Analysis and Design Security Primitives Based on Chaotic Systems for eCommerce

MAQABLEH, MAHMOUD,MOHAMMAD

Analysis and Design Security Primitives Based on Chaotic Systems for eCommerce

Mahmoud Mohammad Maqableh

A Thesis presented for the degree of

Doctor of Philosophy

School of Engineering and Computing Sciences

Durham University

United Kingdom

2012

To My Father: ***Mohammed Maqableh,***

Who has been a great source of motivation, inspiration and endless support in my life

Thank you for all your sacrifices for me to help me become what I am now

To My Mother: ***Maymonah Al-Douri***

For all the support, prayers, sacrifices and faith in me

Thank you for everything you have done for me since I was born

To the Soul of My Grandfather: ***Mousa Maqableh***

Who was the first to encourage and support me in continuing my education

To My Mother-in-law: ***Muneerah Al-Thebyan***

For her support, encouragement and endless love

To My Wife: ***Huda Karajeh***

For her endless love, support, encouragement and belief in me

Thank you for being there during the hardest of times

To My Lovely Son: ***Ayham***

Who has grown into a wonderful 2-year-old even though my wife and I were spending so much time working on our PhD theses

Abstract

Security is considered the most important requirement for the success of electronic commerce, which is built based on the security of hash functions, encryption algorithms and pseudorandom number generators. Chaotic systems and security algorithms have similar properties including sensitivity to any change or changes in the initial parameters, unpredictability, deterministic nature and random-like behaviour. Several security algorithms based on chaotic systems have been proposed; unfortunately some of them were found to be insecure and/or slow.

In view of this, designing new secure and fast security algorithms based on chaotic systems which guarantee integrity, authentication and confidentiality is essential for electronic commerce development. In this thesis, we comprehensively explore the analysis and design of security primitives based on chaotic systems for electronic commerce: hash functions, encryption algorithms and pseudorandom number generators. Novel hash functions, encryption algorithms and pseudorandom number generators based on chaotic systems for electronic commerce are proposed. The securities of the proposed algorithms are analyzed based on some well-know statistical tests in this filed. In addition, a new one-dimensional triangle-chaotic map (TCM) with perfect chaotic behaviour is presented.

We have compared the proposed chaos-based hash functions, block cipher and pseudorandom number generator with well-know algorithms. The comparison results show that the proposed algorithms are better than some other existing algorithms. Several analyses and computer simulations are performed on the proposed algorithms to verify their characteristics, confirming that these proposed algorithms satisfy the characteristics and conditions of security algorithms. The proposed algorithms in this thesis are high-potential for adoption in e-commerce applications and protocols.

Declaration

The work in this thesis is based on research carried out at School of Engineering and Computing Sciences, Durham University. No part of this thesis has been submitted elsewhere for any other degree or qualification and it all my own work unless referenced to the contrary in the text.

Acknowledgement

First of all thanks to Allah Almighty for giving me the strength to complete the thesis work and making my dream come true. I have for so many years dreamt of the day when I would acquire my PhD and the day has finally come. I pray that this work is accepted by Him and that He guides me to the Straight Path in this life and in the Hereafter, Amen.

This thesis could never have been completed without the guidance, help and support of many people. I would like to thank my supervisor Dr. Stefan Dantchev for all his guidance and support whenever I needed it, for the freedom he has given me to explore my research interests, and for all his advice and encouragement. Special thanks go to Dr. Rifat Shannak for his support and encouragement throughout my study. I would like to take this opportunity to thank Dr. Azman Samsudin for his guidance and encouragement to continue my PhD study. Many thanks and grateful acknowledgement for proofreading and correcting the English edition go to John R. Coast.

I would like to thank deeply my parents, who have helped, guided and supported me all my life. My deepest gratitude goes to my wonderful brothers, sisters and their families, Dr. Ayman, Iman, Wafa'a, Nisreen, Fatemah, Ahmad, Ghadeer and Mousa, for all the love, prayers and best wishes during my studies. I would also like to thank my best friend and brother Amjed and his family for their prayers, support and encouragement.

I would like to thank Jordan University for providing me with the funding and support for my work.

Last, but not least, I would like to thank all my friends, both in my country Jordan, and in Durham city, for their prayers and support, especially during the difficult times of the PhD journey. Thank you all.

Mahmoud M. Maqableh

List of Publications

- Mahmoud M. Maqableh and S. Dantchev, Cryptanalysis of Chaos-Based Hash Function (CBHF) in First International Alternative Workshop on Aggressive Computing and Security - iAWACS. 2009: France-Laval. (**Chapter 4**)

- Mahmoud M. Maqableh, Secure Hash Functions Based on Chaotic Maps for E-Commerce Application. International Journal of Information Technology and Management information System (IJITMIS), 2010. 1(1): p. 12-19. (**Chapter 3**).

- Mahmoud M. Maqableh, Fast Hash Function Based on BCCM Encryption Algorithm for E-Commerce (HFBCCM), in 5th International Conference on e-Commerce in Developing Countries: with focus on export. 2010: Kish Island - Iran. (**Chapter 7**)

- Mahmoud M. Maqableh, Fast Parallel Keyed Hash Functions Based on Chaotic Maps (PKHC), in Western European Workshop on Research in Cryptology. 2011: Weimar - Germany. (**Chapter 5**)

- Mahmoud M. Maqableh and S. Dantchev, Fast Encryption Algorithm Based on Chaotic Maps for E-commerce (BCCM). Journal of Telecommunication Systems, Springer, under review. (**Chapter 6**)

- Mahmoud M. Maqableh, A Novel Pseudorandom Number Generator Based on New Triangle-Chaotic Map for High Security Applications. Journal of Communications in Nonlinear Science and Numerical Simulations, Elsevier, Submitted 2011.(**Chapter 8**)

- Mahmoud M. Maqableh, Fast Parallel Hash Functions Based on Chaotic Maps for eCommerce (PHFC), in Western European Workshop on Research in Cryptology, under review. Weimar- Germany: Lecture Notes in Computer Science, Springer. (**Chapter 4**)

Table of Content

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Thanks to recent advanced developments in communications and computer technologies, the Internet has become widespread and is used for the purpose of supporting client and server services. The Internet has enabled collaboration and supporting interactivities between individuals, government agencies, academic institutions and businesses of all sizes [1]. People have become dependent on the Internet for personal and professional usages. Many people perform their shopping, payments, money transfers, and many other electronic commerce activities over the Internet. With the increased usage of and dependence on the Internet, information security and privacy have become major problems, and researchers have been motivated to solve these problems to protect Internet users [2-5].

Electronic commerce services have grown rapidly to become a core element of the Internet and web environment, resulting in the development of new strategies and new eCommerce applications [2, 3]. Many people still prefer to engage in physical commerce instead of electronic commerce due to the electronic commerce security problems, which lead to customers' lack of trust in electronic commerce. Electronic commerce transactions' security is considered one of the crucial factors in ensuring the success of electronic commerce [4]. Therefore, it is very necessary to have security to protect the data on computers, distribution systems, individual organizations, and other organizations that communicate with each other [6]. Nowadays, cryptography plays a significant role in protecting systems and personal and secret information such as payment systems and credit card information [7].

1.2 Cryptography and Cryptanalysis

Cryptography is the art and science of encrypting and decrypting data to be protected while it is stored or transferred over insecure networks; this can be achieved by designing cryptographic techniques. On the other hand, *cryptanalysis* is the art and science of studying and analyzing cryptographic techniques to break them [7-11]. It is very clear that there are strong relations between cryptography and cryptanalysis. Cryptography has a very long history dating back over 4000 years [10]. In 1976, cryptography underwent a remarkable development after Diffie and Hellman published a paper entitled "new directions in cryptography" in which they introduced the public key concept and provided different methods of key exchange [12]. In general, cryptography algorithms are used mainly for encrypting and/or signing data. Cryptography algorithms are categorised into three main categories: symmetric cryptosystem, asymmetric cryptosystem, and cryptography hash function. A cryptosystem is a cryptographic algorithm that depends on certain parameters called keys [13]. A symmetric cryptosystem uses one key to encrypt and decrypt messages. On the other hand, an asymmetric cryptosystem uses two keys, a public key and a private key, to encrypt/decrypt an input message. A symmetric key primitive can be further divided into two main categories: block cipher and stream cipher. In addition, a cryptographic hash function can be further divided into two main categories: keyed hash function and unkeyed hash function.

A cryptographic hash function is a deterministic procedure that processes variable-length input messages and produces fixed-length hash values. Hash values are used in many different applications such as verifying message integrity, message identity, message authentication code, key derivation, password derivation, password verification, and pseudorandom number generator. A block cipher is a symmetric key encryption method that encrypts/decrypts one group of data at the same time and generates output of the same size based on encryption/decryption method and shared secret key. A block cipher can be used to construct other cryptography primitives such as stream cipher using OFB and CTR modes, hash function, message authentication code, and secure pseudorandom number generator. A cryptographic pseudorandom number generator is a deterministic procedure generating a random statistical independent sequence of bits using a mathematical formula that is used to provide high security for many cryptographic applications such as key generation, one-time pads, prime numbers in RSA, digital signature, Nonces (numbers used once), and Salts. In general, cryptography algorithms are considered computationally secure if they cannot be attacked using different types of analysis and available resources.

There are three basic cryptanalysis (attack) techniques in cryptology: known plaintext attack, chosen plaintext attack, and ciphertext only attack [6-10, 14]. Under known plaintext attack (KPA), the attacker has pairs of plaintext and ciphertext, which can be used to deduce the used secret key or the original text. Under chosen plaintext attack (CPA), the attacker is assumed to have the ability to encrypt random plaintext using encryption algorithm to obtain its corresponding ciphertext or decrypt random ciphertext to obtain its corresponding plaintext. In the case of ciphertext only attack (COA), the attacker has access to a set of ciphertext only and tries to find corresponding plaintext of full or partial ciphertext or figure out the used key to reduce encryption scheme security. Moreover, another type of attack (brute-force attack) is to try all possible keys until one finds the right key that converts the ciphertext to intelligible plaintext.

1.3 Cryptography and Electronic Commerce

Over the last few years, electronic commerce has become very popular and has grown very rapidly [15]. Furthermore, different types of electronic commerce services have been developed, such as e-banking, e-shopping, e-bills, and e-payments. Most electronic payment systems utilize cryptography algorithms in several ways to guarantee the security and confidentiality of secure information [15-21]. Some of the most popular electronic payment systems are credit cards, e-cash and e-cheques. Electronic Commerce transactions are usually performed over insecure and un-trusted networks over which the trading parties have no control. Likewise, merchants' websites are liable to remote attack via the Internet by numerous attackers worldwide. In cryptography algorithms, a message digest is used to provide data integrity, digital signature is used to provide authentication, secret key is used for privacy, and public key is used for privacy and authentication. Therefore, cryptography can provide the capability of hiding electronic transactions, detecting message changes, and confirming the source of electronic transactions.

A strong and fast encryption algorithm for eCommerce transactions is considered an essential requirement. There are two types of cryptography encryption algorithms: symmetric and asymmetric key encryption algorithms. Symmetric key encryption is considered faster than asymmetric key encryption. Therefore, in eCommerce a combination of public key and symmetric key technologies is used to provide fast and confidential eCommerce transactions, where public key is considered confidential and slow but symmetric key is considered a fast and simple algorithm [22]. In addition, hash function and digital signature are used to detect message modifications, to guarantee message source, and to prevent message denial [16, 22, 23]. In electronic commerce systems, customers and merchants have many concerns that can be solved using cryptography algorithms and certificates methods [6]. The integrity of e-payments is achieved by using hash function and encryption algorithm within SSL and SET technologies [24]. An effective digital signature can be proved by a combination of hash function and public key encryption algorithm. Therefore, we need to utilize cryptography algorithms to offer high-security services to encourage customers to use eCommerce. We will explain the basic principles of cryptography in electronic commerce by taking an online bank account as an example.

1- Authentication is to allow the account owner only to login to the online bank account.
2- Authorization is to allow the account owner only to manipulate the account information and perform certain operations.
3- Encryption is to hide all account information and transactions from spying.
4- Auditing is to keep a record of transactions as proof of purchase.

5- Confidentiality is to allow authorized people only to read protected data.

6- Integrity is to ensure that the sent information is received as it was sent.

7- Availability is to ensure that the authorized people are able to access the data resources.

1.4 Chaotic Cryptography

Over the last few years, many researchers have studied chaos theory in several fields, such as electronic systems, fluid dynamics, lasers, weather, climate and cryptography [25-29]. Chaos theory has attracted the cryptography field due to its characteristics, such as its deterministic nature, unpredictability, random-look nature and its sensitivity to initial value [30]. Cryptographers have utilized dynamic chaotic systems to develop new cryptographic primitives by exploiting chaotic maps, such as logistic maps, Henon maps, and Tent maps.

There are similarities and differences between cryptography algorithms and chaotic maps [31]. The parameters in chaotic maps are meaningful, mostly if they on real numbers, which can be used in the cryptographic algorithms as encryption and decryption keys. Chaotic systems are sensitive to any change or changes in the initial condition(s) and are unpredictable in the long term, thus representing the diffusion in cryptographic encryption algorithms. Iterations of a chaotic map lead to the spreading of the initial region over the entire phase space, and this can be achieved in cryptographic algorithms by designing the algorithm based on rounds. The main difference between chaotic systems and cryptographic algorithms is that encryption transformations are defined on finite sets, whereas most of chaotic systems have meaning on real numbers [31]. There are few chaotic maps are defined on complex numbers such as complex squaring map, which to the best of my knowledge they have not been exploited in chaotic cryptography.

Since 1990, many studies on digital chaotic cryptography have been proposed to provide secure communications based on chaotic maps include chaotic block ciphers [13, 32-68], chaotic cryptography hash functions [30, 63, 69-89], and chaotic pseudorandom number generators [34, 62, 68, 90-117]. In general, chaos theory has been proved a secure algorithm against known cryptanalysis techniques. Recently, various studies have been conducted on chaotic cryptographic algorithms [118-155]. Some of the proposed chaotic cryptographic algorithms that have been analyzed have had weak internal designs and incorrect exploitation of chaotic maps. In this research, we will focus on studying and analysing chaotic cryptography hash function, chaotic block cipher encryption, chaotic maps, and chaotic cryptographic secure pseudorandom number generators.

1.5 Original Key Contributions of this Thesis

Recently, collisions of well-known hash functions such as PIREMD, MD4, MD5 and SHA-1 have occurred [156-158]. Moreover, security of full or reduce version of some block cipher algorithms are analyzed such as DES, RC6, RC5, IDEA and AES [14, 159-164]. SHA-1 is one of the most widely used hash functions employed in numerous security applications and protocols. Since SHA-1 was attacked in 2005, many researchers have been working on designing new alternative secure hash functions [165]. Designing a secure hash function based on chaos theory has attracted the interest of researchers due to its characteristics that are analogous to hash function requirements [30, 70, 76, 81, 87, 88]. In 2008, we designed a new hash function using a logistic map CHA-1 [30]. CHA-1 is a secure chaotic hash function, but it is at least three times slower than SHA-1. In this thesis, research on chaotic cryptography was initially motivated by our interest in designing a fast and secure hash function for electronic commerce applications. Later on, we realized that other secure chaotic cryptographic systems are needed for electronic commerce and other applications.

This thesis involves the following aspects of analysis and design of chaotic cryptographic algorithms for electronic commerce: cryptographic hash functions, block cipher, pseudorandom number generators and chaotic maps. The original key contributions of this thesis are listed as follows:

1- During our review of the chaotic hash function literature, we reviewed one of the proposed hash function algorithms based on a chaotic system that was called CBHF. We realized that the proposed hash function has a simple and weak design based on tent map. Therefore, we carefully studied its design and analyzed its security. In this research, we show in strong sense how to break keyed and unkeyed versions of CBHF theoretically and we give real collision examples.

2- After we analyzed the security of the CBHF hash function, we designed new parallel hash function based on chaotic maps for electronic commerce applications. Several analyses and computer simulations are performed to show the security and performance of our proposed hash function. Moreover, the proposed hash function is compared with other proposed hash function algorithms. Overall comparison results shows that the proposed hash function algorithm outperforms other algorithms.

3- In recent years, several chaotic block cipher encryption algorithms have been proposed. Some of the proposed algorithms have been proved to be insecure and/or having slow encryption speed. This motivated us to design and implement a new block cipher encryption algorithm by utilizing chaotic maps for electronic commerce applications. The security and performance of the proposed algorithm is analyzed using well-known computer simulations and theoretical analysis in this field. Both experimental results and computer simulations confirm that the proposed chaotic block cipher encryption algorithm satisfies the cryptographic properties.

4- The main function of the proposed block cipher encryption algorithm shows very high sensitivity to the input message and/or secret key. Characteristics of our proposed block cipher encryption function include the possibility of adapting its design to build and design a secure hash function. Therefore, our proposed chaotic block cipher encryption function is modified to design and build a new fast and secure chaotic hash function. We performed some statistical analysis that confirmed this hash function satisfies the cryptographic hash function properties. The proposed hash function algorithm has been compared with other cryptographic hash functions. These characteristics confirm that this hash function is high-potential to be adopted for secure and fast applications.

5- Most of the well-known chaotic maps show chaotic behaviour over small regions with certain parameter values. Moreover, within the chaotic areas there are some regions of n-periodic window with small values of n (2, 3, 6…), which can be exploited by attackers to help them attack the cryptographic systems that fall in these regions. Therefore, we designed a new triangle-chaotic map with full chaotic population and a very large value of periodic window. Triangle-chaotic map analysis shows its great sensitivity to initial conditions, unpredictability and intensive chaotic population. These properties confirm that the proposed triangle-chaotic map can be adopted for many applications in many disciplines such as computer science, engineering, mathematics, physics and economics.

6- A cryptographic secure pseudorandom number generator is used to provide high security for several cryptographic applications such as key generation and digital signature. In this research, a new cryptographic secure pseudorandom number generator based on the proposed triangle-chaotic map is put forward. The proposed generator was tested using the well-known NIST 800-22 suite test, which is designed to test cryptographic random and pseudorandom number generators, and compared with two other generators based on logistic map and modified version of logistic map, respectively. The tests and comparison results showed randomness of proposed generator output and non-randomness of the other two logistic generators' outputs. Therefore, the proposed generator is a high-potential candidate for high-security applications such as e-payments and online banking systems. Moreover, we proved experimentally that one of the proposed pseudorandom number generators based on logistic map is not secure at all.

1.6 Organization of this Thesis

Here is a brief chapter summary of this thesis.

- Chapter 2: gives an introduction to electronic commerce, electronic commerce security protocols and cryptography primitives. This chapter is intended as a solid introduction to cryptography in general and focuses on cryptographic hash functions, block cipher algorithms and random number generators. Hash function principles, properties and applications are given. Moreover, block cipher principles, evaluation, modes of operation and multiple modes encryption are discussed. In addition, different cryptanalysis techniques are presented.

- Chapter 3: introduces chaos theory, explains some of the chaotic maps and their characteristics, and focuses on analysis and design of a logistic map in detail. In addition, this chapter presents a comprehensive survey of chaotic cryptography. All of the proposed chaotic cryptographic algorithms are classified into categories. Several proposed chaotic cryptographic algorithms are studied in detail.

- Chapter 4: focuses on one of the proposed hash function algorithms based on chaotic maps, called CBHF. The security of keyed and unkeyed versions of the CBHF hash function is analyzed. Theoretical and practical analysis of the CBHF hash function algorithm is presented in detail. Examples of real collisions are given.

- Chapter 5: describes in detail the design and analysis of a novel developed parallel chaotic hash function. Comparisons between the proposed hash function and other proposed hash functions are presented. Several analyses and computer simulations of the proposed hash function are performed.

- Chapter 6: depicts details of our new proposed image/text block cipher encryption algorithm based on chaotic maps. Analysis of other proposed chaotic block cipher encryption algorithms is discussed. A comparison between the proposed algorithm and other block cipher algorithms is given. Experimental analysis results on security and performance of the proposed algorithm are presented in detail.

- Chapter 7: describes the details of design and analysis of a fast and secure hash function based on a modified version of our proposed block cipher encryption algorithm in chapter 7. The details of experimental results on the proposed hash function are depicted. A comparison between the proposed hash function and other hash functions is given in this chapter.

- Chapter 8: demonstrates the limitations of some existing chaotic maps, such as small size of periodic and partial chaotic population. In this chapter, the design and analysis of a new proposed triangle-chaotic map with full chaotic population is explained in detail. Several test and simulation results on the proposed chaotic map are given. Moreover, we propose novel cryptographic pseudorandom number generator based on a triangle-chaotic map. It is called PRGBT in this thesis. The proposed generator and two other pseudorandom number generators based on a logistic map and a modified version of a logistic map are tested and analyzed using NIST 800-22 test suite. Each performance of the NIST 800-22 test is briefly explained and the results of NIST statistical test suite for each generator are given.
- Chapter 9: gives a quick summary and conclusion of this research, followed by a list of open problems and future research issues.

Chapter 2

Basic Concepts

We dedicate this chapter to explaining the fundamental concepts of cryptography primitives. We start by discussing principles of security, electronic commerce, electronic commerce security protocols, cryptographic terms and cryptographic primitives. Furthermore, we discuss in detail block ciphers, cryptographic hash functions and random number generators. We briefly explain three well-known block cipher algorithms (DES, IDEA, and AES) and two well-known hash function algorithms (MD5, SHA-1). *Finally, confusion and diffusion properties and cryptanalytic techniques are presented.*

2.1 Principles of Security

It is very important to understand the principles of security in order to start thinking about the various methods of solving the security problems that occur in security systems such as attacks. Therefore, we will simplify the principles of security by taking a simple example from a real-life scenario. So, let us assume that a person A (**Alice)** lives in London and she wants to send a cheque worth £20,000 by post to another person B (**Bob)** who lives in Durham. What would Alice and Bob do in such a case [1]? Normally, Alice (Sender) would write the cheque (£20,000), put it inside an envelope and then send it by post to Bob (Receiver). The following security issues would be involved in this transmission:

A. Alice would like to ensure that only Bob will be able to access the envelope, so Alice will try to protect the cheque from unauthorized parties, preventing them reading or disclosing information. This is the principle of *confidentiality***.**

B. Bob would like to be sure that the cheque has really come from Alice and that it is not a falsified cheque sent by someone else claiming to be Alice. The receiver should know that "YOU ARE WHO YOU ARE". This is the principle of *authentication***.**

C. Alice and Bob would like to make sure that no one can forge the contents of the cheque (such as its amount, date, signature, name of the payee, etc.). Bob should receive the cheque "AS IT IS". This is the principle of *integrity***.**

From the previous example we note the following as major security issues that could occur in electronic commerce systems [3, 6, 7]:

- *Confidentiality* (Privacy): ensuring that the transmitted data are accessible only by authorized parties for reading. This is maintained by the use of the secret key, so the information will make sense only to those who have the secret key.

- *Authentication*: ensuring that the origin of the received message is the genuine sender, not an attacker. Thus, the communication is authentic and the received message is from the source whose identity is not in doubt.

- *Integrity*: ensuring that only the authorized parties are able to perform any modification (duplication, insertion, deletion and/or reordering) of the message - "message received as sent". Thus, the integrity ensures that no-one can change the message content without its detection by the sender or the receiver, as each different message has a different fingerprint encrypted by secret key.

2.2 Electronic Commerce

Electronic commerce (ecommerce or e-commerce) is defined as the process of buying, selling or exchanging products or services over the Internet [3]. Over the last few years, e-commerce has become very popular; it is growing rapidly, improving business efficiency and reducing business process costs [15]. Nowadays, e-commerce is a main channel for sales and services. Furthermore, various types of e-commerce services have been developed such as e-banking, e-shopping, e-bills and e-payments. E-commerce payment is defined as transferring an amount of money from payer account to payee account over the Internet. Online merchants can use various types of online payment systems such as e-cash, e-check, and digital wallet [22]. Moreover, another type of third-party online payment system is also available to complete consumers' online transactions.

E-commerce has to confront many problems and challenges. Providing high security for e-commerce systems is considered a major challenge to e-commerce success, and running e-commerce over an insecure network is considered a major problem in e-commerce. Insufficient security in e-commerce systems leads to the leakage of secure and personal information. Many e-commerce consumers are worried about using online payment systems due to the increase in online fraud and risks [24]. Thus, security of electronic payment systems is considered one of the most important barriers to e-commerce development [17, 22, 166]. Consequently, providing high security for electronic payment transactions will make e-commerce successful. The existing security technologies have failed to meet customers' needs; thus adapted security technologies need to be developed [16]. Providing high security for electronic commerce applications can be achieved using cryptography algorithms such as hash function and encryption algorithms. In e-commerce systems, customers and merchants have many concerns that can be solved using cryptography algorithms and certificates methods [6]. In general, customers need to ensure the following:

1. They are communicating with the intended merchant.
2. Only the merchant can read the sent message.
3. The message is received as it was sent.
4. They can provide evidence of exactly what they sent.

5. Guaranteed delivery of what they ordered.

On the other side, merchants need to ensure the following:

1. They are communicating with the intended customer.
2. Only the intended customer can send messages.
3. Identity of the customer is unmistakable.
4. Content of received message is correct.
5. They send acknowledgement of the received message.

2.3 Electronic Commerce Security Protocols

In e-commerce, various different security protocols are used, such as payment secure protocol and web secure protocol [16, 20, 22]. We will give brief details of the three most popular security protocols in e-commerce.

1. SSL

Secure Socket Layer (SSL), and its successor Transport Layer Security (TLS), is used to protect communication between running clients and servers [19]. It encrypts the outgoing data from sender to receiver and decrypts the incoming data on the receiver side. In general, SSL protocol helps the average user to handle various security issues such as encryption, digital signature and digital certificates by web browsers and web servers [20]. This protocol uses the public key technique to exchange a secret key that is used in encryption transaction to provide an efficient encryption algorithm [22]. Clients use SSL server authentication to prove server identity. One-way hash function is used in this protocol to detect message modifications and to protect the shared secret key.

2. SET

Secure Electronic Transaction (SET) was designed to provide security for electronic payment transactions and authentication of the parties involved in the transactions [22]. SET was developed by Visa and MasterCard with the collaboration of other software companies such as Microsoft and VeriSign [20]. Cryptographic algorithms and digital certificates are used in this protocol to ensure payment integrity. These provide the trust required by consumers and ensure the security and confidentiality of information. The sent message is encrypted using a randomly generated secret key that is encrypted using receiver public key. The receiver decrypts the encrypted secret key using his/her private key and then decrypts the encrypted message using a decrypted secret key. Some of the proposed protocols for electronic payment systems are based on security of cryptographic mechanisms [17]. One-way hash function is used in SET protocol to detect message modifications.

3. HTTPS

Hypertext Transfer Protocol Secure (HTTPS) is designed to add security and support at application level and various security mechanisms [19]. HTTPS protocol is a combination of HTTP and SSL/TLS protocols used to encrypt the transactions and provide security for web server and web browser. This protocol is mainly used to provide a secure transmission channel over an insecure network [22]. HTTPS connection is normally used with sensitive information transmission, such as payment transactions. In general, with https several protection mechanisms are used between client and host, such as message encryption, digital signature and digital certificate.

2.4 Cryptographic Terms

Cryptology is the mathematical science that studies cryptography and cryptanalysis sciences [7]. Cryptography is the art and science of encrypting and decrypting data to protect them while they are stored or transferred over insecure networks; this can be achieved by designing cryptographic techniques. On the other hand, cryptanalysis is the art and science of studying and analyzing cryptographic techniques to break them [8-11]. *Cryptographers* are people working to develop new cryptography algorithms to provide security services, while *Cryptanalysts* are people working to develop and find methodologies to break the cryptographic techniques [7, 167].

The original goal of cryptography is to protect data from unauthorized people by encrypting them; the encryption process is very necessary in communication to protect data from eavesdroppers and hackers [11]. In cryptographic terms [6, 11], the *plaintext* is the original text which can be read and understood by humans and the *ciphertext* is the original message after the encryption process; the ciphertext is apparently random and ambiguous to humans. The basic idea of encryption is to scramble the secret information in such a way that it cannot be understood by unauthorised people [13]. *Encryption* is the process of encoding data in such a way that hides them from any outsider; it is a simple transfer of plaintext to ciphertext while *decryption* is the reverse process of encryption, transferring ciphertext to plaintext [6, 7].

2.5 Cryptography Main Areas

In recent years, cryptography has been employed to deal with some important security issues, such as confidentiality, authentication, message integrity and non-repudiation [6]. According to Menezes et al. (1997), cryptography can be divided into three areas of study: symmetric primitive, asymmetric primitive and unkeyed primitive. *Symmetric* cryptosystems use one key to encrypt and decrypt messages. On the other hand, *asymmetric* cryptosystems use two keys; the first key is called the public key and is used to encrypt a message, while the second key is called the private key and is used to decrypt the scrambled message into its original state. The cryptographic hash functions can be divided into keyed hash functions, as part of symmetric primitive, and un-keyed hash functions, as part of unkeyed primitive. *Hash functions* is a part of the unkeyed primitive that is mainly a mechanism to calculate hash value from a given message for cryptography applications, such as message integrity [7]. In the following subsections we will illustrate asymmetric, symmetric and hash function primitives in more detail.

2.5.1 Asymmetric Key Primitives

Asymmetric encryption, also called two-key encryption or public key encryption, uses two keys: the first key is called public key and the second key is called private key. Typically, the public key is used to encrypt a message and the private key is used to decrypt the encrypted message into its original state; the two keys can also be used vice versa. The private key is used to encrypt the message then it will be decrypted by the public key, which is used to verify the sender signature by decrypting the message successfully using his/her public key. In general, the sender produces the ciphertext from the plaintext based on an encryption algorithm and public key and then sends the ciphertext to the receiver. On the receiver side, the receiver produces the plaintext from the ciphertext based on the decryption algorithm and his/her private key [6, 7]. If the decrypted message is understood, the receiver will assume that the received message has been encrypted by the sender's public key; otherwise, the receiver will decline the received message (see Figure 2-1) [8, 11]. The simple encryption and decryption equations are as follows:

$$E_{k_U}(M) = C \tag{2-1}$$

$$D_{k_R}(C) = M \tag{2-2}$$

where E is the encryption function, D is the decryption function, M is the plaintext, C is the ciphertext, K_R is the private Key, and K_U is public Key.

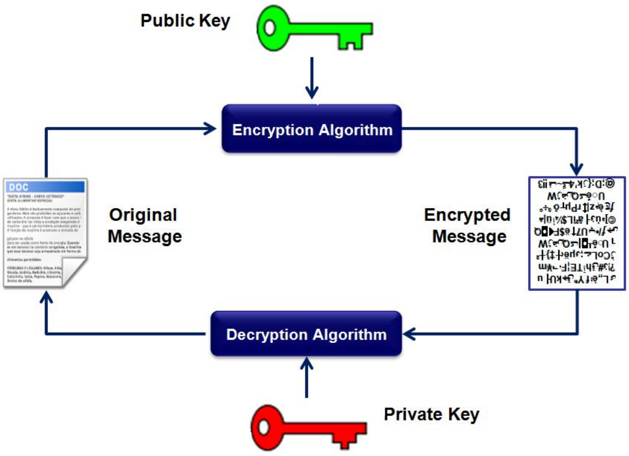


Figure 2-1: Encryption and decryption processes using asymmetric key primitives

In asymmetric key primitives, each party in the network should generate a pair of keys, the private key and the corresponding public key, to communicate with others; each party publishes his/her public key in the public register and keeps the private key secret and hidden from any other party [6-8]. Therefore, the security of asymmetric encryption is based on the algorithm itself and on the private and public keys. Examples of asymmetric encryption algorithms are RSA, Elgamal public key, and Rabin's public key [168-170].

2.5.2 Symmetric Key Primitive

Symmetric encryption, also called single-key encryption or conventional encryption, uses the same secret key to encrypt and decrypt the information (see Figure 2-2) [6]. Typically, the sender produces the ciphertext from the plaintext based on encryption algorithm and secret key; the receiver produces the plaintext from the ciphertext based on the decryption algorithm and the secret key [6, 7]. Therefore, the sender and the receiver should use the same encryption and decryption algorithm and share the same secret key before they start the communication. The security of symmetric key encryption is based on the algorithm itself and the shared secret key [7]. The main drawback of this primitive concerns how to agree on the shared secret key. Some well-known symmetric encryption algorithms are DES, IDEA, AES, and Blowfish [171-174]. The simple encryption and decryption equations are as follows:

$$E_k(M) = C \qquad (2\text{-}3)$$

$$D_K(C) = M \qquad (2\text{-}4)$$

where E is the encryption function, D is the decryption function, M is the plaintext, C is the ciphertext, and K is the secret key.

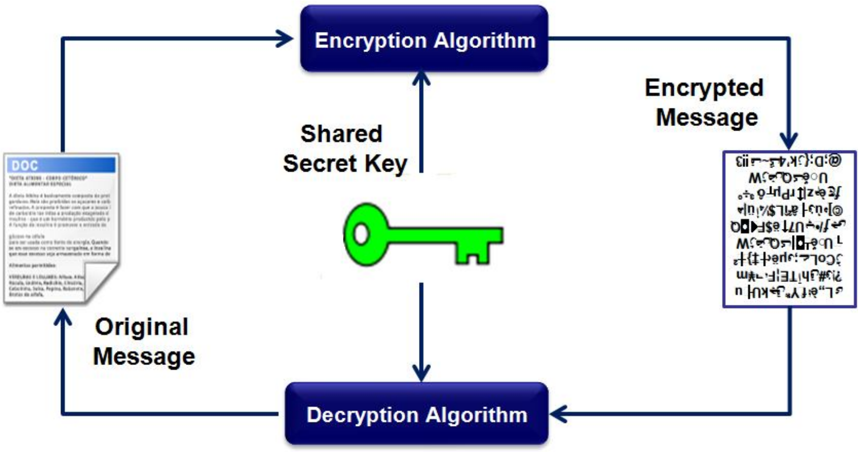


Figure 2-2: Encryption and decryption processes using symmetric key primitives

After the sender and the receiver agree on the secret key, the sender will prepare the message that he/she is willing to send and process it as an input to the encryption algorithm with the secret key to generate the ciphertext; then the ciphertext will be sent to the receiver side [7, 8]. On the receiver side, the receiver will process the ciphertext as input to the decryption algorithm with the secret key to generate the plaintext. If the decrypted message is understood by the receiver, the receiver will assume the received message is from the intended sender. Otherwise, the receiver will decline the received message, as shown in Figure 2-2 [11]. The symmetric key encryption can be further divided into block cipher and stream cipher.

Stream cipher is one of the encryption methods based on the secret key which encrypts a stream of data, such as one bit or one byte at a time [13]. It encrypts individual characters at the same time, while a block cipher encrypts groups of characters simultaneously. Ordinarily, block cipher is used to encrypt large blocks (e.g. $n \geq 64$), while stream cipher is used to encrypt small units [175]. Stream cipher is compulsory for certain applications that use small buffers and/or require no error propagations. In general, stream cipher generates pseudorandom bit sequences that can be XORed with plaintext to produce the correspondence ciphertext; the same sequence can be XORed with ciphertext to produce the corresponding plaintext. Stream cipher is much simpler than block cipher and faster in hardware implementation. A *block cipher* is an encryption method based on the secret key which encrypts one block (ex. 64-bit) of data at the same time; a block cipher processes blocks of n-bit (group of bits) plaintext to produce n-bit of ciphertext [6]. Thus, the block cipher encryption process should be reversible in order to decrypt the encrypted data.

2.5.3 Cryptography Hash Function

Cryptography hash function is mainly a mechanism to produce a hash value of a given message for cryptography applications such as message integrity, authentication, and other security services [6]. Hash function accepts variable input message lengths and produces fixed-length output as hash value, which is also called hash code or message digest [7, 8]. The hash value depends on all bits in the input message and any change or changes in the input message will affect and change the hash value. The cryptographic hash functions can be divided into two main parts: hash functions depending on secret key are called keyed hash functions, and hash functions not depending on secret key are called un-keyed hash functions [8, 11]. The general idea of hash function is to use an iteration procedure that processes whole messages to produce hash value (see Figure 2-3).

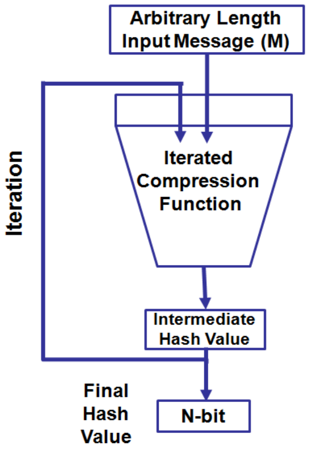


Figure 2-3: General model for an iterated hash function

Having explained the fundamental principles and concepts of cryptography, in the following sections we will focus on and explain in detail block cipher, cryptography hash function, and random number generators.

2.6 Block Cipher

Block cipher is an encryption method that divides input plaintext into blocks of equal lengths and then encrypts one block of data simultaneously under the control of the secret key [6, 176]. In another words, block cipher is a transformation function that maps units of plaintext bits to ciphertext bits of the same unit size under the control of the secret key. The decryption method divides the input ciphertext into blocks of equal length and then applies the decryption function to each block using the same shared secret key [10]. In general, the decryption process is the reverse of the encryption process. The lengths of the plaintext block and the corresponding ciphertext block are equal. Different modes of operations are applied if the size of input message is longer than the block size.

To evaluate block cipher security, we assume that the attacker can access all transmitted cipher and knows encryption algorithm details without knowing the shared secret key [10]. A block cipher is considered totally broken if the shared secret key is recovered, and it is considered partially broken if part of the plaintext is retrieved [177]. In a well-designed encryption algorithm, the plaintext and the ciphertext should be statistically independent to provide high security. The encryption algorithm aims to hide and complicate the relation between the ciphertext and the plaintext, and between the ciphertext and the secret key. Therefore, block cipher encryption algorithms should have very high confusion and diffusion properties [177]. The *confusion* is the process of hiding the relation between ciphertext and secret key, and the *diffusion* is the process of hiding the relation between plaintext and ciphertext [14].

Symmetric key encryption primitive is considered one of the most important parts of many cryptographic systems. In many cryptographic systems, block ciphers are considered the most popular and important encryption technique [8]. Some well-known block cipher encryption algorithms are DES, 3-DES, IDEA, FEAL, Blowfish, and AES [178]. The adaptability of block ciphers can be used to construct stream ciphers, MACs, hash function, and pseudorandom number generator [6]. Moreover, they can be used as core components in data integrity mechanisms, digital signature schemes and authentication protocols.

1.6.1 Block Cipher Evaluation

The following criteria are used to evaluate block ciphers [8, 10, 14]:

1- Key size: longer key size is more secure than shorter key size, but this will add to the cost of creation, storage, and communication transmission.

2- Block size: block size affects the security, complexity and performance levels. In general, longer block size will be more secure, but implementation cost will be higher.

3- Algorithm complexity: a more complex algorithm adds to implementation cost and reduces the algorithm’s performance.

4- Throughput: the algorithm complexity and the implementation flexibility directly affect the throughput.

5- Security level: in general, the confidence level of security increases the longer the algorithm is studied by cryptanalysis without any successful attacks.

6- Data expansion: the process of increase the size of encryption data by processing it through expansion function.

7- Error propagation: the error bits may affect the decryption result of current and succeeding blocks of data. Some error propagation characteristics can be tolerated in some applications.

2.6.1 Modes of Operation

The simplest block cipher approach partitions the input message into equal fixed-length blocks and then encrypts/decrypts each block individually. The process of partitioning the input message into equal blocks and then encrypting each block separately is called the electronic codebook (ECB) mode of operation. There are several different modes of operation such as electronic codebook (ECB), cipher-block chaining (CBC), cipher feedback (CFB), output feedback (OFB) and propagating cipher-block chaining (PCBC) [10, 179-181].

A. Electronic Codebook (ECB)

Electronic codebook is the most basic and simplest mode of operation that encrypts each block separately, without overlapping between blocks (see Figure 2-4) [182]. Thus, with a large input message size the patterns of input information/image will be shown [183]. Therefore, this mode of operation is not recommended for messages larger than block size. The main advantage of this mode is that the encryption/decryption function can be implemented in sequential mode only. ECB mode of operation has the following properties [10, 177, 179]:

1- Identical pair of plaintext and shared secret key produces identical ciphertext.

2- Each block is encrypted independently of other blocks.

3- Single or multiple errors in an encrypted block affect the deciphered result of that block only.

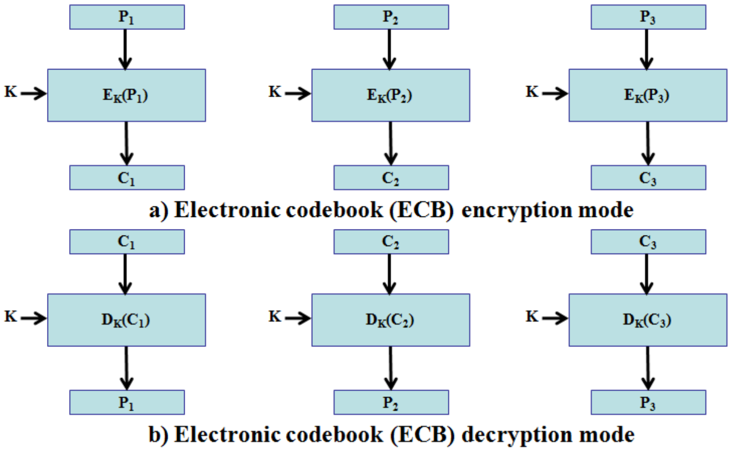
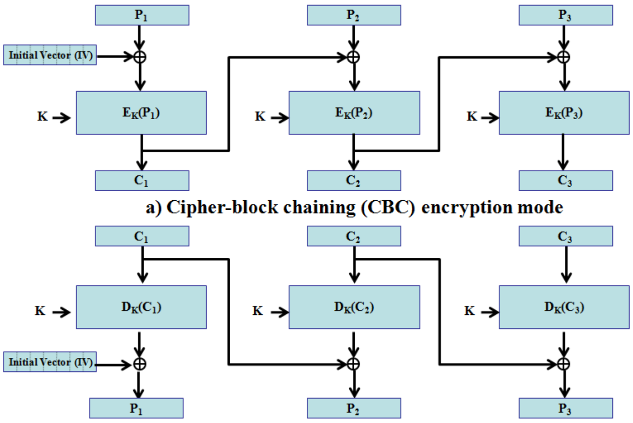

P1
P2
P3
K
EK(P1)
EK(P2)
EK(P3)
C1
C2
C3
a) Electronic codebook (ECB) encryption mode
C1
C2
C3
K
DK(C1)
DK(C2)
DK(C3)
P1
P2
P3
b) Electronic codebook (ECB) decryption mode

Figure 2-4: Electronic codebook (ECB) mode

B. Cipher-block Chaining (CBC)

In CBC mode, each plaintext block is XORed with a ciphertext block of a former block [182]. Thus, each block cipher depends on all previous blocks and every unique message has a unique corresponding ciphertext (see Figure 2-5). In this mode of operation, changing one bit or multiple bits in plaintext blocks will affect the results of the ciphertext of this block and the following blocks. Moreover, changing one bit in ciphertext affects the current block and the following block. This mode of operation is the most widely used. The main disadvantage of this mode is that the encryption function can be implemented in sequential mode only, but the decryption function can be implemented in parallel. CBC mode of operation has the following properties [10, 179]:

1- Identical plaintext with pair of key and IV producing identical ciphertext.
2- Each block is encrypted depending on results of previous encrypted blocks. Therefore, rearranging the cipher blocks’ positions will affect the decryption result.
3- Single error in encrypted block affects the following encrypted results and causes corruption in the complete plaintext.



a) Cipher-block chaining (CBC) encryption mode

b) Cipher-block chaining (CBC) decryption mode

Figure 2-5: Cipher-block chaining (CBC) mode

C. Cipher Feedback (CFB)

Cipher feedback mode was designed to implement stream cipher by block cipher [184]. In this mode, the initialization vector (IV) and secret key are processed through encryption function to produce keystream bits that are XORed with plaintext blocks (see Figure 2-6) [182]. Decryption operations of cipher feedback mode are very similar to reverse encryption operations of CBC mode. The ciphertext of the previous block is used as input for the encryption algorithm of the current block. In this mode, the encryption can be implemented in sequential mode only, but the decryption can be implemented in parallel [175]. Changing one bit in a ciphertext block of CFB mode of operation will affect two blocks only. Moreover, the effect of changing one bit in plaintext will propagate to all the following ciphertext blocks.

CFB mode of operation has the following properties [10, 179, 184]:

1- Changing the value of IV to a different value using the same plaintext input results in enciphering the input plaintext to a different ciphertext.
2- Changing one bit in plaintext will affect all the following ciphertext blocks. This means that the ciphertext depends on all the previous plaintext.
3- Single error in ciphertext block affects the results of two blocks (current and following).

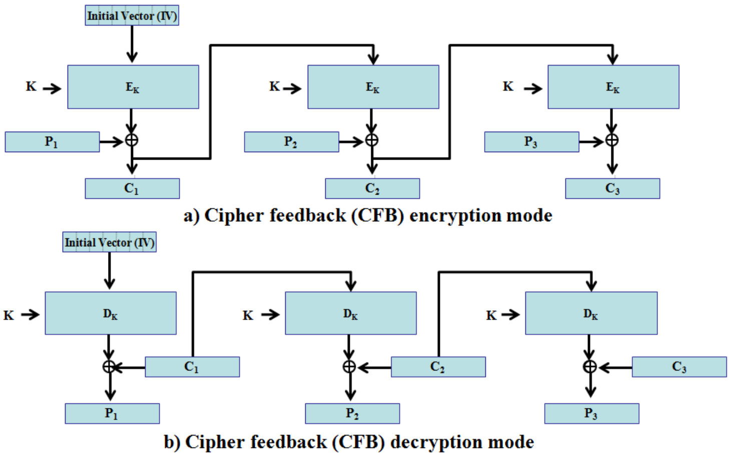
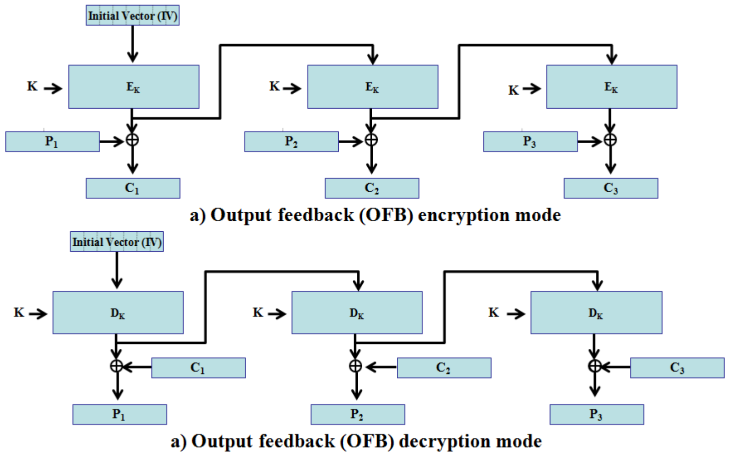

Initial Vector (IV)
K
E_K
P_1
C_1
P_2
C_2
P_3
C_3
a) Cipher feedback (CFB) encryption mode
D_K
b) Cipher feedback (CFB) decryption mode

Figure 2-6: Cipher feedback (CFB) mode

D. Output Feedback (OFB)

The output feedback (OFB) mode of operation is illustrated in Figure 2-7. In this mode, the initialization vector (IV) and the secret key are processed through the encryption function to produce keystream bits that are XORed with the plaintext blocks to obtain the ciphertext. The encryption and the decryption operations are exactly the same. In the encryption function the generated keystream is XORed with plaintext to obtain the ciphertext, and in the decryption function the keystream is XORed with the ciphertext to obtain the plaintext. The main difference between CFB and OFB modes is that, in OFB mode, the output of encryption function is used as feedback [182]. This mode is used in many applications that need to avoid error propagations. The main disadvantage of OFB mode is that it can be implemented in sequential mode only. OFB mode of operation has the following properties [10, 179]:

1- Identical plaintext with pair of key and IV producing identical ciphertext.
2- The keystream is not affected by the plaintext, where the output of the previous keystream is the input of current keystream before being XORed with plaintext.
3- Changing one bit or multiple bits in ciphertext affects only one plaintext block, as the keystream is independent of plaintext and ciphertext.



a) Output feedback (OFB) encryption mode

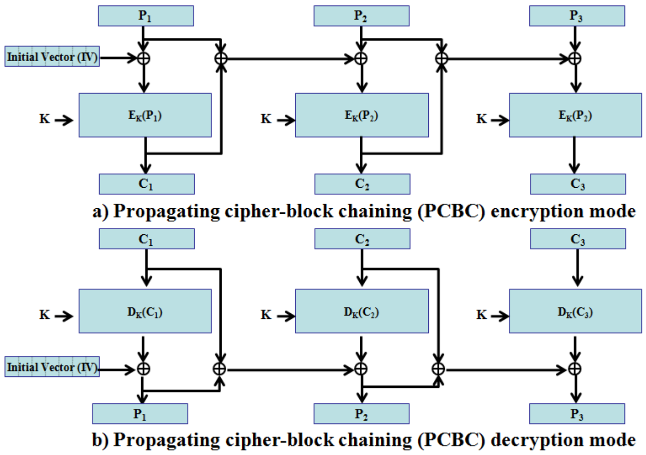
a) Output feedback (OFB) decryption mode

Figure 2-7: Output feedback (OFB) mode

E. Propagating Cipher-block Chaining (PCBC)

In PCBC mode, the first plaintext block is XORed with the initialization vector and is then processed through the encryption function to obtain the corresponding ciphertext block [181]. In the second block, the plaintext and the ciphertext blocks are XORed together and the result is then XORed with plaintext of the current block (see Figure 2-8) [185]. In this mode of operation, changing one bit or multiple bits in plaintext blocks will affect the results of the ciphertext of this block and all the following blocks. Moreover, changing one bit in ciphertext affects the current block and the following block. This mode was designed to infinitely propagate the effect of changing one bit or multiple bits in the encryption/decryption process. Therefore, encryption and decryption of this mode can be implemented in sequential mode only. PCBC mode of operation has the following properties [10, 179]:

1- Identical plaintext with pair of key and IV producing identical ciphertext.
2- Each block is encrypted depending on plaintext and ciphertext of the previous block. Therefore, rearranging the cipher blocks' positions will affect the encryption/decryption result.
3- Single errors in encrypted/decryption block affect the following encrypted/decryption results and cause corruption to the complete result.



a) Propagating cipher-block chaining (PCBC) encryption mode

b) Propagating cipher-block chaining (PCBC) decryption mode

Figure 2-8: Propagating cipher-block chaining (PCBC) mode

2.6.2 Encryption Using Multiple Modes of Operation

Multiple modes of operation are constructed by combining more than one single mode of operation together. For example, concatenation of two single ECB modes provides double-ECB mode of operation, and concatenation of three single ECB modes provides triple-ECB mode (see equations 2-5 and 2-6). The double and triple modes of operation can be implemented in parallel mode to provide better encryption performance [179]. Using multiple modes of operation does not always provide higher security than single mode of operation. For instance, under some types of attacks multiple ECB modes are not stronger than a single mode of operation [10].

$$E(x) = E_{K2}(E_{k1}(x)) \qquad (2\text{-}5)$$

$$E(x) = E_{k3}(E_{K2}(E_{k1}(x))) \qquad (2\text{-}6)$$

2.6.3 Block Cipher Encryption Algorithms

In the following subsections, we will give brief details of three well-known block cipher encryption algorithms (DES, IDEA, and AES) [171-173].

A. Data Encryption Standard (DES)

Data encryption standard is a block cipher algorithm based on the symmetric key principle that encrypts 64-bit amounts of plaintext and generates ciphertext of the same size using secret key of size 56-bit [171]. DES was developed by IBM and approved by the National Bureau of Standards [159]. It was designed based on product cipher and Feistel cipher. *Product cipher* performs various simple operations to build a complex encryption function. The internal function of block cipher is called a round function that is iterated a number of times sequentially. *Feistel cipher* is an iterated block cipher that maps n-bits of plaintext into n-bits of ciphertext by dividing the plaintext into two equal halves (L_i, R_i) and then processing it through t-round using several subkeys (K_1, K_2, ..., K_n) (see Figure 2-9) [10]. In encryption function, the plaintext is processed through 16 rounds and each round has a subkey of size 48-bit that is generated from the secret key of size 56-bit. The input plaintext is divided into two equal halves each of size 32-bit (L_i, R_i) (see equations 8 and 9). The decryption process is performed using the same number of rounds with reverse order of the keys (K_n, K_{n-1}, ..., K_1).

