

[132] I M Llorente. Cloud interoperability and portability with opennebula, 2010. [Online] Available at https://opennebula.io/cloud-interoperability-and-portability-with-opennebula. Accessedon6-Aug-2022.

[133] David JC MacKay, David JC Mac Kay, et al. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[134] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 3–26. Springer, 2016.

[135] George Marsaglia. Random numbers fall mainly in the planes. *Proceedings of the National Academy of sciences*, 61(1):25–28, 1968.

[136] George Marsaglia and Arif Zaman. Monkey tests for random number generators. *Computers & mathematics with applications*, 26(9):1–10, 1993.

[137] James L Massey and Jimmy K Omura. Method and apparatus for maintaining the privacy of digital messages conveyed by public transmission, January 28 1986. US Patent 4,567,600.

[138] MathWorks. Ai with matlab, 2020. [ONLINE] Availabile at https://uk.mathworks.com/campaigns/offers/ai-with-matlab.html.

[139] MathWorks. Ai with matlab, 2020. [ONLINE] Availabile at https://uk.mathworks.com/campaigns/offers/machine-learning-with-matlab.html.

[140] MathWorks. Introducing deep learning with matlab, 2020. [ONLINE] Availabile at https://uk.mathworks.com/campaigns/offers/deep-learning-with-matlab.html.

[141] Robert Matthews. On the derivation of a "chaotic" encryption algorithm. *Cryptologia*, 13(1):29–42, 1989.

[142] J McNair. *The binary derivative: a new method of testing the appearance of randomness in a sequence of bits*. PhD thesis, MSc thesis, University of Western Ontario, 1989.

[143] P Mell and T Grance. The nist definition of cloud computing recommendations of the national institute of standards and technology, 2011. [ONLINE] Accessed on 6-Aug-2022 https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf.

[144] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 2018.

[145] Trend Micro. Security threats to evolving data centers. virtualization and cloud computing, 2022. [Online] Available at http://www.trendmicro.com/cloud-content/us/pdfs/securityintelligence/reports/rpt_security-threats-to-datacenters.pdf.

[146] Microsoft. Microsoft cloud platform - advantages of cloud computing, 2022. [ONLINE] Accessed on 6-Aug-2022 https://azure.microsoft.com/en-gb/?ocid=cloudplat_hp.

[147] Paul C Mogensen and Jesper Glückstad. Phase-only optical encryption. *Optics Letters*, 25(8):566–568, 2000.

[148] Gabriela Mogos. Ciphertext-policy attribute-based encryption using quantum multilevel secret sharing scheme. *IAENG International Journal of Computer Science*, 45(4), 2018.

[149] Robert H Morelos-Zaragoza. *The art of error correcting coding*. John Wiley & Sons, 2006. [Online] Available at `http://the-art-of-ecc.com/`.

[150] NN Mosola, MT Dlamini, JM Blackledge, JHP Eloff, and HS Venter. Chaos-based encryption keys and neural key-store for cloudhosted data confidentiality. 2017.

[151] NN Mosola, MT Dlamini, JHP Eloff, and MM Eloff. Evolutionary neural cryptosystem for cloud-bound data. Southern Africa Telecommunications Networks and Applications Conference ..., 2016.

[152] Suyel Namasudra and Ganesh Chra Deka. Introduction of dna computing in cryptography. In *Advances of DNA computing in cryptography*, pages 1–18. Chapman and Hall/CRC, 2018.

[153] Suruchee V Nandgaonkar and AB Raut. A comprehensive study on cloud computing. *International Journal of Computer Science and Mobile Computing, a Monthly Journal of Computer Science and Information Technology*, 3:733–738, 2014.

[154] Anshika Negi, Mayank Singh, and Sanjeev Kumar. An efficent security farmework design for cloud computing using artificial neural networks. *International Journal of Computer Applications*, 129(4):17–21, 2015.

[155] MA Nielsen and Isaac L Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010. "[Online] Available at `http://csis.pace.edu/ctappert/cs837-18spring/QC-textbook.pdf`".

[156] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[157] NIST. announcing request for nominations for public-key post-quantum cryptographic algorithms, 2016. [ONLINE] Available at https://www.federalregister.gov/documents/2016/12/20/2016-30615/announcing-request-for-nominations-for-public-key-post-quantum-cryptographic-alg

[158] NIST. cryptographic algorithm validation program, 2022. [ONLINE] Available at https://www.nist.gov/programs-projects/cryptographic-algorithm-validation-program.

[159] NIST. cryptographic algorithm validation program, 2022. [ONLINE] Available at url https://csrc.nist.gov/Projects/cryptographic-algorithm-validation-program/Random-Number-Generators.

[160] Mark Badger (NIST), Tim Grance (NIST), Robert Patt-Corner (Global Tech), and Jeffrey Voas (NIST). National institute of science and technology, 2012. [Online] Available at https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-146.pdf.

[161] Bank of England. Take a closer look: Your simple guide to checking banknotes, 2022. [ONLINE] Available at https://www.bankofengland.co.uk/-/media/boe/files/banknotes/take-a-closer-look.pdf.

[162] The University of Texas at Austin. The fbi apple security vs. privacy, 2015. [ONLINE] Availabile at https://ethicsunwrapped.utexas.edu/case-study/fbi-apple-security-vs-privacy.

[163] Diego Perez-Botero, Jakub Szefer, and Ruby B Lee. Characterizing hypervisor vulnerabilities in cloud computing servers. In *Proceedings of the 2013 international workshop on Security in cloud computing*, pages 3–10, 2013.

[164] Steve Pincus and Burton H Singer. Randomness and degrees of irregularity. *Proceedings of the National Academy of Sciences*, 93(5):2083–2088, 1996.

[165] SM Pinkus. Approximate entropy as a measure of system complexity. *Proc. Natl. Acad. Sci. USA*, 88(6):2297–2301, 1991.

[166] Krešimir Popović and Željko Hocenski. Cloud computing security issues and challenges. In *The 33rd international convention mipro*, pages 344–349. IEEE, 2010.

[167] The Open Quantum Safe Project. Software for prototyping quantum-resistant cryptography, 2016. [ONLINE] Available at `https://openquantumsafe.org/`.

[168] N Ptitsyn. *Cryptography using Deterministic Chaos*. PhD thesis, 2005.

[169] Nikolai V Ptitsyn, Jonathan M Blackledge, and Valery M Chernenkiy. Deterministic chaos in digital cryptography. In *Fractal Geometry*, pages 189–222. Elsevier, 2002.

[170] Random.org. random.org, 2022. [ONLINE] Available at `https://www.random.org/`.

[171] MS Ratsoma, MT Dlamini, JHP Eloff, and HS Venter. A conflict-aware placement of client vms in public clouds. In *10th International Conference on Cyber Warfare and Security. Reading: Academic Conferences and Publishing International Limited*, pages 501–503, 2015.

[172] Ingo Rechenberg. Case studies in evolutionary experimentation and computation. *Computer methods in applied mechanics and engineering*, 186(2-4):125–140, 2000.

[173] Eric Rescorla. The transport layer security (tls) protocol version 1.3. Technical report, 2018.

[174] Joshua S Richman and J Randall Moorman. Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology-Heart and Circulatory Physiology*, 278(6):H2039–H2049, 2000.

[175] John W Rittinghouse and James F Ransome. *Cloud computing: implementation, management, and security*. CRC press, 2017.

[176] Scott Rose, Oliver Borchert, Stu Mitchell, and Sean Connelly. Zero trust architecture. Technical report, National Institute of Standards and Technology, 2020.

[177] AL Ruhkin. Testing randomness: A suite of statistical procedures. *Theory of Probability & Its Applications*, 45(1):111–132, 2001.