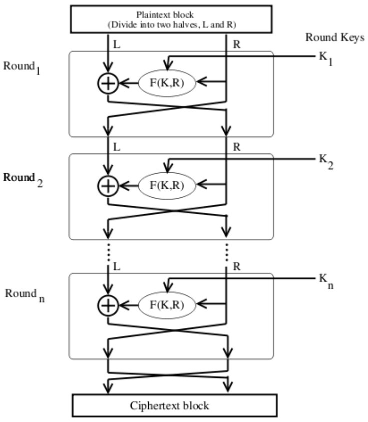


Figure 2-9: Overall DES feistel structure

$$L_i = R_{i-1} \tag{2-5}$$

$$R_i = L_{i-1} \oplus P(S(E(R_{i-1}) \oplus K_i)) \tag{2-6}$$

where E is the expansion permutation map R_{i-1} from 32 to 48 bit and P is permutation on 32-bit.

The left and right halves (L_{15} and R_{15}) in the last round change positions and the final output is the inverse permutation of the last round result. The decryption process applies the same function and rounds, but applies the subkey in reverse order (K_n, K_{n-1}, ..., K_1). In secure block cipher algorithm, changing one bit in the plaintext or the key should change each bit in the ciphertext with a probability of 0.5, and changing one bit in the ciphertext should result in unpredictable changes in the plaintext [8]. Thus, secure block cipher encryption algorithms should have high confusion and diffusion properties (see section 2.10).

Several analyses and attacks have been mounted on DES. The first demonstrated successful differential cryptanalysis on a reduced version of DES was described in 1991 with eight rounds [159]. Later on, three other types of attacks on a reduced version of DES were proposed [14, 160, 161]. Moreover, there have been many proposals for designing DES-machine secret key attack to search all possible keys within a short period of time [186-188]. Therefore, an alternative encryption algorithm was needed by the end of the1990s. In 2001, the Rijndael block cipher algorithm (AES) was approved as the Federal Information Processing Standards (FIPS) symmetric key block cipher algorithm [173]. One of the other suggested alternatives was to apply DES encryption three times for each block (triple-DES), but this would make the encryption and decryption processes slow.

B. International Data Encryption Algorithm (IDEA)

International data encryption is a block cipher algorithm that was developed as a new block cipher encryption algorithm to overcome DES problems [183, 189]. IDEA converts a plaintext block of 64-bit to a block of ciphertext of the same size using a secret key of size 128-bit (see Figure 2-10) [172]. It is designed based on the Feistel structure with eight identical rounds, each round using six subkeys of 16-bit size. The last round of output is an input of the transformation function. Transformation function uses 4 subkeys that are used to produce the final cipher. All subkeys are derived from the main secret key K. The main three mathematical operations are bitwise XOR, addition mod 2^n, and multiplication mod 2^{n+1}. The decryption algorithm is the same encryption algorithm using the same secret key K to derive the decryption subkeys with a few changes. IDEA was designed to be immune to differential cryptanalysis under certain assumptions and other types of attacks [8]. Since 1991, it has no longer been recommended for many applications because of the availability of other, faster encryption algorithms.

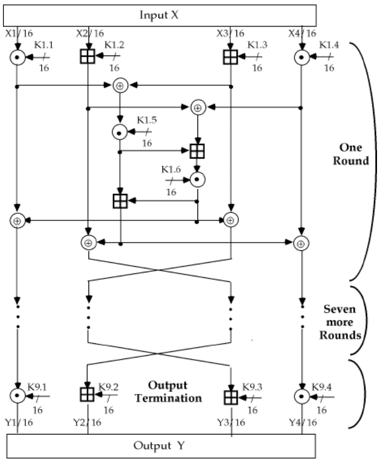


Figure 2-10: Overall IDEA structure

C. Advanced Encryption Standard (AES)

In 1997, after DES was proved to be an insecure encryption algorithm, NIST announced the need to design a new alternative cipher encryption algorithm. Fifteen candidate algorithms were accepted and, later on, in October 2000, the Rijndael algorithm was selected [183]. The Rijndael algorithm is a block cipher algorithm based on symmetric key primitive. It has three versions, AES-128, AES-192 and AES-256, of 128-bit block size and key sizes of 128, 192 and 256 bits respectively. Since October 2000, it has been used globally in many applications and many cryptanalysis studies have focused on its security [8]. This algorithm was designed based on iterative rounds procedure and, in each round, several operations are performed on the entire block of data [10]. It has a very simple design with high speed and resistance to known types of attacks. The reverse round operations are applied using the same encryption key to transfer the ciphertext into the plaintext.

A high-level description of AES-128 encryption process is illustrated in Figure 2-11. The first step is to calculate the rounds' subkeys from the shared secret key using the Rijndael key schedule and to load state array values; then each round subkey is

XORed with byte of state [14]. Subsequently, four transformation functions (SubBytes, ShiftRows, MixColumns, and AddRoundKeys) are performed in each round. The number of rounds depends on the AES version: AES-128 has 10 rounds, AES-192 has 12 rounds and AES-256 has 14 rounds [8]. In the SubBytes step each value of state array is updated using Rijndael S-box. The values of state array are updated in the ShiftRows step by shifting to the left each row value by different amounts based on the number of the row. In the MixColumns step, four vertical values of state array are multiplied by fixed polynomial. ShiftRows and MixColumns steps are intended to provide high diffusion to the ciphertext. The subkeys are mixed with the state array in the AddRoundKeys step. In the last round, all steps are performed except the MixColumns step; then the values of the state array are unloaded as ciphertext. In 1998, a reduced version of Rijndael was attacked with 6 rounds [162]. Two years later, another attack with 7 rounds was described [163]. In 2000, an improvement in the cryptanalysis of the Rijndael algorithm with 8-rounds of AES-198 and 9-rounds of AES-256 was described in detail [164].

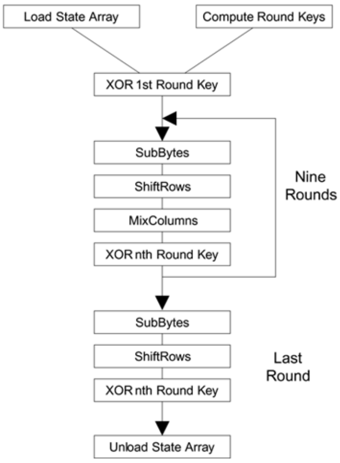


Figure 2-11: High-level description of AES-128 encryption process

2.7 Hash Functions

In modern cryptography, hash functions are playing a primary role due to their efficiency and performance [10]. A well-designed hash function with slight differences in the input messages will produce different hash values. Thus, any change or changes in the input message will produce different hash values. The cryptography hash value is analogous to the fingerprint where we can authenticate a person's identity by verifying fingerprint, Id, name, height, weight, age, facial characteristics and so on [6, 7]. Like the human fingerprint, which is a unique and small piece of data which authenticates the human identity, the hash value is used to authenticate the whole message [7]. Cryptography hash functions are different from the conventional hash functions (henceforth, the cryptography hash function will simply be called the hash function). Hash functions are used to produce a fixed-length code from any variable-length message; this is known as a hash value where a large domain is mapped onto a small range [8]. Hash value is known as message digest, hash code, digital fingerprint and cryptography checksum [7, 8, 11]. The cryptographic hash functions can be divided into keyed hash functions and un-keyed hash functions.

Hash functions are one-way functions used to produce fixed-length hash values from variable-length messages, where it is very hard to find the original message from a given hash value [11]. One-way function is a function *f* from a set X to a set Y that is easy to calculate $f(x)$ for all $x \in X$ but it is computationally infeasible for all y in the range *f* to find any $x \in X$ such that $f(x)=y$. There are two main types of one-way function: weak and strong one-way functions [8]. The weak one-way function is always easy to compute, but it is sometimes hard to find the inverse. On the other hand, it is always impossible or virtually impossible to find the inverse of a strong one-way function. One-way functions play significant roles in cryptography where they are fundamental to most cryptography algorithms, such as public key and hash function [7-10]. In cryptography, hash functions are one-way functions; thus we are unable to derive the original message for a given hash value [8].

2.7.1 Un-keyed Hash Function

As we mentioned before, there are two types of cryptography encryption: public key encryption and secret key encryption. The public key encryption can be up to 1,000 times slower than the secret key encryption and that is not ideal for e-commerce applications [7]. On the other hand, the secret key is very difficult to manage when one is communicating with many people, and one should share different secret keys with each person each time [8]. Therefore, cryptographers have introduced a new cryptography primitive called the un-keyed hash function to make private key signing much faster and more efficient. In general, the hash value is much smaller than the message and needs a much shorter time to be encrypted [6].

Un-keyed hash functions are cryptographic checksums used to provide message integrity and other security services, such as authentication using digital signature. The two most popular hash functions are Message Digest-5 (MD5) and Secure Hash Algorithm–1 (SHA-1) [7]. In general, the hash function accepts a single input message *M* of variable length to produce a fixed-length hash value *H(M)* of *X* bits. Hash function is a one-way encryption function in the sense that the hash value can be derived from the input message, but the original message can't be derived from the given hash value. Figure 2-12 shows that the one-way hash function can be used to provide message integrity and digital signature [6, 10].

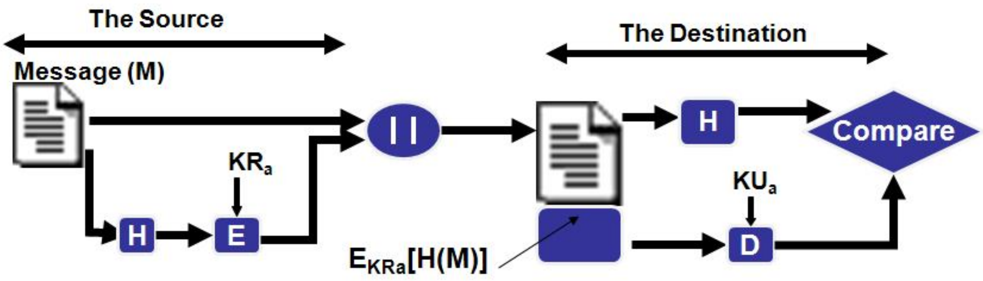


Figure 2-12: Hash function and digital signature

According to Menezes et al. (1997), un-keyed hash functions can be divided into three broad categories based on the internal structure. The first type is hash functions based on block cipher which are reusing an existing system, such as MDC-2 (with DES) and MDC-4 (with DES). The second type is customized hash functions based on MD4, such as MD5, Secure Hash Algorithm (SHA-1), RIPEMD-128 and RIPEMD-160. The third type is hash functions based on modular arithmetic, such as MASH-1 and MASH-2 [10]. Henceforth, un-keyed hash function will simply be called hash function.

2.7.2 Keyed Hash Function: Message Authentication Code (MACs)

Keyed Hash Function (Message Authentication Code) is a combination of two inputs, the message and secret key, to produce a small fixed-length value known as checksum or $MAC = C_K(M)$ that serves as authenticator [7]. This method assumes that the sender and the receiver share the secret key K to calculate the correct MAC value [7, 8]. The sender calculates the MAC value based on the message and secret key, and then sends the message appended with encrypted MAC to the receiver. After the receiver receives the message combined with encrypted MAC value, he/she will compute the MAC value based on the received message and shared secret key K. Finally, the receiver will compare the calculated result with the decrypted MAC value, as shown in Figure 2-13 [8, 11]. If the received MAC(K, M) value and the computed MAC value are equal the receiver can be assured that the received message has not been altered and that it is definitely from the supposed sender. The MAC does not provide the digital signature because both the communication parties are sharing the secret key [11]. To conclude, the security of keyed hash functions depends on the secret key, while the security of the un-keyed hash functions depends on the internal structure.

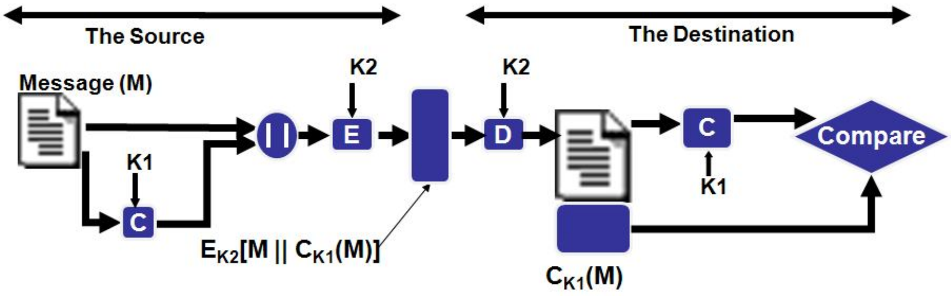


Figure 2-13: Message Authentication Code (MAC)

2.7.3 Hash Function Properties

A *hash function H* is a function that accepts an input message *M* of arbitrary bits length and produces a hash value *h*, such as *h=H(M)*. A secure cryptography hash function must satisfy the following properties [6, 8, 10]:

1. *H* (Hash Function) can apply to any block size of data.

2. *H* (Hash Function) should produce a fixed length of output data (ex.160 bits).

3. *H(M)* (Hash Value) is quite easy to calculate for any given message *M* of any size; both hardware and software implementations are applied.

4. One-Way (2^n) property, aka *Pre-image Resistance*, means it is computationally infeasible to find the message *M* for any given hash value *h.* In other words, it is easy to produce the hash value *h* of any given message *(M)*, but it is almost impossible to produce a message from a given code.

5. Weak Collision Resistance (2^n), aka *Second Pre-image Resistance* means that, given block M_1 , it is computationally infeasible to find M_2 where $M_1 \neq M_2$ with $H(M_1) = H(M_2)$.

6. Strong Collision Resistance ($2^{n/2}$), aka *Collision Resistance* means it is computationally infeasible to find any pair of blocks M_1, M_2 where $H\ (M_1) = H\ (M_2)$.

Collision resistance means it is computationally infeasible to find two different input messages that are hashed to the same hash value [190]. The weak collision resistance guarantees that the opponents will not be able to find another message hashing to the same hash value of the given message. This property prevents message falsification, as the calculated hash code is encrypted (see Figure 2-14). In general, the sender will send the message concatenated with its encrypted hash value to the sender's private key. Therefore, an attacker can read and modify the message, but can't read or modify the hash value. The attacker could calculate the hash value of the sent message, whereas the sent message and the hash function could be known. Moreover, if the private key is known, the attackers are able to modify the original message and calculate the new hash value to be encrypted with the private key, and then will send the modified message with its hash value to the receiver [6].

The strong collision resistance differs from the weak collision resistance. In strong collision resistance, the attacker would be able to choose both x and y, not just y, and could compute the $h_x = H(x)$ and $h_y = H(y)$. If h_x is equal to h_y, the attacker can send to one side *x* and to the other side *y*; when the other side checks the *H(y)* they will find it equal to h_x. So, the receiver will think the message is authenticated, but in reality the received message is different from the original one, as shown in Figure 2-15 [6].

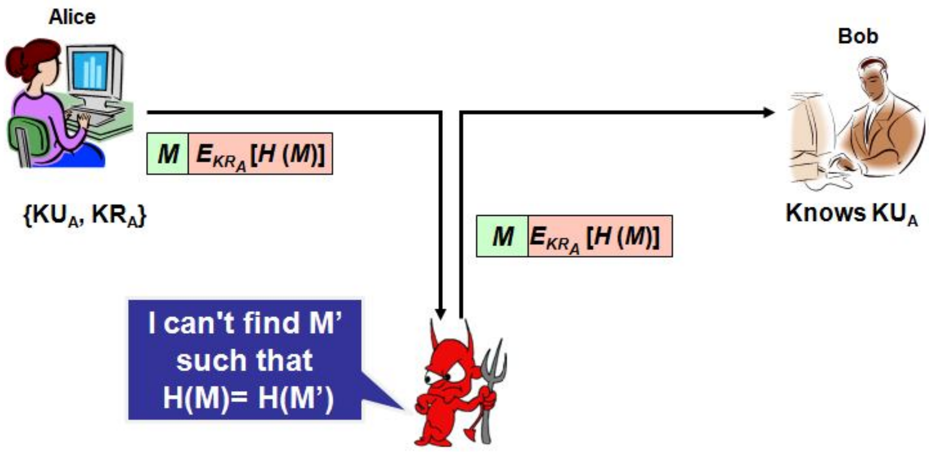


Figure 2-14: An example of a weak collision resistance.

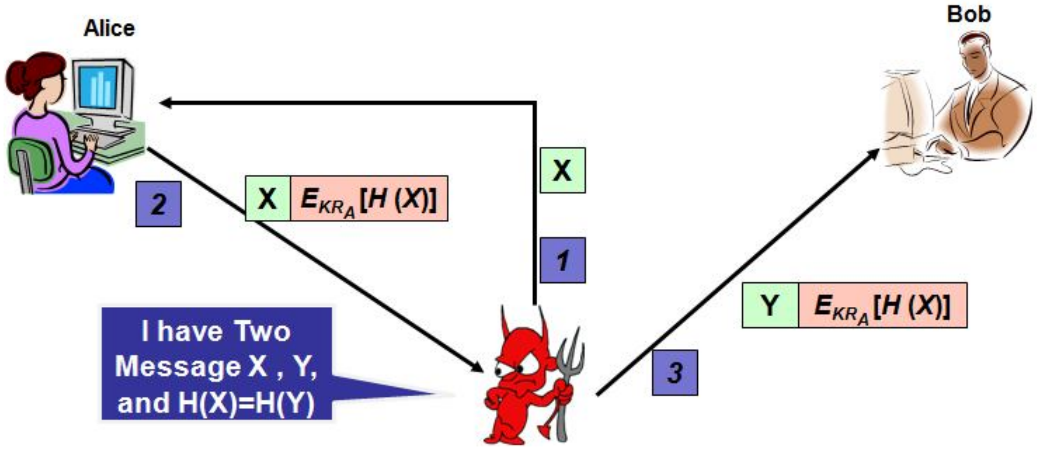


Figure 2-15: An example of a Strong collision resistance.

2.7.4 Hash Function Applications

Hash functions are used to produce “Fingerprints” of the message, which can be useful for message authentication [6]. The hash value should be protected before it is sent to another party where the hash functions are not considered to be secret. There are various ways of using hash value to provide message authentication. Digital signature is an important type of authentication, which can be achieved by computing the hash value of the message on the sender’s side and then encrypting it by the sender’s private key to provide digital signatures. Then, the sender will append the digital signature to the message and send it to the receiver. The receiver will separate the digital signature from the message and decrypt it with the sender’s public key. After that, he/she will re-compute the hash value of the received message and check whether the two are matched; then he/she can verify whether the message is authenticated or not [6].

Hash function plays an elementary role in modern cryptography [10]. It is employed in many different cryptography protocols and security issues such as encryption, data integrity and simple digital signature schemes. A typical use of hash function is data integrity to detect message modification, thus providing error detection capability. In a well-designed hash function, any bit or bits change in the input message will produce a totally different hash value (final output) [6]. Hash functions can be used to generate a pseudorandom number generator that is used to generate shared secret keys [11]. Moreover, the security of digital signatures mainly depends on the security of the hash functions [158]. Hash functions are widely used in many different standards and applications such as message integrity, password verification, message authentication, SSL/TLS, IPSec and S/MIME [191]. Therefore, the Internet mainly depends on hash functions to provide web security, key management, password login, e-payment and many other cryptography protocols. Simply put, without hash functions the Internet will not work.

2.8 Brief History of Hash Functions

In 1990 [192], Ralph Merkle designed a one-way hash function called Snefru, which accepts an arbitrary message length and produces 128-bit or 256-bit hash values. In 1993 [193], Biham and Shamir attacked the two-pass of Snefru-128 using differential cryptanalysis by finding a collision of two messages within minutes. They can attack two, three and four passes with less than the birthday attack. In addition, the same people were able to attack Snefru-256 two, three and four passes with less than the birthday attack [8]. Later, Merkle recommended using Snefru with at least 8 passes to provide higher security, but that would be slower than other hash functions.

Modifications Detection Codes (MDC-2 and MDC-4) are hash function algorithms developed by IBM which allow the use of any n-bit block cipher *E* and provide hash values of *2n* bits length [10]. Both MDC-2 and MDC-4 were originally designed to use DES as the block cipher, but they can be used with other block ciphers. MDC-2 required 2-block cipher operations per block of hash input, while MDC-4 required a 4-block cipher. *Modular Arithmetic Secure Hash Algorithm-1* (MASH-1) is a hash function algorithm based on modular arithmetic [167]. NASH-1 compression function is based on a modular squaring operation. MASH-1 uses the modulus M, where M is long enough for it to be very difficult or infeasible for anyone to factorize it [10]. MASH-2 is the same as MASH-1, with a slight difference in the modular exponent, e = 2^8 +1 instead of e = 2; the security against collision attack of MASH-1 is 2^n and for MASH-2 it is $2^{n/2}$ [167]. In general, the hash functions that are based on modular arithmetic are slower than the other hash functions.

RIPEMD-160 is another hash function algorithm which accepts arbitrary message length and produces 160-bit hash value [190]. It is a more developed and strengthened version of RIPEMD-128 after the first two rounds of RIPEMD-128 were attacked by H. Dobbertin [6, 194]. The general design and structure of RIPEMD-160 follow the design and structure of MD5 in a different way. Therefore, RIPEMD-160 has 10 rounds and each round consists of 16 steps. RIPEMD-160 is more secure than RIPEMD-128, but its performance is reduced by a factor of two [190].

Message Digest-4 (MD4) is a hash function algorithm designed by Ron Rivest in 1990 and published as an RFC 1321. In 1992, a revised version of MD4 was published in RFC 1320 with slight modification of the original version [6]. MD4 has three rounds and each round has 16 steps; it is based on a simple operation performed on 32-bit words. In 1996 [195], H. Dobbertin found a collision attack on MD4 with a complexity of 2^{20} hash operations. In 2005, Wang et al. found a collision attack on MD4 with a complexity of less than 2^8 hash operations [196]. In the following two subsections, we will describe in detail two popular hash functions: Message Digest-5 (MD5) and Secure Hash Algorithm-1 (SHA-1).

2.8.1 Message Digest-5 (MD5)

Message Digest-5 (MD5) was developed by Ron Rivest, the ‘R’ in RSA [Rivest-Shamir-Adleman] [168], which is a further development and improvement of MD4 with higher complexity and slower speed than MD4 [11, 197, 198]. RSA is a public cryptosystem that very slowly encrypts a large message instead of encrypting its hash value. MD5 accepts an arbitrary-length message as an input and produces 128-bit hash value [198]. After development of the MD5 algorithm, it was widely used and known as a secure hash algorithm [6]. The algorithm divides the input message into 512-bit blocks after which the last block is padded by (100...0) to be of size 448-bit; the last 64-bit is padded by the message size mod 2^{64} [11]. MD5 has a compression function consisting of four rounds, and each round has 16 operations to mix the state of the message with the message block (see Figure 2-16). MD5 uses 32-bit words and each round has non-linear mixing functions: XOR, AND, OR, Addition mod 32, and Rotation operation on 32-bit words [11].

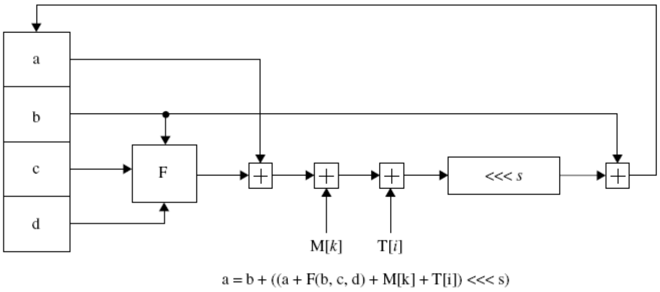


Figure 2-16: Basic Operations of One MD5 round [199]

There have been several attacks on MD4 and MD5 [158, 195, 196, 200, 201]. In 1993 [202], B. Boer and A. Bosselaers found two collision attacks for MD5 with two different Initial Values (IV) of the same message. Later, in 1996 [195], H. Dobbertin found a collision attack of MD5 compression function. Then, in 1998 [201], H. Dobbertin found that the first two rounds of MD5 are not one-way. In 2005, Wang et al. found a differential attack on MD5 with 15-60 minutes of computations [158]. MD4 and MD5 were two very popular hash functions, but their popularity declined as a result of the development of computing power and cryptanalysis techniques [6]. Therefore, in 1993 the National Institute of Standards and Technology (NIST) designed Secure Hash Algorithms (SHA) with bigger hash values to be used as standard hash function. In the following subsection, we will explain in detail Secure Hash Algorithms (SHA).

2.8.2 Secure Hash Algorithm-1 (SHA-1)

NIST designed the Secure Hash Algorithms (SHA) for use with Digital Signature Standard [8]. SHA-0 was the first member of the SHA family to be developed by the NIST and was the original algorithm to be published as a Federal Information Processing Standard (FIPS 180) in 1993 [6]. The National Security Agency (NSA) withdrew it shortly after it was published. In 1995, FIPS published SHA-1 algorithm as a replacement, having corrected the flaw in the original algorithm [203]. SHA-1 was recommended by governments and cryptographers as it provides greater resistance to attacks than other algorithms [6]. SHA-1 has been adopted by many governments and industry security standards.

Table 2-1: Secure Hash Algorithm Properties

	Algorithm	Message Size (bits)	Block Size (bits)	Word Size (bits)	Message Digest Size	Security (Operations)
	SHA-1	2^{64}	512	32	160	2^{80}
SHA-2 Family	**SHA-224**	2^{64}	512	32	224	2^{114}
	SHA-256	2^{64}	512	32	256	2^{128}
	SHA-384	2^{128}	1024	64	384	2^{192}
	SHA-512	2^{512}	1024	64	512	2^{256}

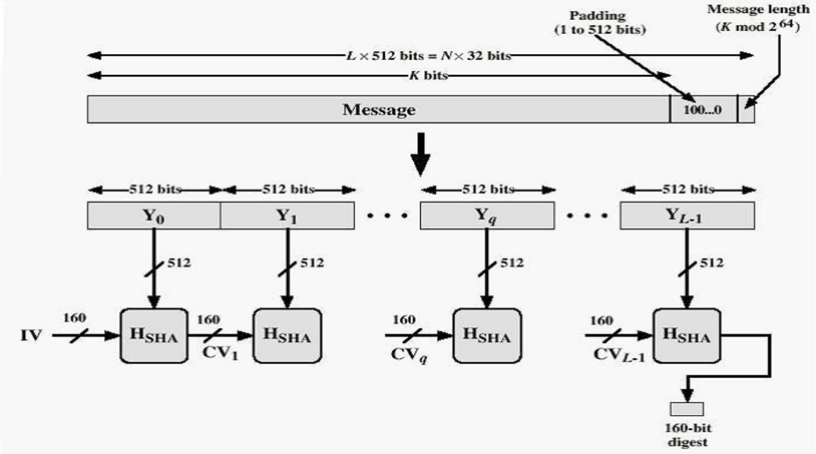


Figure 2-17: Message Digest Generation Using SHA-1 [6]

SHA-1 is a one-way hash function that processes an input message of length less than 2^{64} bits to generate a hash value of 160-bit length. It is used to provide message integrity as it is very sensitive to any change/changes in input message. This property is useful for digital signature, message authentication codes, and generating random bits. Four more SHA variants have been published with longer hash values and a slightly different design: SHA-224, SHA-256, SHA-384, and SHA-512, sometimes collectively referred to as SHA-2 (see Table 2-1) [7, 204].

SHA-1 algorithm processing steps are as follows [6, 8]:

1. Append padding bits.

There are many padding modes are employed in cryptography algorithms such as Zero padding, Bit padding and Byte padding modes. Zero padding mode pads the message with the required number of zeros. The Bit padding mode pads the message by one $(1)_2$ followed by number of zeros $(00\ldots0)_2$ and the last 64-bit with the length of the message mod by 2^{64}. In Byte padding mode the message is padded by zeros and last bit with number of padded bytes, with random bytes, or with number of padded bytes. Now we will explain the Bit padding mode that is used in many cryptography algorithms including SHA-1 and MD5.

This process pads the input message (*M*) before hash computations begin. The message is padded to ensure that it will be a multiple of 512 or 1024 based on the algorithm [6]. Suppose the length of the input message M is *L* of length less than 2^{64} bits; the first step is to append a single 1-bit to the end of the original message followed by k zeros bits where k+1+L = (448 mod 512), then k = (448 mod 512) – (L+1) (see Figure 2-17) [8].

2. Append length.

In this step, a 64-bit block is appended to the end of the message. This block is treated as an unsigned 64-bit integer, which contains the length of the message before the padding.

3. Initialize MD buffer.

In SHA-1, the five registers (A, B, C, D, and E) are initialized with the 32-bit hexadecimal values 0x67452301, 0xEFCDAB89, 0x98BADCFE, 0x10325476, and 0xC3D2E1F0, respectively.

4. Process the message in 512-bit blocks.

The main part of the SHA-1 algorithm is the module which consists of four rounds, each round consisting of 20 steps and one different function of (f_1, f_2, f_3, f_4). These functions are non-linear functions on three buffers (B, C, and D) [8]. In each round, the algorithm processes a 512-bit as one block from the existing message input that is being processed (Y_q), and continues according to how many 512-bits there are in the input message. Each round takes input of 160-bit buffer values (A, B, C, D, and E) and updates the contents of the buffers [6].

5. Output

The final output is 160-bit hash value (CV_{q+1}) that comes from the processing of all 512-bit blocks [6].

Now we will give details of the SHA-1 compression function. We will look at the function logic in each of the 80 steps, each of which operates on three 32-bit words (B, C and D) and produces a 32-bit word as output, F (t, B, C, D). We can summarize this as follows [6, 8]:

$$F(t; B, C, D) = (B \wedge C) \vee ((\neg B) \wedge D) \qquad (0 \leq t \leq 19)$$

$$F(t; B, C, D) = B \oplus C \oplus D \qquad (20 \leq t \leq 39)$$

$$F(t; B, C, D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) \qquad (40 \leq t \leq 59)$$

$$F(t; B, C, D) = B \oplus C \oplus D \qquad (60 \leq t \leq 79)$$

A sequence of constant words K(0), K(1)...K(79) are used in the SHA-1. In hexadecimal these are given as follows [8]:

K (t) = 0x5A827999 $(0 \leq t \leq 19)$

K (t) = 0x6ED9EBA1 $(20 \leq t \leq 39)$

K (t) = 0x8F1BBCDC $(40 \leq t \leq 59)$

K (t) = 0xCA62C1D6 $(60 \leq t \leq 79)$

The 16 32-bit word values are transformed into 80 32-bit words(W_0 to W_{79}) using the following algorithm $W_t = (W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3})$ [8]. A, B, C, D and E are 32-bit words of the state, *F* is a non-linear function that varies, $<<<_s$ denotes a left bit rotation by *s* places, s varies for each operation, (∃) is addition modulo 2^{32}, and K_t is a constant (as shown in Figure 2-18).

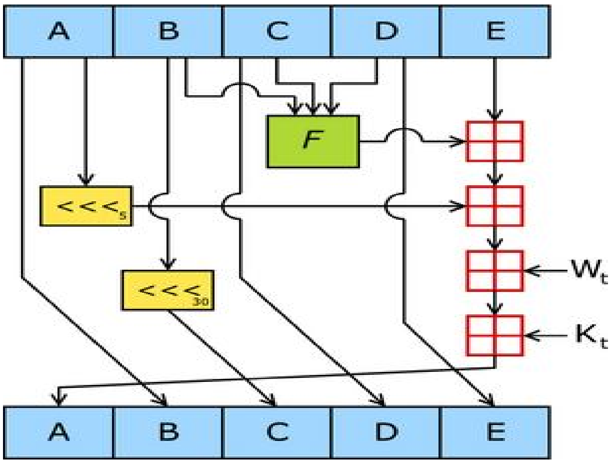


Figure 2-18: SHA Operation of Single Step [205]

- **SHA-1 Attack**

Recently, significant advanced studies and analyses of hash functions have developed a set of very effective new techniques for searching for collisions in SHA-1 [165]. In 2005, Rijmen and Oswald wrote a paper to describe an attack on a reduced version of SHA-1; in 53 out of 80 rounds they found a collision attack in less than 2^{80} hash operations [206]. Wang et al. (February, 2005) found collisions in the full version of SHA-1 with less than 2^{69} hash operations [165]. In August 2005, another research group of cryptanalysts found an improved attack on SHA-1 with complexity of 2^{63} hash operations [207]. This attack on SHA-1 had a complexity of less than the 2^{80} theoretical bound. Researchers use the term ‘broken’ whenever they can demonstrate the vulnerability rather than using a simple brute-force attack.

According to Wang et al. (2005), the analysis of SHA-1 was based on the original differential attack on SHA-0. The SHA-0 attack was based on an algebraic method, whereby the researchers were able to find a real collision with less than 2^{39} hash operation [208]. A simple and important observation made about SHA-0 was that it has a 6-step local collision that can start at any step and that also applies to SHA-1 [165]. In addition, the group used the message modification techniques of collision search attacks on HAVAL-128, RIPEMD, MD4 and MD5 [156, 158, 196, 208].

Since August 2005, researchers have been looking for a new hash function that is more secure than SHA-1 and that would represent a good alternative. There have been many different suggestions from many researchers on finding a new hash function to replace SHA-1. Some researchers have pointed out that a new hash function based on chaos theory would be a good candidate to replace SHA-1. In January 2007, the NIST announced a competition to create and design a new hash function to replace the current hash function SHA-1 in the year 2012 [209, 210].

2.9 Random Number Generators

There are two main types of random number generators: random number generator and pseudorandom number generator. *Random number generator* is an algorithm or device that generates a sequence of binary bits that are statistically independent [10]. A *pseudorandom number generator* (PRNG) is a deterministic procedure that produces a binary bits sequence that is approximately random [6, 211]. The input parameter of PRNG is called seed and the output is called binary sequence. Random number generators are used in many cryptography algorithms and applications [212, 213]. They are used to provide high security for many cryptographic systems such as DES secret key, prime numbers in RSA algorithm, and prime numbers in digital signature [214]. The PRNG output of length l is not random but takes a small truly random bit and expands it to a larger sequence of length l [10]. In this way the PRNG sequence cannot be distinguished from the truly random sequence. To confirm the randomness of PRNG output, specific statistical tests and analysis should be applied. Several statistical tests and analyses have been designed to test random and pseudorandom number generators [211].

Generating truly random sequences using hardware devices or software programs is considered a difficult task [10, 213]. Hardware-based bit generators exploit the randomness of physical phenomena, such as energy and input sound from a microphone. Many processes can be used with software-based bit generators, such as system clock, mouse movement, and content of input/output buffer as a seed [213]. The behaviour of these processes depends on several factors. Designing a software-based bit generator is harder than designing a hardware-based bit generator. One-way function is used to generate a sequence of pseudorandom bits by choosing random seeds and then generating the sequence for these seeds (s, s+1, s+2, ...) [211]. Moreover, hash function and block cipher algorithms can be implemented to generate random bit sequences [6, 8].

2.10 Confusion and Diffusion

In 1949, Claude Shannon introduced the terms ‘confusion’ and ‘diffusion’, which are considered the foundation for designing a secure cipher [215]. Shannon’s theory aims to deduce the possibility of ciphertext attack based on plaintext statistical analysis. Some cryptanalysts base their attacks on knowledge of plaintext statistical characteristics. In some languages, plaintext of different letters or words has a frequency distribution, which could be the starting point to find the used key or part of it [6]. Therefore, Shannon suggested that the ciphertext be independent of the used key and that the ciphertext be independent of the plaintext. *Diffusion* is hiding the relationship between the plaintext and ciphertext; changing one bit in plaintext affects more than half of the ciphertext bits. C*onfusion* is hiding the relationship between the statistics of ciphertext and the used key so that it is sufficiently complicated to foil any attempt to find the key [7, 11].

The principle of diffusion prevents the cryptanalyst from finding any relationship between the plaintext and the ciphertext, while confusion prevents the cryptanalyst from finding any relationship between the ciphertext and the used key [7]. In general, cryptography algorithms are designed based on confusion and diffusion [70, 216].

Hash functions are similar to conventional encryption methods in that they require the influence of the whole input message to be spread into the hash value space [217]. In an ideal hash function, the relationship between bits in input message and corresponding bits in hash value should be complex. Therefore, any bit change in the input message should affect at least half the hash value bits, and each bit has a 50% probability of changing.

2.11 Cryptanalytic Techniques

Cryptanalysis, aka code-breaking, is the science of studying the methods of deciphering the ciphertext without knowing the used key; this depends on the nature of the algorithm, and some knowledge of the properties of the plaintext and/or pairs of plaintext and ciphertext. Cryptanalysis exploits the properties of the algorithm to conclude a specific plaintext or used key. Once the opponent succeeds in concluding the identity of the used key, all past and future communication will be vulnerable. A MAC algorithm will require a cryptanalysis effort greater than or equal to the brute-force effort. It would be impracticable to try to find a collision with a very large key space. Cryptanalysis of the hash functions focuses on the internal structure of the compression algorithm (f) and is based on attempts to find efficient techniques for producing collisions for a single execution of the compression. In hash functions, collisions must exist if the length of the message is longer than the length of the hash code; while it is computationally infeasible to try to find a collision [6]. In cryptology, various attacking techniques are applied to cryptographic algorithms. We will explain the types of attack briefly in the following subsections.

2.11.1 Brute-force Attacks

Brute-force attack, aka exhaustive search attack, is one of the attacking techniques that aim to break the cryptography algorithms by searching all possible keys until the right key is found. A longer key size is more difficult to attack than the shorter key size, as the key size affects the needed resources exponentially. In general, the attackers use this kind of attack when it is very difficult to find weaknesses in cryptography algorithms. The security of a cryptography algorithm against this kind of attack depends on the length of the key and the time required to find all possible keys using available resources. Therefore, the length of key size is very important in symmetric key, public key, and hash function algorithms. In cryptography systems any technique that can find the secret key faster than the brute-force attack is considered a successful attack.

A collision in hash function means to have two different input messages (*M1, M2*) that produce the same hash value [*H(M1) = H(M2)*] [11]. If a strong collision is required, then the value $2^{n/2}$ determines the strength of the hash code against brute-force attacks. Oorschot and Wiener presented a design for a $10 million collision search machine for MD5; they found that the same machine with a hash code length of 160 bits would require over four thousand years to find a collision [6]. Hash function is never collision-free because the number of possible inputs of the hash function is infinite and the number of possible outputs is finite [11]. The length of the hash value makes the job of the opponents more difficult as the opponents need to find two messages of different lengths that hash to the same hash value [6]. Hash function strength against brute-force attack depends on the length of the algorithm hash value [6].

2.11.2 Birthday Attack

Birthday attack is a type of cryptography attack that is based on the birthday problem. The birthday problem is the probability of finding at least two equal values within one group of values [218]. The goal of this attack is to find two different inputs (*x1, x2*) to a given function *f(x)* that produce the same output value *f(x1) = f(x2)*; this is known as collision and the fastest way to find a collision is the birthday attack [8, 11, 190]. If the function is collision-resistance, then it is computationally infeasible to look for two different inputs that have the same output. A simple definition of birthday attack is to find one value two times within the same set of elements, which is simply collision [11]. Cryptographers are interested in the birthday attack, because it is computationally infeasible to find collisions for most of the hash functions with large hash values, such as 160-bit and greater hash value size. According to Mihir et al. 2004, this is true only if the hash function is random or regular; regular means that all values in the range have the same number of pre-images [219]. Birthday attacks on cryptography hash function require on average $2^{n/2}$

operations where n is the size of the hash value in bits [190]. The secure hash algorithm-1 (SHA-1) produces a 160-bit message digest; thus the birthday attack needs 2^{80} hash operations.

2.11.3 Meet-in-the-middle Attack

Meet-in-the-middle attack, aka plaintext attack, is a type of cryptographic attack that is used to deduce double encryption secret key of block cipher algorithms. The attacker is assumed to have access to pairs of plaintext and ciphertext. The attacker then starts encrypting the plaintext by various keys and at the same time decrypting ciphertext by various keys, seeking to match intermediate values of the encrypted plaintext and the decrypted ciphertext. If the intermediate value is found, then it is highly probable that the secret keys are the used key in the double encryption process. This type of attack exponentially reduces the time required in a brute-force attack to deduce the double encryption keys.

2.11.4 Other Attacking Techniques

In this section, we recap other three well-known attacking techniques: ciphertext- only attack, known-plaintext attack, and chosen-plaintext attack [6, 8, 10, 14]. Attacks on cryptography algorithms aim to break the algorithm to add, delete, change, and/or read important information. The attacker performs a certain type of attack based on available information and resources.

1- Known-plaintext attack is one of the attacking models that can be applied to encryption algorithms. In this mode, the attacker is assumed to have access to pairs of plaintext and ciphertext. The attacker uses the pairs of plaintext and ciphertext to retrieve the secret key.

2- Chosen-plaintext attack is one of the attacking models used to collect information to reduce the security of the encryption scheme. In this model of attack, the attacker is assumed to be able to encrypt plaintext and obtain the corresponding ciphertext. The attacker encrypts different plaintext and obtains the corresponding ciphertext to study and analyse these pairs in order to retrieve the secret key.

3- Ciphertext-only attack is a cryptanalysis attacking technique for breaking encryption algorithms. In this type, the attacker is assumed to have access to ciphertext only. The attack is considered successful if the attacker can retrieve the some information about the plaintext or the secret key.

Differential cryptanalysis and linear cryptanalysis are two general forms of attacking techniques based on known-plaintext attack. The *differential cryptanalysis* is a method used mainly for breaking block ciphers, but it can be used for breaking stream cipher and hash function [159, 220]. In general, this method studies the relationship of the differences between the input and output for the function. For encryption algorithm, this method studies the relationship of differences of pairs of plaintext with differences of corresponding pairs of ciphertext. For the hash function, a collision of hash function can be found if the difference between the input and output of the compression function is equal to zero. *Linear cryptanalysis* is a known-plaintext attack that uses linear approximation for non-linear operations [188]. This type of cryptanalysis is used to break block cipher and stream cipher algorithms.

Many studies on digital chaotic cryptography have been conducted to design new cryptography primitives based on chaotic map [30, 52, 60, 63, 64, 221-227]. In the next chapter, we will discuss in detail chaos theory, cryptographic systems based on chaos theory, and a literature review of chaotic cryptography.

2.12 Summary

This introductory chapter has presented the basic concepts and principles of security, electronic commerce and cryptography primitives. The focus of this chapter is on block ciphers and hash functions. Block cipher principles, evaluation, modes of operation, and encryption using multiple modes of operation are explained in detail. Three well-known block cipher algorithms (DES, IDEA, and AES) are discussed. Cryptographic hash functions' principles, characteristics, applications, and security are also explained in detail. Descriptions of two popular hash functions (MD5 and SHA-1) are given. Moreover, confusion and diffusion properties and well-known attacking techniques on block ciphers and hash functions are discussed. Finally, we discussed concepts, types and applications of random number generators.

Chapter 3

Chaos and Cryptography

3.1 Introduction

Over the last several years, many researchers have studied chaos theory in numerous fields, such as electronic systems, fluid dynamics, lasers, weather and climate [25-29]. Chaos theory is a branch of mathematics that studies the behaviour of complex dynamic systems which are highly sensitive to change in their parameters and give unpredictable results. There are several popular examples of chaotic systems such as Lorenz attractor, Rossler attractor, Logistic map, Henon map, Tent map, and Piecewise linear chaotic map. In general, the security of cryptography systems was built based on difficult or unsolved mathematical problems. Chaos theory has attracted the cryptography field due to its characteristics, such as deterministic nature, unpredictability, random-look nature and its sensitivity to initial value [30]. Chaotic systems have potential applications in such cryptography algorithms as block cipher, stream cipher, hash function, and pseudorandom number generator. Over the last two decades, there has been tremendous interest in utilizing chaotic systems to design secure cryptographic algorithms [13, 30, 32-117, 228-236].

The organization of this chapter is as follows. In section 3.2, we will introduce chaos theory, chaotic maps and the Lyapunov exponent. We will focus on and explain the details of a logistic map as a simple non-linear chaotic map using several tests and analyses. We will briefly introduce Lorenz attractor, Rossler attractor, Henon map, Tent map, and Piecewise linear chaotic map. In section 3.3, we will explain the relationship between chaotic systems and cryptography algorithms. In section 3.4, we will provide a comprehensive review of chaotic block ciphers, chaotic hash functions and chaotic pseudorandom number generators. In sections 3.5 and 3.6, we discuss implementation issues and give a summary of this chapter, respectively.

3.2 Chaos Theory

Chaos is derived from a Greek word ‘Χαos’, meaning a state without order or predictability [237]. A chaotic system is a simple, non-linear, dynamical, and deterministic system that shows completely unpredictable behaviour and appears random [26]. Moreover, it is a deterministic system with great sensitivity to initial conditions, such that a computer system can give an amazingly different result when the value of an input parameter is changed. On the other hand, in classical science small changes in an initial value might generate small differences in the result [238]. A system is called a chaotic system if it is sensitive to initial conditions, topology mix, and if periodic orbits are dense.

According to Alligood et al. (1996), *a dynamical system* contains all the possible states and regulations that control the next state from the current state. On the other hand, the *deterministic* regulations are those that determine the current state uniquely from the previous states, whereas there is always a mathematical equation to determine the system evolution [239, 240]. From the previous definitions of deterministic and dynamical systems, we cannot say that the randomness is not allowed. The *bifurcation* in dynamic differential equation changes the number of solutions as the parameters are changed [241].

In 1890 Poincaré published his article [242] (On the equations of the dynamics and the three-body problem) of 270 pages, which simplified the way of looking at the complicated continuous trajectories from differential equations [239]. Then, in 1898 Hadamard observed the sensitivity to initial conditions and unpredictability of special systems, calling this the geodesic flow [26]. Later, in 1908, Poincaré noted that chaos sensitivity depends on initial conditions and gives unpredictable results [26]. Later on, Edward Lorenz (1963) examined chaos theory and described a simple mathematical model of weather prediction [240, 243]. Lorenz's model was the first numerical model to detect chaos in a non-linear dynamical system [241]. Lorenz's findings were very interesting in that some equations rise to some surprisingly complex behaviour and chaos behaviour dependent on the initial condition [27, 244]. In 1975, Li and Yorke were the first to introduce the word 'chaos' into mathematical literature, where system results appear random [26].

Chaotic maps have been the subject of an extremely active research area due to their characteristics, such as sensitivity to the initial value, complex behaviour, and completely deterministic nature. The chaotic behaviour can be observed in many different systems, such as electronic systems, fluid dynamics, lasers, weather, climate and economics [25-29, 245, 246]. Our intuition tells us that a small change in input parameters should give a small change in output, but chaotic systems show us that this is not necessarily the case. Usually, chaotic maps define infinitely large fields of real numbers. The most important characteristics of chaotic systems are as follows:

1. *Apparently random behaviour but completely deterministic*: the behaviour of chaotic systems seems to be random but actually it is purely deterministic. Hence, if we run the chaotic system many times with the same initial value, we will obtain the same set of output values. Furthermore, the chaotic systems are dynamical systems that are described by differential equations or iterative mappings, and the next state is specified from the previous state (see equation 3-1 [26, 238, 239, 247].

$$\frac{d}{dt}x_i = F_i(x_1, \ldots.., \; x_n) \quad i = 1, 2, \ldots, \; n \qquad (3\text{-}1)$$

2. *Sensitivity dependence on the initial conditions (The state from which the system starts)*: Dynamical systems evolve completely differently over time with slight changes in the initial state [29, 248, 249].

3. *Unpredictable (difficult or impossible to predict the behaviour in the long term)*: In chaotic maps, even if one knows the current state of the chaotic system it is useless trying to predict the next state of the system. In other words, it is very difficult to predict the future states of the chaotic system in the long term [238, 240].

3.2.1 Lyapunov Exponents

A Lyapunov number is the divergence rate average of very close points along the orbit, which is the natural algorithm of the Lyapunov number (see Figure 3-1) [239, 250]. Therefore, the Lyapunov exponent may be used with chaotic behaviour to measure the sensitive dependence on the initial condition [241]. This means that, in one-dimensional chaos maps, the Lyapunov numbers are used to measure separation rates of nearby points along the real line. The Lyapunov exponent is used to help choose the initial parameters of chaotic maps that fall in chaotic areas. For example: suppose we have two nearby points (x_1, x_2) in a one-dimensional map f and $f'(x_1) = z, z > 1$, then x_2 orbit will separate from x_1 at approximately rate z per iteration, until the orbit of x_2 moves significantly far away from x_1 [239]. The Lyapunov exponent has three different cases of dynamics as follows [238]:

1. If all Lyapunov exponents are less than zero, the orbit is attracted to a fixed or stable point.

2. If the Lyapunov exponents are zero, there is an ordinary attractor, which is simpler than a fixed point. This mean that the system is neutrally stable and in steady state mode, where the attractor maintain constant separation.

3. If at least one of the Lyapunov exponents is positive, the dynamical is chaotic and vice versa.

Lyapunov number =

$$L(x_1) = \lim_{n\to\infty} (|f'(x_1)|....|f'(x_n)|)^{1/n} , \qquad (3\text{-}2)$$

where $\{x_1, x_2,.... x_n\}$ is the orbit of the map f on the real line *R*.

If the limit of the Lyapunov number exists, the Lyapunov exponent is defined as follows:

Lyapunov exponent =

$$h(x_1) = \lim_{n\to\infty} (1/n) [\ \ln|f'(x_1)| + ... + \ln|f'(x_n)|\ ] \qquad (3\text{-}3)$$

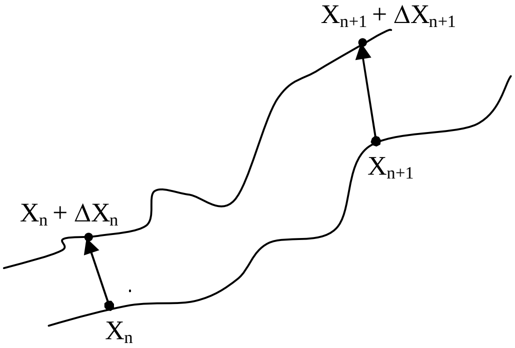


Figure 3-1: Lyapunov exponent principle

3.2.2 Chaotic Maps

According to Alligood et al. (1996), a chaotic map is a function of its domain and range in the same space, and the starting point of the trajectory is called the initial value (condition). Chaotic dynamics have a unique attribute that can be seen clearly by imagining the system starting twice with slightly different initial conditions [241]. Chaos theory attempts to explain the result of a system that is sensitive to initial conditions, complex, and shows unpredictable behaviour. Chaotic dynamical systems increase communication security with higher dimensions and more than one positive Lyapunov exponent [251]. A Lyapunov exponent is used to help select the initial parameters of chaotic maps that fall in chaotic areas. A chaotic system exhibits some chaotic behaviour and often occurs in the study of dynamical systems. In the following subsections, we will give a brief induction to some chaotic systems: Logistic map, Lorenz attractors, Rossler attractors, Henon map, Tent map, and Piecewise linear chaotic map.

3.2.2.1 Logistic Map

In 1845, Pierre Verhulst proposed a logistic map, which is a simple non-linear dynamical map. A logistic map is one of the most popular and simplest chaotic maps [241]. Logistic map became very popular after it was exploited in 1979 by the biologist Robert M. May [252]. The logistic map is a polynomial mapping, a complex chaotic system, the behaviour of which can arise from very simple non-linear dynamical equations, as shown in Figure 3-2 [239]. The logistic map equation is written as:

$$g(x_n) = x_{n+1} = rx_n(1 - x_n)\,, \tag{3-4}$$

where x_n is a number between zero and one, x_0 represents the initial population, and r is a positive number between zero and four.