[178] A Rukhin, S Juan, N James, S Miles, and B Elaine. A statistical test suite for random and pseudorandom number generators for cryptographic applications. *NIST Special Publication 800-22, Gaithersburg, MD, US*, 800:163, 05 2001.

[179] Andrew L Rukhin. Approximate entropy for testing randomness. *Journal of Applied Probability*, 37(1):88–100, 2000.

[180] Mark D Ryan. Cloud computing privacy concerns on our doorstep. *Communications of the ACM*, 54(1):36–38, 2011.

[181] Boose S. Cloud computing adoption by federal agencies increases, 2021. [Online] Available at `http://www.tripwire.com/state-of-security/latest-security-news/cloud-computing-adoptionfederal-agencies-increases-400`.

[182] Northcutt S. The attack surface problem, 2012. [ONLINE] Accessed on 6-Aug-2022 `https://www.sans.edu/research/security-laboratory/article/did-attack-surface`.

[183] Safescan.com. Safescan counterfeit detectors, 2022. [ONLINE] Available at `https://www.safescan.com/en-gb/store/counterfeit-detectors`.

[184] Kouichi Sakurai and Hiroki Shizuya. A structural comparison of the computational difficulty of breaking discrete log cryptosystems. *Journal of Cryptology*, 11(1):29–43, 1998.

[185] salesforce. salesforce cloud packages. [ONLINE] Accessed on 6-Aug-2022 `https://salesforce.com`.

[186] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[187] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[188] Bruce Schneier. *Applied cryptography: protocols, algorithms, and source code in C*. john wiley & sons, 2007.

[189] Amazon Web Services. Amazon linux security center, 2022.

[190] R. Shaltiel. An introduction to randomness extractors. In Luca Aceto, Monika Henzinger, and Jiří Sgall, editors, *Automata, Languages and Programming*, pages 21–41, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[191] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[192] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. IEEE, 1994.

[193] Yakov Sinai. Kolmogorov-sinai entropy. *Scholarpedia*, 4(3):2034, 2009.

[194] VA Stafford. Zero trust architecture. *NIST Special Publication*, 800:207, 2020.

[195] W. Stallings. *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 5th ed edition.

[196] Mark Stamp. *Introduction to machine learning with applications in information security*. Chapman and Hall/CRC, 2017.

[197] Katarina Stanoevska, Thomas Wozniak, and Santi Ristol. *Grid and cloud computing: a business perspective on technology and applications*. Springer Science & Business Media, 2009.

[198] Ken Stavinoha. Cloud computing update: Trends, risks, and mitigations. *Petroleum Accounting and Financial Management Journal*, 32(3):1, 2013.

[199] symbolab. Symbolab algebra calculator, 2020. [Online] Available at `https://en.wikipedia.org/wiki/Retroreflector`.

[200] M Sys, Z Riha, V Matyas, K Marton, and A Suciu. On the interpretation of results from the nist statistical test suite. *Romanian Journal Of Information Science AND Technology*, 18:18–32, 2015.

[201] Jakub Szefer and Ruby B Lee. A case for hardware protection of guest vms from compromised hypervisors in cloud computing. In *2011 31st International Conference on Distributed Computing Systems Workshops*, pages 248–252. IEEE, 2011.

[202] Leo Szilard. On the decrease of entropy in a thermodynamic system by the intervention of intelligent beings. *Behavioral Science*, 9(4):301–310, 1964.

[203] Rapid tables. Text to binary translator. [ONLINE] Available at `https://www.rapidtables.com/convert/number/ascii-to-binary.html`.

[204] Talkin'. takin'things, 2021. [ONLINE] Available at `https://talkinthings.com/`.

[205] Songpon Teerakanok, Tetsutaro Uehara, and Atsuo Inomata. Migrating to zero trust architecture: Reviews and challenges. *Security and Communication Networks*, 2021:1–10, 2021.

[206] Velvet-Belle Templeman and Velvet-Belle Templeman. The security threat of quantum computing, 2022.]ONLINE] Available at `https://www.digitalnationaus.com.au/news/the-security-threat-of-quantum-computing-580559#:~:text=A%20quantum%20computer%20can%20slash,larger%20keys%20or%20larger%20digests.`

[207] David Mathew Thomas and Sandeep Mathur. Data analysis by web scraping using

python. In *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 450–454. IEEE, 2019.

[208] P Tobin, J M Blackledge, and S Bezobrasov. Personalized encryptor generation for the cloud using evolutionary computing. In *International Workshop on Nonlinear Maps and their Applications (NOMA-2015), Dublin, Ireland*, 2015.

[209] Paul Tobin, Mick McKeever, Lee Edward Tobin, and Jonathan Blackledge. Secrecy and randomness: Encoding cloud data locally using a one-time pad. *URL: http://www. iariajournals. org/security/tocv10n34. html*, 2018.

[210] Alan M Turing and J Haugeland. Computing machinery and intelligence. *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, pages 29–56, 1950.

[211] Martin J Turner, Jonathan M Blackledge, and Patrick R Andrews. *Fractal geometry in digital imaging*. Academic Press, 1998.

[212] V. Desai, V. Deshmukh, and D. Rao. Pseudo random number generator using elman neural network. *International Journal of Computer Science Issues*, 9:324–334, 2012.

[213] Gilbert S Vernam. Cipher printing telegraph systems: For secret wire and radio telegraphic communications. *Journal of the AIEE*, 45(2):109–115, 1926.

[214] A.J. Viterbi. A personal history of the viterbi algorithm. *IEEE Signal Processing Magazine*, 23(4):120–142, 2006.

[215] J Walker. Ent: A pseudorandom number sequence test program, 2008. [ONLINE] Available at https://www.fourmilab.ch/random/.

[216] Daniel D Wheeler. Problems with chaotic cryptosystems. *Cryptologia*, 13(3):243–250, 1989.

[217] Wikibooks. Artificial neural networks/neural network basics, 2019. [ONLINE] Availabile at `https://en.wikibooks.org/wiki/Artificial_Neural_Networks/Neural_Network_Basics`.

[218] wikipedia. Radio frequency identification (rfid), 2020. [ONLINE] Available at `https://en.wikipedia.org/wiki/Radio-frequency_identification`.

[219] wikipedia. Retroreflector, 2021. [ONLINE] Available at `https://en.wikipedia.org/wiki/Retroreflector`.

[220] wikipedia. List of random number generators, 2022. [ONLINE] Available at `https://en.wikipedia.org/wiki/List_of_random_number_generators`.

[221] H Wortmann. Business consequences of cloud computing. In *Innovate IT 2010 Conference*, pages 91–95, 2010.

[222] X-Ways. Winhex: Computer forensics & data recovery software, hex editor & disk editor, 2022. [ONLINE] Available at `https://www.x-ways.net/winhex`.

[223] Keita Xagawa, Akira Ito, Rei Ueno, Junko Takahashi, and Naofumi Homma. Fault-injection attacks against nist's post-quantum cryptography round 3 kem candidates. In *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part II 27*, pages 33–61. Springer, 2021.

[224] Pengtao Xie, Misha Bilenko, Tom Finley, Ran Gilad-Bachrach, Kristin Lauter, and Michael Naehrig. Crypto-nets: Neural networks over encrypted data. *arXiv preprint arXiv:1412.6181*, 2014.

[225] S Yadav. Comparative study on open source software for cloud computing platform: Eucalyptus, openstack and opennebula. *International Journal Of Engineering And Science*, 3(10):51–54, 2013.

[226] Wei Zhang, Kazuyoshi Itoh, Jun Tanida, and Yoshiki Ichioka. Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Applied optics*, 29(32):4790–4797, 1990.