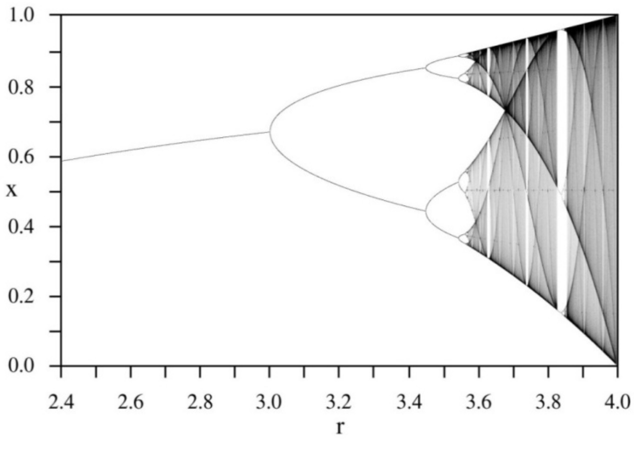


Figure 3-2: Bifurcation diagram of the Logistic map [253]

The logistic map is one of the simplest chaotic maps; it is highly sensitive to change in its parameter value, where a different value of the parameter r will give a different map f [254]. Its transformation function is F: $[0,1] \rightarrow [0,1]$ which is defined in the above equation. From the onset of chaos, a seemingly random jumble of dots, the behaviour of the logistic map depends mainly on the values of two variables (r, x_0); by changing one or both variables' values we can observe different logistic map behaviours. The population of a logistic map will die out if the value of r is between 0 and 1, and the population will be quickly stabilized on the value (1- r)/r if the value of r is between 1 and 3 [239]. Then, the population will oscillate between two values if the value of parameter r is between 3 and 3.45. After that, with values of parameter r between 3.45 and 4 the periodic fluctuation becomes significantly more complicated. Finally, most of the values after 3.57 show chaotic behaviour.

In the logistic map $g(x_n) = rx_n(1 - x_n)$, the function result depends on the value of parameter r, where different values of r will give quite different pictures. We can note that $g(x_1) = x_2$ and $g(x_2) = x_1$, that mean g($g(x_2)) = x_1$ and g($g(x_1)) = x_2$. According to Alligood et al. (1996) the periodic fluctuation between x_1, x_2 is steady and attracts orbits (trajectories). Therefore, there are a minimum number of iterations of the orbit to repeat the point. There are obvious differences between the behaviour of the exponential model and the logistic model's behaviour. To illustrate the difference between the two functions, we take an example of the exponential function $f(x_{n+1}) = 2x_n$ and an example of logistic function $g(x_{n+1}) = 4x_n(1 - x_n)$; the initial value for both functions is 0.0090, and we then calculate the population for n = 0, 1, 2, ….., 10 resulting in an accuracy of five decimal places. We can notice that the output values of the exponential function are always increasing as time progresses, while the output values of the logistic function are fluctuating between zero and one [239].

3.2.2.2 Lorenz Attractor

The Lorenz attractor is one of the most popular three-dimensional chaotic attractors; it was examined and introduced by Edward Lorenz in 1963 [238, 239, 243]. He showed that a small change in the initial conditions of a weather model could give large differences in the resulting weather. This means that a slight difference in the initial condition will affect the output of the whole system, which is called sensitive dependence to the initial conditions. The non-linear dynamical system is sensitive to the initial value and is related to the system's periodic behaviour [26]. Lorenz's dynamic system presents a chaotic attractor, whereas the word chaos is often used to describe the complicated manner of non-linear dynamical systems [240]. Chaos theory generates apparently random behaviour but at the same time is completely deterministic, as shown in Figure 3-3. The Lorenz attractor is defined as follows:

$$\begin{aligned} dx/dt &= a(y-x) \\ dy/dt &= rx - y - xz \\ dz/dt &= xy - bz \end{aligned} \tag{3-5}$$

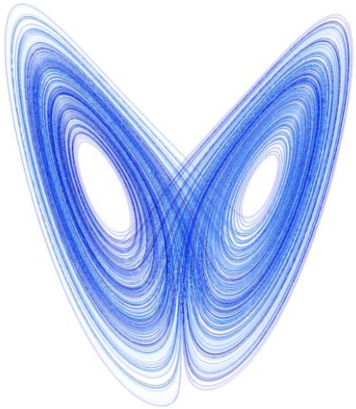


Figure 3-3: A plot of the trajectory of the Lorenz system, (Modified from [255])

3.2.2.3 Rossler Attractors

In 1976 [256], O. Rossler created a chaotic attractor with a simple set of non-linear differential equations [238, 239]. Rossler attempted to write the simplest dynamical system that exhibited the characteristics of a chaotic system [238]. The Rossler attractor was the first widely-known chaotic attractor from a set of differential equations; defined by a set of three non-linear differential equations, the system exhibits a strange attractor for $a = b = 0.2$ and c = 5.7 (see equation 3-6) [239, 257]. Rossler attractor is a system of three non-linear differential equations, which is a quite nice but is not famous attractor, see Figure 3-4.

$$\begin{aligned} dx/dt &= -y - z \\ dy/dt &= x + Ay \\ dz/dt &= B + xz - Cz \end{aligned} \qquad (3\text{-}6)$$

A, B, and C are constants.

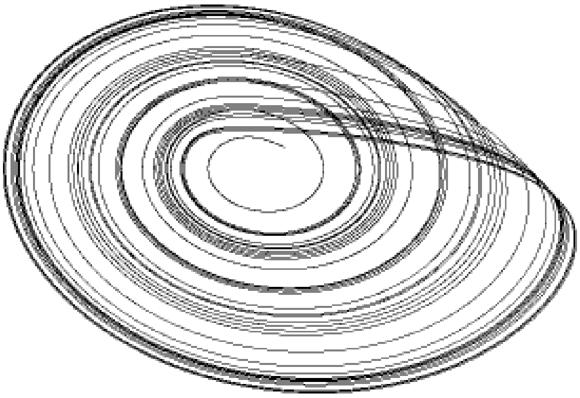


Figure 3-4: Rossler Attractor [257]

3.2.2.4 Henon Attractors

The Henon map is one of the dynamical systems that exhibit chaotic behaviours. The Henon map is defined by two equations; the map depends on two parameters *a*, *b*, and the system exhibits a strange attractor for *a* = 1.4 and *b* = 0.3 (see equation 3-7). A Henon map takes one point (*x*, *y*) and maps this point to a new point in the plane, as shown in Figure 3-5 [239].

$$\begin{aligned} x_{n+1} &= y_n + 1 - ax_n^2 \\ y_{n+1} &= bx_n \end{aligned} \qquad (3\text{-}7)$$

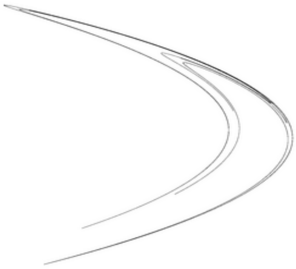


Figure 3-5: Henon attractor for a = 1.4 and b = 0.3 [257]

3.2.2.5 Tent Map

A Tent map is an iterated function of a dynamical system that exhibits chaotic behaviours (orbits) and is governed by equation 3-8. It has a similar shape to the logistic map shape with a corner (Figures 3-6 and 3-7) [239]. The Tent map exhibits the Lyapunov exponents on the unit interval $T(x) \in [0,1]$ and $\mu \in [0,2]$. It is a simple one-dimensional map generating periodic chaotic behaviour similar to a logistic map.

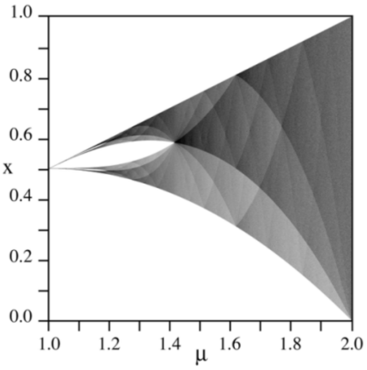


Figure 3-6: Bifurcation diagram for the tent map [258]

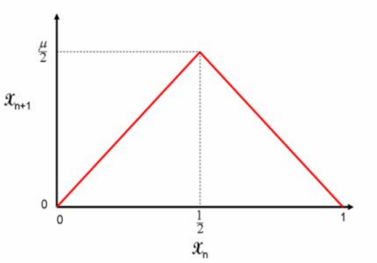


Figure 3-7: Graph of tent map function

$$T_{\mu}(x)=\begin{cases}\mu x & , \quad x\leq 1/2 \\ \mu(1-x) & , \quad 1/2\leq x\end{cases} \tag{3-8}$$

3.2.2.6 Piecewise Linear Chaotic Maps

Piecewise linear chaotic maps (PWLCMs) are simple non-linear dynamical systems with large positive Lyapunov exponents. In [259], the authors show that PWLCMs have several brilliant chaotic properties that can be exploited in chaotic cryptographic algorithms. PWLCM has perfect behaviour and high dynamical properties such as invariant distribution, auto-correlation function, ergodicity, large positive Lyapunov exponent, and mixing property [260]. Iterations of PWLCM with initial value and control parameters generate a sequence of real numbers between 0 and 1, which is called an orbit. A large positive Lyapunov exponent means that the system shows chaotic behaviour over large orbits [261]. Correlation functions are a very important test of the correlation over time and space between random variables at two different points, thus indicating correlation statistical properties [259] .

$$f(x_n,p)=\begin{cases} x_n/p & , \quad x_n\in[0,\ p) \\ (1-x_n)/(1-p) & , \quad x_n\in[p,\ 1]\end{cases} \tag{3-9}$$

$$f(x_n,p)=\begin{cases} x_n/p & , \quad x_n\in[0,\ p) \\ (x_n-p)/(0.5-p) & , \quad x_n\in(p,\ 0.5] \\ f(1-x_n,p) & , \quad x_n\in[0.5,\ 1]\end{cases} \tag{3-10}$$

where x_0 is the initial condition value, *P* is the control parameter, $x_n\in[0,\ 1]$, and *P* $\in(0,\ 0.5)$.

PWLCMs are the simplest kind of chaotic systems, which need one division and few additions. A skew Tent map is a PWLCM defined by a generalized form of Tent map that is very similar to a Tent map with small differences (see equation 3-9). A more complex example of PWLCMs is defined by equation 3-10. It is very clear from equation 3-8 that f(0, p) = 0, $f^2(0.5, p) = 0$, $f^3(1, p) = 0$ for any P $\in$ (0, 0.5). Thus, we should avoid those values as initial parameters of x_n.

3.3 Chaos-based Cryptography

Chaotic systems and cryptography algorithms have similar properties including sensitivity to any change or changes in the initial parameters, unpredictability over long periods, and random-like behaviour [225, 262, 263]. Therefore, understanding the relationship between conventional cryptography algorithms and chaos-based cryptography algorithms is very important [264]. The similarities and differences between cryptography algorithms and chaotic systems are shown in Table 3-1 [31, 224, 264, 265]. The main difference between chaotic systems and cryptography algorithms is that encryption transformations are defined on finite sets of integers, but chaotic systems are defined on floating point numbers [224]. The parameters in chaotic maps are meaningful if they are real numbers, which can be used in the cryptographic algorithms as encryption and decryption keys.

Table 3-1: Comparison between chaotic systems and cryptographic algorithms [31, 224, 264, 265]

Chaotic Systems	Cryptographic Algorithms
• Parameters (Real)	• Key (Boolean)
• Sensitive to change the control parameters or initial conditions	• Diffusion
• Ergodicity	• Confusion
• Using set of real numbers	• Finite set of integers
• Iterations	• Rounds
• Deterministic dynamical	• Deterministic Pseudorandom
• Structure complexity	• Algorithm complexity

In well-designed cryptographic algorithms, the influence of changing one bit of the plaintext or the key spreads over the ciphertext. On the other hand, chaotic systems' iterations are used to spread the initial region over the entire chaotic system space. Cryptography confusion and diffusion properties aim to complicate the statistical relation between the ciphertext and the key and between the plaintext and the ciphertext, respectively [14, 177]. The chaotic mixing property and sensitivity to initial condition are very close to the diffusion property of a cryptographic encryption system [266]. The ergodicity property indicates that it is very difficult to predict the system behaviour based on the values of the initial conditions, which is similar to the confusion property [60]. Rounds in cryptography algorithms are very similar to iteration in chaotic systems.

3.4 Chaos Applications in Cryptography

Chaos theory has attracted the cryptography field due to its characteristics such as deterministic nature, sensitivity to initial conditions, unpredictability, and complex structure [224]. Over the past two decades, a vast number of papers have been published on chaos-based cryptography including chaotic block ciphers [13, 32-68], chaotic cryptography hash functions [30, 63, 69-89], and chaotic pseudorandom number generators [34, 62, 68, 90-117]. Simultaneously, many cryptanalytic researchers have analyzed the proposed chaos-based cryptographic algorithms and found that some of them are not secure enough and/or are slow algorithms [118-155]. Therefore, the main challenge in this research area is to design secure and fast chaos-based cryptography algorithms. In the last chapter of this dissertation, we will make some suggestions for future research in this area.

The first published paper on a cipher-based dynamical system was by Wolfram in 1985; this is a stream cipher algorithm based on cellular automation [105]. Two years later another paper was published on a public-key cryptosystem based on cellular automation in [267]. In 1989, Matthews published the first chaos-based stream cipher algorithm, which attracted the attention of many researchers [103]. The proposed stream cipher design was based on a generalized logistic map. From 1989 to 1992, many papers on chaos-based cryptography were published [55, 102, 268-270]. Unfortunately, later on some researchers showed that proposed chaotic cryptography algorithms have certain problems [118-120]. Consequently, in the next five years very few chaos-based cryptography algorithms were designed. In 1997, a considerable number of chaos-based cryptographic algorithms were proposed [13, 227-236, 265]. The three most common cryptography primitives are block cipher, hash function and pseudorandom number generator [271]. In the following subsections, we will give a comprehensive review of previous work on chaotic block ciphers, chaotic cryptographic hash functions and chaotic pseudorandom number generators.

3.4.1 Block Cipher Based on Chaotic Systems

As we discussed before, a block cipher is a transformation function that maps units of plaintext bits to ciphertext bits of the same unit size under the control of the secret key. The decryption method divides the input ciphertext into blocks of equal length and then applies the decryption function to each block using the same shared secret key. To evaluate the security of a block cipher, we assume that the attacker can access all transmitted ciphers and knows the encryption algorithm details without knowing the shared secret key. A block cipher is considered totally broken if the shared secret key is discovered, and it is considered partially broken if part of the plaintext is retrieved [177]. A well-designed block cipher encryption algorithm should have very high confusion and diffusion properties.

Over the past two decades, many researchers were utilized a chaotic system to design block cipher algorithms in order to provide high security [13, 32-68]. Unfortunately, some of the proposed algorithms are described as insecure and/or slow algorithms [119, 121-136]. Therefore, further research is still needed to design fast and secure chaotic block cipher algorithms. In this section, we will review chaos-based block cipher cryptosystems and give brief details of insecure and slow algorithms.

In 1991 [55], Habutsu et al. suggested a new secret key cryptosystem based on iterating a one-dimensional chaotic map using a generalized form of tent map (see equation 3-9). They used the parameter α as secret key where the value of $p \in$[0.4, 0.6]. The algorithm encrypts blocks of 64-bits of plaintext each time, and the initial value is a part of the plaintext between zero and one. The encryption process involves iterating the plaintext 75 times to produce the ciphertext. The main disadvantage of Habutsu et al.'s algorithm is that the ciphertext size is much larger than the plaintext size [55]. At the same conference, Eli Biham published a paper to describe the two attacks on Habutsu et al.'s algorithm: chosen-ciphertext attack and known-plaintext attack [119]. In addition, he pointed out some disadvantages of the suggested algorithm, such as the large size of the plaintext and small range of ciphertext, which could help an attacker to find the linear relationship between the plaintext and the ciphertext.

Several modified versions of Habutsu et al.'s algorithm were suggested to improve its security [13, 56-59]. In [56], the use of cellular automata to design a new cryptosystem was suggested. In [57], a new block cipher was designed based on a skew tent map as a one-to-one chaotic map. In [58], the authors proposed a new fast communication system based on a multiple piecewise linear chaotic map instead of using one only. In [59], the authors proposed a chaotic cryptosystem based on a tent map to provide higher security. In 2003, the proposed system in [58] was attacked by G. Alvarez et al. [122] using different attack techniques.

In 1998, M.S. Baptista published a new cryptographic system based on an iterated logistic map [60]. Baptista's method works by dividing the domain of the logistic map from 0.2 to 0.8 to 256-site with each region assigned to one character of the English language. The sender will choose the initial value of the logistic map and iterate it until it reaches the correspondence region of the first character. Then it will take the result of the iterated logistic map of the previous character as the initial value for the next character and iterate it until it falls in the region of the next character and so on, until the whole message has been processed. The number of iterations of each character will be the correspondence ciphertext of the plaintext.

There have been several researchers attacks on Baptista's cryptographic system [123, 131-134]. According to W. Wang et al. [61], this method has two main drawbacks. First, the distribution of ciphertext is not sufficiently spread out and it has too few iterations. The second drawback is that it is too slow and random numbers are repeated early. In 2001, Jakimoski et al. show two attacks on the algorithms of Baptista and Alvarez using known-plaintext attack, and they point out that they are not competitive for standard algorithms [133].

E. Alvarez et al. presented a new symmetric block cipher encryption approach based on a tent map [62]. The encryption algorithm can be described as follows: firstly choose the control parameter k as secret key, then choose plaintext block of size b_{max} size in integer for each plaintext block, $b_i = b_{max}$. Next, choose the threshold U_i to generate the chain C_i according to the rule: $x_n \leq U_i \rightarrow 0$ and $x_n \geq U_i \rightarrow 1$. As the chain is generated, keep looking for the pattern b_i in C; when it is found, record the (U_i, b_i, x_{ni}) as the ciphertext of plaintext of the block size b_i, and so on until the entire plaintext has been encrypted. Thus, this algorithm encrypts each block of plaintext to triple blocks cipher based on a d-dimensional chaotic system. A few months after this method was proposed, G. Alvarez et al. [121] showed that E. Alvarez's method can be easily broken by four different methods and found some other weaknesses. He attacks the method using four methods: chosen-plaintext attack, chosen-ciphertext attack, ciphertext-only attack and known-plaintext attack.

In [63], Wong proposed a new cryptography block encryption based on Baptista's method. He proposed a cryptography system using a dynamical look-up table instead of a static one. Therefore, during the encryption/decryption process, the look-up table contents will be continuously updating. This update will lead to the building of a complicated relationship between plaintext and ciphertext. Wong's method works by building a look-up table by dividing the interval range of the logistic map to match each possible input to equal the width range. Then, the *i*th message block is encrypted by iterating the logistic map until it falls in the corresponding region, as in Baptista's method. If the number of iterations is large enough to be secure, it will be sent as ciphertext of the input plaintext. To encrypt the next block of data, one must change the position of the *i*th with the *j*th block based on a certain formula and so on until the end of the message.

Wong's algorithm is considered fast and secure compared to Baptista's method, but the ciphertext length is still twice that of the plaintext. As the final look-up table does not depend on a key, this is main source of attacks on the system. Later on, G. Alvarez et al. pointed out that Wong's algorithms [63] are inefficient and insecure and Baptista's algorithm's weakness is reproduced in Wong's algorithms [132]. Moreover, using the same secret key twice helps the attacker to break the system.

In 2003, Pareek et al. proposed a new cryptosystem based on a logistic map as a chaotic system [32]. The secret key in this scheme is generated externally to prevent successful attacks on the chaotic system. The initial conditions and the control parameters of the chaotic system are not used explicitly in the subkeys generation to provide a more secure cryptosystem. In [124], the authors explained how to attack Pareek et al.'s block cipher cryptosystem using a known-plaintext attack, choosing plaintext attack and ciphertext attack. Moreover, they pointed out that Pareek et al.'s cryptosystem has low encryption speed, and it would be inexpensive to attack it.

In [64], the author presented a chaotic encryption scheme for digital communication based on a Baker map that encrypts wave signal instead of symbolic sequence at the physical level. In 2005, Guan et al. proposed an image encryption algorithm using an Arnold map to shuffle the positions of the image pixels and Chen's chaotic system to change the grey levels of the shuffled pixels [65]. Later on, Cokal and Solak wrote a short paper that described chosen-plaintext and known-plaintext attacks on Guan et al.'s algorithm [135]. Another two image encryption algorithms based on chaotic maps have been proposed with permutation operation and XOR-like transformations of shuffled pixels in encryption functions [66, 67]. In [136], Arroyo et al. have discussed and shown the weakness of the two schemes and described how to attack them using the chosen-plaintext attack method. A chaotic block cipher algorithm for wireless sensor network was proposed; it was claimed to be a very secure algorithm that can be used in confidential communications [35]. Afterwards a group of researchers attacked the proposed algorithm using differential cryptanalysis and proved it to be a very weak algorithm [129].

In 2005, Pareek et al. proposed another cryptosystem using multiple one-dimensional chaotic systems with external secret key of variable length [33]. This scheme is a block cipher-based encryption algorithm of variable-length block size; the number of chaotic iterations and initial conditions depend on the session key and cipher of the previous block. In 2007, Wei et al. [125] explained the fundamental flaw of Pareek et al.'s cryptosystem and showed how to attack it through its flaw. Moreover, they suggested some modification to avoid the flaw and to overcome security problems. Later on, Chengqing pointed out that there are still some security problems in the modifications by Wei et al. [125], such as the existence of weak keys, the fact that some intermediate ciphers are not random, and that the secret key can be attacked using known-plaintext attack technique [126]. Moreover, they showed that the proposed modification is not secure and suffers from the same problem as the original scheme.

In 2006 [34], Xianga et al. proposed a novel block cipher algorithm using a chaotic logistic map to provide a fast and secure encryption algorithm. In the same year, Wang et al. [127] explained how to break this encryption algorithm using chosen-plaintext attack. Furthermore, they suggested a remedial solution to avoid this type of attack. In 2009 [128], Wang and Yu showed that the original algorithm in [34] and the modified version in [127] are not secure and suffer from the same fatal flaws. They could recover the plaintext by applying chosen-plaintext attack. Furthermore, they proposed an improvement to avoid this flaw and enhance the security without losing the original advantages.

In 2008 [35], Shuai et al. proposed a new block cipher based on chaotic systems. The subkeys are generated through logistic mapping and Feistel discrete encryption structure based on chaos operations. One-turn permutation is used to encrypt/decrypt every single block (8-bit). They used small encryption and decryption block size to be visible for implementing a wireless sensor network. The proposed algorithm is described in [129] as an insecure algorithm due to the small number of rounds and because the calculation precision is too small. They presented how to attack the encryption scheme using differential cryptanalysis and deduced the related subkeys of r rounds using chosen-plaintext attack and brute-force attack together.

In 2009 [43], Yang proposed a novel block cipher based on iterating chaotic map with output feedback. He used the output feedback instead of simply mixing the chaotic signal to provide better security than other cryptosystems. A logistic map is used as a binary sequence generator with the secret key the initial parameter; he then used the current state of logistic map as the value of three binary values. XOR and mod operations are utilized to provide high confusion and diffusion properties. The first step is to divide the input message into a number of sequences of length 8 bytes, and the value of A_j is computed based on seven simple equations. The next step is to permute the message block p_i by applying circle left shift on D_i bits. The last step is to calculate the cipher by XORing the permutated plaintext with A_j variable value. The decryption process is similar to the encryption process, but in the last step the ciphertext is XORed with A_j instead of the plaintext with Aj (more details can be found in [43]).

In 2010 [50], Huang and Lin presented a new block cipher algorithm based on a combination of logistic map and substitution permutation encryption network. A logistic map is used as a chaotic system to utilize its characteristics in S-boxes and P-boxes. A chaotic system is used with S-boxes and P-boxes to ensure the confusion and the diffusion properties. This algorithm encrypts 64-bits into the same length of ciphertext. The secret key is a combination of logistic map initial condition and user master key, which is used to generate the other eight subkeys. In each iteration, 8 × 8 S-boxes and 64 × 64 P-boxes are used as substitute transformation and replacement transformation, respectively (more details can be found in [50]).

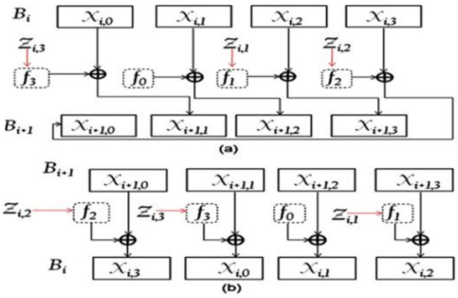


Figure 3-8: Encryption and decryption processes of [68]

In 2011, a Mexican research group proposed a new block cipher encryption algorithm based on a tent map [68]. The tent map was chosen to generate a sequence of bits as it does not have stability islands such as the islands in a logistic map. Stability islands are the non-chaotic areas in the chaotic systems with some values of the control parameter over the chaotic interval. A tent map is scaled to [0, 2^n] to provide higher precision and easier computer implementation; this means that, with a bigger value of n, a better encryption approach will be given. The encryption function consists of r rounds of chaotic map that processes blocks of size 64-bit. The encryption result is assigned to variables $x_{i+1,0}$, $x_{i+1,1}$, $x_{i+1,2}$, $x_{i+1,3}$ as shown in Figure 3-8 (a). The decryption process consists of inverse r rounds of the chaotic map using respective decryption subkeys, and from B_{i+1} the value B_i is calculated. The result of B_i is assigned to the variable $x_{i,0}$, $x_{i,1}$, $x_{i,2}$, $x_{i,3}$ as shown in Figure 3-8 (b).

3.4.2 Hash Function Based on Chaotic Systems

SHA-1 is one of the most widely-used hash functions employed in numerous security applications and protocols. Collisions of well-known hash functions such as PIREMD, MD4, MD5 and SHA-1 have been found [156-158]. Since SHA-1 was attacked in 2005, many researchers have been working on designing a new, alternative secure hash function [165]. Designing a secure hash function based on chaotic systems has increasingly attracted researchers' interest due to its characteristics which are analogous to hash function requirements [30, 70, 76, 81, 87, 88].

Over the past decade, many researchers have utilized chaotic systems to design a cryptography hash function to provide high security [30, 63, 69-89]. Unfortunately, some of the proposed algorithms are described as insecure and/or slow algorithms [132, 137-146]. Therefore, further research is still needed to design a fast and secure chaotic cryptography hash function. In this section, we will review the chaos-based cryptography hash functions and give brief details of insecure and slow algorithms.

To the best of the present researcher’s knowledge, in 2003 Wong was the first to propose hash function-based chaotic systems. Wong published a paper that combined chaotic cryptosystem with cryptography hash function [69]. The proposed chaotic encryption scheme with dynamic look-up table in [63] uses a dynamical look-up table instead of a static one. He found the final look-up table depends only on the input message. Therefore, he suggested using the final look-up table as the hash value of the encrypted message to check the authentication and completeness of the input message. Later on, G. Alvarez et al. point out that Wang’s algorithms [63, 69] are inefficient and insecure, and Baptista’s algorithm weakness was reproduced in Wang’s algorithms [132]. They showed how to break the system in special cases without knowing the secret key. Moreover, they pointed out that the proposed hash is not considered as a MAC function which should depend on the key, as the final look-up table does not depend on the key.

In 2005 [86], a hash function based on a chaotic tent map was proposed; this processes arbitrary input messages to produce hash values of twice the length of the input message block length. Several statistical security analyses were applied, and these showed that the proposed hashing scheme is strong enough against all types of attacks and more efficient than Wang’s hashing scheme [69].

Xiao et al. (2005) proposed a new keyed hash function based on chaotic map with changeable parameters [70]. They proposed a new keyed hash function based on a one-dimension piecewise chaotic system in which the secret key is X_0, $H_0 \in [0, 1]$. The algorithm accepts any message with different sizes to produce 128-bits as the final hash value. It calculates the final hash value without padding to produce a faster keyed hash function than other functions. The hash value in Xiao et al.'s algorithm is produced directly from three fixed positions in chaotic trajectory. Therefore, it is consider to have weak collision resistance where the hash space is not fully covered from these three points; also, if the message is too short the key could be attacked [87].

In 2007, J. Zhang et al. proposed a chaotic keyed hash function based on a feedforward–feedback non-linear digital filter [87]. They designed a feedforward–feedback non-linear filter as a chaotic dynamical system with uniform distribution in which they introduced chaotic shift keying (CSK) and cipher block chaining (CBC) modes to expedite diffusion and confusion. This filter is exploited in designing the proposed keyed hash function. They claimed that the proposed hash scheme has good cryptography properties and is easy to implement by hardware and software. (For more details about this hashing scheme, please see reference [87]).

In 2008, a parallel keyed hash function was proposed based on piecewise linear chaotic map and 4-dimension cat map (see Figure 3-9) [76]. In 2009, another research group proposed an improvement on the original hashing scheme to provide higher security [84]. In the same year, another research group analyzed the security of the original parallel algorithm [140]. They showed how to attack the scheme with differential cryptanalysis and pointed out the problem of five special pairs' weak keys. Wang and Zhao proposed a parallel keyed hash function based on a chaotic neural network [81]. One year later, the security of the proposed algorithm was analyzed [143]. In 2011, Huang analyzed the hashing scheme vulnerability and proposed an enhancement to remove the security problems [84].

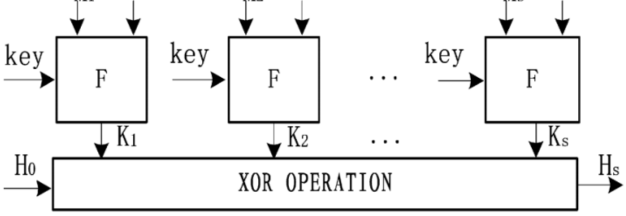


Figure 3-9: Simplified structure of encryption scheme in [76]

In 2008, a research group published a paper on designing a new hash function using chaos theory (CHA-1) built based on two simple functions and chaos theory [30]. They proposed an unkeyed hash function called CHA-1 that produces 160-bit hash value, with a security factor equal to 2^{80} brute-force attacks. CHA-1 accepts any message length shorter than 2^{84} bits and it is sensitive to any change in the input message. On average, SHA-1 execution time is three times faster than CHA-1, but CHA-1 provides 2^{17} more security than SHA-1. In terms of input message length, CHA-1 accepts any message shorter than 2^{84} while SHA-1 accepts only messages shorter than 2^{64}. From the present researcher's point of view, CHA-1 may have two disadvantages. Firstly, it has simple functions similar to SHA-1 functions and it could be vulnerable to Wang et al.'s attack. Secondly, CHA-1 is three times slower than SHA-1 and will become slower as the message size increases; therefore it is unsuitable for e-commerce applications. Thus far, there have been no known successful attacks on CHA-1.

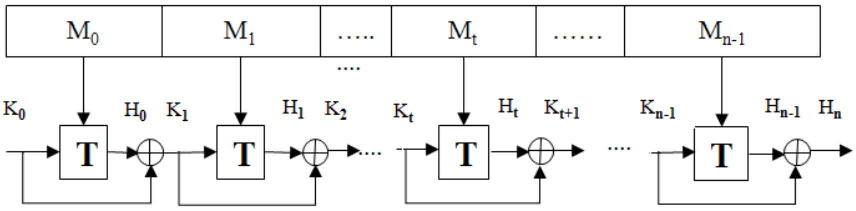


Figure 3-10: Overview of CBHF structure

In March 2009, M. Amin et al. suggested a simple implementation of un-keyed hash function using a tent map (see Figure 3-10) [88]. They explained the general idea of their algorithm in one paragraph without giving sufficient details of how it works. The general idea shows that it was designed based on a tent map and XOR operation. In general, the algorithm works by dividing the input message into n blocks of size1024-bits. If the last message block is less than 1024-bits, it will be appended by one bit '1' followed by n number of '0's. The tent map accepts two main inputs: the initial value K_i and the message block M_i. Each block in the message will process through the chaotic tent map to produce an intermediary hash value and so on until it has processed all message blocks. The final output will be 128-bits representing the hash value of the input message.

In June 2009, we described how to attack the keyed and unkeyed versions of CBHF [137]. The details of our attack are described in chapter 4 .We proofed the attack very robustly, both theoretically and practically. In case of keyed hash function, we were able to append many blocks to the end of any original message without changing the final hash value. On the other hand, we were able to attack the un-keyed version of CBHF by having two totally different messages hashing to the same hash value. In conclusion, CBHF is a very weak hash function and the keyed and unkeyed functions are totally insecure. In 2010, another research group analyzed a computational collision problem in the proposed algorithm [144].

In 2010 [89], Xiao et al. proposed a hash function based on a piecewise linear chaotic map, which has modification detection and localization capabilities. It supports a parallel processing mode to provide higher performance. The input message is divided into 2048-bits and each block is inserted into a look-up table of 256-blocks. PWLCM chaotic map is iterated to process the look-up table array. The look-up table values are grouped into 16 vertical and 16 horizontal groups. The final hash value is the result of XORing the 16 vertical and 16 horizontal groups with each other. In 2011, the security of the proposed algorithm was analyzed [145]. It was shown that this hashing scheme is not secure, with weak confusion and diffusion properties, and the parallel structure resulted in collision. Moreover, the cost of breaking this scheme using the birthday attack is low.

3.4.3 Random Number Generators Based on Chaotic Maps

Since chaotic systems generate unpredictable results, many researchers have been attracted by chaotic systems to design pseudorandom number generators [34, 62, 68, 90-117]. Pseudorandom number generators' (PRNGs) results are mainly used on stream cipher algorithms as key streams that simply XOR with plaintext to generate the correspondence ciphertext using any mode of operation [108].

Moreover, it is very important to generate the secret keys and initialization variables by PRNGs [272]. In the literature, many cipher algorithms have been implemented based on chaotic pseudorandom number generators (CPRNGs) to generate the keystream. In CPRNGs, many chaotic systems have been utilized including Piecewise non-linear chaotic map, Logistic map, Tent map, and Henon attractor. Some researchers have proposed using multiple chaotic systems to enhance the PRNG security [105].

Over the past two decades, many researchers have utilized chaotic systems to design pseudorandom number generators to provide high security [34, 62, 68, 90-117]. Unfortunately, some of the proposed generators are described as insecure and/or slow algorithms [118, 120, 121, 136, 147-155, 270]. Therefore, further research is still needed to design fast and secure chaotic pseudorandom number generators. In this section, we will review chaotic pseudorandom number generators and give brief details of insecure and slow algorithms.

As we mentioned before, the first published paper on ciphers based on a dynamical system was that of Wolfram in 1985; this was a stream cipher algorithm based on cellular automation [105]. Cellular automation is used to generate a random binary sequence that is XORed with the plaintext to produce the correspondence ciphertext. In 1989, Matthews published the first chaos-based stream cipher algorithm, which attracted the attention of many researchers [103]. He suggested using a chaotic function to generate a random sequence as system keys instead of pads. Matthews utilized chaotic system characteristics to generate a random (unpredictable) sequence with sensitivity to any change in the initial conditions or system parameters.

In 1999, E. Alvarez et al. presented a new symmetric block cipher encryption approach based on chaotic systems [62]. The proposed algorithm explained the use of a tent map as a chaotic system. They used chaotic systems to generate a pseudorandom sequence from its orbit using a certain threshold. Then, they searched for the position of the plaintext in the generated sequence and took its information to represent the correspondence ciphertext. A few months later, G. Alvarez et al. [121]

pointed out that E. Alvarez et al.’s method with a tent map can be easily broken by four different methods, and also found certain other weaknesses.

In 2001, Shujuna et al. proposed a pseudorandom binary sequence generator based on coupled chaotic systems (see Figure 3-11) [108]. They claimed that they were using two different chaotic systems instead of one just to provide higher security. In the same year, another new stream cipher based on a logistic map was proposed [110]. This uses two or more chaotic systems to generate pseudorandom sequences. The authors used two nearby logistic map trajectories to generate the pseudorandom sequence with high complexity. The plaintext is XORed with the generated sequence to give the ciphertext. In 2007, Skrobek showed how to break the proposed system and pointed out that using binary representation of some chaotic systems with XOR operation would help the attacker to predicate the ciphertext [151].

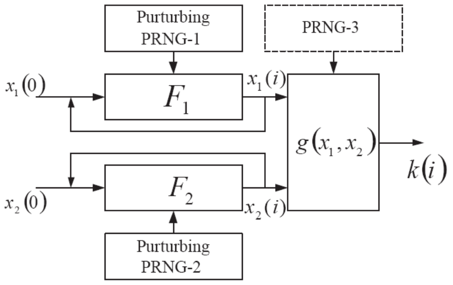


Figure 3-11: Pseudorandom number generator based on couple chaotic systems [108]

In 2003, Lee et al. proposed a chaotic stream cipher based on the composition of multiple chaotic systems [106]. The proposed algorithm generates pseudorandom bytes sequences based on a chaotic system, and then applies certain permutations using a two-dimensional chaotic map. Thus far, there have been no successful attacks on this algorithm. In the same year, new chaos-based pseudorandom number generators were proposed for cryptography applications [91]. In 2005, another research group proposed a pseudorandom number generator derived from a discrete chaotic map that defined over long interval [116].

In 2006, three chaotic pseudorandom number generators were proposed for cryptography applications [34, 109, 111]. Wang et al. proposed a new pseudorandom binary generator based on n-dimensional non-linear digital filter and chaotic systems [109]. They used this filter to increase the randomness and security of the proposed generator. In the same year, Xiang proposed a cipher encryption algorithm based on a logistic map combined with XOR operation, which can be considered an improved version of Baptista's scheme [34]. They used a combination of XOR and circular bit shift in the encryption and the decryption processes. A modified version of Xiang's scheme was proposed by Yu and Cao; they replaced the logistic map with a chaotic neural network with time-varying delay and some other modifications [111]. They generated a pseudorandom sequence using a chaotic neural network.

In 2007, Li et al. showed how to break Yu and Cao's scheme and Yiang's scheme using chosen-plaintext attack and differential known-plaintext attack [155]. The core of the two schemes' security is the pseudorandom number generator. They proved that the pseudorandom number generator does not have uniform distribution and sufficient randomness. In 2007, chaotic image encryption was proposed based on high-dimensional cat map and tent map as chaotic systems to generate a pseudorandom key stream with stream cipher architecture [113]. Thus far, there have been no successful attacks on this algorithm.

In 2008, a new chaotic stream cipher for digital communication was proposed using one-dimensional chaotic systems such as Tent map and Logistic map [112]. This scheme utilized the symbolic dynamics of chaotic system-based synchronization to generate a pseudorandom sequence as a keystream based on the value of the secret key. The plaintext is encrypted using the symbolic dynamics of the logistic map or tent map with certain values of its parameters and initial conditions. In 2011 [153], a research group analyzed the proposed stream cipher encryption scheme. They were able to deduce and estimate chaotic systems' parameters with low error rate, and pointed out that a tent map is not a good source for a pseudorandom number generator and that the logistic map key stream has to be generated from a positive Lyapunov exponent.

Later on, researchers proposed two pseudorandom number generators based on a logistic map to generate pseudorandom binary sequence for cryptography stream cipher [117]. The first generator is based on one logistic map and the second is based on two logistic maps. In the second generator, the initial conditions of the two maps should be independent ($r_0 \neq r_0$, $x_0 \neq y_0$), where x_0, $y_0 \in (0, 1)$ and r_0, $r_0 \in (3.99996, 4]$. They calculate the remainder by dividing the sum of the two output values (x_{n+1}, y_{n+1}) by 1 (see equation 3-12). The algorithm generates the binary pseudorandom sequences based on equation 3-13. They tested their proposed pseudorandom number generator using Beker and Piper's suite [273] and FIPS 140-1 suite [274]. They claimed that the proposed random number generator passed all FIPS 140-1 and Beker and Piper's suites with sequence length of 100,000 bits and significance level $\alpha = 0.05$. In this thesis, we refer to logistic map pseudorandom number generator as LPRNG. The randomness of this generator is analyzed in this research in chapter 8.

$$y_{n+1} = ry_n(1 - y_n)\,. \tag{3-11}$$

$$Sum_i = (y_i + y_i') \mod 1. \tag{3-12}$$

$$Z_i = F(Sum_i) = \begin{cases} 1, & if \quad Sum_i >= 0.5 \\ 0, & if \quad Sum_i < 0.5 \end{cases} \tag{3-13}$$

In 2009 [114], Patidar et al. proposed a novel chaos-based cryptosystem with simple mixing operation. Intermediate chaotic keystreams are generated based on a logistic map and chaotic standard map to provide high confusion and diffusion properties. In 2010 [154], Rhouma et al. analyzed Patidar et al.'s chaotic cryptosystem with only one pair of plaintext or ciphertext. In the same year [115], Patidar et al. proposed a modified version of the proposed algorithm. They claimed to have overcome the security problems in the original algorithm. In 2011 [155], Li et al. analyzed the modified version and showed that it is still insecure to known-plaintext and known-ciphertext attack. They showed that the generated sequence based on logistic map is not random and very weak.

In 2011, a new encryption algorithm based on a tent map was proposed [68]. As the tent map does not has Stability Island, it was chosen as a chaotic map to generate a sequence of pseudorandom bits in this algorithm. The tent map was scaled to [0, 2^n] to have a higher precision and easier computer implementation; this means that, with a bigger value of n, a better encryption approach will be given. This approach encrypts blocks of 64-bit length, which is divided into 4 equal sub-blocks. The encryption process consists of r rounds of chaotic map using encryption key. The result will be assigned to variables $x_{i+1,0}$, $x_{i+1,1}$, $x_{i+1,2}$, $x_{i+1,3}$. The result of the proposed chaotic pseudorandom number generator was tested using NIST statistical test suite, and it confirmed its randomness by passing all the tests.

3.5 Implementation Issues

Implementing chaotic systems on digital computers using finite-precision may cause dynamical degradation as the result of implementing dynamical system on a computer is not exactly the same with the theoretical one [264]. Therefore, chaotic systems' properties may become non-ideal as the cycle length is shorter and degrades orbit distribution and correlation [236, 275, 276]. Consideration of this issue is very important for the performance and security of the chaotic cryptography. This issue has been studied intensively in the last two decades and many different solutions have been proposed in the literature [59, 118, 120, 259, 270, 275, 277-283]. The three main solutions were proposed by several researchers using higher precisions [59, 118, 120, 282, 283], cascading multiple chaotic maps [275, 281], and random perturbation of the chaotic systems [278-280]. The GNU multiple-precision arithmetic library (GMP) is used with chaotic cryptography systems to solve the deterioration problem by providing higher precisions [284, 285]. The perturbation of system variables is found to be better than the perturbation of control parameters, which is performed by generating pseudorandom sequences and then XORing with variable parameters every iteration to increase the cycle length or using other mask operations [259]. In order to obtain a longer periodic, we suggest using two chaotic maps that are independent of the initial values and the control parameters, and then XORing the results of the two systems. Moreover, cascading multiple chaotic maps could affect the performance of the cryptographic systems and would not be appropriate for many applications.

The two most important requirements of any cryptographic system are high security and low implementation cost. Therefore, a well-designed cryptographic system is one with simpler hardware and software implementation at low cost with very high security. A secure cryptography system with very high implementation cost could be easily achieved with a combination of very complicated mathematical equations, but many applications require a high-security cryptography system at low implementation cost. The following issues should be considered in designing and implementing chaotic cryptography systems:

1. A lower implementation cost is achieved by selecting a simpler chaotic system.
2. Chaotic system parameters should be carefully chosen with a large positive Lyapunov exponent and no stability islands.
3. Using one or two simple operations in chaotic cryptography systems could be useful to break the systems more easily.
4. A well-designed cryptography system provides high security and fast computational speed.
5. Parallel implementation will be useful to implement complex chaotic systems that provide fast and secure cryptography systems for many applications such as e-commerce and online banking.
6. Higher precisions, cascading multiple chaotic maps, or random perturbation of chaotic systems could be used to solve the digital degradation problem.

3.6 Summary

Chaos theory has attracted the cryptography field due to its characteristics, such as deterministic nature, sensitivity to initial conditions, unpredictability, and complex structure. In recent years, several cryptographic systems based on chaotic systems have been proposed, such as cipher encryption algorithms, hash functions, and pseudorandom number generators. Unfortunately, some of the proposed algorithms suffer from security problems or slow performance. The main challenge in this research area is to design a secure and fast chaos-based cryptography algorithm. In this chapter, we gave an overview of chaos theory and presented a literature review of chaotic cryptography. We explained in detail about chaos theory, chaotic maps and Lyapunov exponent. Moreover, we explained the relationship between chaotic systems and cryptography algorithms. In the rest of this chapter, we briefly reviewed the literature and past efforts to utilize chaotic maps in cryptography to design cipher encryption algorithms, cryptography hash functions, and cryptography pseudorandom number generators. Finally, we discussed some implementation issues.

To conclude, the existing work on chaos-based cryptography by other researchers is still suffering from security and performance problems due to the incorrect choice of chaotic system, chaotic system stability, simplicity of design at the expense of security, and complexity of design at the expense of performance. Therefore, there is still room for further improvement by finding new chaotic cryptography algorithms with high security and performance. Additional advanced research is needed on new chaotic cryptography algorithms, especially block ciphers, hash functions, and pseudorandom number generators for e-commerce and other applications. In this chapter, we explored the literature on chaotic cryptography algorithms. In the next chapters, we propose novel chaotic cryptography algorithms, including chaotic block cipher, hash function, and chaotic pseudorandom number generator, which handle performance and security problems. Moreover, we analyze the security of one of the proposed chaotic hash functions and other chaotic pseudorandom number generators.

Chapter 4

Cryptanalysis of Chaos-based Hash Function (CBHF)

In 2009, Amin et al. proposed a new hash function based on chaotic systems for cryptography applications (CBHF). In the same year (2009), we published a paper that describes how to break keyed and unkeyed versions of Amin et al.'s hash function. In this chapter, we show how to break both keyed and unkeyed versions of CBHF hash function in detail with two real collision examples.

4.1 Introduction

In 2009, Amin et al. suggested a new hash function based on chaotic systems for cryptography applications (CBHF). In the same year (2009), we published a paper that describes how to break keyed and unkeyed versions of Amin et al.'s hash function [137]. One year later (2010), another research group analyzed the security of the unkeyed version of Amin et al.'s hash function [144]. In this chapter, we show how to break both keyed and unkeyed versions of CBHF hash function in detail with two real collision examples.

In this chapter, we analyze the security of the recently proposed hash function (CBHF). The rest of this paper is organized as follows: Section 4.2 describes details of the CBHF hash function. Section 4.3 discusses details of our analysis of CBHF hash function. Section 4.4 describes Xaio et al.'s analysis of CBHF hash function.

Section 4.5 comments on future research on hash function. Finally, the conclusion of this chapter is provided in section 4.6.

4.2 Details Chaos-based Hash Function (CBHF)

In 2009 [80], Amin et al. proposed a new chaos-based hash function for cryptography applications. They suggested a simple design of unkeyed hash function based on a well-known chaotic tent map. They claimed that they could construct a keyed version of the proposed hash function (CBHF) by treating the initial values in the unkeyed hash function (CBHF) as the secret key. The general design structure of the proposed hash function is illustrated in Figure 4-1. Amin et al. explained the general idea of their hash function in one paragraph without giving much detail on how it works. The algorithm's general description shows that the design is mainly based on tent map and XOR operations. In general, any hash function mainly based on XOR operation is not secure at all. A well-designed hash function should be collision-free and public, where the security of the hash function should not depend on hiding the details [8].

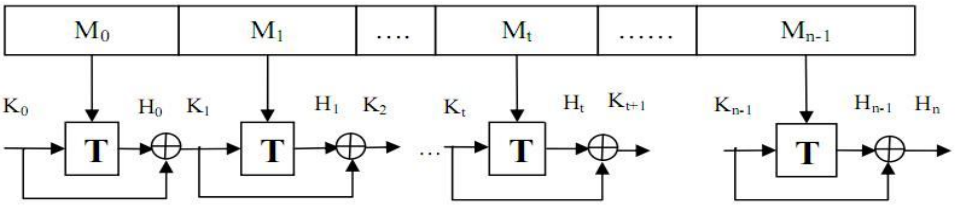


Figure 4-1: Overview of CBHF structure

A tent map is a simple one-dimensional map that generates periodic chaotic behaviour similar to a logistic map; it is governed by equation 4-1. It is an iterated dynamical system function that exhibits chaotic behaviours (orbits).

$$T_r(x) = \begin{cases} rx & , \ 0 \leq x < 1/2 \\ r(1-x), & 1/2 < x \leq 1 \end{cases} \tag{4-1}$$

where r is the control parameter and x is the initial condition, $r \in [0,2]$and $x \in [0,1]$.

The proposed hash function works by dividing the input message into 1024-bit blocks $M_i, 0 \leq i \leq n-1$. If the last message block size M_{n-1} is less than 1024 bits, it will be padded by adding a single one bit followed by the necessary number of zero bits $(100\ldots0)_2$. Then, the value of the secret key is assigned either to the tent map's control parameter (r) or to the initial condition (x), and then the message blocks are assigned to the other variable. Finally, all message blocks are encoded and the final hash value (H_n) is generated (see equation 4-2). The core function of the proposed algorithm is shown in figure 4-2.

$$H_n = K_0 \oplus H_1 \oplus H_2 \oplus \ldots \oplus H_{n-1} \tag{4-2}$$

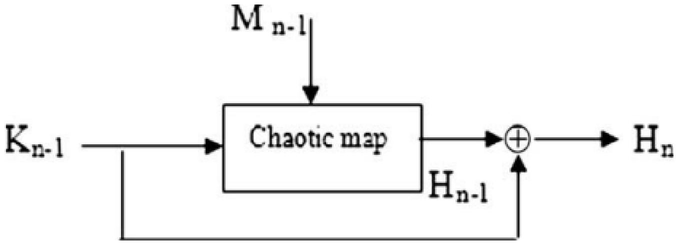


Figure 4-2: Core of chaotic hash function

4.3 Collision Analysis

The tent map has two input variables: the control parameter (x) and the initial condition (r). One of them is assigned by intermediate value K_i and the other is assigned by message block M_i. It is not clear from the paper which one corresponds to x and which one to r so, in our analysis, we shall work the two cases separately. In any case, we work under the following assumptions:

1. The initial value K_0 is public as CBHF is unkeyed hash function.
2. We will study the two cases:
   a) $H_i = T_{M_i}(K_i)$

b) $H_i = T_{k_i}(M_i)$

$for\, 0 \leq i \leq n-1$ as it was explained above.

3. $K_i = K_{i-1} \oplus H_{i-1}\, for\, 1 \leq i \leq n-1$ and inally $H_n = K_{n-1} \oplus H_{n-1}$. Thus, the final hash value is calculated using equation 4-2.

Now, we will show how to break CBHF in a very strong sense. Indeed, given any (partial) message$M'_0M'_1 \ldots \ldots M'_{n-2}$, we can compute the last block M'_{n-1} so that the hash value of M' is the same as the hash value of M. For this end, we first calculate $\mathrm{K}'_{\mathrm{n}-1}$ which depends from $M'_0M'_1 \ldots \ldots M'_{n-2}$, only. Then, we want to have the last block M'_{n-1} satisfying equation 4-3.

$$H_n = K'_{n-1} \oplus H'_{n-1} \qquad (4\text{-}3)$$

Now, both H_n and K'_{n-1} are known, so H'_{n-1} is calculated by the following equation:

$$H'_{n-1} = K'_{n-1} \oplus H_n \qquad (4\text{-}4)$$

We know that H'_{n-1} is obtained by one of the following two ways, which we consider separately.

1. $H'_{n-1} = T_{M'_{n-1}}(K'_{n-1}) = M'_{n-1}K$,

where K (which either K'_{n-1}or $1 - K'_{n-1}$) is known, so we can solve it for M'_{n-1} to get:

$$M'_{n-1} = \frac{H'_{n-1}}{K} \qquad (4\text{- }1)$$

2. $H'_{n-1} = T_{K'_{n-1}}(M'_{n-1}) = K'_{n-1}M$,

where M is either M'_{n-1} or $1 - M'_{n-1}$. In any case, we can solve it for M to get

$$M = \frac{H'_{n-1}}{K'_{n-1}} \qquad (4\text{- }2)$$

and then calculate the value of M'_{n-1}based on equation 4-7.

$$M'_{n-1} = \begin{cases} \mathrm{M} & , \& M \le 1/2 \\ 1-\mathrm{M}, & M > 1/2 \end{cases} \quad (4\text{-}3)$$

Now we will give real collision examples for the unkeyed and keyed versions of the CBHF that have been calculated based on our analysis. The collisions were analyzed on a Laptop Intel Core(TM) 2 Duo 2.00 GHz CPU with 3 GB RAM running on Linux Operating System, using GMP (GNU Multi-Precision Library). The following example shows a real collision attack on the unkeyed version of the proposed hash function (CBHF) as follows:

Example 1: In the case of unkeyed hash function we will suppose the initial values of M_0 and K_0 are equal to the value of the first block sequence; also, when we compute the final hash value we will take the first 128-bit of H_n to be the final hash value. In this case, we select randomly the first sequence as message *M* and then we calculate its hash value based on CBHF hash function. Then, we select randomly another (partial) message $M'_0M'_1 \dots\dots M'_{n-2}$ and based on our analysis we calculate the value of the last block M'_{n-1} to get construct the second message that has the same hash value of the original message *M*, where $H_M = H_{M'}$. Both messages (*M and M')* and the correspondence hash values are given in hexadecimal representation.

Now, we show how to break the keyed version of CBHF. In this setting, the initial value K_0 is the unknown key. However, the final hash value H_n (of the original message $M = M_0 M_1 ... M_{n-1}$) is, of course, known. Therefore, we can append any new part to the end of the original message *M*, say $M' = M_nM_{n+1} \dots\dots M_{n+m-2}$ and then use the procedure above to determine the value of the last block M_{n+m-1}. Thus, the hash value of the new message $MM'M_{n+m-1}$ and the original message *M* are equal, $H_{n+m} = H_n$. In the following example, we will show a real collision on the attack keyed version of the hash function (CBHF).

- Sequence number one (*M*):

```
49 6E 20 74 68 65 20 6C 61 73 74 20 73 65 76 65 72 61 6C 20 79 65 61 72 73 20 72
65 73 65 61 72 63 68 65 72 73 20 75 74 69 6C 69 7A 65 64 20 63 68 61 6F 73 20 74
68 65 6F 72 79 20 69 6E 20 63 72 79 70 74 6F 67 72 61 70 68 79 20 1C 66 69 65 6C
64 20 20 64 75 65 20 74 6F 20 69 74 20 63 68 61 72 61 63 74 65 72 69 73 74 69 63
73 2C 20 73 75 63 68 20 61 73 20 73 65 6E 73 69 74 69 76 69
```

- Sequence number two (*M*'):

```
54 68 65 20 67 65 6E 65 72 61 6C 20 64 65 73 63 72 69 70 74 69 6F 6E 20 73 68 6F
77 73 20 74 68 61 74 20 74 68 65 20 64 65 73 69 67 6E 20 69 73 20 62 61 73 65 64
20 6D 61 69 6E 6C 79 20 6F 6E 20 74 65 6E 74 20 6D 61 70 20 61 6E 64 20 65 78
63 6C 75 73 69 76 65 20 6F 72 20 28 64 65 6E 6F 74 65 74 20 62 79 20 58 4F 52 29
20 68 61 73 68 20 6F 70 65 72 61 74 69 6F 6E 73 2E 2E 2E 2E 2E F1 98 AF DE 1D
A8 A3 70 33 DF D5 5A 6A DA B5 ED 94 65 01 21 95 05 B3 34 53 DD C9 C2 BA
FE F8 55 A9 E2 03 06 83 B1 74 50 6D B9 66 8E 7C 66 B0 E0 A7 74 65 CD 78 61
86 9F D2 E2 6E 91 A0 34 B2 AB F4 D3 43 44 54 7E 66 A0 22 15 D8 0B A5 AB BC
47 C5 2C E4 F8 E5 4E 88 59 B1 D6 BF 80 4E 21 FB B0 22 8A DE 94 29 7E AB 78
0D 13 E1 24 F8 71 B7 DD 10 A7 5F B9 8B 80 8E 50 28 22 ED 3E 94 0D 61 50
```

- Both produce CBHF Hash Value:

```
6C B3 05 F6 FE 9F 9B 9F DF 45 F0 90 CE 0F 5D 7E
```

Example 2: In the case of keyed hash function, we can add two or more blocks at the end of the original sequence without changing the final hash value. In this case, we can find collisions easily without knowing the used secret key. Messages (*M and M'),* secret key, and the correspondence hash values are given in hexadecimal representation.

- Sequence number one (M):

```
49 6E 20 74 68 65 20 6C 61 73 74 20 73 65 76 65 72 61 6C 20 79 65 61 72 73 20 72
65 73 65 61 72 63 68 65 72 73 20 75 74 69 6C 69 7A 65 64 20 63 68 61 6F 73 20 74
68 65 6F 72 79 20 69 6E 20 63 72 79 70 74 6F 67 72 61 70 68 79 20 1C 66 69 65 6C
64 20 20 64 75 65 20 74 6F 20 69 74 20 63 68 61 72 61 63 74 65 72 69 73 74 69 63
73 2C 20 73 75 63 68 20 61 73 20 73 65 6E 73 69 74 69 76 69
```

- Used Key (K):

```
4D 61 6E 79 20 6F 66 20 74 68 65 20 68 61 73 68 20 66 75 6E 63 74 69 6F 6E 73 20 67 65
6E 65 72 61 74 65 20 74 68 65 20 6D 65 73 73 61 67 65 20 64 69 67 65 73 74 20 74 68 72
6F 75 67 68 20 61 20 72 61 6E 64 6F 6D 69 7A 69 6E 67 20 70 72 6F 63 65 73 73 20 6F 66
20 74 68 65 20 6F 72 69 67 69 6E 61 6C 20 6D 65 73 73 61 67 65 2C 20 62 65 66 6F 72 65
20 6D 6F 64 69 66 69 63 61 74 69 6F
```

- Sequence number two (M'):

```
49 6E 20 74 68 65 20 6C 61 73 74 20 73 65 76 65 72 61 6C 20 79 65 61 72 73 20 72
65 73 65 61 72 63 68 65 72 73 20 75 74 69 6C 69 7A 65 64 20 63 68 61 6F 73 20 74
68 65 6F 72 79 20 69 6E 20 63 72 79 70 74 6F 67 72 61 70 68 79 20 1C 66 69 65 6C
64 20 20 64 75 65 20 74 6F 20 69 74 20 63 68 61 72 61 63 74 65 72 69 73 74 69 63
73 2C 20 73 75 63 68 20 61 73 20 73 65 6E 73 69 74 69 76 69 54 68 65 20 67 65 6E
65 72 61 6C 20 64 65 73 63 72 69 70 74 69 6F 6E 20 73 68 6F 77 73 20 74 68 61 74
20 74 68 65 20 64 65 73 69 67 6E 20 69 73 20 62 61 73 65 64 20 6D 61 69 6E 6C 79
20 6F 6E 20 74 65 6E 74 20 6D 61 70 20 61 6E 64 20 65 78 63 6C 75 73 69 76 65
20 6F 72 20 28 64 65 6E 6F 74 65 74 20 62 79 20 58 4F 52 29 20 68 61 73 68 20 6F
70 65 72 61 74 69 6F 6E 73 2E 2E 2E 2E 2E 69 00 91 5A 25 78 55 C2 96 7F BE 8C
B2 3F B4 18 91 DE CE 35 78 97 A6 3F 44 73 0C 63 14 EB 37 02 1A 5A 99 B6 81
37 60 06 2F DD 10 99 3E B7 E8 36 E3 02 58 7B 7B B1 82 FD 35 E9 AF 4D 4E 9E
70 6A 81 8B 6F 4B 26 86 B5 0D EA A9 5F 27 57 ED 79 15 B9 2B 34 B2 99 4E 33
73 BB 89 E0 0F 19 24 86 C7 21 67 FB E7 B1 62 50 E8 A7 D2 D5 7F 69 D3 B8 F3
05 44 49 61 CC A1 43 8C E6 AA CC 78 9B 96 88
```

- Both produce CBHF Hash Value:

```
6A 87 F0 C3 E2 E6 DC 55 4F C3 9F 8A 12 D8 1A 5D
```

4.4 Xiao et al.'s Analysis

We published the result of our analysis a few months after the (CBHF) hash function was published [137]. In 2010 [144], Xiao et al. published a paper that describes how to attack the unkeyed version of Amin et al.'s hash function. Xiao et al. pointed out that the details of the proposed hash function are not clear and they explained how it works in general. They claimed that they would be able to find two different messages as the first two sequences are not equal and the rest of the message sequences are equal, such as $M = (a, z, p_3, \ldots p_{n-1}, p_n)$, ($M' = b, y, p_3, \ldots p_{n-1}, p_n$), p_i represent an 8-bit character, a ≠ b, and z ≠ y. The intermediate hash value is calculated based on a tent map for each 8-bit and then it maps to "0" if it is less than 0.5, otherwise it maps to "1". Moreover, they used the initial condition $x = 0.3$ and the control parameter $r = 1.8$ to clarify the attack.

The first two orbit values of the first message M are as follows:

$x_1 = T^{97}(0.3) = 0.2635$,

$x_2 = T^{122}(0.2635) = 0.8645$.

The first two orbit values for the second message M' are as follows:

$x_1' = T^{98}(0.3) = 0.4743$,

$x_2' = T^{121}(0.4743) = 0.8645$.

Since the current orbit initial value of the first two sequences of the two messages are equal and the rest of the message sequences are the same, the subsequent intermediate hash value and final hash value for both messages would be the same, $H_1 = H_1', H_2 = H_2', \ldots H_n = H_n'$. Therefore, even though the first two sequences of the messages are not equal they were able to find collisions. In addition, they pointed out that the padding process of adding one “1” followed by a number of zeros “0” $(100…0)_2$ could help to find collisions. Our analysis applies for two the descriptions of the CBHF hash function.

4.5 Remarks on Chaos-based Hash Function Future Research

Some important issues should be considered in future for designing new hash functions based on chaotic maps based on CBHF analyses. In general, a well-designed chaotic hash function should satisfy the cryptography requirements and characteristics such as high security and speed. The CBHF hash function and some other proposed chaotic hash functions were found to be insecure or slow algorithms. Therefore, advanced research on chaotic hash functions is needed to design a new family of secure hash functions for cryptography and e-commerce applications. Remarks on future research into chaotic hash functions are listed below:

1. Security and performance of chaos-based unkeyed hash function mainly depend on the chaotic system and the algorithm’s internal structure.

2. Security of chaos-based keyed hash function mainly depends on chaotic system, algorithm internal structure and secret key.

3. Cascading multiple chaotic systems could provide highly secure hash function, but at the expense of computation time. Small computational cost can be achieved by selecting a simple chaotic system.

4. A padding process that add one “1” followed by a number of zeros “0” $(100…0)_2$ could help attackers to find collisions; this could be resolved by padding the length of the input message at the end of the last block.

5. In chaos-based hash function, using a single simple operation such as XOR operation with chaotic systems as a mixture could help attackers to break the system.

4.6 Conclusion

In conclusion, we have shown in strong sense how to break the recently proposed unkeyed hash function based on chaos theory (CBHF). Our attack shows that we can easily find two different messages that have the same hash value. In addition, we have shown how to break the keyed version of CBHF by adding different blocks at the end of the original message without changing the final hash value. We can find a large number of collisions for each message using our analysis technique. One year later, in 2010, Xiao et al. analyzed the security of the unkeyed version of the proposed algorithm. Therefore, neither keyed nor unkeyed hash versions of CBHF are at all secure. Advanced research on chaotic hash function is needed to design a new family of secure hash functions for cryptography and e-commerce applications.

Chapter 5

Fast Parallel Hash Functions Based on Chaotic Maps for eCommerce (PHFC)

In the last decade, various hash functions based on chaotic maps have been proposed. Some of the proposed hash function algorithms have proved to be as insecure or slow speed. In this chapter, we propose novel parallel hash functions based on couple chaotic systems. Hereafter, we will referee to the proposed chaotic hash function in this chapter as PHFC. PHFC consists of four main sub-functions: segmentation and padding, hash round function, subkeys generation, and hash mixing function. It can be implemented in parallel mode to provide fast hashing scheme. Several analyses and computer simulations are applied on PHFC hash function. Moreover, comparisons between the PHFC and other hash functions are presented. The analyses and comparisons confirm that the proposed hash function satisfies the cryptographic hash function characteristics.

5.1 Introduction

Over the last decade, various hash functions based on chaotic systems have been proposed [30, 70, 76, 81, 87, 88, 286, 287]. Some of the proposed chaotic hash functions have been proved as insecure or slow speed algorithms [87, 137, 140, 143, 144]. It would be difficult based on the iteration structure of the hash function to construct a parallel hash function to have faster speed [140, 143]. Therefore, the iteration structure of the hash function needs to be changed to construct a parallel hash function (see equations 5-1 and 5-2). Moreover, in the recent years, most of the personal computers manufactured with 2-core processors. Consequently, designing parallel hash function helps to exploit the 2-core processers more efficiently and construct a faster chaotic hash function. In this chapter, we propose a parallel hash function based on chaotic maps.

The rest of this chapter is organized as follows: Section 5.2 describes details of the proposed keyed hash function. A discussion on performance analysis is presented in section 5.3. Finally, the conclusion is given in section 5.4.

5.2 Parallel Hash Function (PHFC)

The general designs of the sequential and parallel chaotic hash functions are illustrated in Figures 5-1 and 5-2, respectively. In the following subsections, we will explain the details of the proposed hash function.

5.2.1 Message Padding and Segmentation

All the cryptography padding modes can be used, for more details see section 2.8.2. In PHFC, we will use the bit padding mode that is defined in RFC1321 and employed in many cryptography algorithms. The first step in PHFC padding is to partition the input message into *n* number of blocks. The length of each block equal is equal the length of the final hash value *l*. Therefore, the input message should be a multiple of *l*: otherwise, the input message needs to be padded to be a multiple of *l*. Let us assume the input message length is *M* and the hash length *l=160*. Then, the block length is *l=160* and the padded *M* should be a multiple of 160. The message *M* is padded by one $(1)_2$ followed by number of zeros $(00\ldots0)_2$ such that length of padding bits $(100\ldots0)_2$ is *p*, where $(M + p) \bmod l = l\text{-}64$. The last 64 bits are filled up by the length of the input message mod by 2^{64}, $lM = M\ mode\ 2^{64}$.This padding scheme is adopted to prevent collision flaw [144].

$$H_i = h(M_i, H_{i-1}), \qquad i = 1,2,\ldots,\ n-1, n\,, \tag{5-1}$$

$$H_n = h(M_n, H_{i-1}) = h(M_n, h(M_{n-1},\ldots,\ h(M_1, H_0))) \tag{5-2}$$

where *h*() is the round function, M_i is the *ith* message block, H_{i-1} is the intermediate hash value of the previous hash round function, and H_0 is the initial value of the first hash round function.

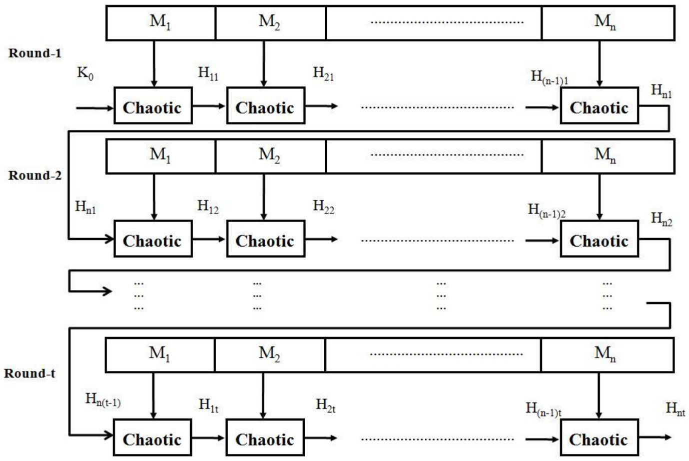


Figure 5-1: Structure of sequential chaotic hash function

5.2.2 Keys Generations

The next step, after partition of the padded message is to generate the rounds' subkeys. In each round one subkey of length *l* is needed, where the minimum number of rounds is defined by the variable *t*, such that $t \geq 8$. Thus, $t+1$ subkeys is needed to be generated using a chaotic map, one subkey for each round and one for the final mixing process, see Figure 5-2. In PHFC, several chaotic maps (Logistic map, Tent map, Skew Tent map, and other PWLCMs) can be used as chaotic systems in subkeys generation function, but we have to consider few important issues such as sensitivity of secret key and subkey space. The chaotic system's initial condition x_0 and the control parameter *r* are used as the secret key of the PHFC. Then, chaotic system is iterated *n* times, where the minimum number of iterations is 50 to prevent predication of generated values, as chaotic system gives unpredictable values in the long-term run. The resulted value (x_{n+1}) is used as the value of the first subkey (K_1), and then the second iteration is applied to generate the second subkey K_2 and so on until all subkeys have been generated.

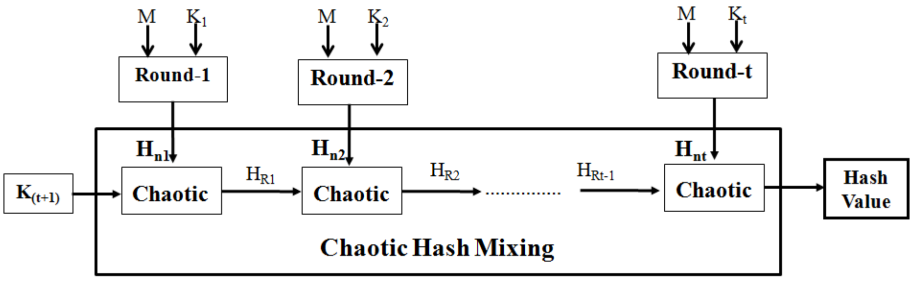


Figure 5-2: Structure of the proposed parallel chaotic hash function

In well-designed hash function, the secret key is sensitive to any change(s) so that resulting two different hash values. Chaotic system is sensitive to the initial conditions and/or the control parameters within the chaotic areas. The non-chaotic areas must be avoided from the secret key space as it not very sensitive to changes. In case the PWLCM, which is defined in equation 5-2, is used as one of the chaotic systems for subkey generation, its initial condition x_0 and the control parameter p are used as the secret key. In the PWLCM, the non-chaotic areas are avoided, since all its control parameters have positive Lyapunov exponent values. Moreover, we must to exclude the $x_n = 0$, $x_n = 0.5$, or $x_n = 1$ from the key space, as the system with these initial parameters will end in the same fixed-point $x_{n+1} = 0$, $f(0, p) = 0, f^3(0.5, p) = 0, f^2(1, p) = 0$ for any $P \in (0, 0.5)$.

5.2.3 Hash Rounds Functions

Hash round functions are processed input messages in parallel mode as shown in Figure 5-2. In each round function, the initial parameters' values of variables x_0 and r are the round secret key K_i and the first message block, respectively (see Figure 5-3). In a round function with a logistic map as a chaotic system, the value of the subkey is assigned to initial condition x_0 and the message block value is added to the chosen value of r such that $r \in (3.9, 4)$. Transform every message block of the input message $M_i = (p_1\ p_2 ... p_n)$ into binary fractions $M_i' = (0.0\ p_1\ p_2 ... p_n)$. The result from the previous message block is set as the initial value of the current block and so on until all the input message blocks have been processed. As the proposed hash function is designed parallel, all hash rounds functions $h()$ process the input message simultaneously.

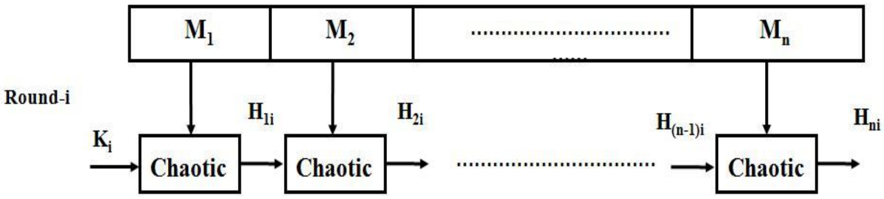


Figure 5-3: Block diagram of round function

5.2.4 Chaotic Hash Mixing

After all rounds have calculated the final intermediate values (H_{n1}, H_{n2},…, H_{nt}), the next step is processing chaotic hash mixing function. In case of using PWLCM as chaotic map in the Hash Mixing function, the value of the first-round intermediate hash H_{n1} is set as the value of parameter r and the secret key K_{t+1} is set as x_0, where $r \in (0, 0.5)$ and $x_i \in (0, 1)$. Then, the PWLCM is iterated to produce the intermediate hash mixing value x_n (H_{R1}), which is used as the initial condition to mix the next round intermediate hash (H_{n1}), and so on until all the intermediate hash values are mixed. The final hash value of the input message is the final resulted value form chaotic hash mixing function based on the all intermediate hash values (H_{n1}, H_{n2},…, H_{nt}), as shown in Figure 5-2.

5.3 Security and Performance Analyses

In this section, the performance and statistical analyses are studied on the calculated hash values based on PHFC hash function. The logistic map is used as the chaotic maps with r=3.999999 and x=0.50, the length of hash value is 160-bit, number of initial iterations is 50, and the number of rounds is 8.

5.3.1 Sensitivity of Hash Values

In order to investigate the sensitivity of generated hash value based on PHFC hash function to the input message, we accomplished 10 different hash simulations tests on the following message: "*A cryptographic hash function is a deterministic procedure that takes an arbitrary length block of data as input and produces fixed-length bits of string as an output.*" [288], see appendix C. The simulation result shows that PHFC are very sensitive to any slight (bit or bits) changes in the input message or the used key and will cause significant changes in the final hash value. The binary hash values under the 10 different conditions are shown in Figure 5-4.

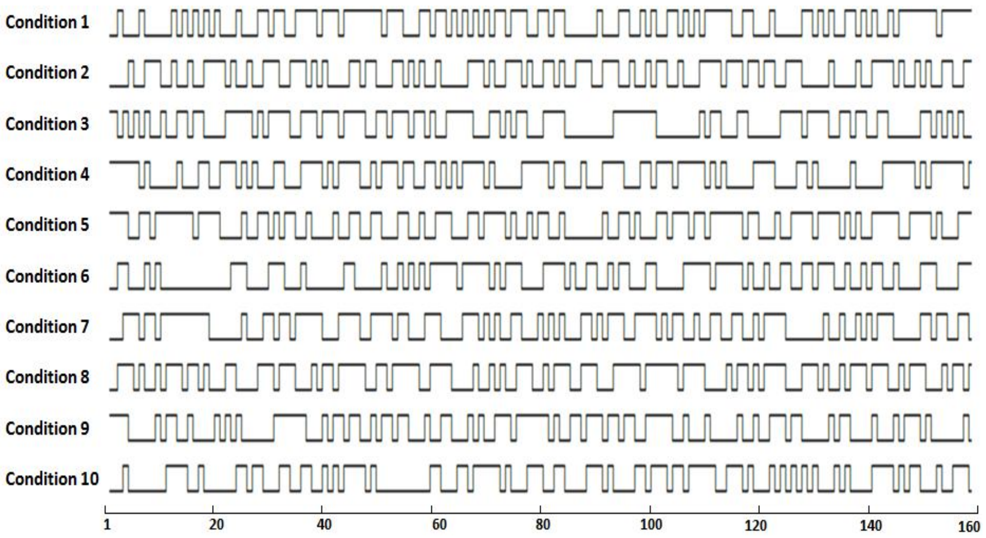


Figure 5-4: Calculated hash values under different conditions

5.3.2 Statistical Analysis of Diffusion and Confusion

Cryptographic algorithms are designed based on the confusion and diffusion properties [216]. Hash functions are similar to the conventional encryption methods, which require the spreading influence of the full input message to be spread into the hash value space. In an ideal hash function, the relationship between input message and correspondence hash value bits must be complex. Therefore, any bit change in the input message should affect at least half of the bits in the hash value, and each bit has a 50% changing probability. A message is chosen randomly and then the correspondence hash value is calculated. Subsequently, one bit is selected and toggled randomly to generate a new hash value in binary format. Afterward, we compare the two hash values (original and new) and count the number of changed bits at the same locations B_i. This test is performed 3000 times and the correspondence distributions of the numbers of bits changes are shown in Figure 5-5.

Similar to [70, 87, 286, 287], the following are six statistics for the proposed hash function algorithm:

Minimum bit number changed:

$$B_{\min} = \min(\{B_i\}_1^N) \ ; \tag{5-3}$$

Maximum bit number changed:

$$B_{\max} = \max(\{B_i\}_1^N) \, ; \tag{5-4}$$

Mean bit number changed:

$$\bar{B} = \frac{1}{N}\sum\nolimits_1^N B_i \ ; \tag{5-5}$$

Mean probability changed:

$$P = (\frac{\bar{B}}{HashSize}) \times 100\% \, ; \tag{5-6}$$

Standard deviation of the changed bit number:

$$\Delta B = \sqrt{\frac{1}{N-1}\sum\nolimits_{i=1}^{N}(B_i - \bar{B})^2} \ ; \tag{5-7}$$

Standard deviation:

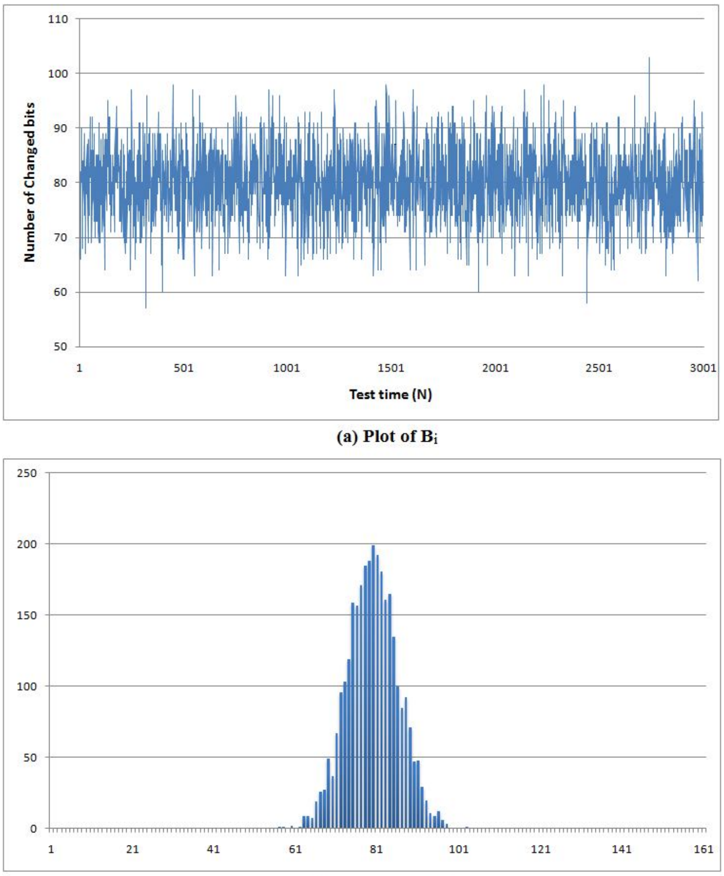
$$\Delta P = \sqrt{\frac{1}{N-1}\sum\nolimits_{i=1}^{N}(B_i / HashSize - P)^2} \times 100\% \ ; \tag{5-8}$$

where N is the total number of statistics, B_i is the number of changed bits in the *i*th test, and ΔB and ΔP indicate the stability of diffusion and confusion.

Table 5-1 shows the statistical results for different numbers of tests N = 3072, 2048, and 1024. From Table 5-1, the mean changed number of bit and the mean changed bit probability are 80.15 and %50.09, respectively, which is very close to the ideal values 80 and 50%. Meanwhile ΔB and ΔP indicate the stability of diffusion and confusion and show that PHFC algorithm is very stable. In conclusion, this analysis shows that the proposed hash function is a secure enough against statistical attacks and it will be very difficult to be attack it.

Table 5-1: Statistic of number of changed bit Bi

	N = 3072	N = 2048	N = 1024	Mean
$\bar{B}$	80.00813802	80.01611328	80.44238281	80.16
P (%)	50.00508626	50.0100708	50.27648926	50.10
ΔB	6.348566433	6.257558918	6.05954492	6.22
ΔP (%)	3.967854021	3.910974323	3.787215575	3.89
Bmax	102	102	98	100.67
Bmin	55	57	62	58



(a) Plot of B_i

(b) Histrogram of B_i

Figure 5-5: Distribution of changed bit number B_i

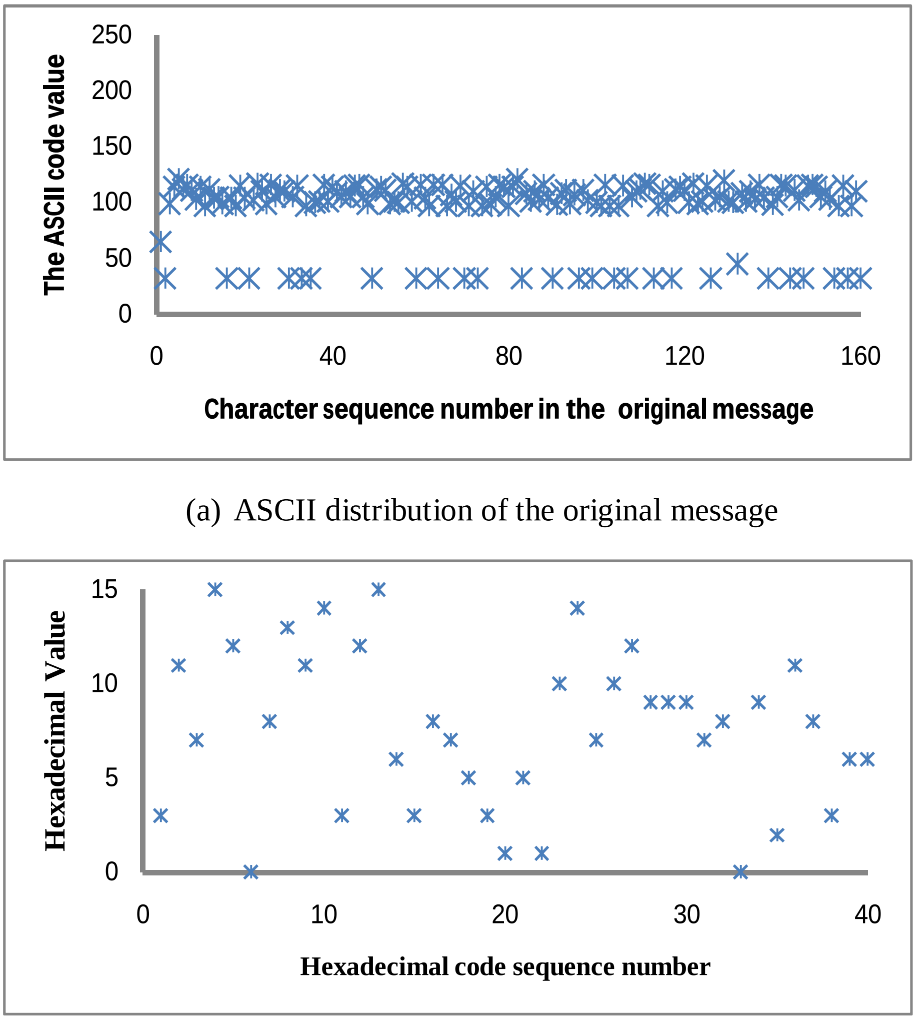
The proposed hash function algorithm is compared with the typical traditional hash function algorithm (SHA-1) and other two chaotic hash functions algorithms (Ref. [287] and Ref. [74]). These hash function algorithms generate the same length of hash value (160-bit). Table 5-2 shows the statistical results of the comparison with a value of N = 2048. From Table 5-2, it is very clear that the means of bit number changes and probability changes are very close to the ideal values, 80 and 50%, respectively. Meanwhile ΔB and ΔP are very small that indicating that the diffusion and confusion capabilities are very stables. In conclusion, the proposed hash function statistical performance is very close to the ideal performance and resistant to against statistical attacks.

5.3.3 Hash Value Distribution

One of the most important conditions in hash functions security is the uniformity distribution [81]. The uniform distribution test is performed on the following randomly selected sentence: "*A cryptographic hash function is a deterministic procedure that takes an arbitrary length block of data as input and produces fixed-length bits of string as an output.*" [288]. Figure 5-6 shows the difference between the ASCII distribution of the original message and the hexadecimal distribution of hash value. The simulation result shows that the ASCII distribution of the original message is focused on very small areas, but the hash value distribution is irregularly distributed and scattered in most areas of the space. Therefore, the simulation shows that it is very difficult for the attackers to find information from the original message after it has been processed through the proposed hash function.

Similar to [87, 286], another hash space uniform distribution is performed to check hash space distribution property. We generated a message randomly and then calculated its hash value. After that, we toggled one bit randomly and calculated the correspondence hash value. Finally, we compared and counted the number of bits toggled in the hash space for the two hash values. This test was performed 3000 times, by fixing the first message, then toggling one bit randomly and calculating the number of toggled bit at the same location in hash space, as shown in Figure 5-7.

The mean of toggled bits is 1501, which is very close to the ideal mean value (1500). Therefore, PHFC is collision-resistant and strong enough against the statistical attacks.



(a) ASCII distribution of the original message

(b) Hash values distribution in hexadecimal format

Figure 5-6: Distribution of original message and hash value

5.3.4 Analysis of Collision Resistance

To investigate PHFC collision resistance capability the following test was performed. First, we selected a message randomly and then calculated its hash value and stored it in ASCII format. Then, we selected and toggled one bit randomly to generate the new hash value and also stored it in ASCII format. After that, the difference between the two ASCII hash values in the same position was calculated. Finally, the summation of all differences for all characters of the two hash values was calculated, which is called absolute difference (d), as shown by Equation 5-11.

Table 5-2: Comparison of Statistics performance

	SHA-1 [81]	Ref. [287]	Ref. [74]	PHFC
$\bar{B}$	79.86	80.04	80.01	80.02
P (%)	49.91	50.02	50.01	50.01
ΔB	6.24	6.18	6.34	6.26
ΔP (%)	3.91	3.86	3.97	3.91
Bmax	60	60	57	57
Bmin	100	100	102	102

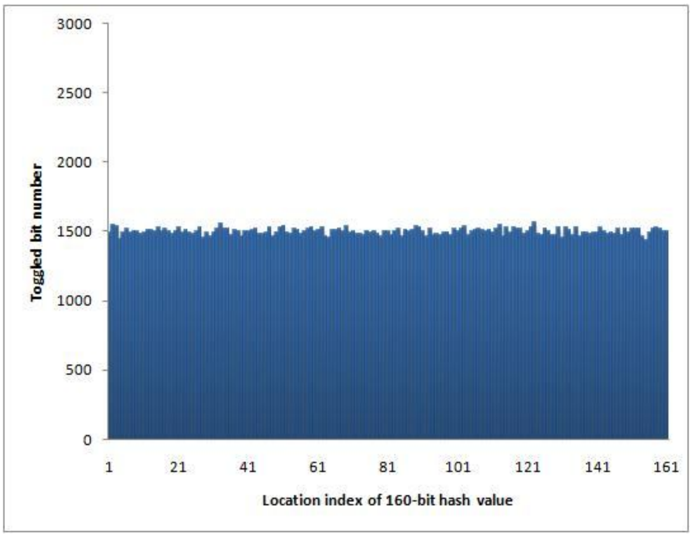


Figure 5-7: Hash value distribution in the hash space with N = 3000 and the mean 1501

$$d = \sum_{i=1}^{N} \left| t(e_i) - t(e`_i) \right| \tag{5-9}$$

$$\omega = \sum_{i=1}^{N} f(t(e_i), t(e`_i)), \quad f(x, y) = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases} \tag{5-10}$$

where e_i and $e`_i$ is the *i*th entry of the original message and *i*th entry of the new message after being toggled, respectively, and t(.) is a function converting the entry to the equivalent decimal value of the ASCII.

This kind of test is performed 3000, 2000, and 1000 times and the maximum, minimum, mean and mean/character for each case are calculated. Figure 5-8 shows the distribution of the number of ASCII characters with the same value at the same location in the hash value with number of tests being 3000, 2000, and 1000 (see equation 5-12). The maximum number of equal ASCII characters in the same location is only two. Therefore, the possibility of collision is very low meaning that most of the entries are different in the ASCII format. The result of this test is compared with other six other different hash function algorithms (see Table 5-3). The comparison result shows that the proposed hash function's absolute differences are very close to the two typical traditional hash function algorithms (MD5 and SHA-1).

Table 5-3: Absolute differences of two hash values

	Maximum	**Minimum**	**Mean**	**Mean/C**
Xiao's et al. scheme [70]	2221	696	1506	94.12
Zhang's et al. scheme [87]	2022	565	1257	78.56
Akhavan's et al. scheme [286]	2431	744	1371	85.68
Akhshani's et al. scheme [287]	2731	866	1706	85.30
MD5 [74]	2074	590	1304	81.5
SHA-1 [287]	2730	795	1603	80.15
PHFC	2602	775	1600	80.02

5.3.5 Number of Hash Rounds

The number of hash rounds (R) was not set as a part of the secret key to avoid timing-attack based on calculating the elapsed time in processing one block of data. It was chosen a very carefully by selecting the input message and the secret key randomly, and then the hash value is calculated using the PHFC with different numbers of rounds. For the resulted hash value, the confusion and the diffusion properties were tested, as it is explained in details in section 5.3.2. We found that the proposed algorithm has high confusion and diffusion properties with number of rounds equal eight.

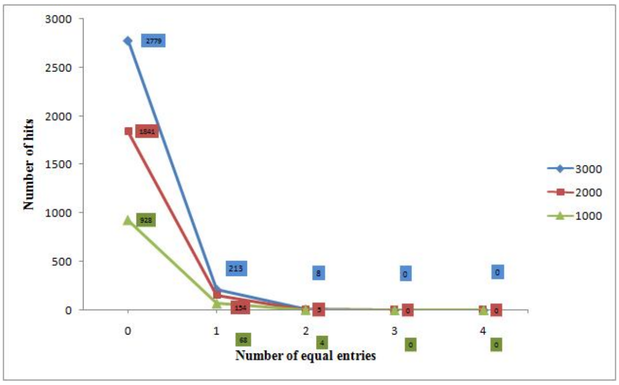


Figure 5-6: Distribution of number of ASCII characters with the same value at the same location in the hash value with number of tests 3000, 2000, and 1000

5.3.6 Speed Analysis

Chaotic hash functions are based on a complex system and typical traditional hash functions (SHA-1 and MD5) are based on simple mathematical operations. Therefore, some of the proposed chaotic hash functions are slower than the typical traditional hash functions. Several speed tests are performed on sequential and parallel modes of PHFC hash functions with different input messages lengths. The experiment's result shows that the parallel mode is demonstrated a much higher efficiency than the sequential mode, as shown in Figure 5-9. Moreover, a comparison in execution time is performed between SHA-1, CHA-1, sequential mode of PHFC, and parallel mode of PHFC.

The execution times of the four hash function algorithms with different message lengths are plotted in Figure 5-10. It can be observed from Figure 5-10, SHA-1 and PHFC with parallel mode show very short average execution times are compared with execution times of the CHA-1 and sequential mode of PHFC. Parallel mode of PHFC and SHA-1 average execution times are very short and close to each other, but SHA-1 shows a slightly better performance with a bigger message size. At the same time CHA-1 speed with the same input message length is much faster than the other three algorithms (SHA-1, parallel mode of PHFC, and sequential mode of PHFC).

The speed tests were analyzed on a Laptop Intel Core(TM) 2 Duo 2.00 GHz CPU with 3 GB RAM running on Linux Operating System, using GMP (GNU Multi-Precision Library). Although SHA-1 with a bigger message length is slightly faster than the parallel mode of PHFC in small differences, the parallel mode of PHFC is much better in terms of security than SHA-1 (see Table 5-4). SHA-1 security reduced down to only 2^{53}, while security of the proposed hash function is 2^{80} (hash operations brute-force) [289]. The qualitative analysis between SHA-1, CHA-1, sequential mode of PHFC, and parallel mode of PHFC is described in Table 5-4. In conclusion the parallel mode of the proposed algorithm demonstrated fast speed and high security, thus PHFC would be a good candidate for use in e-commerce applications over broadband networks.

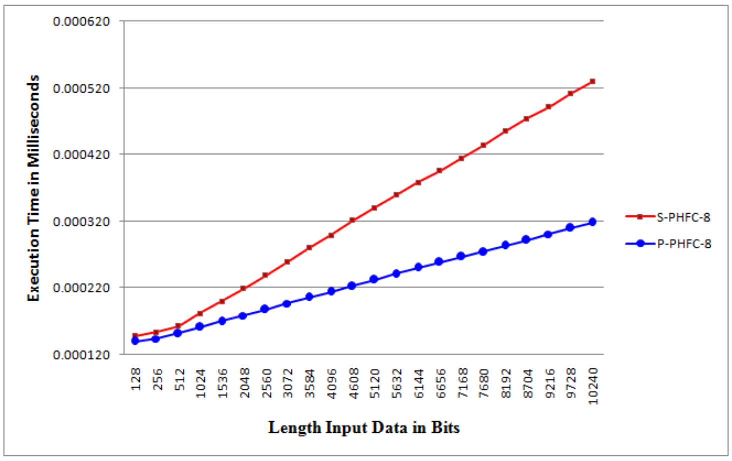


Figure 5-7: Comparison between sequential and parallel versions of PHFC hash algorithm in the execution time (millisecond)

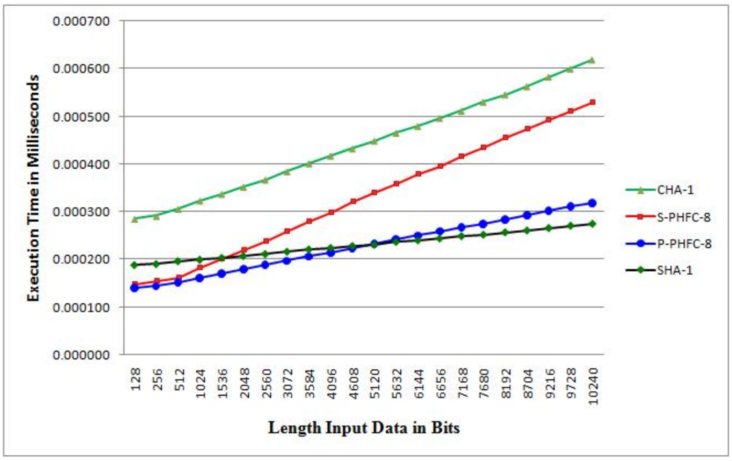


Figure 5-8: Comparison between SHA-1, CHA-1, sequential and parallel version of PHFC with rounds in the execution time (millisecond)

5.3.7 Implementation and Flexibility

In PHFC hash function, the chaotic map is iterated with arbitrary precision floating-point arithmetic. Since GNU multiple precision arithmetic library (GMP) is a free library operating on floating-point numbers, and rational and signed integer number [290], it is suggested that the proposed algorithm is implemented by GMP precision floating-point arithmetic. The GMP is mainly used in algebra systems, Internet security, and cryptography applications and research. GMP is considered as faster big-numbers library than other libraries. It runs over all platforms such as, MAC OS, GNU/Linux, BSD, AIX, and Windows 32-bit and 64-bit mode.

PHFC is a hash function that processes different input message lengths and produces different lengths of final hash value. It was designed as a very fixable hash function algorithm which through simple modifications of the block size, and secret key length it can then produce different lengths of hash values (see Table 5-4). To produce high sensitive hash function to any change(s) in the input message, the entire input message is processed through each hash round function. Moreover, Logistic map, tent map, and PWLCM can be use as chaotic system in one function of the PHFC: subkey generation, round hash function, and chaotic mixing hash function. The proposed hash function produces variable lengths of hash values such as, PHFC-128, PHFC-160, PHFC-224, PHFC-256, PHFC-384, and PHFC-512.

Table 5-4: Comparison between SHA-1, CHA-1, and PHFC Properties

Properties	MD5	SHA-1	CHA-1	PHFC
Message Size (bits)	2^{64}	2^{63}	2^{80}	2^{80}
Block Size (bits)	512	512	160	128, 224,352, 480
Word Size (bits)	32	32	32	8, 16, 32
Message Digest Size	128	160	160	160, 256,384, 512
Security (bits)	2^{24}[291]	2^{51}[289]	2^{80}	$2^{80}, 2^{128}, 2^{192}, 2^{256}$
Found Collisions	**Yes**	**Yes**	**No**	**No**

The proposed hash function provides high security, speed and adaptability comparing with MD5, SHA-1, and CHA-1 hash functions. In addition, different versions of the proposed hash functions are proposed using different chaotic maps, block length and number of rounds. PHFC is aimed to construct keyed hash function based on chaotic systems, meanwhile we can construct unkeyed hash function by processing the input message through one of the chaotic maps and the result is used as the secret key. The proposed hash function is high-potential candidate in many different standards and applications due to its characteristic such as security, flexibility, and performance.

5.4 Conclusion

In this chapter, we proposed a fast parallel chaos-based hash function called PHFC. A couple of chaotic maps with high chaotic behaviour are used in designing PHFC functions: subkey generation, round hash function, and chaotic hash mixing function. PHFC is a very flexible hash function that generates different lengths of hash values using different chaotic maps: PHFC-128, PHFC-160, PHFC-224, PHFC-256, PHFC-384, and PHFC-512. The proposed hash function can be implemented as keyed and unkeyed hash function based on chaotic systems. We compared the proposed hash function with two well-know hash function (SHA-1, MD5) and other five chaos-based hash functions; the comparison results showed that the PHFC is outperform than many other existing hash functions. Several computer simulations and theoretical analysis on PHFC hash function are performed which showed that the it is satisfying the characteristics and conditions of cryptography hash functions such as, collision resistance, high bit confusion and diffusion, uniform distribution, flexibility, and fast speed. PHFC is high-potential for adoption in e-commerce applications and protocols.

Chapter 6

Fast Encryption Algorithm Based on Chaotic Maps for eCommerce (BCCM)

In recent years, many researchers have proposed new block cipher encryption algorithms based on chaotic systems. Some of the proposed chaotic encryption algorithms had major problems such as lack of security against different types of attacks, poor flexibility and slow speed. In this chapter, a novel fast block cipher encryption algorithm based on chaotic maps (BCCM) is proposed with high bit confusion and diffusion, sensitivity and flexibility. BCCM encryption and decryption functions are designed based on product cipher structure, that is similar to design of DES, IDEA, MD5 and SHA-1 algorithms. BCCM encryption and decryption functions use simple non-commutative operations: XOR, addition mod 2^n, subtraction mod 2^n, and w-bit left and right circular shifts. Chaotic key-generation function is used to generate eight subkeys for each round that are also used to generate the values of the rows shifting and the columns mixing to provide high confusion and diffusion properties. Therefore, the encryption and decryption functions architectures are depend on the secret key. Analyses and computer simulations were applied on the BCCM encryption algorithm. Moreover, comparisons between the BCCM and well-known hash functions are presented. BCCM is a high-potential candidate for practical texts encryption, images encryption and e-commerce applications and protocols.

6.1 Introduction

Recently, many researchers have developed chaotic block cipher encryption algorithms by utilizing chaotic maps' characteristics [35, 62, 65-67]. Some of the proposed encryption schemes based on chaotic maps have been found to be insecure or very slow [31]. In the present chapter, a novel block cipher encryption algorithm based on chaotic systems is proposed; it is called BCCM. The rest of this chapter is organized as follows: Section 6.2 gives details of the proposed chaotic block cipher. Experimental results are discussed in section 6.3. Finally, the conclusion is given in section 6.4.

6.2 Details of the Proposed Block Cipher Algorithm (BCCM)

The overview of the BCCM is illustrated in Figure 6-1. BCCM is a simple block cipher algorithm designed based on chaotic systems for texts/images encryption. The proposed algorithm works with variable numbers of round R, word size w, and key length (see Table 6-1). In each round there are eight registers of plaintext as input and eight registers of ciphertext as output of the same size (w-bit). It has two main functions: one for encryption and the second for decryption (see Table 2.) In the following subsections, we will explain the details of the BCCM algorithm functions.

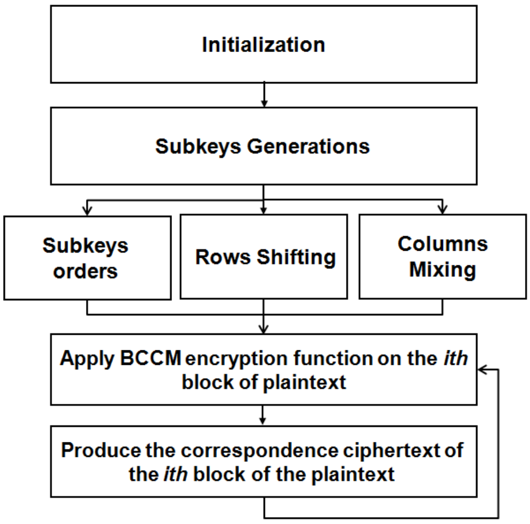

Initialization
Subkeys Generations
Subkeys orders
Rows Shifting
Columns Mixing
Apply BCCM encryption function on the *ith* block of plaintext
Produce the correspondence ciphertext of the *ith* block of the plaintext

Figure 6-1: Overview of BCCM encryption function

Table 6-1: BCCM parameters

Parameter	Meaning	Values Range
R	Number of rounds	1,2,3,........512
W	Word size in bits	8, 16, 32, 64
SK	Secret key in bytes	1,2,3,........512
C	Chaos theory	1D Chaotic maps

Table 6-2: BCCM primitive operations

Notation	Meaning
X + Y	The 2nd complement addition of words
X - Y	The 2nd complement subtraction of words
X >>> Y	Circle right shift Y bits
X <<< Y	Circle left shift Y bits
X ⊕ Y	Bit wise exclusive OR of words
X % Y	The modulus, or remainder, operator

6.2.1 Initialization

In this step we need to choose the chaotic map and then set the values of the initial condition, the control parameter and the number of initial iterations. In BCCM, several chaotic maps (Logistic map, Tent map, Skew Tent map, and other PWLCMs) can be used as chaotic systems in subkeys generation function. The chaotic system initial condition x_0 and the control parameter r within chaotic interval are used as the secret key of the BCCM.

6.2.2 Subkeys Generations

In a well-designed block cipher encryption algorithm, the secret key is sensitive to any change(s). Subkeys generation function is considered the most important function in BCCM algorithm. The generated subkeys values are also used in subkeys orders, shifting rows and mixing columns functions. Subkeys generation is designed based on one of the chaotic system to be very sensitive to any change(s) in the secret key (initial conditions and/or control parameters). The next step after one of the chaotic maps has been chosen as the chaotic system and the values of initial condition, the control parameter and the numbers of initial iterations are set is to apply the Subkeys generation function. The first step in it is iterating the chaotic system *n* times to produce a chaotic sequence and hide any relationship between the initial value and subsequent values, where the minimum number of iterations is 50. Then, the value of x_n of the chosen chaotic map is used as the initial value to generate the next value (x_{n+1}). The binary transformation of the resulting value (x_{n+1}) is used as the value of the first subkey and initial value to the following iteration ($n+2$) and so on until all subkeys have been generated (see the below pseudo-code of subkeys generation).

In the encryption/decryption function each round needs eight different subkeys. Therefore, the total number of subkeys is eight times the number of rounds (8*R). The size of subkeys is equal to the size of sub-block in the encryption function. The algorithm generates eight different subkeys for each round and the same set of subkeys are used twice, at the beginning of each round and at end of each round in different orders. Subkeys orders (SO_i) at the end of each round are calculated based on equation 6-1.

$$SO_i = (i \times SK[R][i] \times R)\%\beta + 1; \tag{6-1}$$

where *i* the subkey index, *R* is rounds number, and β is number of sub-blocks.