Appendix A

Function to Compute Some Basic Statistics for the Relative Entropy Test

The MATLAB function RBET (Relative Binary Entropy Test) given in this Appendix has not been exhaustively tested and has no data error checks. It is provided to give the reader a guide to the basic programming required to implement the computational procedures discussed in Section IV. The code given has been commented but is highly condensed in order to comply with the prescribed page limit for this publication.

```
function [Mean,Std,Median,Mode]=RBET(M);
%INPUTS: (int) M - length of the
%Relative Entropy Signal (RES).
%OUTPUTS: Mean - Mean of the RES.
%Std - Standard Deviation.
%Median - Median of the RES.
%Mode - Mode of the RES.
%Shuffle random number generator.
rng('shuffle');
```

```
%Read binary string from default file.
fid1=fopen('binary_string.txt','r');
bstring=fread(fid1); fclose(fid1);
%Compute length of string
L=size(bstring,1);
%and convert binary string to a bit array B with elements equal to 0 and 1.
for n=1:L
temp=bstring(n); if temp==48, B(n)=0;
else B(n)=1; end
end %Compute binary histogram
h=hist(B,2);%and evaluate
%probabilities of bits
p(1)=h(1)/L; p(2)=h(2)/L;
%Compute relative entropy signal.
for m=1:M
%Return random bits using function rand,
RB=round(rand(1,L));
%compute binary histogram
h=hist(RB,2);%and evaluate
%probabilities of bits.
q(2)=h(2)/L; q(1)=h(1)/L;
%Compute relative entropy.
RES(m)=-sum(p.*log2(q/p)); end
%Plot the RES in figure 1.
figure(1), plot(RES);
%Compute the 100-bin histogram
h=hist(RES,100);%and plot a bar
%graph of the result in figure 2.
figure(2), bar(h);%Compute the
```

```
%Mean, Standard Deviation, Median and Mode of the RES.
Mean=mean(RES); Std=std(RES);
Median=median(RES); Mode=mode(RES);
```

Appendix B

Functions for Implementation of a Three-pass Protocol using Phase-only Encryption