Pseudo-code of Subkeys Generation

```
//An example of Subkeys generation with logistic map as chaotic map,
number of initial iterations=50, r=0.59 and $x_0$ =3.95.
secret-key_1=3.95; // within chaotic interval
secret-key_2=0.95; //within chaotic interval
r= secret-key_1;
$x_0$ =secret-key_2;
```

```
n=0;
    for n from 1 to 50 // 50 initial iterations
        do $x_{n+1} = x_n r(1 - x_n)$;
    endfor
//To generate 8×R subkeys
$x_0 = x_{50}$;
R=8;
n=0;
    for R from 1 to 8 // 8 rounds
    for i from 1 to 8  // 8 subkeys for each round
        do  $x_{n+1} = x_n r(1 - x_n)$;
            $SK[R][i] = x_{n+1}$;
    endfor
```

6.2.3 Rows Shifting and Columns Mixing

The values of rows shifting (RS_i) and columns mixing (CM_i) are generated based on the generated subkeys using equations 6-2 and 6-3, respectively. The BCCM algorithm performs left circular shift and right circular shift on a number of sub-blocks in encryption and decryption functions, which we refer to it as rows shifting (RS_i). The number of shifting bits is calculated based on equation 6-2. Furthermore, the last step in each round of the encryption function is mixing and swapping between different sub-blocks to increase output randomness, which we refer to it as columns mixing (CM_i).

$$RS_i = (i \times SK[R][i] \times R)\%L + 1; \qquad (6\text{-}2)$$

$$CM_i = (i \times SK[R][i])\%\beta + 1; \qquad (6\text{-}3)$$

where *i* the subkey index, *R* is round number, L is half sub-block size, and β is number of sub-blocks.

6.2.4 Encryption Process

BCCM encryption function internal design is similar to DES, IDEA, MD5 and SHA-1 algorithms design. The encryption function is designed based on four non-linear operations: addition, exclusive-OR, circle left shift, and circle right shift (see Figure 6-2). In BCCM, after we generate the subkeys, subkeys orders, shifting rows and mixing columns values, the next step is to apply the encryption function. The first step is to divide the input message into *n*-bit blocks after which each block will be further divided into eight sub-blocks. The next step is to encrypt each block of the input message by applying the encryption function R times to generate the correspondence ciphertext. Finally, the encryption process will continue until the entire input message has been processed through the encryption function. In case the last block of the input plaintext is less than the full block, we can use any of the cryptography padding modes that explained in section 2.8.2. The recommended block size for BCCM algorithm is 256. The variable X[i] is the plaintext sub-block number i, Y[i] is the ciphertext sub-block number i, SK[i] is the subkey number i, R is the number of rounds, RS_i is the value of row shifting number i, CM_i is the value of column mixing number i and SO_i is the value of subkey order number i.

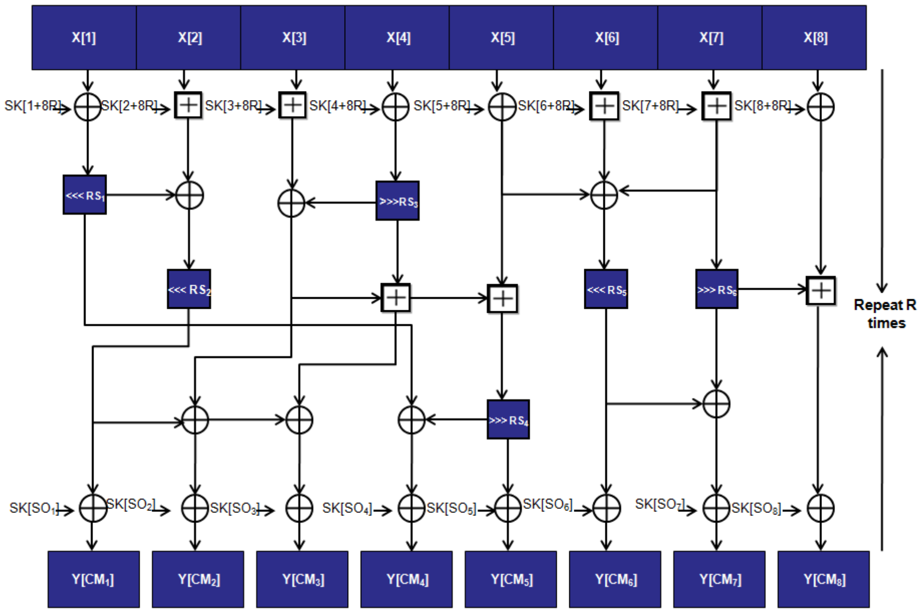


Figure 6-2: BCCM encryption function

6.2.5 Decryption Process

To decrypt the encrypted message using the BCCM encryption function, we need to use the BCCM decryption function and shared secret key. The decryption function is the inverse of the encryption function, using four non-linear operations: subtraction, exclusive-OR, circle left shift, and circle right shift, as shown in Figure 6-3. In BCCM decryption function, we need to use the same selected chaotic map and the same set values of the initial condition, the control parameter and the number of initial iterations in the encryption function. Then, we generate the subkeys, subkeys orders, shifting rows and mixing columns values. Now, we can apply the decryption function. The first step in the decryption function is to divide the input message into *n*-bit blocks after which each block will be further divided into eight sub-blocks. The next step is to decrypt each block of the input message by applying the decryption function R times to generate the correspondence plaintext. Finally, the decryption process will continue until the entire input message has been processed through the decryption function. Any parameters values are used in the encryption function should be the same that will be used in the decryption function.

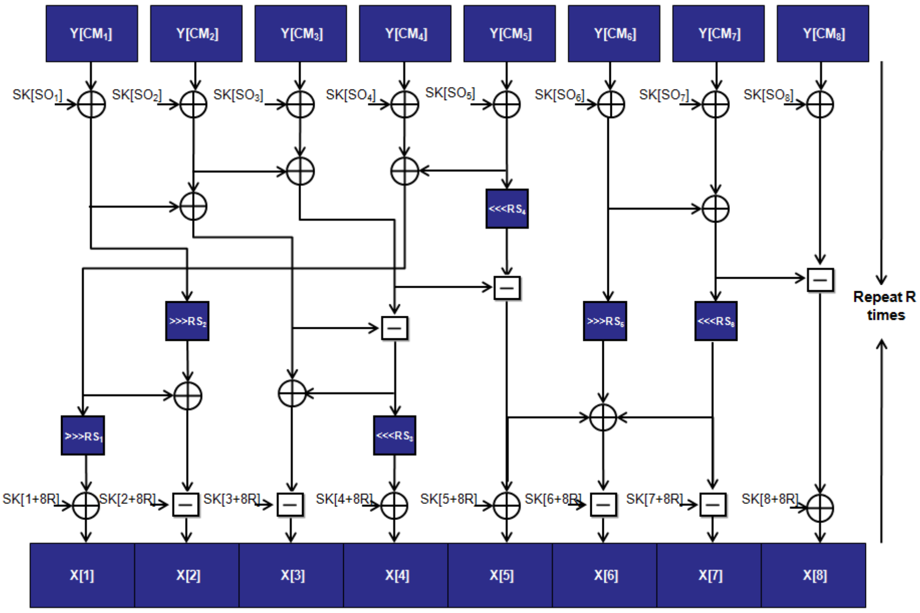


Figure 6-3: BCCM decryption function

6.3 BCCM Parameters and Structure

The number of encryption rounds (R) was not set as part of the secret key in order to avoid a timing-attack based on a calculation of the elapsed time in an encryption/decryption single block of data. It was chosen very carefully by selecting input message and secret key randomly, and then encrypting it using the BCCM with different numbers of rounds. Then, several tests were performed: histogram analysis, correlation coefficient analysis and information entropy analysis, as explained in detail in 6.4.3, 6.4.4, and 6.4.5 subsections, respectively. The proposed algorithm has high confusion and diffusion properties with number of rounds equal to or greater than 4. In case of chosen the Logistic map as the chaotic system the value of initial condition x_0 should be $x_0 \in [0.15, 0.95]$ and $r \in [3.87, 4]$ and in case of chosen the PWLCM as the chaotic system the value of initial condition x_0 should be $x_n \in (0, 1)$, and $P \in (0, 0.5)$, where we must to exclude the some of the x_n values ($x_n \neq 0$, 0.5, and 1) to make sure it would be within the chaotic intervals. The minimum length of the secret key should be 160-bit, which would be very hard to attack using brute-force attack (see section 2.11.1). The recommended block length for BCCM algorithm is 128 or 256, as it has 8 sub-blocks of size 16 or 32 bits.

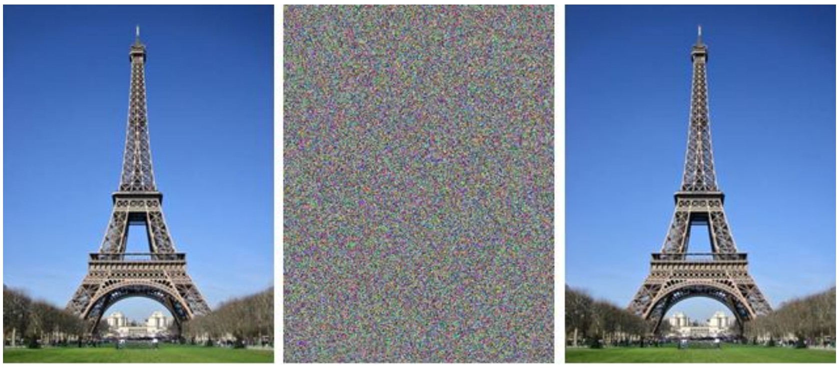
BCCM encryption and decryption functions are designed based on product cipher structure. The product cipher structure is used in the design several cryptography algorithm such as DES, IDEA, MD5 and SHA-1 algorithms. Many brute-force attacks have been proposed for DES algorithm because of using short length secret key (64-bit). Therefore, we proposed to use secret key of 160-bit minimum length to be very hard to attack using brute-force attack. IDEA has very simple key schedule as they derived the subkeys directly from the secret key and using circle left shift, which would be subjective to weak key attack. In BCCM the key generation is based on chaotic system to have unpredictable subkeys as chaotic systems are deterministic, unpredictability and random-look systems with very high sensitivity to initial values. In AES (Rijndael) algorithm the row shifting values are pre-calculated and it would be easier for attacker to track the encryption process rather than using dynamical row shifting. Moreover, SHA-1 and MD5 are using pre-calculated circle shift values that have helped the researchers to attack these algorithms. The row shifting and column mixing in BCCM algorithm are mainly depend on the values of the generated subkeys, therefore different secret key would generate different row shifting and column mixing values and would be very hard to figure-out these values with a large number possibilities without knowing the value of the secret key.

6.4 Experimental Results

In this section, analyses and computer simulations on the BCCM block cipher algorithm is presented. The following analyses and simulations are performed on original, encrypted and decrypted images. In these experiments, the algorithm parameters are set as follows: the block length is 256-bit, sub-block length is 32-bit, secret key length is 160-bit, number of rounds is 8, and logistic map is used as the chaotic map with $x_0 = (0.548756235417889549946554458 75)_{10}$ and $r = (3.99999999)_{10}$.

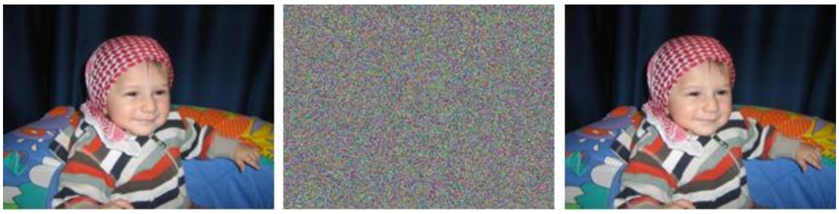
6.4.1 Images Encryption and Decryption Using BCCM Algorithm

In this test, several images of different types and patterns are encrypted with the BCCM algorithm using cipher-block chaining (CBC) block cipher mode of operation. Then, the same set of images are decrypted with the BCCM decryption algorithm using the same parameters and mode of operation as shown in Figures 6-4, 6-5, 6-6, 6-7, 6-8 and 6-9. This test shows that the proposed algorithm encrypts and decrypts images successfully using the same parameters. Moreover, the details of the encrypted images are totally invisible. BCCM algorithm is very sensitive to any change/changes in input parameters such as value of secret key, number of rounds, and word size. Figures 6-10, 6-11 and 6-12 show the decrypted images' sensitivity to any change in word size, number of rounds and value of secret key, respectively.



A) Original B) Encrypted C) Decrypted

Figure 6-4: Application of BCCM algorithm with CBC to Eiffel Tower plainimage/cipherimage with repeated patterns and large areas of the same colour abound the Tower in the picture

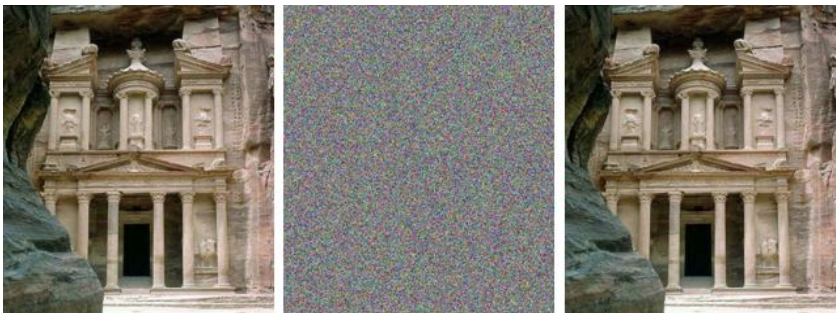


A) Original B) Encrypted C) Decrypted

Figure 6-5: Application of BCCM algorithm with CBC to Ayham plainimage/cipherimage with similar details in the Boy blouse (lines)

A) Original B) Encrypted C) Decrypted

Figure 6-6: Application of BCCM algorithm with CBC to Taj Mahal plainimage/cipherimage repeated patterns and large areas of the same colour around the castle

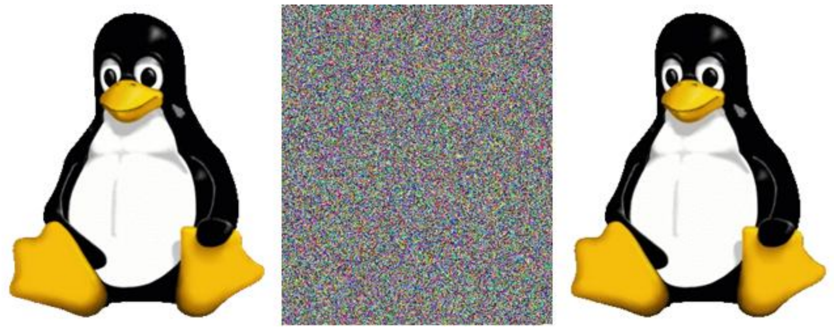


A) Original B) Encrypted C) Decrypted

Figure 6-7: Application of BCCM algorithm with CBC to Petra plainimage/cipherimage with overlap texture

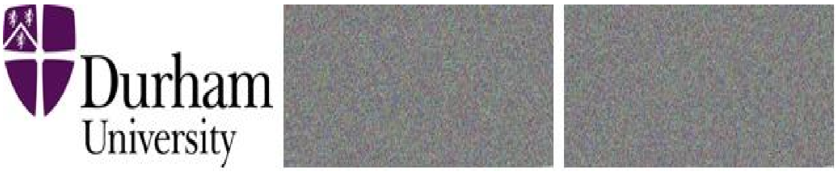
A) Original B) Encrypted C) Decrypted

Figure 6-8: Application of BCCM algorithm with CBC to Lion plainimage/cipherimage with repeated patterns and large areas of the same colour



A) Original B) Encrypted C) Decrypted

Figure 6-9: Application of BCCM algorithm with CBC to Penguin plainimage/cipherimage with repeated patterns and large areas of the same colour

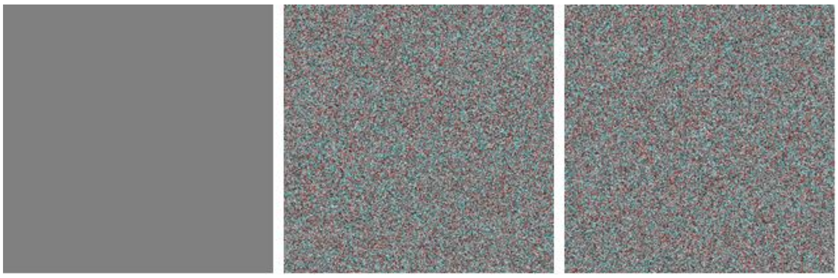


A) Original B) Encrypted C) Decrypted

Figure 6-10: Application of BCCM algorithm with CBC to Durham University Logo plainimage/cipherimage with decreasing the value of secret key during the decryption process by 10^{-144}

A) Original B) Encrypted C) Decrypted

Figure 6-11: Application of BCCM algorithm with CBC to Fruit plainimage/cipherimage with changing number of rounds from 8 to 4 during the decryption process



A) Original B) Encrypted C) Decrypted

Figure 6-12: Application of BCCM algorithm with CBC to Grey plainimage/cipherimage with changing the word size during the decryption from 32 to 16 bits.

6.4.2 Modes of Operation Effects on BCCM Algorithm

The BCCM encryption algorithm is tested with the main five modes of operation: electronic codebook (ECB) mode, cipher-block chaining (CBC) mode, propagating cipher-block chaining (PCBC) mode, cipher feedback (CFB) mode, and output feedback (OFB) mode. To show the proposed encryption algorithm with different modes of operation, Sunflower and Lion images were selected of sizes 500 × 500 and 500 × 375 pixels; these have repeated patterns and large areas of the same colour (see Figures 6-13 and 6-14). The parameters values in the encryption function are exactly the same values that mentioned in section 6.4. The BCCM encryption algorithm using ECB mode did not hide all the information and it gave a template and general idea about the encrypted image, which is similar to many other encryption algorithms. On the other hand, the result of the encrypted images showed that the BCCM algorithm hid all the information and is completely secure using CBC, PCBC, CFB or OFB modes. It can thus be concluded that these modes of operation are suitable for very highly secure applications.

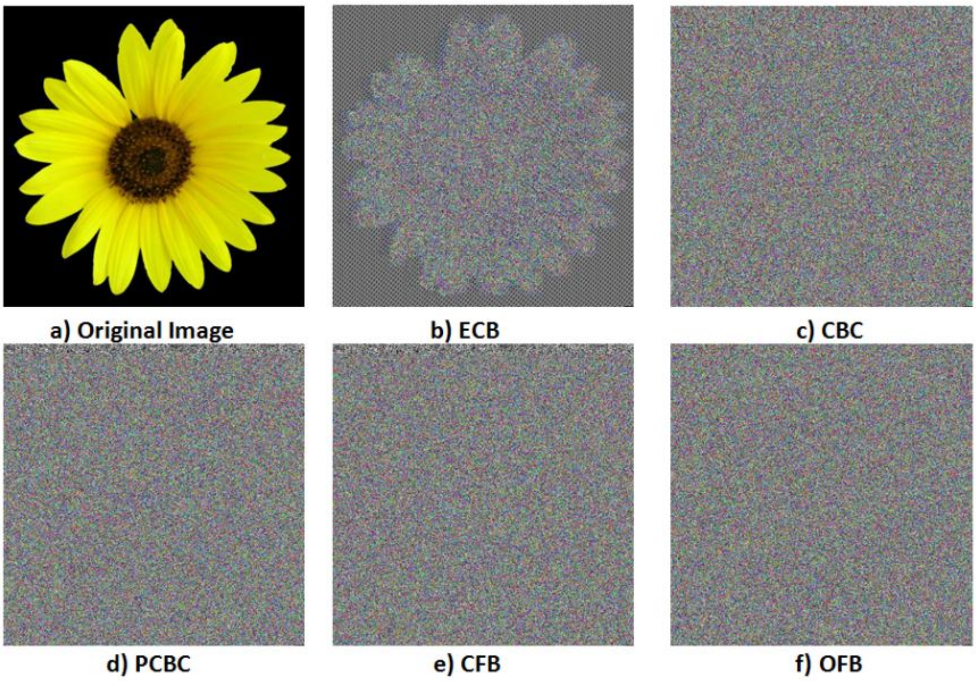

a) Original Image
b) ECB
c) CBC
d) PCBC
e) CFB
f) OFB

Figure 6-13: Encryption of Sunflower original image by BCCM algorithm with the five modes of operation

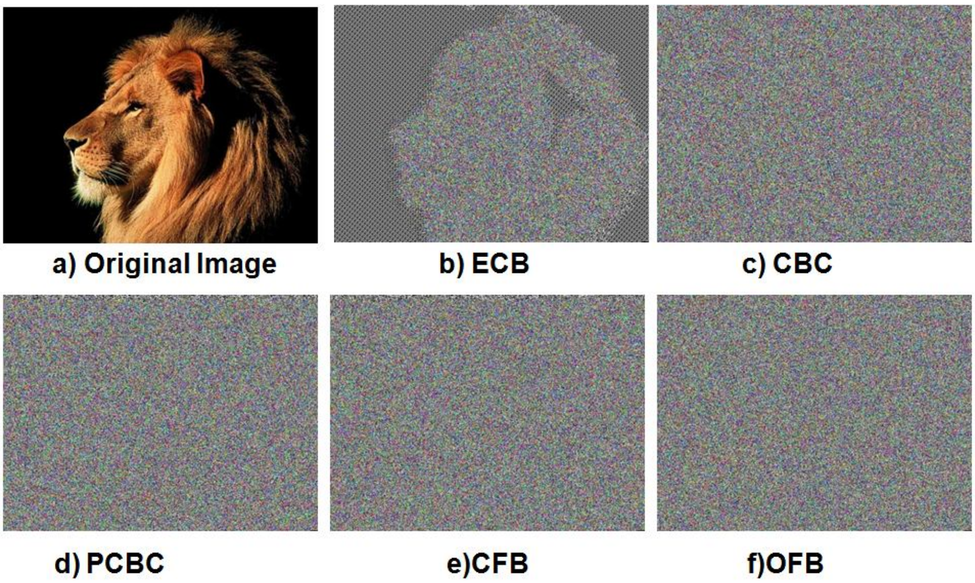
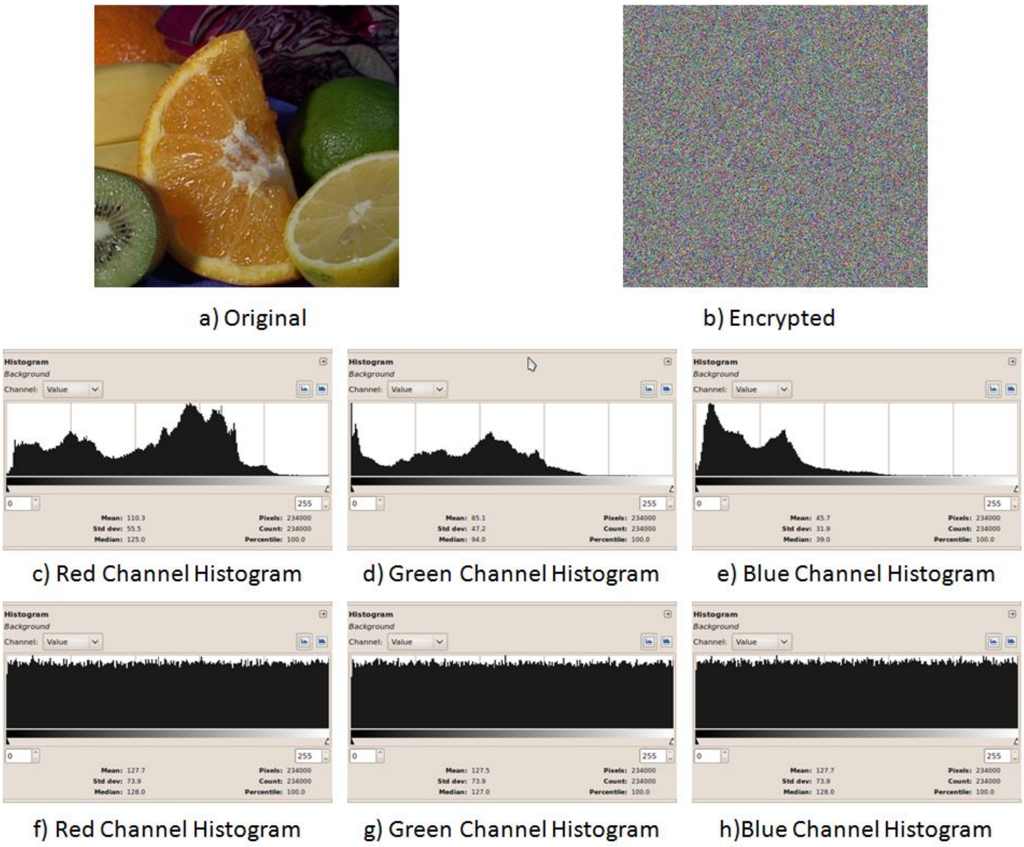


Figure 6-14: Encryption of Lion original image by BCCM algorithm with the five modes of operation

6.4.3 Histogram Analysis

A histogram analysis test was performed on several coloured images by showing the pixel intensity histogram distribution of plainimage and cipherimage. In general, highly secure encryption algorithms produce uniform histogram distribution regardless of the plainimage distribution to prevent the statistical analysis of plainimage and cipherimage. Figure 6-15 shows the histogram analysis of original and encrypted images using the same secret key. Frames (c), (d) and (e), respectively, show the histograms of red, green and blue channels of the original image. Frames (f), (g) and (h), respectively, show histograms of red, green and blue channels of the encrypted image. It is very clear from Figure 6-15 that the cipherimage has a fairly uniform distribution contrast with distribution of plainimage.



a) Original

b) Encrypted

Histogram
Background
Channel: Value
0 255
Mean: 110.3 Std dev: 55.5 Median: 125.0
Pixels: 234000 Count: 234000 Percentile: 100.0

c) Red Channel Histogram

Histogram
Background
Channel: Value
0 255
Mean: 85.1 Std dev: 47.2 Median: 94.0
Pixels: 234000 Count: 234000 Percentile: 100.0

d) Green Channel Histogram

Histogram
Background
Channel: Value
0 255
Mean: 45.7 Std dev: 31.9 Median: 39.0
Pixels: 234000 Count: 234000 Percentile: 100.0

e) Blue Channel Histogram

Histogram
Background
Channel: Value
0 255
Mean: 127.7 Std dev: 73.9 Median: 128.0
Pixels: 234000 Count: 234000 Percentile: 100.0

f) Red Channel Histogram

Histogram
Background
Channel: Value
0 255
Mean: 127.5 Std dev: 73.9 Median: 127.0
Pixels: 234000 Count: 234000 Percentile: 100.0

g) Green Channel Histogram

Histogram
Background
Channel: Value
0 255
Mean: 127.7 Std dev: 73.9 Median: 128.0
Pixels: 234000 Count: 234000 Percentile: 100.0

h)Blue Channel Histogram

Figure 6-15: Histogram analysis of plainimage and cipherimage: (a) shows original image, (b) shows encrypted image, (c-e) show channels histogram of original image, (f-h) show channels histogram of encrypted image.

6.4.4 Correlation Coefficient Analysis

Cryptography algorithms should have high capabilities of confusion and diffusion properties [292]. The confusion is the process of hiding the relationship between ciphertext and used secret key while the diffusion is the process of hiding the relationship between plaintext and ciphertext [14]. The cipherimage shows high diffusion property by normal distribution of the pixels with small correlation and it shows confusion property by hiding the relationship between cipherimage and used key. Coloration coefficients and distribution tests were carried out on the Fruits image of size 500 × 468 pixels with plainimage and cipherimage. Firstly, 2000 pairs of adjacent pixels have been selected randomly in each of the three directions, horizontal, vertical and diagonal. Then, correlation coefficients have been calculated with two different keys defined by formula 6-8. The coloration distributions of the red channel of two horizontally adjacent pixels of plainimage and cipherimage are shown in Figure 6-16. From Table 6-3, it is very clear that the coloration distribution of cipherimage is nearly uniform and much better than the histogram of plainimage; the correlation coefficients of the cipherimage are very small. Therefore, the correlation between plainimage and cipherimage is negligible and the experiment confirmed that the proposed algorithm has high confusion and diffusion properties with high security against statistical analysis attack.

$$r_{xy} = \frac{\operatorname{cov}(x, y)}{\sqrt{D(x)}\sqrt{D(y)}}, \qquad (6\text{-}4)$$

$$\operatorname{cov}(x, y) = E\{(x - E(x))(y - E(y))\}, \qquad (6\text{-}5)$$

$$E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad (6\text{-}6)$$

$$D(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x_i))^2. \qquad (6\text{-}7)$$

where x and y are grey level values of two adjacent pixels in the input image, that can be performed on horizontal, vertical, and diagonal.

6.4.5 Information Entropy Analysis

One of the most important tests of randomness is the entropy test. The entropy *H(x)* of the input message is defined by equation 6-8. The entropy values are calculated for different encrypted images with BCCM algorithm using different modes of operation. Most of the calculated entropy values for different greyscale images with different modes are very close to the optimal value (8) as shown in Table 6-4. Therefore, the BCCM encryption algorithm shows a high security factor, and the possibility of obtaining secret information is negligible, especially with CBC, PCBC, CFB, OFB modes of operation.

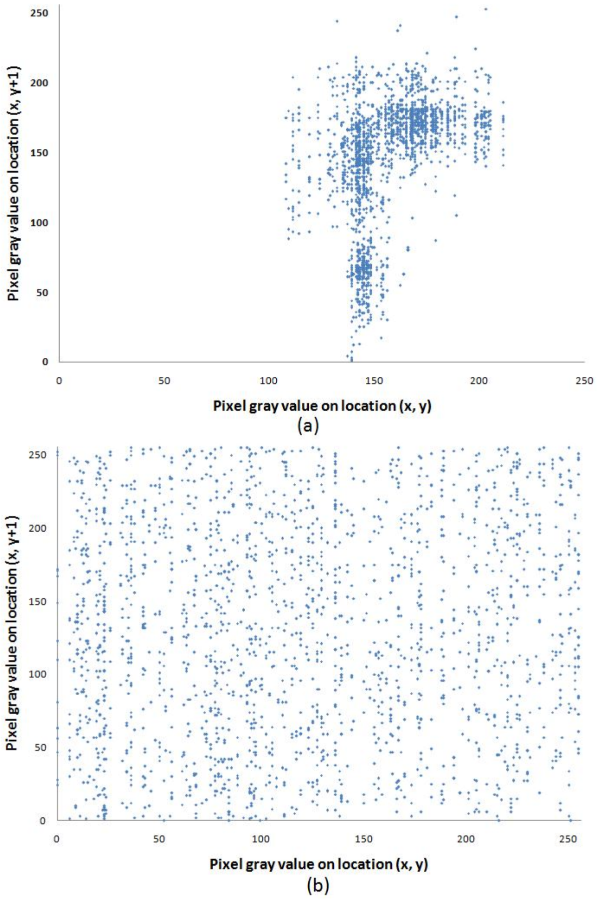


Figure 6-16: Correlation of two horizontally adjacent pixels in plainimage and cipherimage, respectively

Table 6-3: Correlation coefficients of two adjacent pixels in plainimage and cipherimage

Direction	Plainimage	Cipherimage K1	Cipherimage K2
Horizontal	0.4153518	0.0519106	0.0171848
Vertical	0.4117491	-0.0244040	0.0338735
Diagonal	0.4069532	0.0054513	-0.0079450

Table 6-4: Entropy analyses of original and encrypted images with different modes of operation

Image	Original	ECB	CBC	PCBC	CFB	OFB
Eiffel Tower	7.2908	7.9989	7.9988	7.9989	7.9989	7.9988
Lion	5.3953	7.6459	7.9991	7.9989	7.9989	7.9991
Fruit	7.5494	7.9990	7.9991	7.9991	7.9991	7.9992
Flowers	4.8413	7.6903	7.9991	7.9992	7.9992	7.9993
Ayham	7.2835	7.9989	7.9991	7.9991	7.9990	7.9992
Petra	7.8719	7.9991	7.9991	7.9992	7.9992	7.9993

$$H(x) = \sum_{i=1}^{2^N-1} p(x_i)\log_2 \frac{1}{p(x_i)} \tag{6-8}$$

where $p(x_i)$ is the probability of occurrence the symbol x_i.

6.4.6 Execution time of BCCM with Different Parameters

Some of the proposed encryption algorithms provide high security at the expense of execution time. A well-designed block cipher encryption algorithm demonstrates very high security and speed. Several tests were performed on BCCM encryption/decryption execution time with different parameters, such as number of rounds and size of input texts/images. The speed of the proposed algorithm was analyzed on Laptop Intel Core(TM) 2 Duo 2.00 GHz CPU with 3 GB RAM running on Linux Operating System, using GMP (GNU Multi-Precision Library) and OpenCV-2.0.0. As shown in Tables 6-5 and 6-6, the execution time average is very short for different texts/images with different parameters, which confirmed that the proposed encryption algorithm can be described as a fast encryption algorithm. Therefore, the BCCM algorithm demonstrated that the encryption/decryption execution time is very short, and it would be a good candidate for use in e-commerce applications over broadband networks.

Table 6-5: BCCM text execution encryption/decryption time(s)

Text size (in bits)	BCCM Encryption/Decryption Time(s) in Seconds			
	R = 4	R = 8	R = 12	R = 16
1024	0.000074	0.000089	0.000108	0.000121
6656	0.000173	0.000195	0.000218	0.000273
10752	0.000245	0.000272	0.000298	0.000321
19200	0.000394	0.000430	0.000463	0.000494
29952	0.000582	0.000632	0.000673	0.000714
64512	0.001190	0.001280	0.001348	0.001423
92160	0.001676	0.001798	0.001888	0.001992
114688	0.002072	0.002225	0.002328	0.002454

Table 6-6: BCCM image execution encryption/decryption time(s)

Image size (in pixels)	Image size on disk (in	BCCM Encryption/Decryption Time(s) in			
		R = 4	R = 8	R = 12	R = 16
64 x 64	629	0.000143	0.000203	0.000262	0.000325
128 x 128	881	0.000433	0.000638	0.000852	0.001057
256 x 256	1649	0.001580	0.002414	0.003204	0.003992
512 x 512	4721	0.006384	0.009420	0.012620	0.015752
1024 x 1024	17009	0.025065	0.038630	0.050303	0.063127

6.4.7 Comparison between BCCM and Some Existing Schemes

BCCM is compared with two well-know encryption schemes (DES, RC6) and one chaotic encryption scheme (Chen S. et al.'s scheme). Table 6-7 summarizes the comparison results for word size, block size, maximum block size, secret key length, number of subkeys, number of rounds, mathematical operations and algorithm structure. The security of BCCM relies on the sensitivity of secret key, chaotic subkeys, chaotic subkeys order, chaotic mixing column, chaotic shifting rows, and non-linear operations. The combination of these security factors produces a very highly random, sensitive, fixable encryption function with high confusion and diffusion properties. Moreover, the simple non-linear operations - addition, exclusive-OR, circle left shift, and circle right shift - are very fast on the computer. A chaotic system is slower than some other systems on a computer, but we utilized simple chaotic systems on subkey generation and the resulting values are employed in other functions to provide high randomness, performance and security. DES, RC6 and Chen et al.'s encryption schemes were proved as insecure encryption algorithms.

Table 6-7: Comparison between DES, RC6, Chen S. et al. algorithm, and BCCM

Parameters	Encryption Algorithm			
	DES	**RC6**	**Chen et al.**	**BCCM**
w (word size in bits)	8	16, 32, 64	8	8, 16, 32, 64
Block size in words	8w	4w	w	8w
Block size in bits	64	64, 128, 256	8	64, 128, 256, 512
Max. block size in bits	64	256	8	512
s (key length in bytes)	7	0, 1, 2, ..., 255	4	1, 2, 3, ..., 512
No. of keys derived from key	16	2r + 4	4	8r
r (No. Of rounds)	16	0, 1, 2, 3, ..., 255	4	4, 6, 8, ..., 512
Used operations	+, -, *, >>>, <<<, ⊕	+, -, *, >>>, <<<, ⊕	+, -, *, >>>, <<<, ⊕	+, -, >>>, <<<, ⊕ , %
Algorithm structure	Feistel cipher	Feistel cipher	Feistel cipher	Feistel cipher
Chaotic maps	No	No	Logistic map, 8-bit chaos	1D chaotic maps
Existing attacks	linear cryptanalysis, differential cryptanalysis, Brute-force attack	linear cryptanalysis, Statistical attacks,	differential cryptanalysis, brute-force attack	No
Security	Proven inadequate	Proven inadequate	Proven inadequate	Under study

6.5 Conclusion

In this chapter, we proposed a novel fast block cipher encryption algorithm based on chaotic systems. The proposed algorithm (BCCM) works with different parameters such as block size, number of rounds, secret key length and chaotic maps. BCCM processes n-bit of plaintext and produces the same size of ciphertext with k-bit key size. It generates several subkeys from a single secret key using a simple one-dimensional chaotic map. The proposed algorithm has chaotic subkeys generation, chaotic columns mixing, and chaotic rows shifting based on the value of the secret key to provide high confusion and diffusion properties. Therefore, encryption and decryption functions architectures are depend on the value of the secret key. We performed several analyses and computer simulations which confirmed that the BCCM algorithm satisfies the characteristics and conditions of cryptography block cipher. Moreover, we have compared the BCCM with two well-known block ciphers (DES, RC6,) and one chaos-based block cipher; the comparison results showed that BCCM work better than these algorithms. BCCM is confirmed as a good candidate for texts/images encryption with high flexibility, confusion and diffusion, and fast speed. These characteristics show that the proposed algorithm has high potential for adoption in e-commerce.

Chapter 7

Fast Hash Function Based on BCCM Encryption Algorithm for eCommerce (HBCCM)

In chapter 6, we proposed a fast block cipher encryption algorithm based on chaotic maps (BCCM) with high sensitivity, flexibility, performance, confusion and diffusion properties. Analyses and experimental results confirmed that the whole encryption process is extremely sensitive to any change or changes in the input message and/or secret key. These properties are promising for the adoption of the BCCM encryption function in designing a new cryptography hash function. In this chapter, we propose a fast chaotic hash function based on BCCM encryption algorithm with changeable parameters (HBCCM). The proposed hash function produces different lengths of hash values based on different chaotic systems with variable parameters. A comparison between HBCCM and other hash functions is presented. Analyses and computer simulations confirmed that the proposed algorithm satisfies all cryptography hash function requirements and it is high potential for adoption in e-commerce.

7.1 Introduction

Over the past decade, many researchers have utilized chaotic systems to design cryptography hash functions to provide high security [30, 70, 76, 81, 87, 88, 286, 287, 293, 294]. Unfortunately, some of the proposed algorithms are described as insecure and/or slow [137, 140, 143, 144]. Therefore, further research is still needed to design a fast and secure chaotic cryptography hash function. In chapter 6, we proposed a fast block cipher encryption algorithm based on chaotic maps (BCCM) with high sensitivity, flexibility, performance, bit confusion and diffusion properties [295]. In this chapter, we propose a fast hash function based on BCCM encryption algorithm with changeable parameters (HBCCM) [296]. The rest of this chapter is organized as follows: Section 7.2 gives details of the proposed hash function (HBCCM). Section 7.3 discusses the experimental results of HBCCM. Finally, the conclusion is given in section 7.4.

7.2 Details of HBCCM Hash Function

BCCM is a fast block cipher encryption algorithm based on chaotic maps with changeable parameters. BCCM is an encryption algorithm that provides high flexibility, confusion, diffusion, and fast encryption speed. We performed several analyses and computer simulations on BCCM which confirmed that the algorithm satisfies the characteristics and conditions of a cryptography encryption algorithm. Experimental results confirmed that the whole encryption process is extremely sensitive to any change or changes in the input message and/or secret key. These properties are promising for the adoption of the BCCM encryption function to design a new cryptography hash function. In this chapter, a fast hash function based on BCCM encryption algorithm is proposed; it is called HBCCM.

An overview of the HBCCM hash function algorithm is illustrated in Figure 7-1. The new hash function algorithm is exactly same as BCCM encryption function with few changes. HBCCM compression function is implemented by adding new compression function at the end of BCCM encryption as shown in Figure 7-1(a). The Initialization, key generation, subkeys' orders (SO_i), rows shifting (RS_i) and columns-mixing (CM_i) are exactly the same in BCCM algorithm (for more details see section 6.2). In the proposed hash function, the total number of subkeys is eight times the number of (rounds+1) [8*(R+1)]: eight subkeys for each round and eight subkeys for mixing the result of the previous block with the current block in the addition compression function (see Figure 7-2(a)).

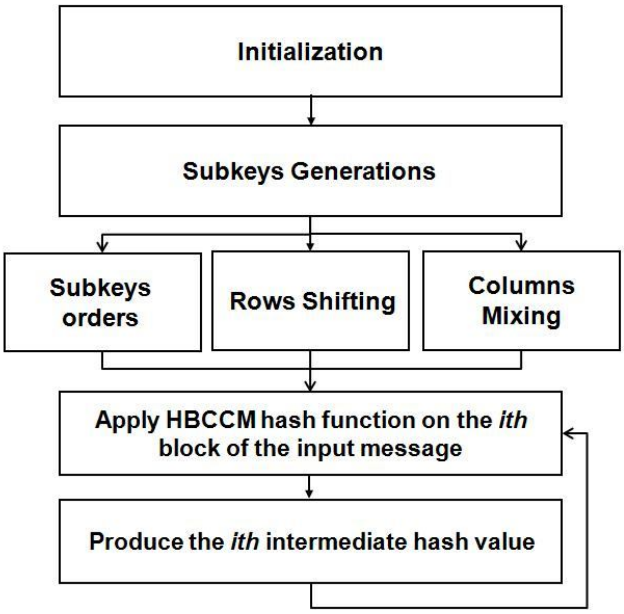
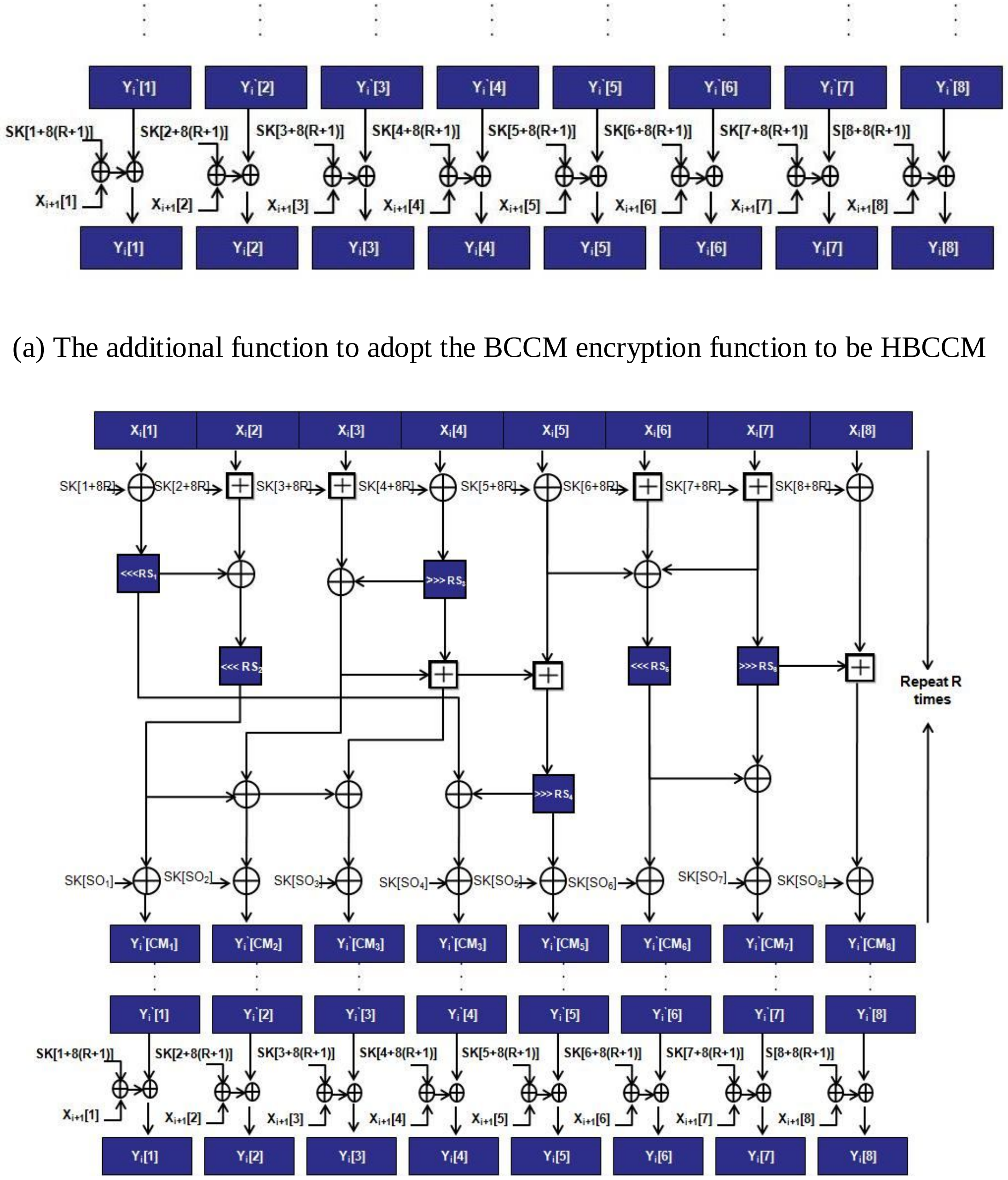

Initialization
Subkeys Generations
Subkeys orders
Rows Shifting
Columns Mixing
Apply HBCCM hash function on the *ith* block of the input message
Produce the *ith* intermediate hash value

Figure 7-1: Overview of HBCCM

In the HBCCM algorithm, after all subkeys, subkeys orders, shifting rows and mixing columns are generated, the next step is to apply the hash compression function. The blocks of the input message are processed throughout the compression function by iterating each block “R” times and in each round applying compression function operations to calculate the intermediate result of the sub-block Y_i`. Then, the values of the next sub-blocks $X_{i+1}$[n] are XORed with n+8(R+1) subkeys SK[n+8(R+1)], and are then XORed with the resulting values from processing the previous sub-blocks through the compression function $Y_i$`[n] to produce the correspondence intermediate sub-block hash value (Y_i[n]). After that, the resulted intermediate sub-block hash values Y_i will be the next input to the hash compression function. HBCCM keeps running until it has processed the entire input message and then generated the final hash value.



(a) The additional function to adopt the BCCM encryption function to be HBCCM

(b) Full HBCCM compression function

Figure 7-2: HBCCM compression function

The number of hash rounds (*R*) was chosen very carefully by selecting the input image and secret key randomly; then the hash value is calculated using the HBCCM with different numbers of rounds. For the resulting hash value, the confusion and diffusion test and correlation coefficient analysis are applied as explained in detail in section 7.3.4 and 7.3.5, respectively. We found that the proposed algorithm has high confusion and diffusion properties with the number of rounds equal $R \geq 8$. HBCCM aimed to construct a keyed hash function; at the same time we can construct an unkeyed hash function by processing the complete input message through one of the chaotic maps and using the result as the secret key, although this will add to computation costs depending on input message size.

HBCCM has a very flexible design that can be customized using different parameters for different applications as follows:

1- HBCCM can construct a keyed hash function as the chaotic system's initial condition x_0 and the control parameter r is used as the secret key.

2- HBCCM can construct an unkeyed hash function by processing the complete input message through one of the chaotic maps and using the result as the secret key.

3- HBCCM processes different types of input messages of variable lengths.

4- HBCCM can be constructed using several chaotic maps as chaotic systems: Tent map, Skew Tent map, Logistic map, and other PWLCMs.

5- HBCCM produces different hash value lengths in bits, such as 64, 128, 192, 256, and 512.

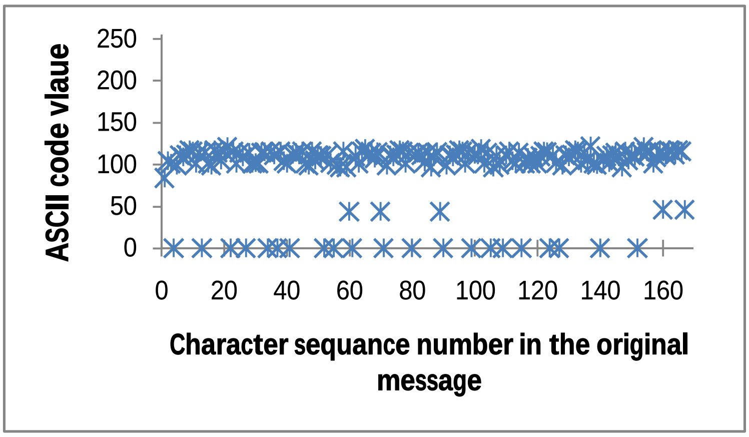
6- HBCCM is based on variable number of rounds R, $R \in [8, N]$.

7.3 Experimental Results

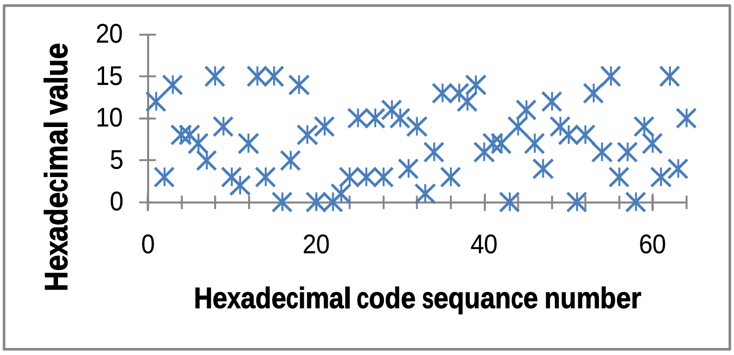
In this section, performance and statistical analysis of HBCCM hash function are discussed. Several tests and simulations are performed on the computed hash values. The hash value length is 265, the block length is 256-bit, the sub-block length is 32, the secret key length is 160-bit, the number of rounds is 16 and logistic map is used as the chaotic map with x_0 = $(0.80660971808844940000)_{10}$ and r = $(3.99999999)_{10}$.

7.3.1 Hash Value Distribution

Uniform distribution of hash value is a very important condition in hash function security [81]. We performed the uniform distribution test on the following message: *“The computer security term refers to the protection of data, networks, computer programs, computer power and other elements of computerized information systems.”* The differences between ASCII distribution of original message and hexadecimal distribution of hash value are shown in Figure 7-3. The experiment result shows that the ASCII distribution of the original message focuses on a very small space, while the hash value distribution is irregular and spread over a large hash space. The simulation result confirms that the proposed hash function is secure enough against statistical attack and it is difficult to retrieve information about the input message from the calculated hash value.



(a) ASCII distribution of the original message



(b) Hash values distribution in hexadecimal format

Figure 7-3: Distribution of original message and hash value

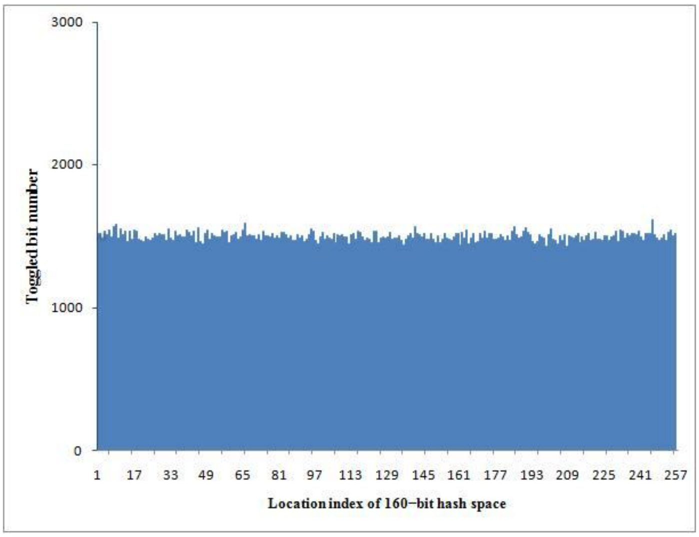


Figure 7-4: Hash value distribution in hash space with N = 3000 and mean 1498.03

Similar to [87, 286], another hash space uniform distribution is performed to check hash space distribution property. A message is generated randomly and then the correspondence hash value is calculated to ensure hash space distribution. One bit is toggled randomly and then the new correspondence hash value is calculated. Later on, the number of toggled bits in the calculated hash space for the two hash values is counted. This test is performed 3000 times by fixing the first input message and toggling one bit randomly to calculate the number of toggled bits at same location in the hash space, as shown in Figure 7-4. The mean of toggled bits is 1498.03, which is very close to the ideal mean value (1500). Therefore, HBCCM will be strong enough against statistical attacks and in terms of collision resistance.

7.3.2 Hash Value Result of Text Input Message

In order to evaluate HBCCM hash value sensitivity to text input message, simulation tests are performed on the following text: “*The computer security term refers to the protection of data, networks, computer programs, computer power and other elements of computerized information systems.*” Then, several changes are performed using secret key K = $(2F050FE938943ACC45F65567FFFFFFFE)_{16}$. The correspondence hash value for each scenario is calculated in hexadecimal format (see Appendix D). The simulation results show that HBCCM are very sensitive to any slight change(s) in the input text message or used key by producing significantly different hash value.

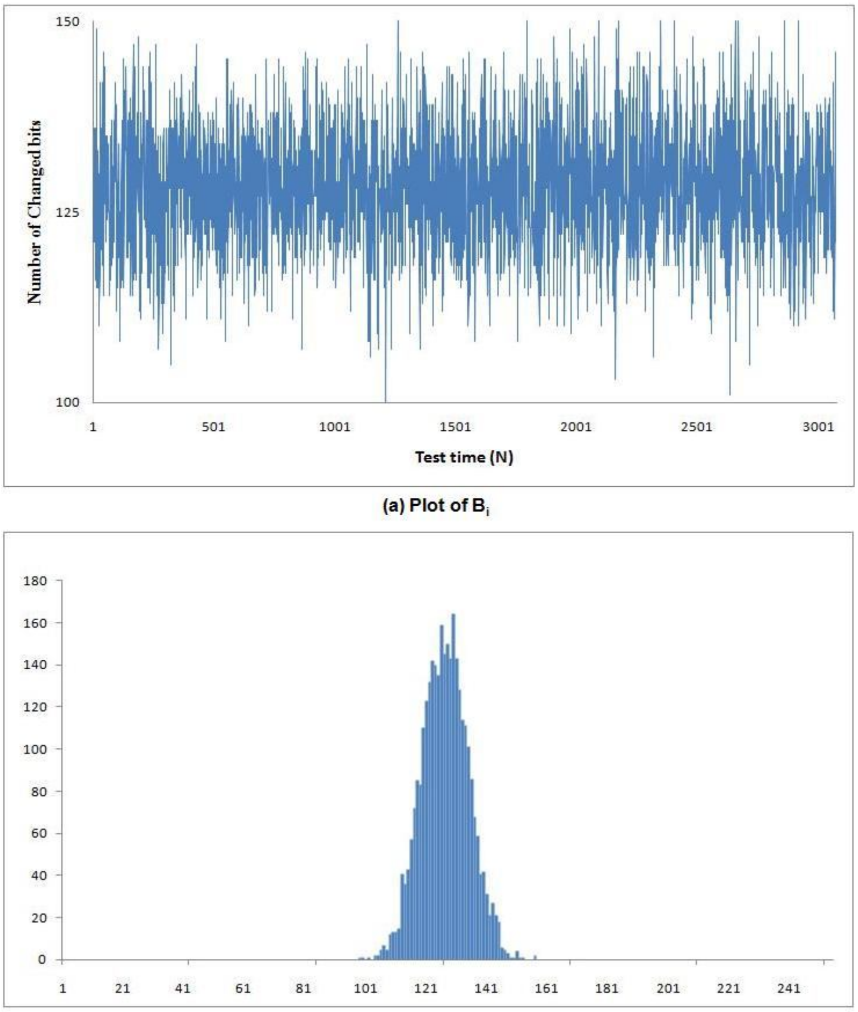
7.3.3 Hash Value Result of Images Input Message

In order to evaluate HBCCM hash value sensitivity to image input message, simulation tests were performed on Ayham image of size 500 × 375 pixels with 8-bit greyscale; the used key is k = $(2F050FE938943ACC45F65567FFFFFFFFF)_{16}$ (see Figure 7-4). Then, several changes were performed and the correspondence hash values were calculated in hexadecimal format (see Appendix E). The simulation results show that HBCCM are very sensitive to any change(s) in input image or used key by producing significantly different hash value.

Figure 7-5: Ayham 8-bit grayscale input image

7.3.4 Statistical Analysis of Diffusion and Confusion

Confusion and diffusion properties are very important for many cryptography algorithms such as block cipher encryption algorithms and hash functions [297]. A hash function requires the spreading influence of the whole input message into the hash value space. A well-designed hash function must have a complex relationship between the bits of input message and the correspondence bits in the hash value. Consequently, changing any bit or bits in input message must affect at least half the hash value bits with a 50% probability of each bit in the hash value being changed. A message is chosen randomly and correspondence hash value is calculated. Then, one bit in the input message is selected and toggled randomly, and the binary hash value is calculated. Now, a comparison between the original and the new hash value is performed by counting the number of changed bits in the same locations called B_i. This test is performed 3000 times and the correspondence distributions of the changed bits are calculated (see Figure 7-6). Similar to [70, 87, 286, 287], six statistics for HBCCM hashing scheme (see section 5.3.2 and equations 5-3 to 5-8)



(a) Plot of B_i

(b) Histogram of B_i

Figure 7-6: Distribution of changed bit number B_i

Table 7-1 shows the statistical results for different numbers of tests: N = 3072, 2048, 1024, and 512. The mean number of changed bits and the mean changed bit probability are 128.14 and 50.05, respectively, which are very close to the ideal values of 128 and 50%. ΔB and ΔP indicate the stability of diffusion and confusion which show that the HBCCM hashing scheme is very stable. We conclude that HBCCM is secure enough against statistical attacks and very difficult to attack, so it can be adopted in many different applications and protocols such as e-commerce, e-business and Internet banking.

Table 7-1: Statistic of number of changed bit Bi

	N = 3072	N = 2048	N = 1024	N = 512	Mean
$\bar{B}$	128.20898	127.93505	127.9463	128.47656	128.14
P	50.08163	49.97463	49.97902	50.18616	50.055
ΔB	7.8296714	7.7925710	8.176215	7.9626966	7.9402
ΔP	3.057968	3.043973	3.193834	3.110428	3.1015
Bmin	100	103	104	108	103.75
Bmax	158	153	157	145	153.25

7.3.5 Analysis of Collision Resistance

HBCCM collision resistance capability is tested. First, a message is selected randomly and the associated hash value is calculated in ASCII format. Then, one bit is selected and toggled randomly to generate a new hash value in ASCII format. Now, the difference between the two ASCII hash values in the same position is calculated. Finally, absolute difference (d) is calculated by calculating the summation of all differences of all characters for two hash values (see equation 5-9). This kind of test is performed 3000, 2000 and 1000 times and maximum, minimum, mean and mean/c, where c is number of hexadecimal character of hash value for each case, are calculated, as shown in table 7-2. Figure 7-7 shows the distribution of number of ASCII characters with same value at the same location in the hash value, with number of tests 3000, 2000 and 1000 (see equation 5-10). It is clear that the maximum number of equal ASCII characters in the same location is only three, the possibility of collision is very low, and most of the entries are different in ASCII format. Figure 7-7 shows that the peaks values at one while Figure 5-6 shows that the peaks at zero, which means the probability to find a collision in PHFC is less than HBCCM.

Table 7-2: Absolute differences of two hash values

	N = 3000	N = 2000	N = 1000
Maximum	5436	5417	5533
Minimum	2630	2693	2585
Mean	3956	3936	3959
Mean/C	123.6	123	123.7

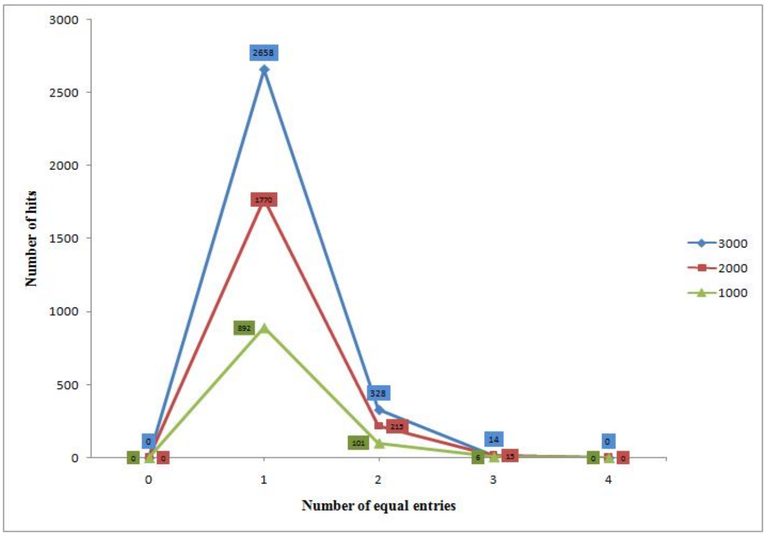


Figure 7-7: number of ASCII characters distribution of the same value at the same location in the hash value with 3000, 2000, and 1000 number of tests

7.3.6 Execution time of HBCCM with Different Parameters

The main idea behind designing a hash function is to provide a fast security algorithm based on a small amount of data compared with some public key encryption algorithms for security applications. An ideal hash function algorithm should provide high security with high speed. Several speed tests have been performed on HBCCM algorithm execution time with different parameters, such as number of rounds and size of the input text/image. This test has been performed for the proposed hash algorithm on Laptop Intel Core(TM) 2 Duo 2.00 GHz CPU with 3 GB RAM running on Linux Operating System, using GMP (GNU Multi-Precision

Library) and OpenCV-2.0.0. Speed average is very small for calculating hash values using HBCCM algorithm with different parameters (see Tables 7-3 and 7-4).

Table 7-3: Execution times for HBCCM to generate hash value of Images

Image size (in pixels)	**Image size on disk (in bytes)**	**HBCCM Encryption/Decryption Time(s) in Seconds**			
		R = 4	**R = 8**	**R = 12**	**R = 16**
64 x 64	629 bytes	0.000121	0.000198	0.000231	0.000269
128 x 128	881 bytes	0.000446	0.000484	0.000524	0.000564
256 x 256	1649 bytes	0.001298	0.001338	0.001392	0.001464
512 x 512	4721 bytes	0.004754	0.004832	0.004924	0.005056
1024 x 1024	17009 bytes	0.019006	0.01926	0.019482	0.019702

Table 7-4: Execution times for HBCCM to generate hash value of texts

Text size (in bits)	**HBCCM Encryption/Decryption Time(s) in Seconds**			
	R = 4	**R = 8**	**R = 12**	**R = 16**
2048	0.000131	0.000145	0.000165	0.000181
5888	0.000189	0.000204	0.000230	0.000263
10240	0.000265	0.000281	0.000315	0.000352
20480	0.000445	0.000463	0.000515	0.000562
32768	0.000660	0.000681	0.000755	0.000814
57344	0.001093	0.001118	0.001235	0.001318
81152	0.001511	0.001541	0.001700	0.001806
94208	0.001741	0.001773	0.001955	0.002074

A comparison of execution times between SHA-1, CHA-1, parallel version of PHFC (P-PHFC), sequential version of PHFC (S-PHFC), and HBCCM hashing schemes is performed. The experiment result shows that the CHA-1 and sequential mode of PHFC are very slow compared with the other hashing schemes (see Figure 7-8). Moreover, the parallel mode of PHFC and SHA-1 average execution times is very short and close to each other, but SHA-1 shows a slightly better performance with bigger message size. Furthermore, the average execution times of HBCCM with 8 rounds is shorter than all the other algorithms with different input message lengths, and HBCCM with 16 rounds is slightly higher than SHA-1 with all input message lengths. The overall result shows that HBCCM is a very fast and secure hashing scheme compared with many of the existing algorithms. Therefore, HBCCM is considered a very fast and secure hashing scheme that would be a good candidate for use in e-commerce applications over broadband networks. SHA-1 code was downloaded from Requests for Comments (RFC) website (http://www.rfc-editor.org/rfc/rfc2841.txt).

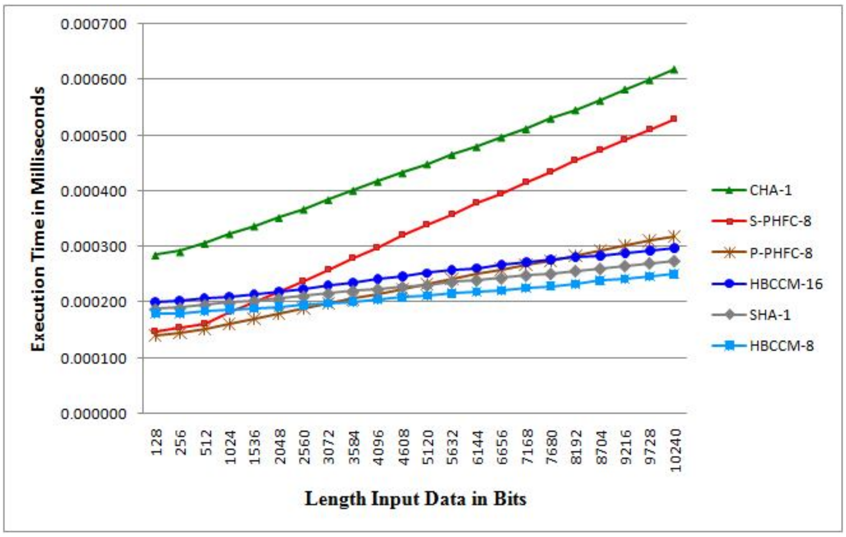


Figure 7-8: Comparison between SHA-1, CHA-1, sequential and parallel versions of PHFC with 8-round, and HBCCM with 8 and 16 rounds in the execution time (millisecond)

7.4 Conclusion

In this chapter, we proposed HBCCM as a new hash function based on BCCM block cipher encryption algorithm. HBCCM is very simple, flexible, and sensitive to any change (s) in the input message of the secret key. It processes input messages of different lengths to generate different hash value lengths based on different chaotic systems. Moreover, it can be used to calculate the correspondence hash value of input messages as texts or images with high confusion and diffusion properties. Comparisons between HBCCM and other hash functions show that HBCCM outperforms all the other algorithms. Several analyses and computer simulations have been performed that confirm the new hash function algorithm satisfies the characteristics and conditions of cryptography hash hashing scheme, which would have high potential for adoption in e-commerce applications.

Chapter 8

A Novel Pseudorandom Number Generator Based on New Triangle-Chaotic Map for High Security Applications

In this chapter, a new one-dimensional Triangle-chaotic map (TCM) with full intensive chaotic population is proposed. It shows a great sensitivity to initial conditions, has unpredictability, is uniformly distributed, random-like and has an infinite range of intensive chaotic population with large positive Lyapunov exponent values. In addition, a novel pseudorandom number generator based on TCM map is proposed; it is called PRNGT. PRNGT generator is designed based on a combination of two TCM maps running side by side and starting from independent initial conditions. Generated pseudorandom sequences based on logistic map, modified logistic map and PRNGT generators are tested using NIST 800-22 suite standard statistical tests. The experimental results confirm high randomness of generated sequences using PRNGT by passing all NIST 800-22 suite tests. Moreover, non-randomness of generated sequences using original and modified versions of logistic map is confirmed. The statistical analysis results confirm that PRNGT is a high-potential candidate for high-security applications such as e-commerce and e-banking. Moreover, TCM characteristics are very promising for possible utilization in designing and developing new security primitives.

8.1 Introduction

Random number generator is an algorithm or a device that generates a statistically independent sequence of bits. Random number generators play a significant role in many applications such as computer simulation, games, science, statistical sampling and cryptography [272, 298]. Generating a high-randomness bit sequence is considered a big challenge in the literature. There are two main types of random numbers generators: truly random number generator (TRNG) and pseudorandom number generator (PRNG). TRNG is a non-deterministic procedure for extracting random bits from physical phenomena such as a user's mouse movements. PRNG algorithm is a deterministic algorithm that produces random statistically independent bits from mathematical formulae [10].

PRNG's initial set of input parameters are called seeds and output values are called random bit sequence. PRNG produces multiple pseudorandom bits based on one or more input parameters. Ideally, we can regenerate exactly the same sequence only if we use the same seeds. In a generated pseudorandom sequence, the subsequent bits should be independent and unpredictable from the current bits or generated bits. The input parameters should be random and unpredictable from the generated sequence and the output sequence is deterministic on the same input parameters. Unpredictable sequences are used to provide high security for many cryptography systems such as digital signatures, secret key in DES encryption algorithm, and prime numbers in RSA [10].

A logistic map shows a chaotic behaviour that can arise from very simple non-linear dynamical equations (see Figure 8-1) [239]. Logistic map behaviour seems to be a random jumble of dots and mainly depends on two parameters (x_0 and r). We can observe different logistic map behaviours by changing the value(s) of one or both of these parameters. The general idea of a logistic map was built based on an iterations function, where the next output value depends on the previous output value. Figure 8-2 shows the calculated Lyapunov exponent value of a logistic map with different values of parameter $r \in [0, 4]$. In a logistic map equation, x_0 and r represent the initial conditions, $x_0 \in [0, 1]$ and $r \in [0, 4]$. Chaotic behaviour is exhibited with $3.57 > r \geq 4$, but it shows non-chaotic behaviour with some values of parameter r (see

Figures 8-1 and 8-2). In this chapter, we refer to x_0 and r parameters as the initial conditions of a logistic map.

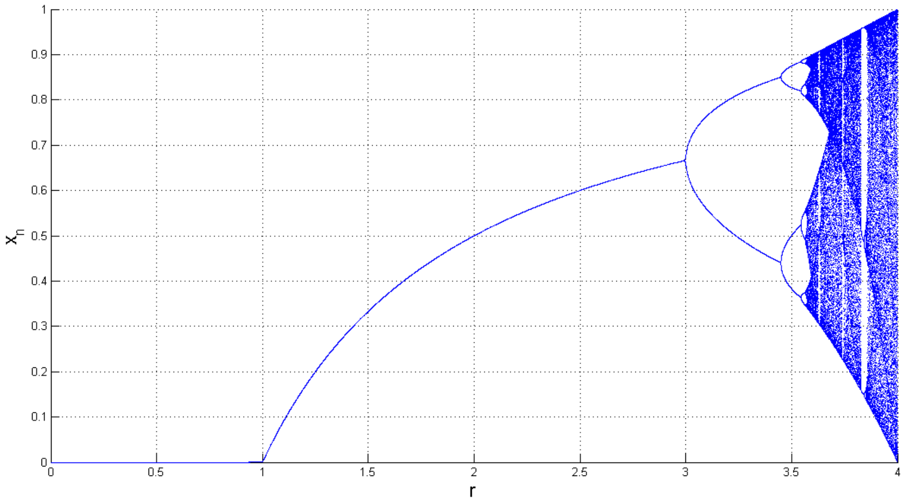


Figure 8-1: Bifurcation diagram of logistic map

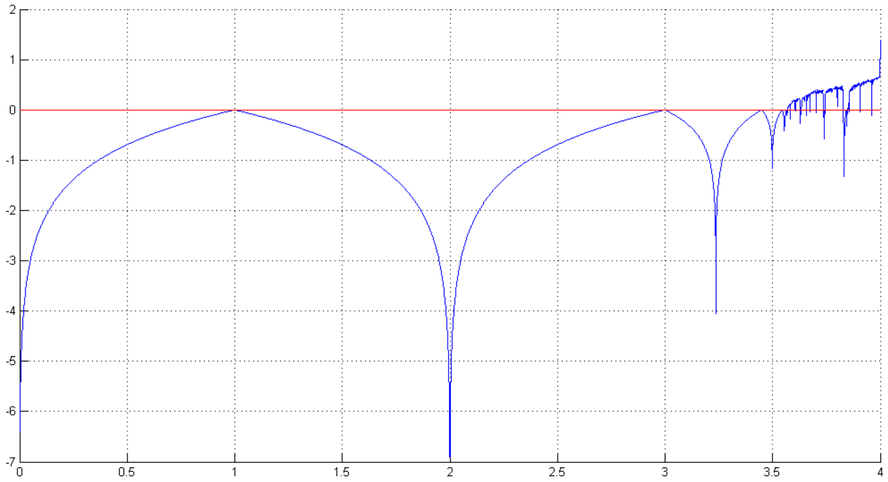


Figure 8-2: Lyapunov exponent of Logistic map with t ∈ [0, 4]

A small range of logistic map parameters are consider as valid values to show chaotic behaviour [299]. In general, chaotic behaviour is exhibited with values of parameter r greater than 3.57 and less than or equal to 4. It is very clear from Figure 8-1 that the logistic map periodic window becomes significantly complicated with 3.57 > r ≥ 4. In Figures 8-3 and 8-4, we plotted a portion of logistic map bifurcation and its Lyapunov exponent, respectively, using MATLAB software, to give a clear picture of the chaotic areas [300, 301]. There are non-chaotic areas with some values of parameter r over the chaotic interval, which are called stability or islands. It is very clear that there is a 3-periodic window between 3.828429 and 3.841037 [124]. The value of r = 3.840 falls in the 3-periodic window and the value of r = 3.845 fall in the 6-periodic window. Therefore, after a small number of iterations with different initial values of x, (x_0) will end up in one of these periodic. The cryptosystems fall within the 3-perodic and the 6-periodic windows with r = 3.840 and 3.845, respectively, and were utilized for the purpose of attacking them [124]. Moreover, the logistic map population will cover the full interval of x, ([0, 1]), only with r = 4.

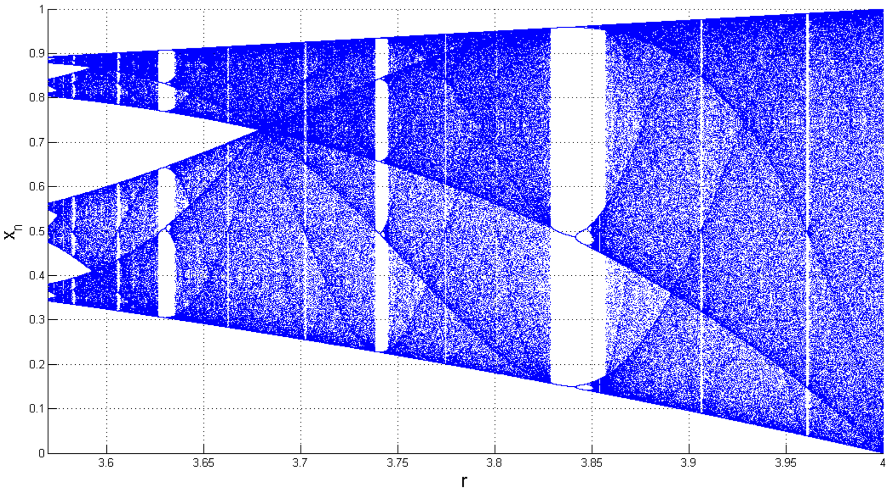


Figure 8-3: Logistic map bifurcation diagram of a periodic window

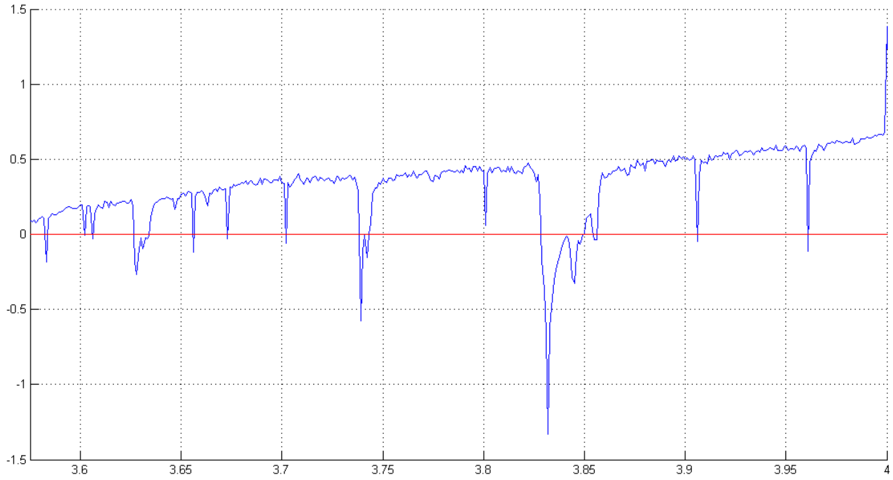


Figure 8-4: Lyapunov exponent of Logistic map with t ∈ [3.575, 4]

In this chapter, we propose a new one-dimensional triangle-chaotic map (TCM) of degree two with full chaotic population over infinite interval of parameters.

Moreover, a novel pseudorandom bits sequence generator based on TCM chaotic map is proposed for high security applications. The rest of this chapter is organized as follows. In section 8.2 details of the new triangle-chaotic map are given. Section 8.3 describes details of pseudorandom number generators. High level of NIST statistical suite test details are given in section 8.4. Experimental results are discussed in Section 8.5. Finally, the conclusion is given in section 8.6.

8.2 New Triangle-Chaotic Map (TCM)

In this chapter, a new triangle-chaotic map (TCM) is proposed. TCM is a one-dimensional chaotic map of degree two with full chaotic population over infinite interval of parameter *t* values (see equation 8-1). The triangle-chaotic map behaviour mainly depends on the initial values of parameters y_0 and *t*. TCM behaviour seems to be a random jumble of dots, and depends on initial conditions (y_0 and *t*). The y_0, y_n are positive real numbers between 0 and 1, $y_n \in [0, 1]$, and *t* can be any positive real number $t \in [0, \infty]$. Figures 8-5 and 8-6 show a TCM map bifurcation diagram and the calculated Lyapunov exponent value over r $\in [0, 4]$. It is very clear from the figures that TCM shows perfect chaotic behaviour over the full interval. Figure 8-7 shows a TCM diagram with initial value of *t* very close to zero and random number of y_0, iterating TCM map many times, and then plotting the *t* series of values of y_n using MATLAB software. In other words, we plotted corresponding points of y_n to a given value of *t* and increased *t* to the right. TCM is very sensitive to any change(s) in one or both initial conditions and is unpredictable in the long term, as shown in Figures 8-7 and 8-8. In this paper, we refer to y_0 and *t* parameters as the initial conditions of TCM map.

$$f(y_n) = y_{n+1} = \begin{cases} (t\,y_n(1-y_n)) \bmod 1 & n \bmod 2 = 0 \\ & ; \\ (\pi\,y_n\beta) \bmod 1 & n \bmod 2 != 0 \end{cases} \qquad (8\text{-}1)$$

where y_n is a number between zero and one, y_0 represents the initial population, *t* is a positive real number, *n* is a number of iterations, β: is a positive odd number between 3 and 99.

The general idea of a TCM map was built based on an iteration function. The result of the next output value (y_{n+1}) in TCM depends on the previous output value (y_n) (see equation 8-1). A TCM map over a different range of parameter *t* values will give different f maps. To show TCM sensitivity we plotted the behaviour of three nearby initial values of y_0 and three nearby initial values of *t*. Three nearby initial values of y_0 (0.990000, 0.990001, and 0.990002) for t =1 started at the same time and rapidly diverged exponentially over time with no correlation between each of them (see Figure 8-7). Moreover, we plotted populations of three slightly different parameter values of *t* (4.000000, 4.000001, and 4.000002) and $y_0 = 0.5$ to show great sensitivity to initial conditions of the TCM map (see Figure 8-8).

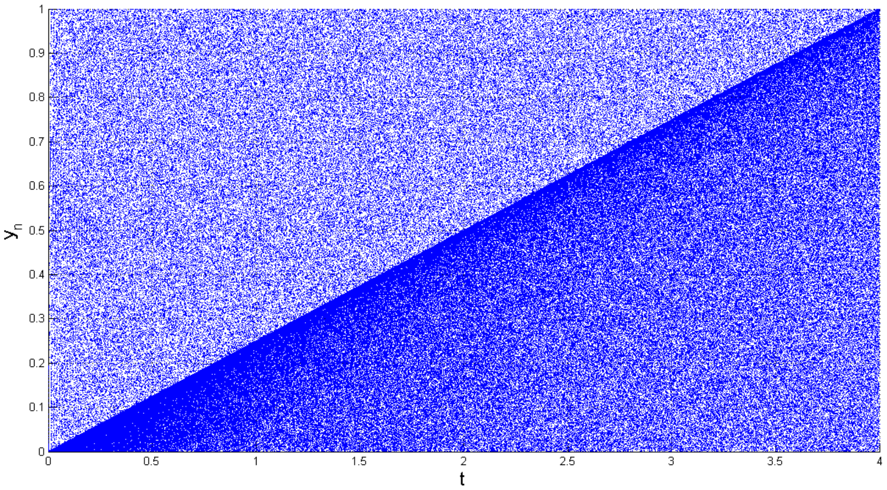


Figure 8-5: TCM chaotic map bifurcation diagram with t ∈ [0, 4]

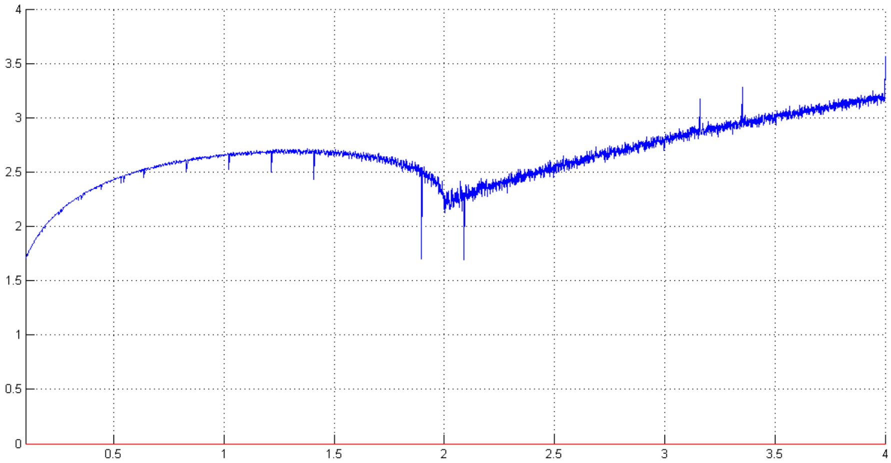


Figure 8-6: Lyapunov exponent of TCM chaotic map with *t* ∈ [0, 4]

TCM diagram and population distribution histograms have been plotted for population of TCM over the $t \in [32, 36]$. TCM iterated 43686 times with initial conditions values of t_0 = 32 and y_0 = 0.5. We draw the TCM diagram by plotting corresponding points of y_n to a given value of t and increasing t to the right (see Figure 8-9). TCM population interval, [0, 1], is divided into 10 equal sub-intervals and the number of points in each interval has been counted for each sub-interval and plotted (see Figures 8-10). It is very clear from Figures 8-9 and 8-10 that TCM population is uniformly distributed over the interval [0, 1] with $t \in [32, 36]$. We draw the TCM diagram and population distribution histogram with different initial t values (0, 4, 8, 12....etc.) and the overall results confirm that the TCM population distributions are uniformly distributed with $t \geq 12$ and interval size 4. In conclusion, TCM is a new one-dimensional chaotic map with perfect chaotic behaviour over infinite interval, high positive Lyapunov exponent value, uniform distribution, and great sensitivity to any change(s) in the initial condition or the control parameter (see Appendix B).

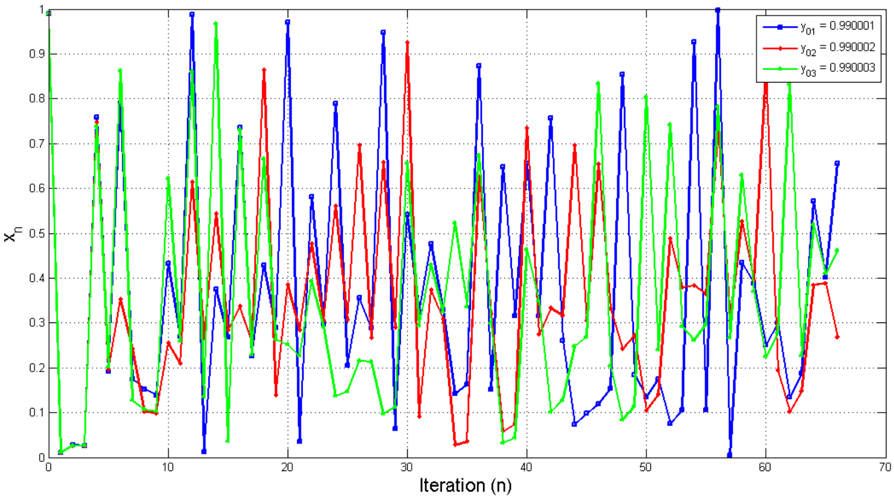


Figure 8-7: TCM iterations with t = 1 and three different initial values of y_0

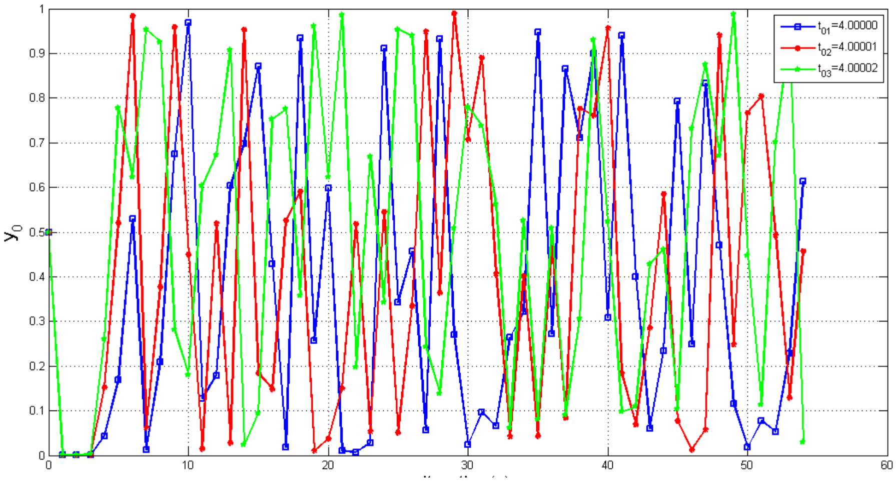


Figure 8-8: TCM iterations with $y_0 = 0.5$ and three different initial values of t

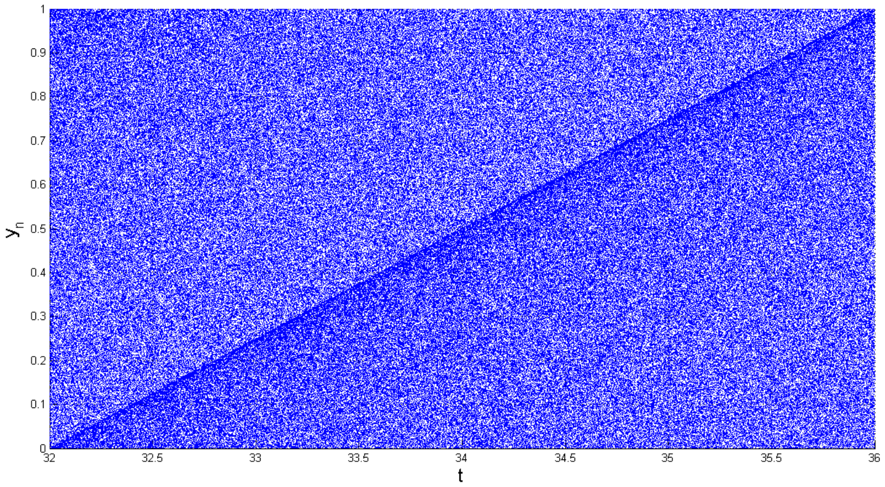


Figure 8-9: TCM chaotic map bifurcation diagram with $t \in [32, 36]$

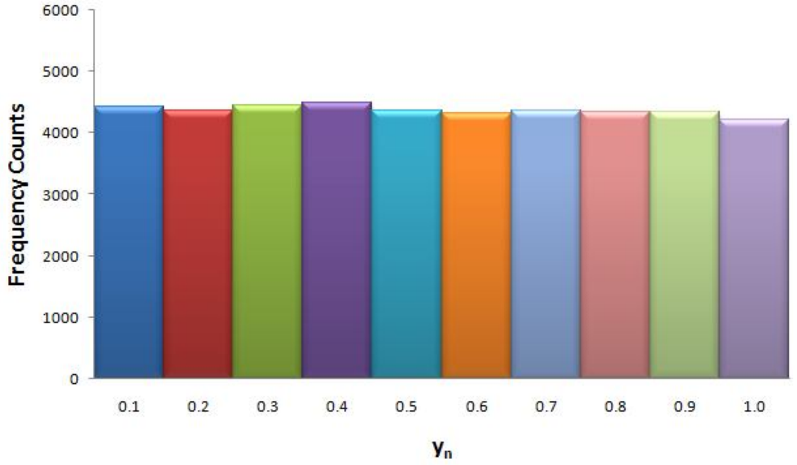


Figure 8-10: TCM distribution of y_n values over $t \in [32, 36]$

8.3 Pseudorandom Number Generators

8.3.1 Constructing a Novel PRNG based on TCM Map (PRNGT)

In this section, we discuss the details of the novel proposed pseudorandom number generator based on TCM (PRNGT). The proposed pseudorandom number generator is based on two TCM maps running side by side, as shown in Figure 8-11. We employ two TCM maps to increase the complexity of the PRNGT generator and prevent information extraction. The initial conditions of the two TCM maps are independent: $r_0 \neq r'_0$, $y_0 \neq y'_0$. After setting the initial values of each map, the TCM maps are iterated a number of times. Each time, the summation mod by one of the output results of the two TCM maps is calculated (see equation 8-3). Threshold is applied to convert the TCM real output value to zero and one as binary bits. We apply equation 8-4 with threshold value 0.5. If the obtained value from equation 8-3 is less than 0.5, the PRNGT will generate zero (0); otherwise it will generate one (1) as binary bit.

$$y'_{n+1} = \begin{cases} (t' y'_n(1 - y'_n)) \bmod 1 & n \bmod 2 = 0 \\ (\pi y'_n \beta') \bmod 1 & n \bmod 2 != 0 \end{cases} \tag{8-2}$$

$$Sum_i = (y_i + y'_i) \ \bmod\ 1, \tag{8-3}$$

$$Z_i = F(Sum_i) = \begin{cases} 1, & if \quad Sum_i >= 0.5 \\ 0, & if \quad Sum_i < 0.5 \end{cases} \tag{8-4}$$

We can briefly express the PRNGT algorithm as follows:

1. Set values ($y_{0i},\ t_{0i}, \beta_i$) and ($y'_{0i},\ t'_{0i}, \beta'_i$).
2. Calculate the two TCM output values (y_{n+1}, y'_{n+1}).
3. Apply equation 8-3 to calculate sum_{n+1}.
4. Z_{n+1} = F(sum_{n+1}).

5. Set $y_n = y_{n+1}$ and $y'_n = y'_{n+1}$.

6. Increment t_i and t'_i by nt'_i, nt_i ; $nt'_i \neq nt_i < 0.01$

7. Do the next iteration.

The sum function aims to mix two TCM output values to produce a third real number and map it from (0, 2) to (0, 1). The generated pseudorandom bit sequence using PRNGT generator will be denoted as $\{Z_i\}_{i=0}^{\infty}$. In each iteration the variables t_i and t'_i will be incremented by a small real number, $nt'_i \neq nt_i < 0.01$, in each map equation. PRNGT generator can be easily implemented in parallel by running the two TCMs either in a parallel system or in a distributed system to provide a faster generator. The proposed pseudorandom number generator produces an infinite number of random bit sequences.

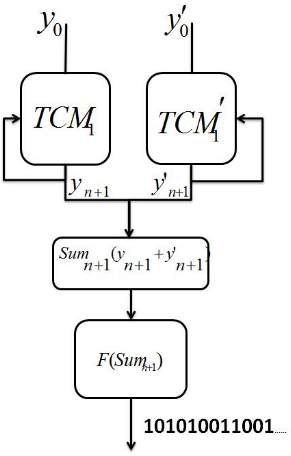


Figure 8-11: Block diagram of the proposed PRNGT generator

8.3.2 Logistic Map Pseudorandom Number Generator (LPRNG)

In this section we briefly explain another proposed pseudorandom number generator based on a logistic chaotic map equation [117]. The authors proposed two pseudorandom number generators based on a logistic map for a cryptography stream cipher. One of the proposed generators is based on two logistic maps to generate a pseudorandom binary sequence. The initial conditions of the two maps should be independent ($r_0 \neq r_0$, $x_0 \neq y_0$), where x_0, $y_0 \in (0, 1)$ and r_0, $r_0 \in (3.99996, 4]$. They calculate the remainder by dividing the sum of the two output values (x_{n+1}, y_{n+1}) by 1 (see equation 8-3). The algorithm generates the binary pseudorandom sequences based on equation 8-4. They tested their proposed pseudorandom number generator using Beker and Piper's suite [273] and FIPS 140-1 suite [274]. They claimed that the proposed random number generator has passed all FIPS 140-1 and Beker and Piper's suites with sequence length 100,000 bits and significance level $\alpha = 0.05$. In this chapter, we refer to the logistic map pseudorandom number generator as LPRNG.

8.3.3 Modified Logistic Map Pseudorandom Number Generator (MLPRNG)

As we explained earlier, a small range of logistic map parameters are considered valid values to show chaotic behaviour $r > 3.57 \geq 4$. In addition, the logistic map population will cover the full interval of x, $x_n \in [0, 1]$, only with $r = 4$. Therefore, we propose to use a modified version of the logistic map defined in equation 8-6. We used the remainder of dividing the logistic map by 1 to ensure that all the output values will be between zero and one, $x_n \in [0, 1]$, and we added a small real number ($\beta \leq 0.001$) to ensure $x_n \neq 0$ or 1. Consequently, in the modified version the value of parameter r can be any value greater than 0, $r \in [0, \infty]$. We plotted the modified version of logistic map bifurcation and its Lyapunov exponent over different intervals using MATLAB software (see Figure 8-12 and Appendix A). It is very clear from Figure 8-12 that the modified version has bigger intervals of chaotic behaviour and it covers the full *x* interval over many different values of parameter *r*. Unfortunately, It still shows non-chaotic areas over different values within the intervals: [0, 4], [4, 8], [8, 12] and [12, 16], which are known as stability or islands. In contrast, the TCM map shows perfect chaotic behaviour and covers the entire range of *y* for every value of t (see Figure 8-13 and Appendix B). In other words, in the triangle-chaotic map at every value of *f(x)* there is at least one image value, but in the logistic map and modified logistic map there are no image values.

$$x_{n+1} = (rx_n(1-x_n)) \bmod 1 + \beta; \qquad (8\text{-}5)$$

where $x_n \in (0, 1)$, $r \in [0, \infty]$, $n \in \mathrm{N}$, $\beta < 0.001$ and x_0 is value of initial population.

In this section, we propose a new pseudorandom number generator based on modified logistic map equation (see equation 8-5). This generator is based on two modified logistic maps running side by side. The initial conditions of the two maps should be independent ($r_0 \neq r_0$, $x_0 \neq y_0$), where x_0, $y_0 \in (0, 1)$ and r_0, $r_0 \in (3.99996, 4]$. We calculate remainder of divide the sum of two output values (x_{n+1}, y_{n+1}) by 1 (see equation 8-3). We used equation 8-3 to generate the binary pseudorandom sequences. Then, we increment the parameters r_x and r_y of the two maps by a small real number and then apply the next iteration. In this paper, we refer to modified logistic map pseudorandom number generator as MLPRNG.

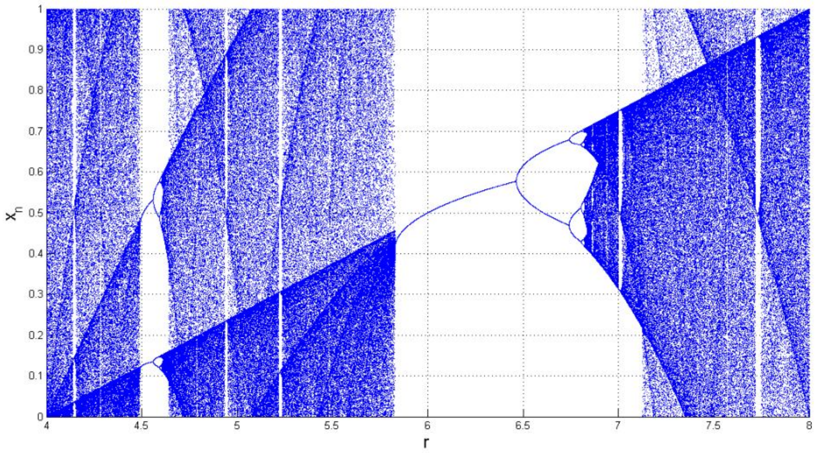


Figure 8-12: Modified logistic map bifurcation diagrams over $r \in [4, 8]$

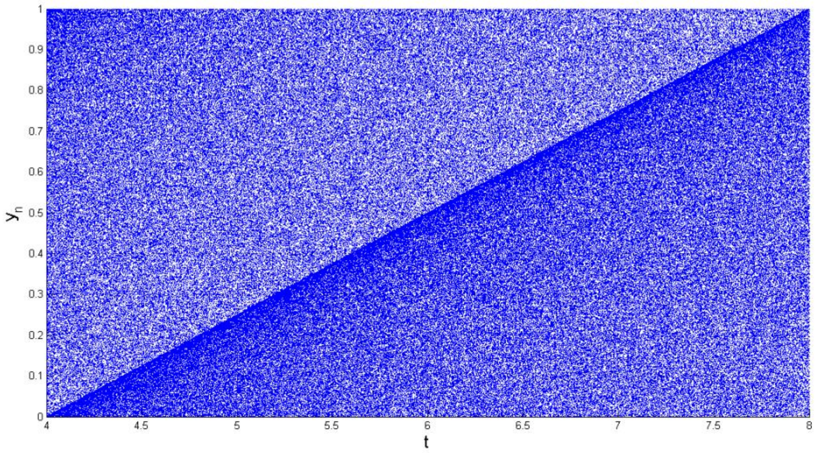


Figure 8-13: TCM map bifurcation diagrams over $t \in [4, 8]$

8.4 NIST Statistical suite test

NIST (National Institute of Standard Technologies) developed a statistical package to test the randomness of generated bit sequences for cryptography applications [211]. It contains 16 tests and each test contains several subtests. The focus of these tests is to confirm the existence of non-randomness in the input bit sequences. A number of tests have chi-square (x^2) and standard normal to refer to distribution. In each test, p-value for each sample in the sequence will be computed, thus m p-values will be computed for each test. A p-value of greater than or equal to $\alpha = 0.01$ would be considered a random sequence with a confidence level of 99%. A p-value of less than $\alpha = 0.01$ would be considered a non-random sequence with a confidence level of 99%.

The recommended significance level (α) value is in the range $0.0001 \leq \alpha \leq 0.01$. For the statistical tests in this paper the value of α has been chosen to be 0.01. If the computed p-value is greater than the chosen significance level ($\alpha = 0.01$), the sequence will be considered a random sequence. In random sequences, if the chosen value of α is 0.01, it would be acceptable to reject only one sequence out of each 100 sequences. The computed p-value$_s$ of p-values should be greater than 0.001 to be considered uniformly distributed. We can broadly classify these tests into two main types: parameterized tests and non-parameterized tests. The parameterized tests have some input variables that the user can change its values, while the variables values of the non-parameterized tests are already set. NIST suite test is described in high-level description in Appendix F.

8.5 Experimental results

In [117], the proposed random number generator was tested using FIPS 140-1 and Beker and Piper’s suites which were published in 1982 and 1994, respectively. The authors claimed that the generated sequences passed all the tests with sequence length of 100,000 bits for several samples and significance level of $\alpha = 0.05$. In 2010 NIST published a package of statistical suite tests of random bits for random and pseudorandom number generators for cryptography applications (NIST 800-22) to check the non-randomness of generated sequences. Typically, they recommended the chosen significance level (α) in the range [0.001, 0.01] and sequence size of at least 20,000 without giving an exact sample size. Therefore, the chosen significance level α = 0.05 is considered an atypical value, which make the test much more easier as it would be acceptable to reject five sequences out of each 100 sequences. Moreover, based on acceptable confidence interval in equation 8-7, the minimum sample size with significance α = 0.05 should be 3250 samples approximately.

Table 8-1: Number of NIST 800-22 test suite sub-tests

No.	Statistical Test	Sub-tests
1	Frequency	1
2	Block Frequency	1
3	Cusum-Forward	1
4	Cusum-Reverse	1
5	Runs	2
6	Long Runs of Ones	1
7	Rank	1
8	FFT	1
9	NonOverlapping Templates	1
10	Overlapping Templates	148
11	Universal	1
12	Approximate Entropy	18
13	Random Excursions	1
14	Random Excursions Variant	1
15	Linear Complexity	1
16	Serial	8

NIST 800-22 suite test consists of 16 main tests aiming to verify the randomness of the input sequences (see table 8-1). Some of these tests are decomposed to sub-tests; thus, the total number of tests is 188 tests in NIST 800-22 suite (see table 8-2). The test aims to confirm the existence of non-randomness in the generated bit sequence. The generated sequences based on the three pseudorandom number generators (LPRNG, MLPRNG, and PRNGT) were tested using NIST SP800-22 suite test to prove their randomness. Significance level (α) and sequence length determine that the pass criteria α is set to 0.01 for all tests of p-values. Each generator is used to generate 1000 samples, $m = 10^3$, of 1-Mbit sequence size, $n = 10^6$. The experimental results confirm that LPRNG and MLPRNG failed in most tests with sample sizes bigger than 10 (see table 8-3). Conversely, the proposed pseudorandom number generator (PRNGT) passed all NIST tests with high p-values that confirm the randomness of the generated sequences (see table 8-4).

Table 8-2: NIST 800 – 22 suite test parameters value

Test Type	Parameter	Value
The Block Frequency	Block Size	128
The Non-overlapping Template	Block Size	9
The Overlapping Template	Block Size	9
Maurer’s “Universal”	Block Size	7
Maurer’s “Universal”	Number of Blocks	1280
The Approximate Entropy	Block Size	10
The Serial	Block Size	16
The Linear Complexity	Sequence Length	500

Table 8-3: NIST statistical test suite for MLPRNG and LPRNG generators with α = 0.01, m = 10^3, and n = 10^6

Statistical Test	LPRNG		MLPRNG	
	P-value	Result	P-value	Result
Frequency	0.000	Failed	0.000	Failed
Block Frequency (m = 128)	0.000	Failed	0.000	Failed
Cusum-Forward	0.000	Failed	0.000	Failed
Cusum-Reverse	0.000	Failed	0.000	Failed
Runs	0.000	Failed	0.000	Failed
Long Runs of Ones	0.000	Failed	0.000	Failed
Rank	0.000	Failed	0.009467	Passed
FFT	0.000	Failed	0.000	Failed
NonOverlapping Templates (m = 9, B = 000000001)	0.000	Failed	0.000	Failed
Overlapping Templates (m = 9)	0.000	Failed	0.000	Failed
Universal	0.000	Failed	0.000	Failed
Approximate Entropy (m = 10)	0.000	Failed	0.000	Failed
Random Excursions (x = +1)	0.213309	Passed	0.593823	Passed
Random Excursions Variant (x = -1)	0.122325	Passed	0.357274	Passed
Linear Complexity (M = 500)	0.000	Failed	0.002447	Passed
Serial (m = 16, $\nabla\Psi\frac{2}{m}$)	0.000	Failed	0.000	Failed

Distribution histograms have been plotted for the resulting p-value from each test of NIST suite test for generated sequences using PRNGT. If the calculated p-value$_p$ of the p-values ≥ 0.001, it will considered uniformly distributed. 1000 samples of binary sequences have been generated by the proposed generator and each one is of size 10^6 ($n = 10^6$ and $m = 10^3$). P-value interval [0, 1] is divided into 10 equal sub-intervals and the number of p-values are counted for each sub-interval and plotted for each test (see Figures 8-14 and 8-15). It is clear from all calculated p-value$_p$ of p-values, p-value$_p$≥ 0.0001 and p-values are uniformly distributed over the interval [0, 1]. Moreover, the proportions of passed test sequences have been calculated for each NIST test. Then, we compare these values with acceptable confidence interval of p-values. The quantitative analysis test results show that proportions lie inside the acceptable confidence interval [0.980561, 0.999439] as shown in Figure 8-16.