B.1 Function for Plaintext (.txt) Exchange

The function povided implements the step-by-step computational procedures Specified in Chapter 6, Section 7.3. Where possible, the notation used for array variables and constants are based on the mathematical notation used.

```
function []=TPP(key,step,c)
%FUNCTION: Exchange Plaintext composed of an array of integers between
%two %user - User_1 and User_2 using the Three-pass Protocol (TPP) with
%Phase-only Encryption
%INPUTS:
%key: Key(s) used to execute TPP where 'key' is a string of integer
%numbers between 0 and 9 with a maximum string length of 10
%(the limiting upper bound for a MATLAB random number generator
```

```
%with a non-negative integer seed <2^32)
%step:
%step=1 - first pass (first encrypt)
%step=2 - second pass (second encrypt)
%step=3 - third pass (first decrypt)
%step=4 - decryption (second decrypt)
%c: Spectral Embedding Constant c>>1, user defined for the
%first pass only. Specification of c is not required
%for execution of steps 2, 3 and 4.
%Apply step 1 - first pass.
if step==1
%Read plaintext P (taken to be an array of  integers from
%0 to 9) from an ASCII delimited file - Plaintext.txt -
%generated by User_1 to be transferred to User_2.
P=dlmread('Plaintext.txt');
%Zero pad the first element of array due to re-normalisation
%condition which needs to be applied in step 4 when the
%first element of the decrypt is eliminated from the output.
zero=zeros(1,1); P=[zero P];
N=size(P',1);%Compute size of P.
%Generate cipher using function
%'rand' seeded for first user
%defined key.
rng(key,'twister'); Theta=rand(1,N);
%Compute phase-only spectrum POS.
POS=exp(i*angle((fft(Theta))));
%Compute phase-only encrypted spectrum E, embed the result
%and return the real component of inverse DFT.
E=(fft(P).*POS)+c*POS;
```

```
E=real(ifft(E));
%Write out first pass ciphertext to file which is then
%sent by User_1 to User_2
dlmwrite('Ciphertext.txt',E, ...
'delimiter',' ','precision',32);
end
%Apply step 2 - second pass.
if step==2
%Read first passed ciphertext file
%(received by User_2 from User_1).
E=dlmread('Ciphertext.txt');
N=size(E',1); %Compute size of array.
%Computer fft of first pass ciphertext.
E=fft(E);
%Generate new cipher using function 'rand' seeded by
%second user defined key.
rng(key,'twister'); Theta=rand(1,N);
%Compute phase-only encrypted spectrum and return
%real component of inverse DFT.
E=E.*exp(i*angle((fft(Theta))));
E=real(ifft(E));
%Write out second pass ciphertext to file which is
%then sent by User_2 to User_1.
dlmwrite('Ciphertext.txt',E, ...
'delimiter',' ','precision',32);
end
%Apply step 3 - third pass.
if step==3
%Read second passed ciphertext file.
```

```
%(received by User_1 from User_2).
E=dlmread('Ciphertext.txt');
N=size(E',1); %Compute size of array.
%Computer fft of second pass ciphertext.
E=fft(E);
%Generate cipher using function 'rand' seeded by first
%user defined key.
rng(key,'twister'); Theta=rand(1,N);
%Decrypt phase-only spectrum for first pass and return
%the real component of inverse DFT.
E=E.*exp(-i*angle((fft(Theta))));
E=real(ifft(E));
%Write out third pass ciphertext to file which is then
%sent by User_1 to User_2.
dlmwrite('Ciphertext.txt',E, ...
'delimiter',' ','precision',32);
end
%Apply step 4 - Decryption of third pass cipher.
if step==4
%Read third pass cipher from file.
%(received by User_2 from User_1).
E=dlmread('Ciphertext.txt');
N=size(E',1); %Compute array size.
%Computer fft of second pass cipher.
E=fft(E);
%Generate cipher using function 'rand' seeded by
%second user defined key.
rng(key,'twister'); Theta=rand(1,N);
%Decrypt phase-only spectrum for second pass,
```

```
%return real component of inverse DFT and re-normalise
%by setting the first element of the array to zero.
E=E.*exp(-i*angle((fft(Theta))));
P=real(ifft(E)); P(1)=0.0;
%Convert return to integer values and eliminate first
%element of the associated array - square brackets.
P=round(P); P(1)=[];
%Write out decrypt to Plaintext.txt file.
dlmwrite('Plaintext.txt',P, ...
'delimiter',' '); end
```