Table 8-4: NIST statistical test suite for PRNGT generator with $\alpha = 0.01$, $m = 10^3$, and $n = 10^6$

Statistical Test	P-value	PRNGT Proportion	Conclusion
Frequency	0.298282	0.985	Passed
Block Frequency (m = 128)	0.190654	0.990	Passed
Cusum-Forward	0.676615	0.985	Passed
Cusum-Reverse	0.166260	0.986	Passed
Runs	0.989425	0.992	Passed
Long Runs of Ones	0.628790	0.986	Passed
Rank	0.025708	0.997	Passed
FFT	0.966626	0.988	Passed
NonOverlapping Templates (m = 9, B = 000000001)	0.587274	0.991	Passed
Overlapping Templates (m = 9)	0.068571	0.988	Passed
Universal	0.461612	0.989	Passed
Approximate Entropy (m = 10)	0.007694	0.993	Passed
Random Excursions (x = +1)	0.587748	0.989	Passed
Random Excursions Variant (x = -1)	0.917766	0.994	Passed
Linear Complexity (M = 500)	0.796268	0.988	Passed
Serial (m = 16, $\nabla\Psi\frac{2}{m}$)	0.911413	0.989	Passed

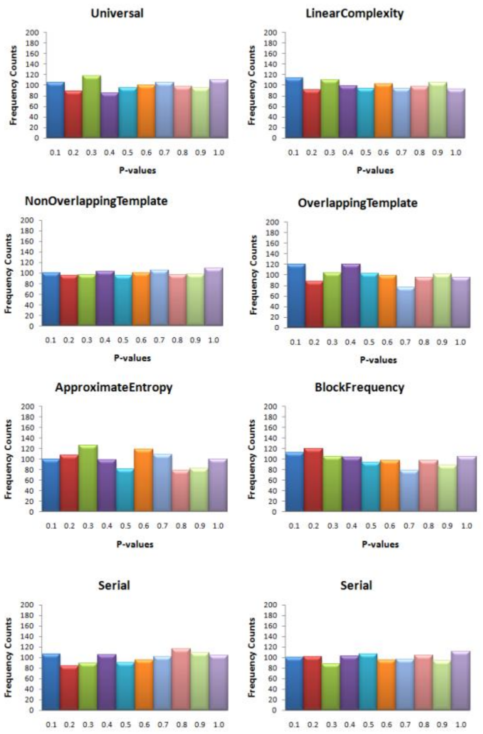


Figure 8-14: P-values histograms of parameterized NIST 800-22 suite tests

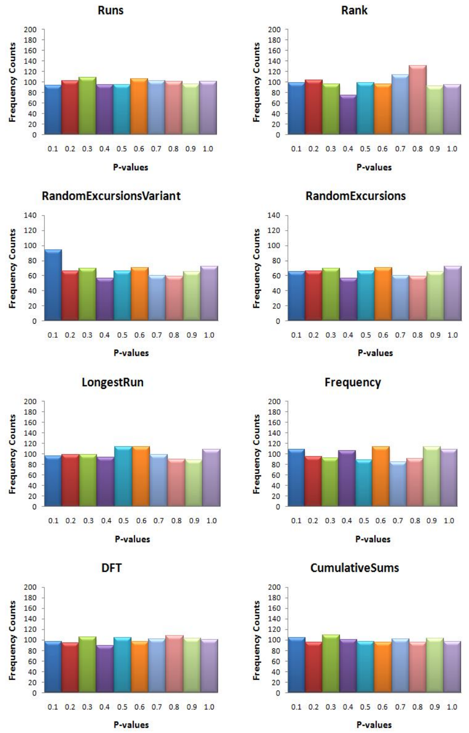


Figure 8-15: P-values histograms of non-parameterized NIST 800-22 suite tests.

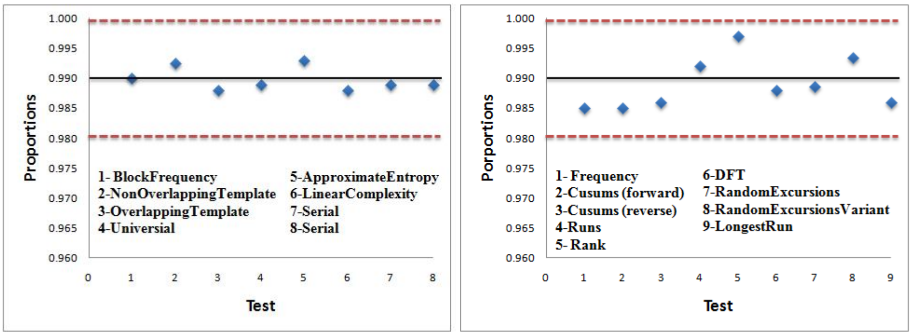


Figure 8-16: PRNGT proportions of sequences passing NIST 800-22 suite test for (a) non-parameterized tests (b) parameterized tests. The acceptable proportions range between the dashed lines.

8.6 Conclusion

In this chapter, we proposed a new triangle-chaotic map (TCM) with high-intensity chaotic areas over infinite interval. The proposed chaotic map has very strong chaotic properties such as very high sensitivity to initial conditions, random-like, uniformly distributed population, deterministic nature, unpredictability, high positive Lyapunov exponent values, and perfect chaotic behaviour over infinite positive interval. TCM chaotic map is a one-way function that prevents the finding of a relationship between the successive output values and increases the randomness of output results. Furthermore, a novel pseudorandom number generator based on TCM map (PRNGT) for high security applications is proposed that can be implemented in parallel to achieve high efficiency. The proposed generator produces high-randomness pseudorandom sequences that were confirmed by passing all NIST 800-22 suite standard statistical tests with high p-values and excellent statistical properties. Moreover, we analyzed the security and statistical properties of generated sequences based on logistic map (LPRNG) and modified logistic map (MLPRNG) using NIST 800-22 suite test. The analysis results confirmed that the generated sequences based on the two generators are non-random with sample size bigger than 10. TCM characteristics are encouraging for possible utilization in designing new security primitives.

Chapter 9

Conclusion and Future Works

In this chapter, a summary of our work in this thesis is given and can be classified into five parts: cryptanalysis of chaotic hash function; designing new chaos-based hash functions; designing new chaos-based encryption algorithm; designing new chaos-based pseudorandom number generator; cryptanalysis of chaotic pseudorandom number generators; and new chaotic system. Moreover, remarks and future directions on designing chaos-based security primitives are given.

9.1 Conclusion of this Thesis

Over recent years, electronic commerce has developed amazingly rapidly; people have become dependent on the Internet to use many electronic commerce services include online shopping, order-tracking, online payments and online money transfers. The security of electronic commerce is very necessary as it is considered the most important requirement for the success of electronic commerce. The integrity of many traditional security algorithms remains at risk including MD4, MD5, SHA1, DES, Triple DES, RC6, AES and many other algorithms. Therefore, designing new security algorithms is needed for electronic commerce to grantee safety of information in transactions over the Internet (integrity), identification of parties involved in online transactions (authentication), ensuring that only authorized people can access the information (confidentiality), and preventing denial of the transaction (non-repudiation).

Customers and merchants are very concerned about the security of electronic commerce applications such as Internet banking, online shopping and online payments. The security of electronic commerce is built based on the security of hash functions, encryption algorithms and pseudorandom number generators. These primitives are used to ensure the integrity of data transactions, privacy of sensitive or personal information of sellers and buyers, non-repudiation of buyers and authentication of sellers. Therefore, the construction of fast and secure security primitives is essential to protect electronic commerce applications and protocols. The construction of security primitives based on chaotic systems would provide a high level of security and performance for electronic commerce.

To this end, designing fast and secure new security algorithms based on chaotic systems to guarantee integrity, authentication and confidentiality is essential for electronic commerce development. In this thesis, we explored comprehensively the analysis and design of security primitives based on chaotic systems for electronic commerce: hash functions, encryption algorithms and pseudorandom number generators. The main purpose of this thesis is to design new, fast and secure hash functions, encryption algorithms and pseudorandom number generators based on chaotic systems for electronic commerce. In addition, we have studied and analyzed the security of chaos-based hash functions and pseudorandom number generators.

9.1.1 Contribution of this Thesis

This thesis involves several aspects of analysis and design of chaos-based cryptography algorithms for electronic commerce: hash functions, block cipher and pseudorandom number generators. One of the proposed hash functions and one pseudorandom number generator are analyzed. Moreover, two hash functions, block cipher encryption algorithm, pseudorandom number generator and chaotic map are proposed. The original key contributions in this thesis are addressed by the following:

1- A chaotic hash function algorithm based on a tent map (CBHF) was proposed. We realized that the proposed hash function has a weak design and directly exploited to tent map. Therefore, we carefully studied its design and then analyzed its resistance to collisions. In this research, we explained in detail how to break keyed and unkeyed versions of CBHF mathematically. Moreover, real collision examples of keyed and unkeyed versions were given.

2- After analyzing the security of the CBHF hash function, we designed a fast parallel hash function based on chaotic maps for electronic commerce applications. Simple chaotic maps are used with the control parameter of a positive Lyapunov exponent value to provide high security and better performance. Several analyses and computer simulations are performed to show the security and performance of our proposed hash function. The proposed hash function is satisfying the cryptographic hash function characteristics. Moreover, the proposed hash function is compared with other proposed hash function algorithms and a comparison between parallel and sequential modes is made. Overall comparison results show that the proposed parallel hash function algorithm outperforms other algorithms.

3- In recent years, several chaotic block cipher encryption algorithms have been proposed. Some of the proposed chaotic encryption algorithms had major problems such as vulnerability to different types of attacks, inflexibility and slow speed. This motivated us to design and implement a new, fast block cipher encryption algorithm based on chaotic maps (BCCM) by utilizing chaotic maps for electronic commerce applications. The security and performance of the proposed algorithm is analyzed using well-known computer simulations and theoretical analysis in this field. Both experimental results and computer simulations confirm that the proposed chaotic block cipher encryption algorithm satisfies the cryptographic properties.

4- Experimental results confirmed that the whole encryption process of BCCM algorithm is extremely sensitive to any change or changes in the input message and/or secret key. These properties suggest that the BCCM encryption function could be adopted in designing a new cryptography hash function. We propose a fast chaotic hash function based on BCCM encryption algorithm with changeable parameters (HBCCM). The proposed hash function produces different lengths of hash values based on different chaotic systems with variable parameters. Comparing the results of the proposed hash function with other cryptographic hash functions shows that the proposed hash function has a performance equivalent to that of SHA-1. These characteristics confirm that this hash function is practical and reliable with high potential to be adopted for secure and fast applications.

5- Most of the well-known chaotic maps show chaotic behaviour over small regions with certain parameter values. Moreover, within the chaotic areas there are some regions of n-periodic window with small values of n (2, 3, 6…), which can be exploited by attackers to help them attack the cryptographic systems that fall in these regions. Therefore, we designed a new triangle-chaotic map with full chaotic population and a very large value of periodic window. Triangle-chaotic map analysis shows its perfect chaotic behaviour, great sensitivity to initial conditions, unpredictability, and intensive chaotic population with full positive Lyapunov exponent values over an infinite interval. These properties confirm that the proposed triangle-chaotic map can be adopted for many applications in many disciplines such as high-security applications, computer science, engineering, mathematics, physics and economics.

6- A cryptographic secure pseudorandom number generator is used to provide high security for several cryptographic applications such as key generation and digital signature. In this research, a new cryptographic secure pseudorandom number generator based on the proposed triangle-chaotic map is presented. The PRNGT generator is designed based on a combination of two TCM maps running side by side and starting from independent initial conditions. The proposed generator was tested using the well-known NIST 800-22 suite test, which is designed to test cryptographic random and pseudorandom number generators, and compared with two other generators based on a logistic map and modified version of a logistic map, respectively.

The tests and comparison results showed randomness of the proposed generator output and non-randomness of the other two logistic generators' outputs. Therefore, the proposed generator is a high-potential candidate for high-security applications such as online payments and online banking systems.

9.1.2 Results and Discussion of Individual Chapters

Here is a brief summary of each chapter of this thesis.

- **Chapter 4: Cryptanalysis of Chaos-based Hash Function**

In 2009, Amin et al. suggested a new hash function based on chaotic systems for cryptography applications (CBHF). In this chapter, we show how to break the recently proposed unkeyed hash function based on chaos theory (CBHF). Our attacks show that we can easily find two different messages that have the same hash value. In addition, we show how to break the keyed version of CBHF by adding different blocks at the end of the original message without changing the final hash value. We can find a large number of collisions for each message using our analysis technique. Therefore, neither keyed nor unkeyed hash versions of CBHF are at all secure. Advanced research on chaotic hash functions is needed to design a new family of secure hash functions for cryptography and e-commerce applications.

- **Chapter 5: Fast Parallel Hash Functions Based on Chaotic Maps for eCommerce (PHFC).**

In this chapter, we propose a fast, parallel, chaos-based hash function called PHFC. Coupled chaotic maps with high chaotic behaviour are used in design PHFC functions: subkey generation, round hash function, and chaotic hash mixing function. PHFC is a very flexible hash function that generates different lengths of hash values using different chaotic maps: PHFC-128, PHFC-160, PHFC-224, PHFC-256, PHFC-384, and PHFC-512. The proposed hash function can be implemented as keyed and unkeyed hash function based on chaotic systems. We have compared the proposed hash function with two well-know hash functions (SHA-1, MD5) and five other chaos-based hash functions; the comparison results showed that PHFC outperforms many other existing hash functions. We performed several analyses and computer simulations on PHFC hash function, including hash value sensitivity to any change(s), hash space distribution, statistical analysis of diffusion and confusion, collision resistance analysis, number of hash rounds, implementation and flexibility, and speed analysis to verify its characteristics. The simulations and analyses results show that PHFC satisfies the characteristics and conditions of cryptography hash functions such as collision resistance, high bit confusion and diffusion, uniform distribution, flexibility and fast speed. PHFC is practical and reliable, and has efficient hash functions with high potential for adoption in e-commerce applications and protocols.

- **Chapter 6: Fast Encryption Algorithm Based on Chaotic Maps for eCommerce (BCCM)**

In this chapter, we proposed a novel, fast block cipher encryption algorithm based on chaotic systems. The proposed algorithm (BCCM) works with different parameters such as block size, number of rounds, secret key length and chaotic maps. BCCM processes n-bits of plaintext and produces the same size of ciphertext with k-bit key size. It generates several subkeys from a single secret key using a simple one-dimensional chaotic map. The proposed algorithm has chaotic subkeys generation, chaotic columns-mixing and chaotic rows-shifting that provide high security and confusion and diffusion properties. We performed several analyses and computer simulations on the BCCM encryption algorithm, including images encryption and decryption, modes of operation effects, correlation coefficient analysis, information entropy analysis, and execution time with different parameters to verify its characteristics. The simulations and analyses results show that the algorithm satisfies the characteristics and conditions of a secure cryptography block cipher. Moreover, we have compared the BCCM with two well-know block ciphers (DES, RC6) and one chaos-based block cipher; the comparison results showed that BCCM outperforms these algorithms. BCCM is confirmed as a good candidate for texts/images encryption with high flexibility, confusion and diffusion, and fast speed. These characteristics confirmed that the proposed algorithm has high potential for adoption for e-commerce.

- **Chapter 7: Fast Hash Function Based on BCCM Encryption Algorithm for eCommerce (HBCCM)**

In this chapter, we propose HBCCM as a new hash function based on BCCM block cipher encryption algorithm. HBCCM is very simple, flexible and sensitive to any change (s) in the input message of the secret key. It processes input messages of different lengths to generate different hash value lengths based on different chaotic systems. Moreover, it can be used to calculate the correspondence hash value of input messages as texts or images with high confusion and diffusion properties. We performed several analyses and computer simulations on HBCCM hash function, including hash value sensitivity to any change(s), hash space distribution for image and text as an input, statistical analysis of diffusion and confusion, collision resistance analysis, number of hash rounds, implementation and flexibility, and speed analysis to verify its characteristics. The simulations and analyses results show that the algorithm satisfies the characteristics and conditions of cryptography hash functions. Comparisons between HBCCM and other hash functions show that HBCCM outperforms all the other algorithms. HBCCM is fast hashing scheme, with high potential for adoption for e-Commerce applications.

- **Chapter 8: A Novel Pseudorandom Number Generator Based on New Triangle-Chaotic Map for High Security Applications**

Some existing chaotic maps have several limitations such as small size of periodic and partial chaotic population. In this chapter, we proposed a new triangle-chaotic map (TCM) with high-intensity chaotic areas over infinite interval. The proposed chaotic map has very strong chaotic properties such as very high sensitivity to initial conditions, random-like, uniformly distributed population, deterministic nature, unpredictability, high positive Lyapunov exponent values, and perfect chaotic behaviour over infinite positive interval. TCM chaotic map is a one-way function that prevents the discovery of the relationship between the successive output values and increases the randomness of output results. Furthermore, a novel pseudorandom number generator-based TCM map (PRNGT) for cryptography application is proposed that can be implemented in parallel to achieve high efficiency. The proposed generator produces highly random pseudorandom sequences that are confirmed by passing all NIST 800-22 suite standard statistical tests with high p-values and excellent statistical properties. Moreover, we analyzed the security and the statistical properties of generated sequences based on a logistic map (LPRNG) and a modified logistic map (MLPRNG) using NIST 800-22 suite test. The analysis results confirmed that the generated sequences based on the two generators are non-random with sample size bigger than 10. TCM characteristics are encouraging in terms of its potential utilization in designing new security primitives.

9.2 Perspective of Future Research

There are three very important challenges in the chaotic cryptography research area: security, performance and stability within chaotic areas. One essential requirement of any cryptographic system is its security and resistance to any kind of attack. Unfortunately, some of the proposed chaotic cryptography algorithms suffer from either security or performance problems, whereas some chaotic cryptography algorithms provide high security at the expense of performance. Therefore, those two requirements must be considered together to design fast and secure chaotic cryptography algorithms. Moreover, most chaotic systems exhibit chaotic behaviour with specific parameter values and, within the chaotic areas, non-chaotic areas still exist. Unfortunately, this could be utilized by researchers to break chaotic cryptography systems. Thus, the choice of chaotic system and its parameters must be made based on deep studies and analysis.

9.2.1 Remarks for Designing a Good Chaotic Cryptography

There are many important issues that should be considered in chaotic cryptography. Providing high security and good preference in chaotic cryptography is very important, and these mainly depend on the selection of a good chaotic system. Evaluating certain dynamical properties, security, and performance of chaotic cryptography using appropriate analyses and computer simulations is essential. Remarks on future research in chaotic cryptography are listed below:

1. A well-designed cryptography system provides high security and fast computation speed. Security and performance of chaos-based cryptography systems mainly depend on the selected chaotic system and the algorithm’s internal structure, and/or secret key. Smaller computation cost is achieved by selecting a simpler chaotic system, but evaluating certain dynamical properties of the chaotic system is crucial. Parallel implementation would be useful to implement complex chaotic cryptography systems that provide fast and secure cryptography for many applications such as e-commerce and online banking.

2. Chaotic system parameters are chosen carefully with a large positive Lyapunov exponent value and no stability islands. A chaotic system with a large exponent value means that the chaotic system with certain parameter values is sensitive to initial condition and behaviour chaotically. A logistic map is one of the simplest chaotic systems, which shows chaotic behaviour over a small parameter range $(3.57 \geq r \geq 4)$ and full chaotic population with $r = 4$. It shows non-chaotic behaviour (stability) within the chaotic interval.

3. Uniform distribution of most chaotic cryptosystems depends on the properties distribution of the chaotic system orbit. A chaotic system with non-uniform distribution reduces the quality of the diffusion property. Moreover, the security of some encryption algorithms is based on chaotic pseudorandom sequence. Thus, if the chaotic system has non-uniform distribution, the relationship between plaintext and cipher text will be deduced.

4. The secret key is composed of several subkeys in some chaotic cryptography systems. Sometimes, partial knowledge of the key can lead to the deduction of all the other subkeys; this can be prevented if the different subkeys are uncorrelated. In some chaotic cryptography systems, the subkey generation is based on a chaotic system, by iterating the chaotic system using the secret key as initial condition.

5. The secret keys of chaotic cryptography systems are the values of initial condition and the control parameter of the chaotic system, or these values are

derived from the secret key. The link between secret key and the control parameters has to be established carefully to ensure that the chaotic system is evolving in a chaotic way. Otherwise, the chaotic system evolves non-chaotically and the security level of the chaotic cryptography system can be reduced significantly. Large secret key parameter space is very important to prevent an exhaustive search attack. Moreover, generation of subkeys based on a chaotic system should be with a parameter value of large Lyapunov exponent value.

6. Implementing chaotic systems on digital computers using finite-precision may cause dynamical degradation as the dynamical properties of computers are different from the theoretical ones. Consideration of this issue is very important as it could affect the performance and security of the chaotic cryptography system. Higher precisions, cascading multiple chaotic systems or random perturbation of the chaotic systems could be used to solve the digital degradation problem. Cascading multiple chaotic systems could provide highly secure chaotic cryptography systems, but at the expense of computation time.

7. In chaos-based cryptography, using a single, simple operation such as XOR operation mixed with chaotic systems could help attackers to break the system. Moreover, the padding process of adding one "1" followed by a number of zeros "0" $(100\ldots0)_2$ could help attackers to find collisions; this could be solved by padding the length of the input message at the end of the last block.

9.2.2 Future Work

In closing this chapter, possibilities for future work directions are summarized as follows:

- **Design and analysis chaos-based public key algorithms**

Cryptography algorithms are divided into three main categories: cryptography hash function, symmetric cryptosystem, and asymmetric cryptosystem. A cryptosystem is a cryptographic algorithm that depends on certain parameters called keys. Cryptography hash function can be further divided into two main categories: keyed hash function and unkeyed hash function. In addition, a symmetric cryptosystem uses one key to encrypt and decrypt messages that can be further divided into block cipher and stream cipher. On the other hand, an asymmetric cryptosystem uses two keys to encrypt/decrypt an input message. In this thesis, we analyse and design chaos-based keyed hash function, unkeyed hash function, block cipher and stream cipher. A few chaos-based asymmetric algorithms have been proposed in the literature, but they are not of the standard level as they lack deep study and analyses. Therefore, the analysis and design of asymmetric key cryptosystem based on chaotic system is a promising route to the designing of very secure and fast algorithms for high-security applications.

- **Cryptanalysis some chaos-based cryptography algorithms**

Over the past two decades, a tremendous number of papers have been published on chaos-based cryptography including block ciphers, cryptography hash functions and pseudorandom number generators. Simultaneously, many cryptanalytic researchers have analyzed some of the proposed chaos-based cryptography algorithms which proved to be insecure and/or slow. Therefore, we believe deep analyses of these algorithms are still needed to attack and/or improve the insecure algorithm; this will be reflected optimistically in the progress of this research area.

- **Adopting Triangle-Chaotic Map**

A novel triangle-chaotic map designed with full chaotic population over infinite interval is proposed. Triangle-chaotic map analysis shows its perfect chaotic behaviour, great sensitivity to initial conditions, unpredictability, and intensive chaotic population with full positive Lyapunov exponent values over infinite interval. Therefore, adopting the triangle-chaotic map in designing new security primitives could provide cryptography algorithms with high security. Moreover, it can be adopted in many disciplines including computer science, engineering, mathematics, physics and economics.

- **Design and analysis chaos-based stream cipher algorithms**

In chapter 8, we propose a novel pseudorandom number generator based on a chaotic system. The proposed generator can be used to provide high security for many cryptography systems such as digital signature, secret key of DES algorithm, and prime numbers in RSA. Moreover, it can be used in stream cipher encryption algorithms as a keystream by simply XORing the keystream with the plaintext to provide ciphertext. Therefore, we believe further research can be conducted to exploit the proposed generator in security systems and application to provide high security.

Bibliography

1. M.Y. Rhee, *Internet Security Cryptographic Principles, Algorithms and Protocols*. 2003, Republic of Korea: Seoul National University: John Wiley & Sons Ltd.

2. Nabi, F., *Secure business application logic for e-commerce systems.* Computers & Security, 2005. **24**: p. 208-217.

3. Turban, E., D.K.D. Viehland, and J. Lee, *Electronic Commerce A Managerial Perspective 2006*. 2006: Pearson Prentice Hall.

4. Belanger, F., J.S. Hiller, and W.J. Smith, *Trustworthiness in electronic commerce: the role of privacy, security, and site attributes.* Journal of Strategic Information Systems 2002. **11**: p. 245-270.

5. Sundt, C., *Information security and the law.* Information Security Technical Report, 2006. **11**: p. 2-9.

6. Stallings, W., *Cryptography and Network Security: Principles and Practice*. Second ed. 2003, USA: Prentice Hall.

7. Mel, H.X. and D. Baker, *Cryptography Decrypted*. First ed. 2001: Addison Wesley.

8. Schneier, B., *Applied Cryptography*. Second ed. 1996, USA: Wiley.

9. Talbot, J. and D. Welsh, *Complexity and Cryptography An Introduction*. First ed. 2006, New York: Cambridge University Press.

10. Menezes, A., P.v. Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. 1997, FL: CRC Press, Boca Raton.

11. Ferguson, N. and B. Schneier, *Practical Cryptography*. First ed. 2003, USA: Wiley.

12. W. Diffie and M.E. Hellman, *New directions in cryptography.* IEEE Transactions on Information Theory, 1976. **IT-22**(6): p. 644-654.

13. Kotulski, Z. and J. Szczepanski, *Discrete chaotic cryptography (DCC).* Annalen der Physik 1997. **6**(5): p. 381-394.

14. Rhee, M.Y., *Internet Security Cryptographic Principles, Algorithms and Protocols*. 2003, Republic of Korea: Seoul National University: John Wiley & Sons Ltd.

15. Liaw, H.-T., J.-F. Lin, and W.-C. Wu, *A new electronic traveler's check scheme based on one-way hash function.* Electronic Commerce Research and Applications, 2007. **6**(4): p. 499-508.

16. Al-Slamy, N.M.A., *E-Commerce security.* IJCSNS International Journal of Computer Science and Network Security, 2008. **8**(5).

17. Alexandris, N., et al., *Secure linking of customers, merchants and banks in electronic commerce.* Future Generation Computer Systems, 2000. **16**(4): p. 393-401.

18. Zhaofu, T., X. Ningning, and P. Wuliang. *E-Commerce Security: A Technical Survey*. in *Intelligent Information Technology Application, 2008. IITA '08. Second International Symposium on*. 2008.

19. Sengupta, A., C. Mazumdar, and M. Barik, *e-Commerce security - A life cycle approach.* Sadhana, 2005. **30**(2): p. 119-140.

20. Zhou, J. and H. Xie. *E-Commerce Security Policy Analysis.* in *Electrical and Control Engineering (ICECE), 2010 International Conference on.*

21. Praveen Gauravaram, Adrian McCullagh, and E. Dawson, *Collision Attacks on MD5 and SHA-1: Is this the "Sword of Damocles" for Electronic Commerce?*, in *AusCERT Asia Pacific Information Technology Security Conference Refereed R & D Stream*. May, 2006.

22. He, Y. and J. Jiang. *E-commerce security payment system research and implementation*. in *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on.*

23. Chen, T. and X. Xu. *Digital signature in the application of e-commerce security*. in *E-Health Networking, Digital Ecosystems and Technologies (EDT), 2010 International Conference on.*

24. Ghesmati, S., M. Sate, and A. Asosheh. *A 2-level model for e-commerce security*. in *Internet Security (WorldCIS), 2011 World Congress on.*

25. Sneyers, R., *Climate Chaotic Instability: Statistical Determination and Theoretical Background.* Environmetrics, 1997. **8**(5): p. 517-532.

26. Zeng, X., R.A. Pielke, and R. Eykholt, *Chaos theory and its application to the Atmosphere.* Bulletin of the American Meteorological Society, 1993. **74**(4): p. 631-639.

27. Wikipedia. *Chaos theory*. [cited January 17, 2009]; Available from: http://en.wikipedia.org/w/index.php?title=Chaos_theory&oldid=264934743.

28. Serletis, A. and P. Gogas, *Purchasing power parity, nonlinearity and chaos.* Applied Financial Economics, 2000. **10**: p. 615-622.

29. SERLETIS, A. and P. GOGAS, *Chaos in East European black market exchange rates.* Research in Economics, 1997. **51**: p. 359–385.

30. Maqableh, M., A.B. Samsudin, and M.A. Alia, *New Hash Function Based on Chaos Theory (CHA-1).* IJCSNS International Journal of Computer Science and Network Security 2008. **8**(2): p. 20-26.

31. Kocarev, L., *Chaos-based cryptography: a brief overview.* Circuits and Systems Magazine, IEEE, 2001. **1**(3): p. 6.

32. Pareek, N.K., V. Patidar, and K.K. Sud, *Discrete chaotic cryptography using external key.* Physics Letters A, 2003. **309**(1-2): p. 75.

33. Pareek, N.K., V. Patidar, and K.K. Sud, *Cryptography using multiple one-dimensional chaotic maps.* Communications in Nonlinear Science and Numerical Simulation, 2005. **10**(7): p. 715-723.

34. Xianga, T., et al., *A novel block cryptosystem based on iterating a chaotic map.* Physics Letters A, 2006. **349**(1-4): p. 109.

35. Chen Shuai, Zhong XianXin, and W. ZhengZhong, *Chaos Block Cipher for wireless sensor network.* Science in China Series F: Information Sciences, 2008. **51**(8): p. 1055-1063.

36. Jun, P., et al. *Research on a Block Encryption Cipher Based on Chaotic Dynamical System*. in *Natural Computation, 2007. ICNC 2007. Third International Conference on*. 2007.

37. Yang, H., et al., *A new block cipher based on chaotic map and group theory.* Chaos, Solitons & Fractals, 2007. **40**(1): p. 50-59.

38. Like, C. and Z. Runtong. *A fast encryption mode for block cipher with integrity authentication*. in *Service Operations and Logistics, and Informatics, 2008. IEEE/SOLI 2008. IEEE International Conference on*. 2008.

39. Xua, S., J. Wang, and S. Yang. *A Novel Block Cipher Based on Chaotic Maps*. in *Image and Signal Processing, 2008. CISP '08. Congress on*. 2008.

40. Jun, P., et al. *An Image Encryption Scheme Based on Chaotic Map*. in *Natural Computation, 2008. ICNC '08. Fourth International Conference on*. 2008.

41. Lian, S., *A block cipher based on chaotic neural networks.* Neurocomputing, 2009. **72**(4-6): p. 1296-1301.

42. Fengjian, W., Z. Yongping, and C. Tianjie. *Research of Chaotic Block Cipher Algorithm Based on Logistic Map*. in *Intelligent Computation Technology and Automation, 2009. ICICTA '09. Second International Conference on*. 2009.

43. Yang, D., et al., *A novel chaotic block cryptosystem based on iterating map with output-feedback.* Chaos, Solitons & Fractals, 2009. **41**(1): p. 505-510.

44. Wang, X.-y. and Q. Yu, *A block encryption algorithm based on dynamic sequences of multiple chaotic systems.* Communications in Nonlinear Science and Numerical Simulation, 2009. **14**(2): p. 574.

45. Zhou, Z., et al. *A Block Encryption Scheme Based on 3D Chaotic Arnold Maps*. in *Intelligent Interaction and Affective Computing, 2009. ASIA '09. International Asia Symposium on*. 2009.

46. Jun, P., et al. *A Block Cipher Based on a Hybrid of Chaotic System and Feistel Network*. in *Natural Computation, 2009. ICNC '09. Fifth International Conference on*. 2009.

47. Amin, M., O.S. Faragallah, and A.A. Abd El-Latif, *A chaotic block cipher algorithm for image cryptosystems.* Communications in Nonlinear Science and Numerical Simulation, 2010. **15**(11): p. 3484-3497.

48. Akhshani, A., et al., *A novel scheme for image encryption based on 2D piecewise chaotic maps.* Optics Communications, 2010. **283**(17): p. 3259-3266.

49. Wang, X.-y., F. Chen, and T. Wang, *A new compound mode of confusion and diffusion for block encryption of image based on chaos.* Communications in Nonlinear Science and Numerical Simulation, 2010. **15**(9): p. 2479-2485.

50. Jian-Hua, H. and L. Yang. *A block encryption algorithm combined with the Logistic mapping and SPN structure*. in *Industrial and Information Systems (IIS), 2010 2nd International Conference on.*

51. Zhao, G., et al., *Block Cipher Design: Generalized Single-Use-Algorithm Based on Chaos.* Tsinghua Science & Technology, 2011. **16**(2): p. 194-206.

52. Masuda, N., et al., *Chaotic block ciphers: from theory to practical algorithms.* Circuits and Systems I: Regular Papers, IEEE Transactions on, 2006. **53**(6): p. 1341.

53. Phan, R.C.W. and D. Wagner, *Security considerations for incremental hash functions based on pair block chaining.* Computers & Security, 2006. **25**(2): p. 131-136.

54. Jun, P., L. Xiaofeng, and Y. Zhiming. *A Novel Feedback Block Cipher Based on the Chaotic Time-Delay Neuron System and Feistel Network*. in *Communications, Circuits and Systems Proceedings, 2006 International Conference on*. 2006.

55. Habutsu, T., et al., *A secret key cryptosystem by iterating a chaotic map*, in *Proceedings– EuroCrypt'91*. 1991: Berlin. p. 127-140.

56. Gutowitz, H.A., *Cryptographywith dynamical systems*, in *Cellular Automata and Cooperative Phenomena*. 1993, Kluwer Academic Press.

57. Masuda, N. and K. Aihara. *Cryptosystems based on space-discretization of chaotic maps*. in *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*. 2001.

58. García, P. and J. Jiménez, *Communication through chaotic map systems.* Physics Letters A, 2002. **298**(1): p. 35-40.

59. Masuda, N. and K. Aihara, *Cryptosystems with discretized chaotic maps.* Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, 2002. **49**(1): p. 28.

60. Baptista, M.S., *Cryptography with chaos.* Physics Letters A, 1998. **240**(1-2): p. 50.

61. Wai-kit, W., L. Lap-piu, and W. Kwok-wo, *A modified chaotic cryptographic method.* Computer Physics Communications, 2001. **138**(3): p. 234.

62. Alvarez, E., et al., *New approach to chaotic encryption.* Physics Letters A, 1999. **263**(4-6): p. 373-375.

63. Wong, K.W., *A fast chaotic cryptographic scheme with dynamic look-up table.* Physics Letters A, 2002. **298**(4): p. 238.

64. Machado, R.F., M.S. Baptista, and C. Grebogi, *Cryptography with chaos at the physical level.* Chaos, Solitons & Fractals, 2004. **21**(5): p. 1265.

65. Guan, Z.-H., F. Huang, and W. Guan, *Chaos-based image encryption algorithm.* Physics Letters A, 2005. **346**(1-3): p. 153.

66. Chen, G., Y. Mao, and C.K. Chui, *A symmetric image encryption scheme based on 3D chaotic cat maps.* Chaos, Solitons & Fractals, 2004. **21**(3): p. 749.

67. Gao, T. and Z. Chen, *Image encryption based on a new total shuffling algorithm.* Chaos, Solitons & Fractals, 2008. **38**(1): p. 213.

68. Martínez-Ñonthe, J.A., et al., *Chaotic block cryptosystem using high precision approaches to tent map.* Microelectronic Engineering. **In Press, Accepted Manuscript**.

69. Kwok-Wo, W., *A combined chaotic cryptographic and hashing scheme.* Physics Letters A, 2003. **307**(5-6): p. 292.

70. Xiao, D., X. Liao, and S. Deng, *One-way Hash function construction based on the chaotic map with changeable-parameter.* Chaos, Solitons & Fractals, 2005. **24**(1): p. 65.

71. Shiguo, L., et al. *Hash function based on chaotic neural networks*. in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*. 2006.

72. Peng, F. and S.-s. Qiu. *One-way Hash Functions Based on Iterated Chaotic Systems*. in *Communications, Circuits and Systems, 2007. ICCCAS 2007. International Conference on*. 2007.

73. Yong, W., et al. *One-Way Hash Function Construction Based on Iterating a Chaotic Map*. in *Computational Intelligence and Security Workshops, 2007. CISW 2007. International Conference on*. 2007.

74. Wang, Y., et al., *One-way hash function construction based on 2D coupled map lattices.* Information Sciences, 2008. **178**(5): p. 1391.

75. Khan, M.K., J. Zhang, and X. Wang, *Chaotic hash-based fingerprint biometric remote user authentication scheme on mobile devices.* Chaos, Solitons & Fractals, 2008. **35**(3): p. 519.

76. Xiao, D., X. Liao, and S. Deng, *Parallel keyed hash function construction based on chaotic maps.* Physics Letters A, 2008. **372**(26): p. 4682.

77. Yurong, S. and J. Guoping. *Hash Function Construction Based on Chaotic Coupled Map Network*. in *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for*. 2008.

78. Deng, S., et al., *A novel combined cryptographic and hash algorithm based on chaotic control character.* Communications in Nonlinear Science and Numerical Simulation, 2009. **14**(11): p. 3889-3900.

79. Bo, Y., et al. *Hash function construction based on coupled map lattice for communication security*. in *Global Mobile Congress 2009*. 2009.

80. Amin, M., O.S. Faragallah, and A.A. Abd El-Latif, *Chaos-based hash function (CBHF) for cryptographic applications.* Chaos, Solitons & Fractals, 2009. **42**(2): p. 767-772.

81. Xiao, D., X. Liao, and Y. Wang, *Parallel keyed hash function construction based on chaotic neural network.* Neurocomputing, 2009. **72**: p. 2288–2296.

82. Guo, X. and J. Zhang, *Secure group key agreement protocol based on chaotic Hash.* Information Sciences, 2010. **180**(20): p. 4069-4074.

83. Guyeux, C. and J.M. Bahi. *Topological chaos and chaotic iterations application to hash functions.* in *Neural Networks (IJCNN), The 2010 International Joint Conference on.* 2010.

84. Huang, Z., *A more secure parallel keyed hash function based on chaotic neural network.* Communications in Nonlinear Science and Numerical Simulation, 2011. **16**(8): p. 3245-3256.

85. Wang, Y., K.-W. Wong, and D. Xiao, *Parallel hash function construction based on coupled map lattices.* Communications in Nonlinear Science and Numerical Simulation, 2011. **16**(7): p. 2810-2821.

86. Yi, X., *Hash function based on chaotic tent maps.* Circuits and Systems II: Express Briefs, IEEE Transactions on, 2005. **52**(6): p. 354-357.

87. Zhang, J., X. Wang, and W. Zhang, *Chaotic keyed hash function based on feedforward-feedback nonlinear digital filter.* Physics Letters A, 2007. **362**(5-6): p. 439.

88. Amin, M., O.S. Faragallah, and A.A. Abd El-Latif, *Chaos-based hash function (CBHF) for cryptographic applications.* Chaos, Solitons & Fractals, 2009. **In Press, Corrected Proof**.

89. Xiao, D., F.Y. Shih, and X. Liao, *A chaos-based hash function with both modification detection and localization capabilities.* Communications in Nonlinear Science and Numerical Simulation, 2010. **15**(9): p. 2254-2261.

90. Anderson, R., *Letter to the editor: Chaos and random numbers.* Cryptologia, 1992. **16**(3).

91. Kocarev, L. and G. Jakimoski, *Pseudorandom bits generated by chaotic maps.* Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, 2003. **50**(1): p. 123-126.

92. Xiao-jun, T., C. Ming-gen, and J. Wei. *The Production Algorithm of Pseudo-Random Number Generator Based on Compound Non-Linear Chaos System*. in *Intelligent Information Hiding and Multimedia Signal Processing, 2006. IIH-MSP '06. International Conference on*. 2006.

93. Chen, S. and X.-x. Zhong, *Chaotic block iterating method for pseudo-random sequence generator.* The Journal of China Universities of Posts and Telecommunications, 2007. **14**(1): p. 45.

94. Zheng, F., et al., *Pseudo-random sequence generator based on the generalized Henon map.* The Journal of China Universities of Posts and Telecommunications, 2008. **15**(3): p. 64.

95. Patidar, V., K.K. Sud, and N.K. Pareek, *A pseudo random bit generator based on chaotic logistic map and its statistical testing.* Informatica, 2008. **33**(4): p. 441-452.

96. Sun, F. and S. Liu, *Cryptographic pseudo-random sequence from the spatial chaotic map.* Chaos, Solitons & Fractals, 2009. **41**(5): p. 2216-2219.

97. Songjiang, L., Q. Shuisheng, and C. Xu. *Investigation on complexity analysis of chaos-based pseudorandom sequence*. in *Image Analysis and Signal Processing, 2009. IASP 2009. International Conference on*. 2009.

98. Youssef, M.I., et al. *Image encryption using pseudo random number and chaotic sequence generators*. in *Radio Science Conference, 2009. NRSC 2009. National*. 2009.

99. Aixue, Q., H. Chunyan, and W. Guangyi. *Design and FPGA realization of a pseudo random sequence generator based on a switched chaos*. in *Communications, Circuits and Systems (ICCCAS), International Conference on*. 2010.

100. Yoon, J.W. and H. Kim, *An image encryption scheme with a pseudorandom permutation based on chaotic maps.* Communications in Nonlinear Science and Numerical Simulation, 2010. **15**(12): p. 3998-4006.

101. Dabal, P. and R. Pelka. *A chaos-based pseudo-random bit generator implemented in FPGA device*. in *Design and Diagnostics of Electronic Circuits & Systems (DDECS), IEEE 14th International Symposium on*. 2011.

102. Forré, R., *The Hénon attractor as a keystream generator*, in *Advances in Cryptology. – EuroCrypt'91*. 1991: Brighton, UK.

103. Matthews, R.A.J., *On the derivation of a "chaotic" encryption algorithm.* Cryptologia, 1989. **13**(1): p. 29–42.

104. Li, S., X. Mou, and Y. Cai, *Improving security of a chaotic encryption approach.* Physics Letters A, 2001. **290**(3-4): p. 127.

105. Wolfram, S. *Cryptography with cellular automata*. in *Advances in Cryptology - Crypto'85, Lecture Notes in Computer Science*. 1985. Spinger-Verlag, Berlin.

106. Lee, P.-H., S.-C. Pei, and Y.-Y. Chen, *Generating Chaotic Stream Ciphers Using Chaotic Systems.* CHINESE JOURNAL OF PHYSICS, 2003. **41**(6).

107. Sang, T., R. Wang, and Y. Yan, *Constructing chaotic discrete sequences for digital communications based on correlation analysis.* Signal Processing, IEEE Transactions on, 2000. **48**(9): p. 2557-2565.

108. Shujuna, L., M. Xuanqinb, and C. Yuanlong. *Pseudo-Random Bit Generator Based on Couple Chaotic Systems and its Applications in Stream-Cipher Cryptography*. in *Progress in Cryptology - INDOCRYPT 2001, LNCS*. 2001. Berlin: Springer-Verlag.

109. Wang, X., et al. *Chaotic Pseudorandom Bit Generator Using n-dimensional Nonlinear Digital Filter*. in *Communication Technology, 2006. ICCT '06. International Conference on*. 2006.

110. Philip, N.S. and K.B. Joseph, *Chaos for Stream Cipher*, in *CoRR*. 2001, journals/corr/cs-CR-0102012.

111. Yu, W. and J. Cao, *Cryptography based on delayed chaotic neural networks.* Physics Letters A, 2006. **356**(4-5): p. 333.

112. Kurian, A.P. and S. Puthusserypady, *Self-synchronizing chaotic stream ciphers.* Signal Processing, 2008. **88**(10): p. 2442.

113. Kwok, H.S. and W.K.S. Tang, *A fast image encryption system based on chaotic maps with finite precision representation.* Chaos, Solitons & Fractals, 2007. **32**(4): p. 1518.

114. Patidar, V., N.K. Pareek, and K.K. Sud, *A new substitution-diffusion based image cipher using chaotic standard and logistic maps.* Communications in Nonlinear Science and Numerical Simulation, 2009. **14**(7): p. 3056-3075.

115. Patidar, V., et al., *Modified substitution-diffusion image cipher using chaotic standard and logistic maps.* Communications in Nonlinear Science and Numerical Simulation, 2010. **15**(10): p. 2755-2765.

116. Addabbo, T., et al. *Long period pseudo random bit generators derived from a discretized chaotic map*. in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*. 2005.

117. Kanso, A. and N. Smaoui, *Logistic chaotic maps for binary numbers generations.* Chaos, Solitons & Fractals, 2009. **40**(5): p. 2557-2568.

118. Wheeler, D.D. and R.A.J. Matthews, *Supercomputer Investigations of a Chaotic Encryption Algorithm.* Cryptologia 1991. **15**: p. 140-152.

119. Biham, E. *Cryptanalysis of the chaotic-map cryptosystem suggested at EUROCRYPT'91*. in *Proceedings of the EUROCRYPT '91*. 1991. Berlin: Springer.

120. Wheeler, D.D., *Problems With Chaotic Cryptosystems.* Cryptologia, 1989. **13**: p. 243–250.

121. Alvarez, G., et al., *Cryptanalysis of a chaotic encryption system.* Physics Letters A, 2000. **276**(1-4): p. 191-196.

122. Alvarez, G., et al., *Cryptanalysis of a chaotic secure communication system.* Physics Letters A, 2003. **306**(4): p. 200-205.

123. Alvarez, G., et al., *Cryptanalysis of an ergodic chaotic cipher.* Physics Letters A, 2003. **311**(2-3): p. 172.

124. Alvarez, G., et al., *Cryptanalysis of a discrete chaotic cryptosystem using external key,* in *Physics Letters A*. 2003. p. 334.

125. Wei, J., et al., *Cryptanalysis of a cryptosystem using multiple one-dimensional chaotic maps.* Communications in Nonlinear Science and Numerical Simulation, 2007. **12**(5): p. 814.

126. Li, C., et al., *Cryptanalysis of a chaotic block cipher with external key and its improved version.* Chaos, Solitons & Fractals, 2008. **37**(1): p. 299.

127. Wang, Y., et al., *Cryptanalysis and improvement on a block cryptosystem based on iteration a chaotic map.* Physics Letters A, 2007. **363**(4): p. 277.

128. Wang, X. and C. Yu, *Cryptanalysis and improvement on a cryptosystem based on a chaotic map.* Computers & Mathematics with Applications, 2009. **57**(3): p. 476.

129. Yang, J., D. Xiao, and T. Xiang, *Cryptanalysis of a chaos block cipher for wireless sensor network.* Communications in Nonlinear Science and Numerical Simulation, 2011. **In Press, Accepted Manuscript**.

130. Zhu, S.-l., Y.-x. Wang, and X. Li. *Design and analysis of variable-length block encryption algorithm based on Chaos Particle Swarm.* in *Computer Science and Education (ICCSE), 2010 5th International Conference on.*

131. Álvarez, G., et al., *Keystream cryptanalysis of a chaotic cryptographic method.* Computer Physics Communications, 2004. **156**(2): p. 205-207.

132. Álvarez, G., et al., *Cryptanalysis of dynamic look-up table based chaotic cryptosystems.* Physics Letters A, 2004. **326**(3-4): p. 211-218.

133. Jakimoski, G. and L. Kocarev, *Analysis of some recently proposed chaos-based encryption algorithms.* Physics Letters A, 2001. **291**(6): p. 381.

134. Li, S., et al., *Performance analysis of Jakimoski-Kocarev attack on a class of chaotic cryptosystems.* Physics Letters A, 2003. **307**(1): p. 22.

135. Çokal, C. and E. Solak, *Cryptanalysis of a chaos-based image encryption algorithm.* Physics Letters A, 2009. **373**(15): p. 1357-1360.

136. Arroyo, D., et al., *Cryptanalysis of an image encryption scheme based on a new total shuffling algorithm.* Chaos, Solitons & Fractals, 2009. **27**(8): p. 1035-1039.

137. Maqableh, M.M. and S. Dantchev, *Cryptanalysis of Chaos-Based Hash Function (CBHF)* in *First International Alternative Workshop on Aggressive Computing and Security - iAWACS*. 2009: France-Laval.

138. Qun-ting, Y., et al. *Analysis of One-way Alterable Length Hash Function Based on Cell Neural Network*. in *Information Assurance and Security, 2009. IAS '09. Fifth International Conference on*. 2009.

139. Xiao, D., X. Liao, and Y. Wang, *Improving the security of a parallel keyed hash function based on chaotic maps.* Physics Letters A, 2009. **373**(47): p. 4346-4353.

140. Guo, W., et al., *Cryptanalysis on a parallel keyed hash function based on chaotic maps.* Physics Letters A, 2009. **373**(36): p. 3201-3206.

141. Deng, S., Y. Li, and D. Xiao, *Analysis and improvement of a chaos-based Hash function construction.* Communications in Nonlinear Science and Numerical Simulation, 2010. **15**(5): p. 1338-1347.

142. Wang, J., et al. *The Analysis for a Chaos-Based One-Way Hash Algorithm*. in *Electrical and Control Engineering (ICECE), 2010 International Conference on*. 2010.

143. Wang, X.-y. and J.-f. Zhao, *Cryptanalysis on a parallel keyed hash function based on chaotic neural network.* Neurocomputing, 2010. **73**(16-18): p. 3224-3228.

144. Xiao, D., et al., *Collision analysis of one kind of chaos-based hash function.* Physics Letters A, 2010. **374**(10): p. 1228-1231.

145. Wang, S., D. Li, and H. Zhou, *Collision analysis of a chaos-based hash function with both modification detection and localization capability.* Communications in Nonlinear Science and Numerical Simulation. **In Press, Corrected Proof**.

146. Li, Y., S. Deng, and D. Xiao, *Corrigendum to "Analysis and improvement of a chaos-based hash function construction" [Commun Nonlinear Sci Numer Simulat 2010;15:1338-1347].* Communications in Nonlinear Science and Numerical Simulation, 2010. **15**(10): p. 3233-3233.

147. Li, C., et al., *Cryptanalysis of an image encryption scheme based on a compound chaotic sequence.* Image and Vision Computing, 2009 **27**(8): p. 1035-1039.

148. Alvarez, G. and S. Li, *Cryptanalyzing a nonlinear chaotic algorithm (NCA) for image encryption.* Communications in Nonlinear Science and Numerical Simulation, 2009. **14**(11).

149. Arroyo, D., et al., *Cryptanalysis of a discrete-time synchronous chaotic encryption system.* Physics Letters A, 2008. **372**(7): p. 1034.

150. Pareschi, F., et al. *Power analysis of a chaos-based Random Number Generator for cryptographic security*. in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*. 2009.

151. Skrobek, A., *Cryptanalysis of chaotic stream cipher.* Physics Letters A, 2007. **363**(1-2): p. 84.

152. Li, C., et al., *Cryptanalysis of two chaotic encryption schemes based on circular bit shift and XOR operations.* Physics Letters A, 2007. **369**(1-2): p. 23.

153. Arroyo, D., et al., *Cryptanalysis of a family of self-synchronizing chaotic stream ciphers.* Communications in Nonlinear Science and Numerical Simulation. **In Press, Corrected Proof**.

154. Rhouma, R., E. Solak, and S. Belghith, *Cryptanalysis of a new substitution-diffusion based image cipher.* Communications in Nonlinear Science and Numerical Simulation, 2010. **15**(7): p. 1887-1892.

155. Li, C., S. Li, and K.-T. Lo, *Breaking a modified substitution-diffusion image cipher based on chaotic standard and logistic maps.* Communications in Nonlinear Science and Numerical Simulation, 2011. **16**: p. 837–843.

156. X.Y. Wang, F.D.G., X.J. Lai, H.B. Yu, *Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD*, in *Rump Session of Crypto’04, E-print*. 2004.

157. Wang, X., Y.L. Yin, and H. Yu, *Finding Collisions in the full SHA-1*, in *Advances in Cryptology* **-** *Crypto'05*. 2005, Springer: California, USA. p. 17-36.

158. X. Wang, H.Y., *How to break MD5 and other hash functions*, in *Eurocrypt 2005*. 2005. p. 19 - 55.

159. Biham, E. and A. Shamir, *Differential Cryptanalysis of DES-like Cryptosystems*, in *Advances in Cryptology-CRYPT0' 90*. 1991, Springer Berlin / Heidelberg. p. 2-21.

160. Kim, J., Y. Lee, and S. Lee, *DES with any reduced masked rounds is not secure against side-channel attacks.* Computers & Mathematics with Applications. **60**(2): p. 347-354.

161. H. Handschuh and B. Preneel. *Blind differential cryptanalysis for enhanced power attacks*. in *Proceedings of SAC'06*. 2006: Springer-Verlag.