B.2 Function for Full Colour JPEG Image Exchange

The function is a prototype MATLAB Functions for the implementation of a No-key(s) Protocol using Phase-only Encryption to exchange a JPEG image

```
function []=NKP(key,step)
%PROCESS: To exchange a full colour jpg image using a No-key(s) Protocol (NKP).
%INPUTS: key - an integer string (0-9) with a max string length of 7 digits.
%step=1: First pass encryption.
%step=2: Second pass encryption
%step=3: Third pass decryption
%step=4: Final decrypt for jpg image.
%EXTERNAL FUNCTIONS:
%POX - Phase-only Encryption/Decryption.
%WTI - Tagged image file (tif) output.
%IF - Attack using Inverse Filter.
if step==1%Apply processing for step 1.
%Read an image I from a jpg file.
I=imread('Image.jpg');
```

```
%Extract RGB components of the colour image and convert RGB arrays to double.
I_R =I(:,:,1);I_G =I(:,:,2);I_B =I(:,:,3);
I_R=im2double(I_R);I_G=im2double(I_G);I_B=im2double(I_B);
%Reconstruct colour image,
I = cat(3,I_R,I_G,I_B);
%Output new MATLAB generated jpg image
imwrite(I,'Plaintext.jpg','jpg');
%and display the image in figure 1.
figure(1), imshow(I);
%Phase-only encrypt the image,
I=POX(I,key,step);
%display ciphertext in figure 2
figure(2), subplot(2,2,1),
imshow(I./max(max(I)));
%and output ciphertext to Ciphertext_1.tif.
WTI(I,1); end
%Apply processing for second pass.
if step==2 %Read first pass ciphertext,
I=imread('Ciphertext_1.tif');
%phase-only encrypt,
I=POX(I,key,2);%display ciphertext
figure(2), subplot(2,2,2),
imshow(I./max(max(I)));
%and output ciphertext to default tif.
WTI(I,2); end
%Apply processing for third pass.
if step==3 %Read second pass ciphertext,
I=imread('Ciphertext_2.tif');
%phase-only decrypt,
```

```
I=POX(I,key,3);%display ciphertext
figure(2), subplot(2,2,3),
imshow(I./max(max(I)));
%and output ciphertext to default tif,
WTI(I,3); end
%Apply processing for step 4, i.e. decryption of third pass ciphertext.
if step==4 %Read third pass cipher.
I=imread('Ciphertext_3.tif');
%Phase-only decrypt,
I=POX(I,key,4);%display in figure 2
figure(2), subplot(2,2,4), imshow(I);
%and output result to Decrypt.jpg.
imwrite(I,'Decrypt.jpg','jpg');
%Compute Least Squares Error between Plaintext and Dercypt jpg images.
A=imread('Plaintext.jpg'); B=imread('Decrypt.jpg');
E=sum(abs(A-B).^2,'all')
%Apply 3-pass intercept cryptanalysis.
%Read intercepts.
I_1=imread('Ciphertext_1.tif');
I_2=imread('Ciphertext_2.tif');
I_3=imread('Ciphertext_3.tif');
%Extract RGB components associated with each intercept & convert
%to type double.
%First intercept.
IR_1 =I_1(:,:,1);IG_1 =I_1(:,:,2);
IB_1 =I_1(:,:,3);IR_1=im2double(IR_1);
IG_1=im2double(IG_1);IB_1=im2double(IB_1);
%Second intercept.
IR_2 =I_2(:,:,1);IG_2 =I_2(:,:,2);
```

```
IB_2 =I_2(:,:,3);IR_2=im2double(IR_2);
IG_2=im2double(IG_2);IB_2=im2double(IB_2);
%Third intercept.
IR_3 =I_3(:,:,1);IG_3 =I_3(:,:,2);
IB_3 =I_3(:,:,3);IR_3=im2double(IR_3);
IG_3=im2double(IG_3);IB_3=im2double(IB_3);
%Recover plaintext using inverse filter.
[J_R,status_R]=IF(IR_1,IR_2,IR_3);
[J_G,status_G]=IF(IG_1,IG_2,IG_3);
[J_B,status_B]=IF(IB_1,IB_2,IB_3);
%Check status of process.
if status_R==0 | status_G==0 | status_B==0
%and print note that attack has failed.
fprintf('Failed to recover plaintext\n');
else%else reconstruct colour image and
I = cat(3,J_R,J_G,J_B);%show in figure 3.
figure(3), imshow(I); end
end
```

B.3 Common Functions

```
function [J]=POX(I,key,step)
%PROCESS:Phase-Only Encrypt/Decrypt.
%INPUTS: Colour Image (I), key & step.
%OUTPUT: Colour Image (J).
%INTERNAL PARAMETER: alpha
alpha=500;%Set value of alpha.
%Extract RGB component of the input colour image, evaluate  image size
%and convert RGB arrays to type double.
```

```
I_R =I(:, :, 1); I_G =I(:, :, 2);
I_B =I(:, :, 3); [N,M]=size(I_R);
I_R=im2double(I_R);  I_G=im2double(I_G);
I_B=im2double(I_B);
%Compute uniformly distributed phase arrays for each RGB component using
%the input key and function 'rand'. Each RGB component key is obtained by
%multiplying the input 'key' with the ASCII decimal integers for R G and B.
key_R=key*82;key_G=key*71;key_B=key*66;
rng(key_R,'twister');Theta_R=rand(N,M);
rng(key_G,'twister');Theta_G=rand(N,M);
rng(key_B,'twister');Theta_B=rand(N,M);
%Compute the two-dimensional discrete Fourier transforms of each array and
%the phase angles angles using function 'angle' to give random phase spactra.
if step==1 | step==2%For step=1 & 2,
    sign=1; end %set sign=1 to encrypt.
if step==3 | step==4%For step=3 & 4,
    sign=-1; end %set sign=-1 to decrypt.
N_R=exp(sign*i*angle(fft2(Theta_R)));
N_G=exp(sign*i*angle(fft2(Theta_G)));
N_B=exp(sign*i*angle(fft2(Theta_B)));
if step==2
%Re-scale data using ESF (alpha).
c=exp(-alpha);N_R=N_R.*c; N_G=N_G.*c;
N_B=N_B.*c;  end
%Re-scale data for step=4.
if step==4
c=exp(alpha); N_R=N_R.*c; N_G=N_G.*c;
N_B=N_B.*c; end
%Encrypt input RGB components using phase only ciphers generated returning
```

```
%the real components of ifft2.
I_R=real(ifft2((fft2(I_R).*N_R)));
I_G=real(ifft2((fft2(I_G).*N_G)));
I_B=real(ifft2((fft2(I_B).*N_B)));
%Reconstruct colour image
J = cat(3,I_R,I_G,I_B);
```

```
function WTI(I,step);
%PROCESS: Write image I to a tif file maintaining floating point values of I.
%The function is based on Writing an image with floating point values, from
%Stackoverflow available at %https://stackoverflow.com/questions/14003402/
%writing-an-image-with-floating-point-values/33353930
%INPUTS: Image array (I) and step
%If step=1 (first pass) write I
%to'Ciphertext_1.tif'
if step==1
t = Tiff('Ciphertext_1.tif','w');end
%If step=2 (second pass) write I to 'Ciphertext_2.tif'
if step==2
t = Tiff('Ciphertext_2.tif','w');end
%If step=3 (third pass) write I to 'Ciphertext_3.tif'
if step==3
t = Tiff('Ciphertext_3.tif','w');end
t.setTag('Photometric',Tiff.Photometric.RGB);
t.setTag('BitsPerSample',64);
t.setTag('SamplesPerPixel',3);
tagstruct.RowsPerStrip=16;
t.setTag('SampleFormat',Tiff.SampleFormat.IEEEFP);
t.setTag('ImageLength',size(I,1));
t.setTag('ImageWidth', size(I,2));
```

```
t.setTag('PlanarConfiguration',Tiff.PlanarConfiguration.Chunky);
tagstruct.Software='MATLAB';
t.setTag(tagstruct);
t.write(I);t.close();
```

```
function [f,status] = IF(I_1,I_2,I_3)
%PROCESS:Recover plaintext using inverse filter for a three-pass attack.
%INPUT: Intercepts for first (I_1), second (I_2) and third (I_3) passes.
%OUTPUT: Plaintext f; if the process fails, f=0 and status=0, else status=1.
%Compute spectra of ciphertexts.
C_1=fft2(I_1);C_2=fft2(I_2);C_3=fft2(I_3);
%Compute spectrum of inverse filter, checking that all elements > 0.
D=abs(C_2).^2; Check = all(D(:) > 0);
if Check==1
F=(C_1.*C_3.*conj(C_2))./D;
%Return real component of ifft2 and set status to value > 0.
f=real(ifft2(F));status=1; else
%Print note on singularity of filter,
fprintf('Inverse Filter is Singular\n');
%assign output to 0 and return status=0.
f=0; status=0; end
```