162. Joan Daemen and V. Rijmen, *AES Proposal: Rijndael*, in *AES Round 1 Technical Evaluation CD-1: Documentation*. 1998, NIST.

163. Henri Gilbert and M. Minier, *A collision attack on 7 rounds of Rijndael*, in *The third Advanced Encryption Standard Candidate Conference*. 2000, NIST: New York - USA.

164. John Kelsey, et al., *Improved Cryptanalysis of Rijndael*, in *Fast Software Encryption*. 2000: New York - USA.

165. X. Wang, Y.L.Y., H. Yu, *Finding collisions in the full SHA1*, in *Eurocrypt 2005*. 2005.

166. Torrubia, A., F.J. Mora, and L. Marti, *Cryptography Regulations for E-commerce and Digital Rights Management.* Computers & Security, 2001. **20**(8): p. 724-738.

167. Preneel, B., *Analysis and Design of Cryptography Hash Functions*, in *DEPARTEMENT ELEKTROTECHNIEK-ESAT*. jan 1993, Katholike Universiteit Leuven: Kasteelpark Arenberg. p. 259.

168. R.L. Rivest, A.S., and L.M. Adleman, *A method for obtaining Digital Signature and Public-Key Cryptosystems.* Communications of the ACM, 1978. **21**(2): p. 120-126.

169. Rabin, M.O., *Digitial Signatures and Public-Key Functions as Intractable as Factorization* in *MIT Laboratory for Computer Science, MIT/LCS/TR-212*. 1979.

170. ElGamal, Y. *A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*. in *Advance in Cryptology: Proceedings of CRYPTO 84*. 1985: Springer Verlag.

171. *Federal Information Processing Standards Publications 46-3, Data Encryption Standard (DES), U.S. Department of Commerce/National Institute of Standards and Technology*. 1999.

172. X. Lai, J.M. *A Proposal for a New Block Encryption Standard*. in *Advance in Cryptology-EUROCRYPT 90 Proceedings*. 1991: Springer Verlag.

173. *Federal Information Processing Standard 197, Advanced Encryption Standard (AES), National Institute of Standards and Technology*. 2001.

174. Schneier, B. *Designing of new variable-length Key, 64-Bit Block Cipher (Blowfish)*. in *Fast Software Encryption, Cambridge Security Workshop Proceesings*. 1994: Springer-Verlag.

175. Heys, H.M., *Analysis of the statistical cipher feedback mode of block ciphers*. Computers, IEEE Transactions on, 2003. **52**(1): p. 77-92.

176. Phan, R.C.W. and M.U. Siddiqi, *A Framework for Describing Block Cipher Cryptanalysis*. Computers, IEEE Transactions on, 2006. **55**(11): p. 1402-1409.

177. Mark Stamp and R.M. Low, *Applied Cryptanalysis: Breaking Ciphers in the Real World*. First ed. 2007, New Jersey: John Wiley & Sons.

178. Federal Information Processing Standards Publications, *Data Encryption Standard (DES)*. 1999, FIPS PUB 46-3.

179. *NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation, National Institute of Standards and Technology*. 2001.

180. Lee, H. and S. Moon, *Parallel stream cipher for secure high-speed communications*. Signal Processing, 2002. **82**(2): p. 259-265.

181. Cusick, T.W. and P. Stanica, *Block ciphers*, in *Cryptographic Boolean Functions and Applications*. 2009, Academic Press: Boston. p. 157-191.

182. Wright, R.N. and A.M. Robert, *Cryptography*, in *Encyclopedia of Physical Science and Technology*. 2001, Academic Press: New York. p. 61-77.

183. Chitu, C. and M. Glesner, *An FPGA implementation of the AES-Rijndael in OCB/ECB modes of operation*. Microelectronics Journal, 2005. **36**(2): p. 139-146.

184. Yang, X., et al., *Stream-based cipher feedback mode in wireless error channel*. Wireless Communications, IEEE Transactions on, 2009. **8**(2): p. 622-626.

185. Rhouma, R. and S. Belghith, *Cryptanalysis of a spatiotemporal chaotic image/video cryptosystem*. Physics Letters A, 2008. **372**(36): p. 5790-5794.

186. Wiene, M.J., *Efficient DES Key Search*, in *Advances in Cryptology - CRYPTO '93*. 1993: California - USA.

187. Jean-jacques Quisquater and F.-x. St, *Exhaustive Key Search of the DES: Updates and Refinements*, in *Special-purpose Hardware for Attacking Cryptographic Systems*. 2005 Paris.

188. Gaël Rouvroy, et al., *Efficient Uses of FPGAs for Implementations of DES and Its Experimental Linear Cryptanalysis*. IEEE Transactions on Computers, 2003. **52**(4): p. 473-482.

189. Curiger, A., et al. *VINCI: VLSI implementation of the new secret-key block cipher IDEA*. in *Custom Integrated Circuits Conference, 1993., Proceedings of the IEEE 1993*. 1993.

190. Hans Dobbertin, A.B., Bart Preneel, *RIPEMD-160 A Strengthened Version of RIPEMD*. Springer-Verlag, 1996: p. 71-82.

191. Cid, C., *Recent developments in cryptographic hash functions: Security implications and future directions*. Information Security Technical Report, 2006. **11**(2): p. 100-107.

192. Merkle, R., *A Fast Software One-Way Hash Function.* Journal of Cryptology, 1990. **3**(1): p. 43-58.

193. Biham, E. and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard.* Springer-Verlag, 1993.

194. Dobbertin, H., *RIPEMD with two-round compress function is not collision free.* Journal of Cryptology.

195. Dobbertin, H., *Cryptanalysis of MD5 compress*, in *Presented at the rump session of Eurocrypt '96*. 1996.

196. X. Wang, D.F., X. Lai, H. Yu, *Cryptanalysis of the hash functions MD4 and RIPEMD*, in *Eurocrypt 2005*. 2005. p. 1 - 18.

197. Rivest, R., *The MD4 Message-Digest Algorithm*, in *RFC 1320*. 1992, MIT and RSA Data Security, Inc. p. 87.

198. Rivest, R., *The MD5 Message-Digest Algorithm*, in *Request for Comments (RFC 1321)*. 1992, MIT and RSA Data Security, Inc. p. 87.

199. Rhee, M.Y., *Internet Security: Cryptographic Principles, Algorithms and Protocols*. 2003: John Wiley & Sons Ltd.

200. Dobbertin, H., *Secure hashing in practice.* Information Security Technical Report, 1999. **4**(4): p. 53-62.

201. Dobbertin, H., *The First Two Rounds of MD4 are Not One-Way*, in *Fast Software Encryption*. 1998.

202. den Boer, B., and Bosselaers,A., *Collisions for the compressin function of MD5*, in *EUROCRYPT*. 1993. p. 293-304.

203. NIST, *SECURE HASH STANDARD.* Federal Information Processing Standard, FIPS-180, 1995.

204. Archives, I.R.S.F.B. *US Secure Hash Algorithm 1 (SHA-1).* 1995 [cited January 30, 2009]; Available from: http://www.faqs.org/rfcs/rfc3174.html.

205. Wikipedia. *SHA hash functions*. 1995 [cited February 15, 2009]; Available from: http://en.wikipedia.org/w/index.php?title=SHA_hash_functions&oldid=264888453.

206. Rijmen, V. and M.E. Oswald, *Update on SHA-1*, in *CT-RSA* Springer, Editor. 2005, Springer -verlag. p. 58-71.

207. Shamir, A. *(On behalf of Xiaoyun Wang), Recent Progress on SHA-1*. 2006 [cited January 18, 2009]; Available from: http://www.iacr.org/conferences/crypto2005/r/2.pdf

208. X. Wang, H.Y., Y.L. Yin, *Efficient collision search attacks on SHA0*, in *Crypto 2005*. 2005.

209. NIST, *Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family*. Federal Register, 2 Nov 2007. **72**(212): p. 62212-62220.

210. NIST, *Announcing the Development of New Hash Algorithm(s) for the Revision of Federal Information Processing Standard (FIPS) 180-2*. Secure Hash Standard, "Federal Register", 23 Jan 2007. **72**(14): p. 2861-2863.

211. Andrew Rukhin, et al., *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, in *NIST Special Publication 800–22 Revision 1a*. 2010.

212. Danger, J.L., S. Guilley, and P. Hoogvorst, *High speed true random number generator based on open loop structures in FPGAs*. Microelectronics Journal, 2009. **40**(11): p. 1650-1656.

213. Güler, Ü. and S. Ergün, *A high speed, fully digital IC random number generator*. AEU - International Journal of Electronics and Communications. **In Press, Corrected Proof**.

214. Alsultanny, Y.A., *Random-bit sequence generation from image data*. Image and Vision Computing, 2008. **26**(4): p. 592-601.

215. Shannon, C., *Communication Theory of Secrecy Systems*. Bell Systems Technical Journal, 1949. **4**.

216. Baris Coskun, N.M. *Confusion/Diffusion capabilities of some robust hash functions*. in *2006 40th annual conference*. March; 2006: IEEE conference proceedings: information sciences and systems.

217. Fan, Z., et al., *Hash Function Based on the Generalized Henon Map.* Chinese Physics B, 2008. **17**(5): p. 1685-1690.

218. DasGupta, A., *The matching, birthday and the strong birthday problem: a contemporary review.* Journal of Statistical Planning and Inference, 2005. **130**(1-2): p. 377-389.

219. Mihir Bellare, T.K., *Hash Function Balance and its Impact on Birthday Attacks*, in *EUROCRYPT*. 2004. p. 401–418.

220. Eli Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*. 1990: Springer-Verlag

221. de Oliveira, L.P.L. and M. Sobottka, *Cryptography with chaotic mixing.* Chaos, Solitons & Fractals, 2008. **35**(3): p. 466.

222. Chee, C.Y. and D. Xu, *Chaotic encryption using discrete-time synchronous chaos.* Physics Letters A, 2006. **348**(3-6): p. 284.

223. Jakimoski, G. and L. Kocarev, *Chaos and cryptography: block encryption ciphers based on chaotic maps.* Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, 2001. **48**(2): p. 163.

224. Dachselt, F. and W. Schwarz, *Chaos and cryptography.* Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, 2001. **48**(12): p. 1498.

225. Kwok-Wo, W., H. Sun-Wah, and Y. Ching-Ki, *A chaotic cryptography scheme for generating short ciphertext.* Physics Letters A, 2003. **310**(1): p. 67.

226. Klein, E., et al., *Public-channel cryptography using Chaos synchronization.* American Physical Society (APS) Journals, 19 July 2005.

227. Yang, T., C.W. Wu, and L.O. Chua, *Cryptography based on chaotic systems.* Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, 1997.

228. Sung-Ming, Y. and L. Kuo-Hong, *Shared authentication token secure against replay and weak key attacks.* Information Processing Letters, 1997. **62**(2): p. 77-80.

229. Alvarez, J., E. Curiel, and F. Verduzco, *Complex dynamics in classical control systems.* Systems & Control Letters, 1997. **31**(5): p. 277-285.

230. Tao, Y., W. Chai Wah, and L.O. Chua, *Cryptography based on chaotic systems.* Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, 1997. **44**(5): p. 469-472.

231. Annovazzi-Lodi, V., S. Donati, and A. Scire, *Synchronization of chaotic lasers by optical feedback for cryptographic applications.* Quantum Electronics, IEEE Journal of, 1997. **33**(9): p. 1449-1454.

232. Delgado-Restituto, M. and A. Rodriguez-Vazquez. *Discrete-time integrated circuits for chaotic communication.* in *Circuits and Systems, 1997. ISCAS '97., Proceedings of 1997 IEEE International Symposium on.* 1997.

233. Ogorzalek, M.J. and H. Dedieu. *Observability and identifiability of chaotic systems-keys to system synchronization and transmission security.* in *Circuits and Systems, 1997. ISCAS '97., Proceedings of 1997 IEEE International Symposium on.* 1997.

234. Fridrich, J. *Image encryption based on chaotic maps.* in *Systems, Man, and Cybernetics, 1997. 'Computational Cybernetics and Simulation'., 1997 IEEE International Conference on.* 1997.

235. Dachselt, F., K. Kelber, and W. Schwarz. *Chaotic coding and cryptoanalysis.* in *Circuits and Systems, 1997. ISCAS '97., Proceedings of 1997 IEEE International Symposium on.* 1997.

236. Hong, Z., L. Xie-Tang, and Y. Jun. *Secure communication via one-dimensional chaotic inverse systems.* in *Circuits and Systems, 1997. ISCAS '97., Proceedings of 1997 IEEE International Symposium on.* 1997.

237. Wikipedia. *Chaos*. [cited 17 February 2009 16:01 UTC]; Available from: http://en.wikipedia.org/w/index.php?title=Chaos&oldid=264486195.

238. Bertuglia, C.S. and F. Vaio, *Nonlinearity, Chaos & Complexity The Dynamics of Natural and Social Systems*. First ed. 2005, United States: Oxford University Press Inc.

239. Alligood, K.T., T.D. Sauer, and J.A. Yorke, *Chaos an Introduction to Dynamical Systems*. First ed. 1996, New York: Springer-Verlag.

240. Solari, H.G., M.A. Natiello, and G.B. Mindlin, *Nonlinear Dynamics A Two-way Trip from Physics to Math*. 1 ed. 1996: Institute of Physics Publishing.

241. Baker, G.L. and J.P. Gollub, *Chaotic dynamics an introduction*. First ed. 1990, New York: Press Syndicate of the University of Cambridge.

242. Poincaré, J.H., *Sur le problème des trois corps et les équations de la dynamique. Divergence des séries de M. Lindstedt* Acta Mathematica, 1890. **13**: p. 1–270.

243. Lorenz, E.N., *Deterministic Nonperiodic Flow*. Journal of Atmospheric Sciences, 1963. **20**.

244. Pritchard, J., *The Chaos cookbook*. 1996, OXFORD: Butterworh-Heinemann.

245. Serletis, A. and P. Gogas, *Chaos in East European Black Market Exchange Rates*. Research in Economics, 1997. **51**: p. 359-385.

246. Serletis, A. and P. Gogas, *The North American natural gas liquids markets are chaotic*. The Energy Journal 1999. **20**(1).

247. Gilmore, R., *Chaos and Attractors*. Encyclopedia of Mathematical Physics, 2004.

248. Tullaro, N.B., T. Abbott, and J.P. Reilly, *An Experimental Approach to Nonlinear Dynamics and Chaos*. Vol. 1. 1992: Addison-Wesley.

249. Parker, T.S. and L.O. Chua, *Practical Numerical Algorithms for Chaotic Systems*. First ed. 1989, New York Berlin Heidelberg: Springer-Verlag New York Inc.

250. Schmitz, R., *Use of chaotic dynamical systems in cryptography*. Journal of the Franklin Institute, 2001. **338**(4): p. 429.

251. Grassi, G. and S. Mascoio, *Synchronizing hyperchaotic systems by observer design*. Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on, 1999. **46**(4): p. 478.

252. May, R.M., *Simple mathematical models with very complicated dynamics*. Nature, 1976. **261**: p. 459-467.

253. Wikipedia. *Logistic map*. [cited February 20, 2009; Available from: http://en.wikipedia.org/w/index.php?title=chaotic_maps&oldid=261864353.

254. Naess, A., *Chaos and nonlinear stochastic dynamics*. Probabilistic Engineering Mechanics, 2000. **15**: p. 37-47.

255. contributors, W. *Chaos theory*. 2005 [cited January 17, 2009]; Available from: http://en.wikipedia.org/w/index.php?title=Chaos_theory&oldid=264934743.

256. RÖssler, O., *An equation for continous chaos*. Phys. Lett. A, 1976. **57**: p. 397.

257. Ho, A. *Chaos Introduction*. 2006 [cited February 25, 2009]; Available from: http://www.zeuscat.com/andrew/chaos/chaos.html.

258. Wikipedia. *Tent map*. [cited February 25, 2009]; Available from: http://en.wikipedia.org/w/index.php?title=Tent_map&oldid=186656075.

259. Li, S., G. Chen, and X. Mou, *On the Dynamical Degradation of Digital Piecewise Linear Chaotic Maps*. International Journal of Bifurcation and Chaos, 2005: p. 3119-3151.

260. Jun, P., et al. *A novel scheme for image encryption based on piecewise linear chaotic map*. in *Cybernetics and Intelligent Systems, 2008 IEEE Conference on*. 2008.

261. Rhouma, R., D. Arroyo, and S. Belghith. *A new color image cryptosystem based on a piecewise linear chaotic map*. in *Systems, Signals and Devices, 2009. SSD '09. 6th International Multi-Conference on*. 2009.

262. Amig ◌̧́ J.M., L. Kocarev, and J. Szczepanski, *Theory and practice of chaotic cryptography*. Physics Letters A, 2007. **366**(3): p. 211.

263. Zhang, H. and J.-x. Dong. *Chaos theory and its application in modern cryptography*. in *Computer Application and System Modeling (ICCASM), 2010 International Conference on*. 2010.

264. Alvarez, G. and S. Li, *Some basic cryptographic requirements for chaos-based cryptosystems*. International Journal of Bifurcation and Chaos, 2006. **16**: p. 2129--2151.

265. Gotz, M., K. Kelber, and W. Schwarz, *Discrete-time chaotic encryption systems. I. Statistical design approach*. Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, 1997. **44**(10): p. 963-970.

266. Kartalopoulos, S.V. *Chaotic Quantum Cryptography*. in *Information Assurance and Security, 2008. ISIAS '08. Fourth International Conference on*. 2008.

267. Guan, P., *Cellular automaton public-key cryptosystem*. Complex Systems, 1987. **1**: p. 51–57.

268. Mossayebi, F., H.K. Qammar, and T.T. Hartley, *Adaptive estimation and synchronization of chaotic systems*. Physics Letters A, 1991. **161**(3): p. 255-262.

269. Berstein, G.M. and M.A. Lieberman, *Method and apparatus for generating secure random numbers using chaos*, in *US Patent No. 5007087*. 1991.

270. Erdmann, D. and S. Murphy, *Henon stream cipher*. Electronics Letters, 1992. **28**(9): p. 893-895.

271. B. Preneel, V. Rijmen, and A. Bosselears, *Recent developments in the design of conventional cryptographic algorithm*, in *Lecturer Notes in Computer Science*. 1998, Springer-Verlag: Berlin, Germany. p. 106-131.

272. Stojanovski, T. and L. Kocarev, *Chaos-based random number generators-part I: analysis [cryptography].* Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, 2001. **48**(3): p. 281.

273. Beker, H. and F. Piper, eds. *Cipher Systems: The protection of communications*. 1982, van Nostrand Reinhold: New York.

274. NIST, *Federal Information Processing Standards Publications (FIPS140-1). Security requirments for cryptographic modules.* 1994.

275. Li, S., et al., *On the security of a chaotic encryption scheme: problems with computerized chaos in finite computing precision.* Computer Physics Communications, 2003. **153**(1): p. 52.

276. Hong, Z. and L. Xie-Ting, *Problems with the chaotic inverse system encryption approach.* Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, 1997. **44**(3): p. 268-271.

277. Li, S., G. Chen, and X. Zheng, *Chaos-Based Encryption for Digital Images and Videos*, in *Multimedia Security Handbook*, B. Furht and D. Kirovski, Editors. 2004. p. 133-167.

278. Fryska, S.T. and M.A. Zohdy, *Computer dynamics and shadowing of chaotic orbits.* Physics Letters A, 1992. **166**(5-6): p. 340-346.

279. Pokrovskii, A.V., A. Kent, and J. McInerney, *Mixed moments of random mappings and chaotic dynamical systems.* Proceedings the Royal Society A 2000. **456**: p. 2465-2487.

280. Zhou, H. and X. Ling, *Realizing finite precision chaotic systems via perturbation of m-sequences.* Acta Eletronica Sinica, 1997. **25**: p. 95-97.

281. Heidari-Bateni, G. and C.D. McGillem, *A chaotic direct-sequence spread-spectrum communication system.* IEEE Trans. Communications, 1994. **42**: p. 1524-1527.

282. Jastrzębski, K. and Z. Kotulski, *On Improved Image Encryption Scheme Based on Chaotic Map Lattices.* ENGINEERING TRANSACTIONS, 2009. **52**(2): p. 69–84.

283. Liu, N., D. Guo, and G. Parr, *Complexity of chaotic binary sequence and precision of its numerical simulation.* Nonlinear Dynamics, 2011: p. 1-8.

284. Xiang, T., K.-w. Wong, and X. Liao, *An improved chaotic cryptosystem with external key.* Communications in Nonlinear Science and Numerical Simulation, 2008. **13**(9): p. 1879.

285. Silva, R.M., R.G. Crespo, and M.S. Nunes, *LoBa128 a Lorenz-based PRNG for wireless sensor networks.* Int. J. Communication Networks and Distributed Systems, 2009. **3**(4).

286. Akhavan, A., A. Samsudin, and A. Akhshani, *Hash function based on piecewise nonlinear chaotic map.* Chaos, Solitons & Fractals, 2009. **42**: p. 1046–1053.

287. Akhshani, A., et al., *Hash function based on hierarchy of 2D piecewise nonlinear chaotic maps.* Chaos, Solitons & Fractals, 2009. **42**(4): p. 2405-2412.

288. contributors, W. *Cryptographic hash function* 2011 [cited 2011 24 JAN. 00:36]; Available from: http://en.wikipedia.org/w/index.php?title=Cryptographic_hash_function&oldid=423934839.

289. Manuel, S.e., *Classification and Generation of Disturbance Vectors for Collision Attacks against SHA-1*, in *International Workshop on Coding and Cryptography*. 2009: Norway.

290. Granlund, T. and G.d. team, *GNU multiple precision arithmetic library* in *GMP manual describtion*. 2011. p. 145.

291. Stevens, M.M.J., *On Collisions for MD5*, in *Department of Mathematics and Computing Science*. 2007, Eindhoven University of Technology: Eindhoven. p. 84.

292. Baris Coskun and N. Memon, *Confusion/Diffusion capabilities of some robust hash functions*, in *40th annual conference*. 2006, IEEE conference proceedings: information sciences and systems. p. 1188-1193.

293. Yang, H., et al., *One-way hash function construction based on chaotic map network.* Chaos, Solitons & Fractals, 2009. **41**(5): p. 2566-2574.

294. Yi, Y., *Hash Function Based on Chaotic Tent Maps.* IEEE Transactions on Circuits and Systems, 2005. **52**(6).

295. Maqableh, M.M. and S. Dantchev, *Fast Encryption Algorithm Based on Chaotic Maps for E-commerce (BCCM).* Telecommunication Systems, Springer, ***to be submitted 2011***.

296. Maqableh, M.M., *Fast Hash Function Based on BCCM Encryption Algorithm for E-Commerce (HFBCCM)*, in *5th International Conference on e-Commerce in Developing Countries: with focus on export.* 2010: Kish Island - Iran.

297. Baris Coskun, N.M. *Confusion/Diffusion capabilities of some robust hash functions*. in *2006 40th annual conference*. March- 2006: IEEE conference proceedings: information sciences and systems.

298. Faraoun, K.M., *A Novel Chaotic Ciphering System for Color Digital Images.* A Novel Chaotic Ciphering System for Color Digital Images, 2009. **22**(2): p. 85-98.

299. Arroyo, D., et al., *Cryptanalysis of a computer cryptography scheme based on a filter bank.* Chaos, Solitons & Fractals, 2009. **41**(1): p. 410-413.

300. Wu, Y. *1D Bifurcation plot.* 2010 [cited 2011 20-01-2011]; Modified Online Matlab Source Code - MATLAB Central]. Available from: http://www.mathworks.com/matlabcentral/fileexchange/26839-1d-bifurcation-plot.

301. Vialar, T. *Complex and Chaotic Nonlinear Dynamics, Lyapunov exponent.* 2009 [cited 2011 20-08-2011]; Modefied Online MATLAB Source Code]. Available from: http://www.isnld.com/.

Appendix A

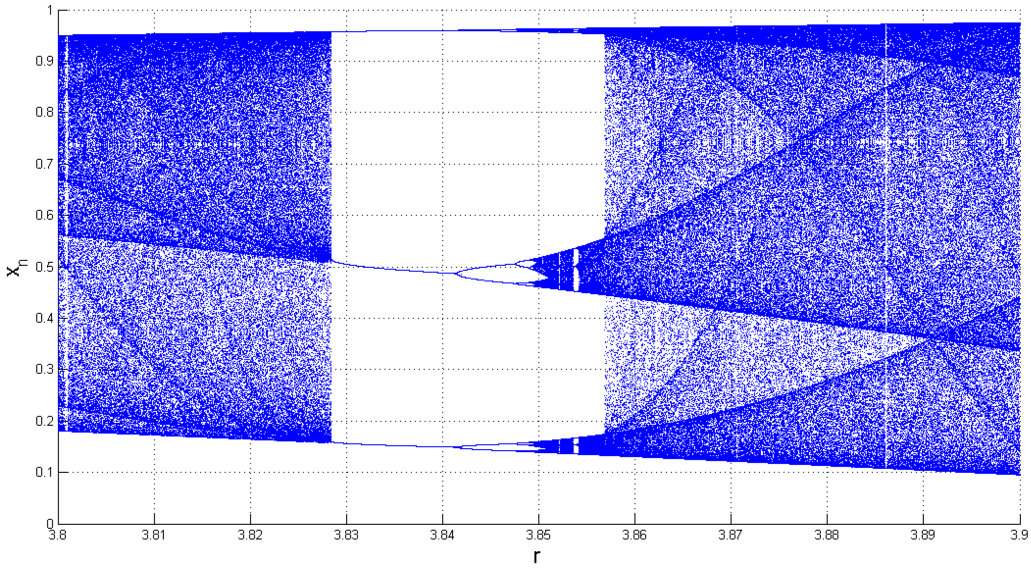


Figure A- 1: Logistic map bifurcation diagram with t∈[3.8, 3.9]

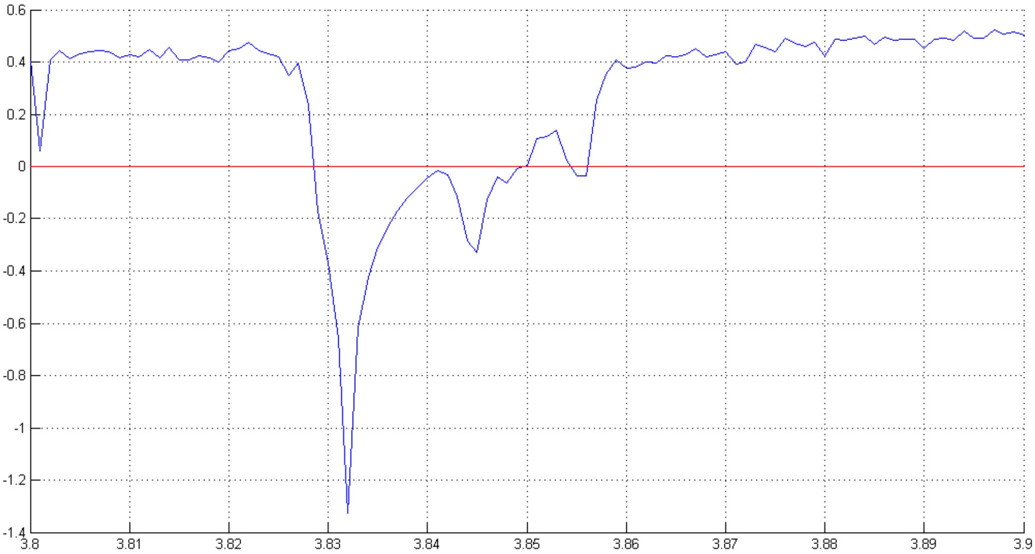


Figure A- 2: Lyapunov exponent of Logistic map with t∈[3.8, 3.9]

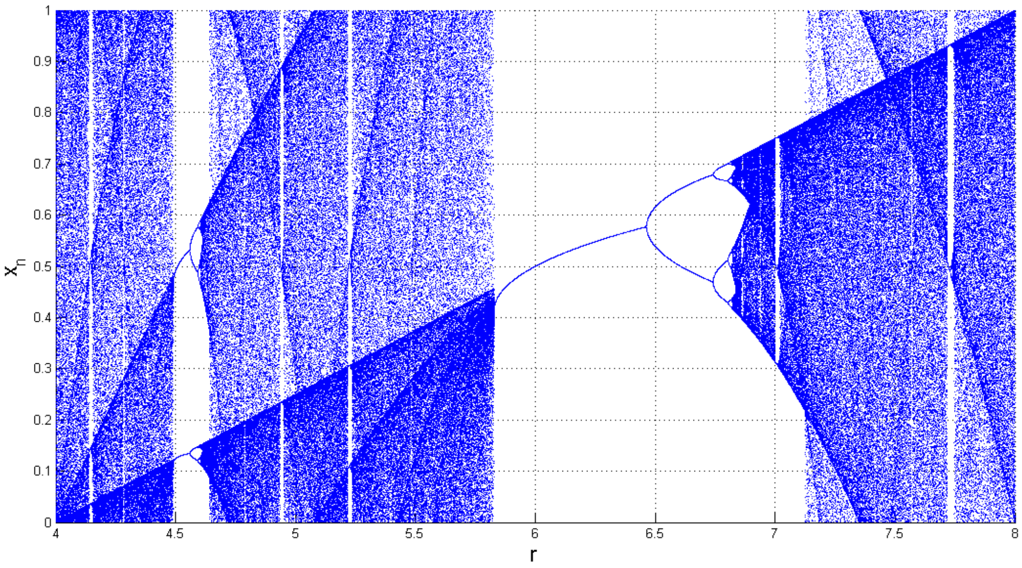


Figure A- 3: Modified logistic map bifurcation diagram with t∈[4, 8]

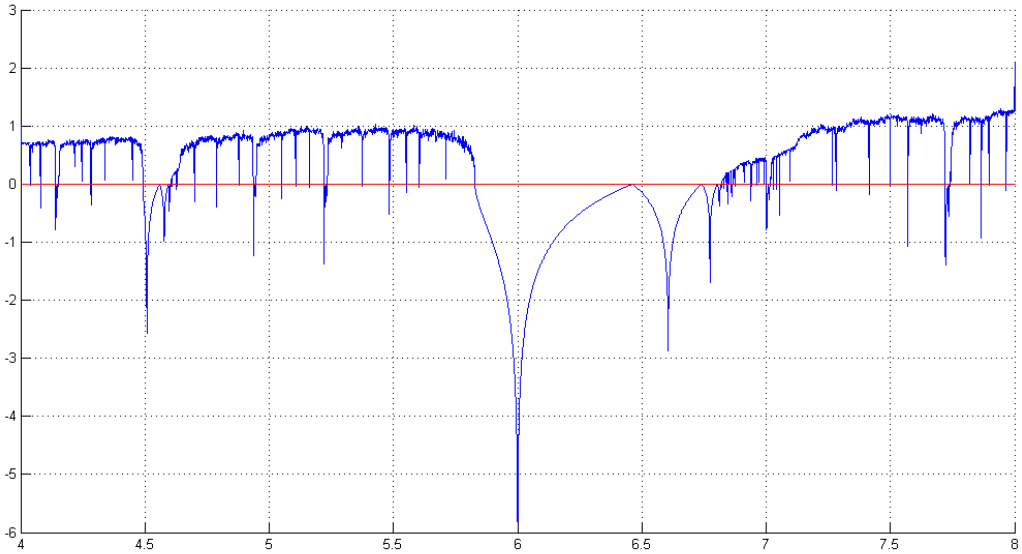


Figure A- 4: Lyapunov exponent of Modified Logistic map with t∈[4, 8]

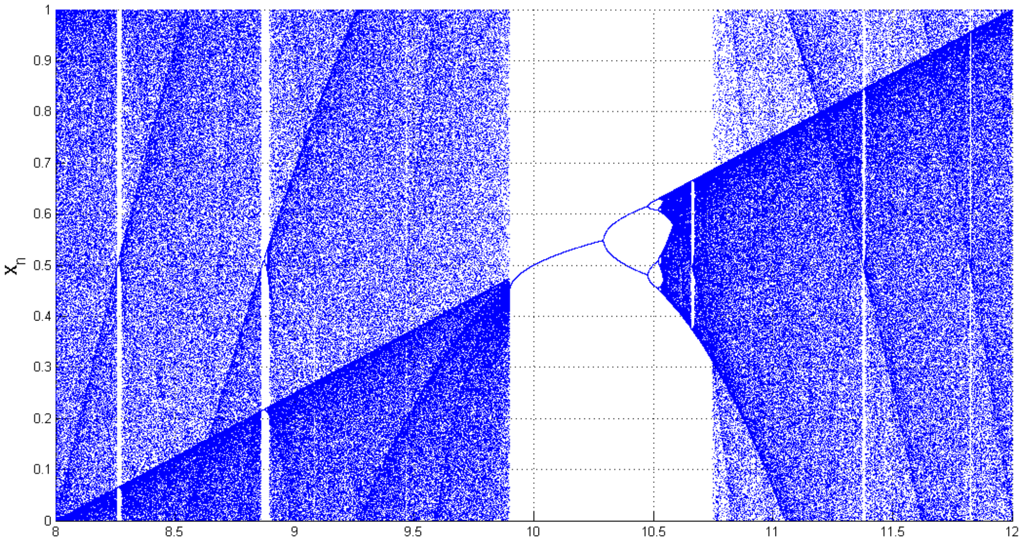


Figure A- 5: Modified logistic map bifurcation diagram with t∈[8, 12]

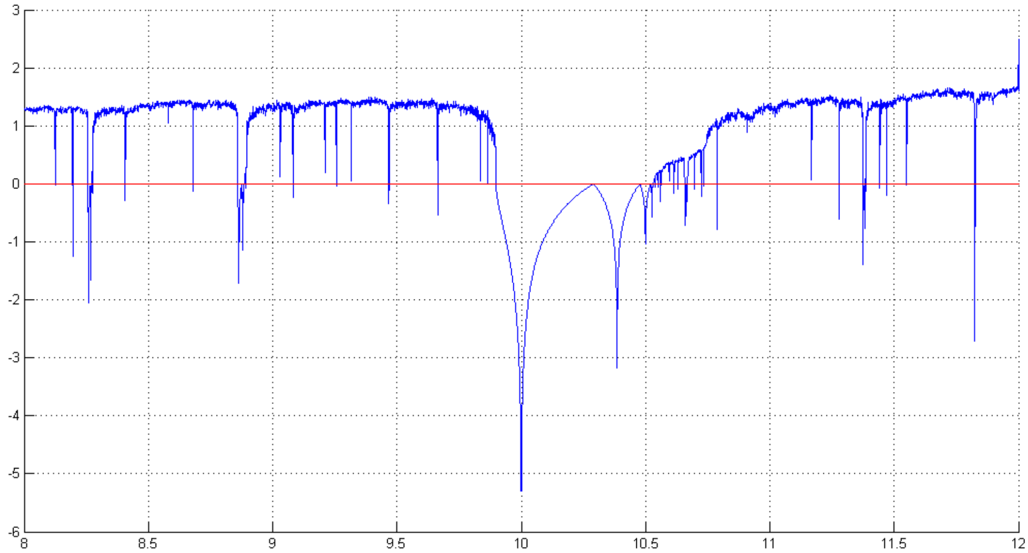


Figure A- 6: Lyapunov exponent of Modified Logistic map with∈[8, 12]

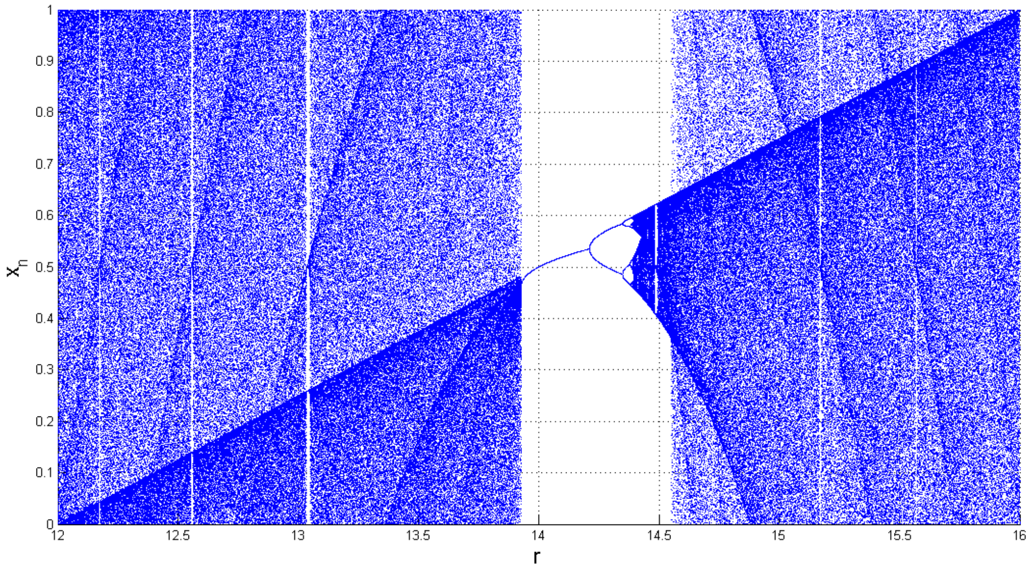


Figure A- 7: Modified logistic map bifurcation diagram with t∈[12, 16]

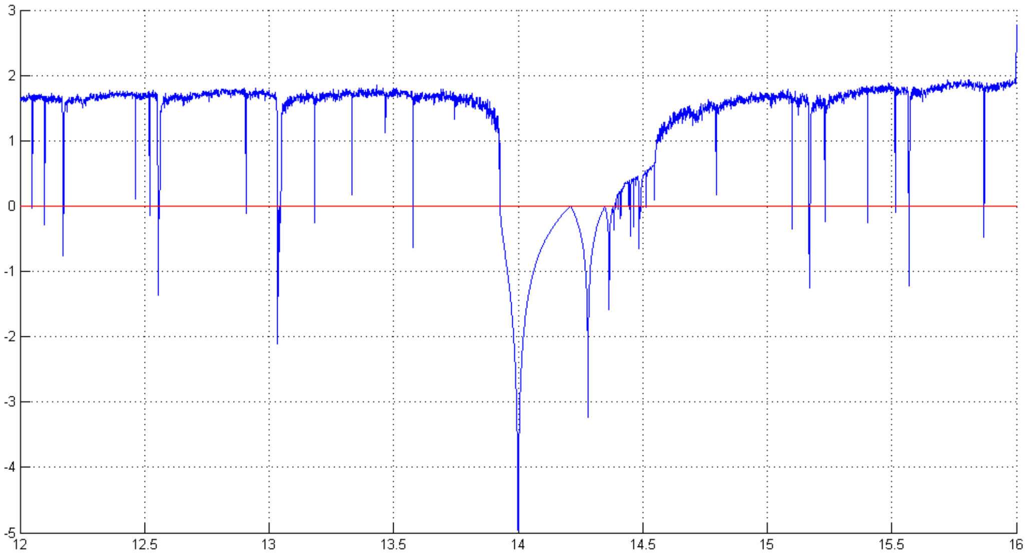


Figure A- 8: Lyapunov exponent of Modified Logistic map with t∈[12, 16]

Appendix B

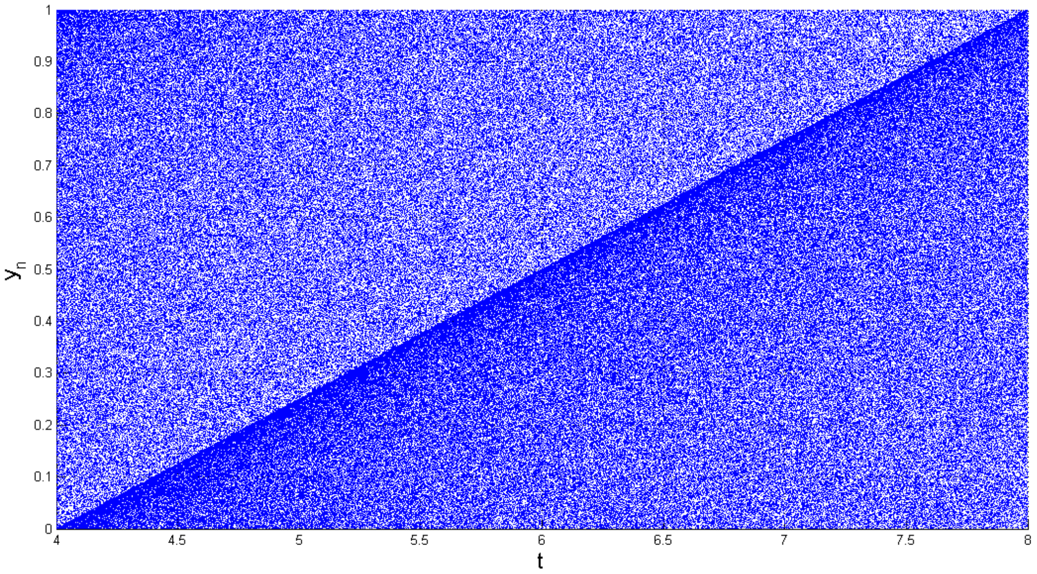


Figure B- 1: TCM chaotic map bifurcation diagram with t ∈ [4, 8]

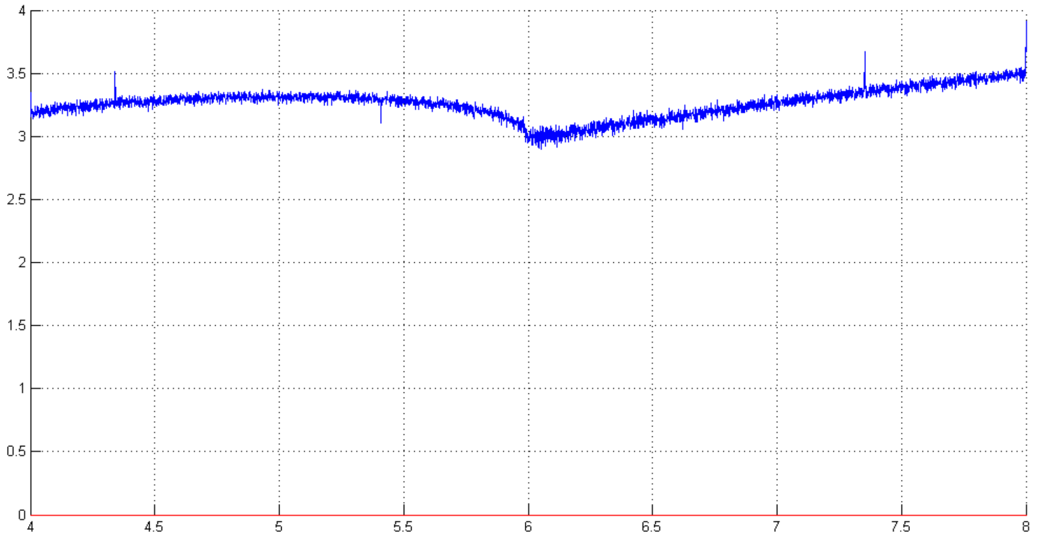


Figure B- 2: Lyapunov exponent of TCM chaotic map with t ∈ [4, 8]

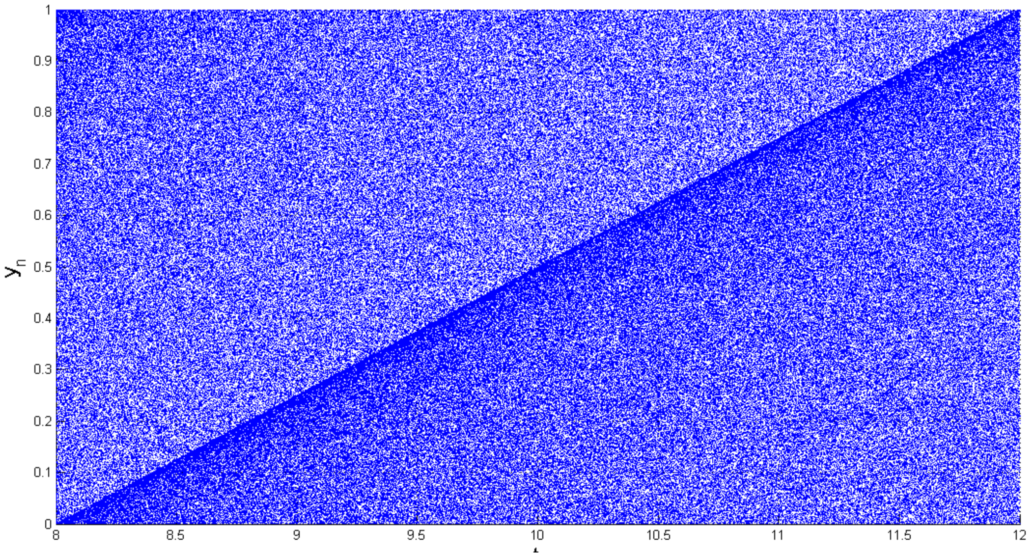


Figure B- 3: TCM chaotic map bifurcation diagram with t∈[8, 12]

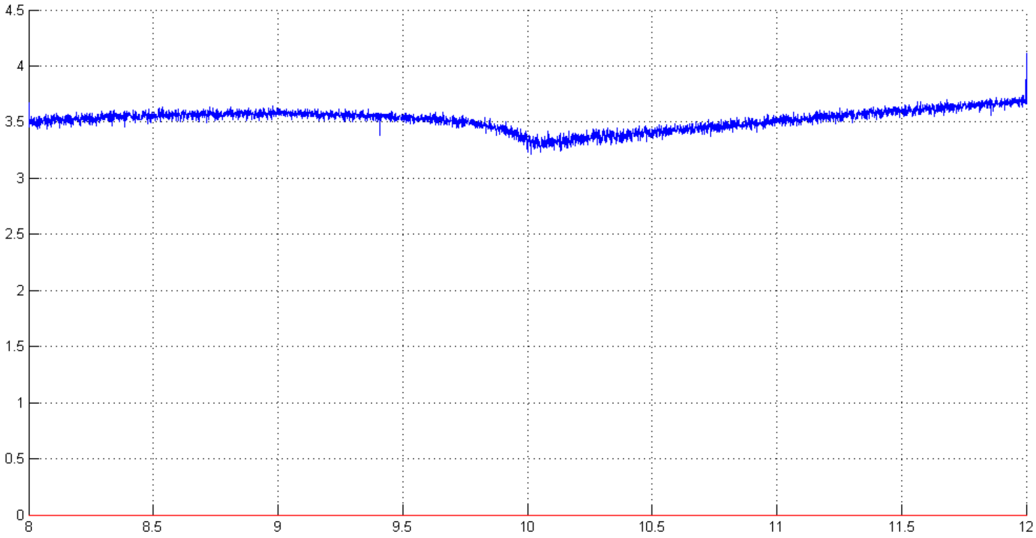


Figure B- 4: Lyapunov exponent of TCM chaotic map with t∈[8, 12]

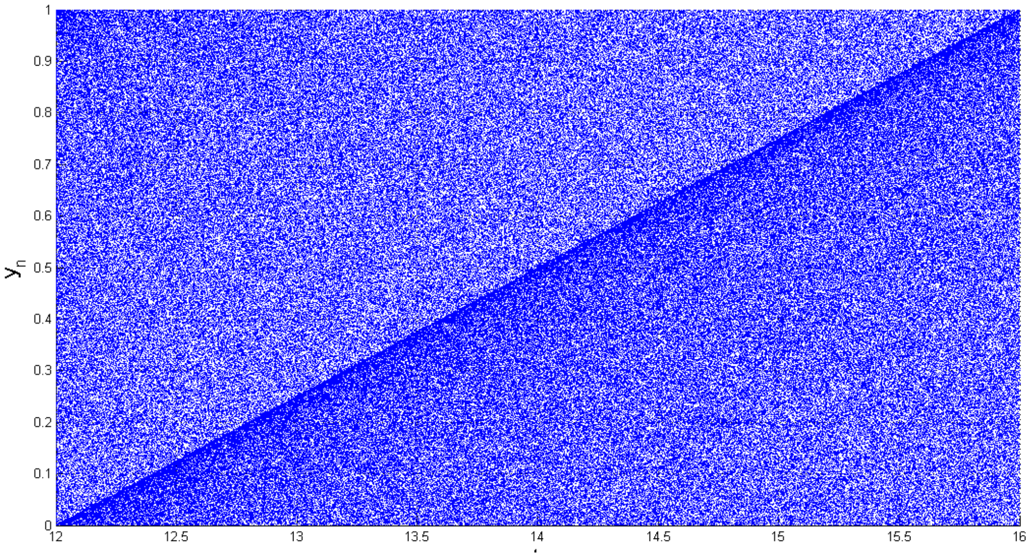


Figure B- 5: TCM chaotic map bifurcation diagram with t∈[12, 14]

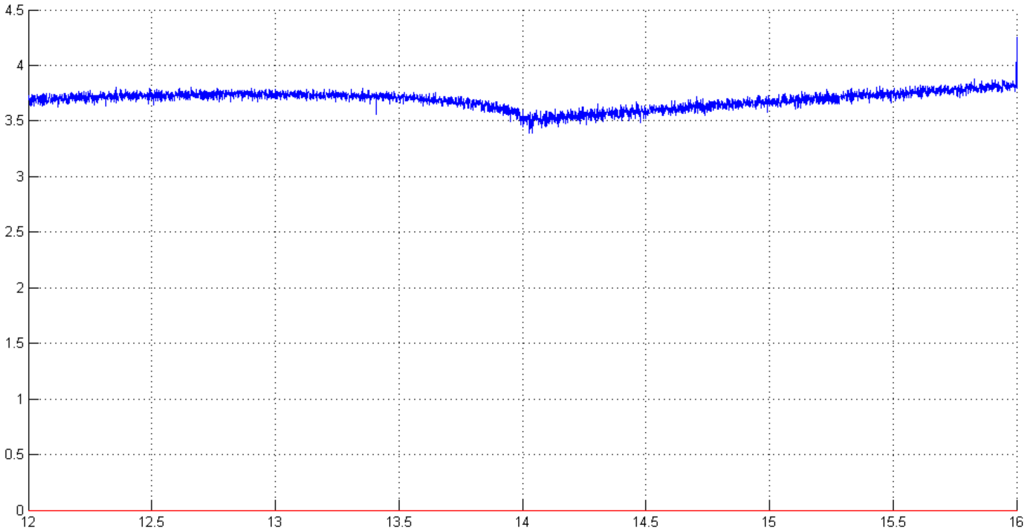


Figure B- 6: Lyapunov exponent of TCM chaotic map with t∈[12, 14]

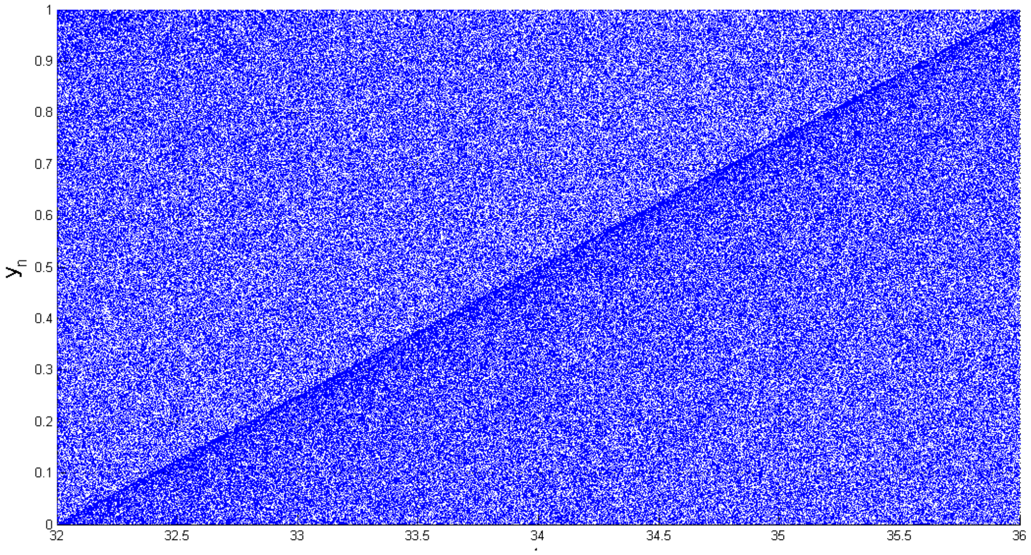


Figure B- 7: TCM chaotic map bifurcation diagram with t∈[32, 36]

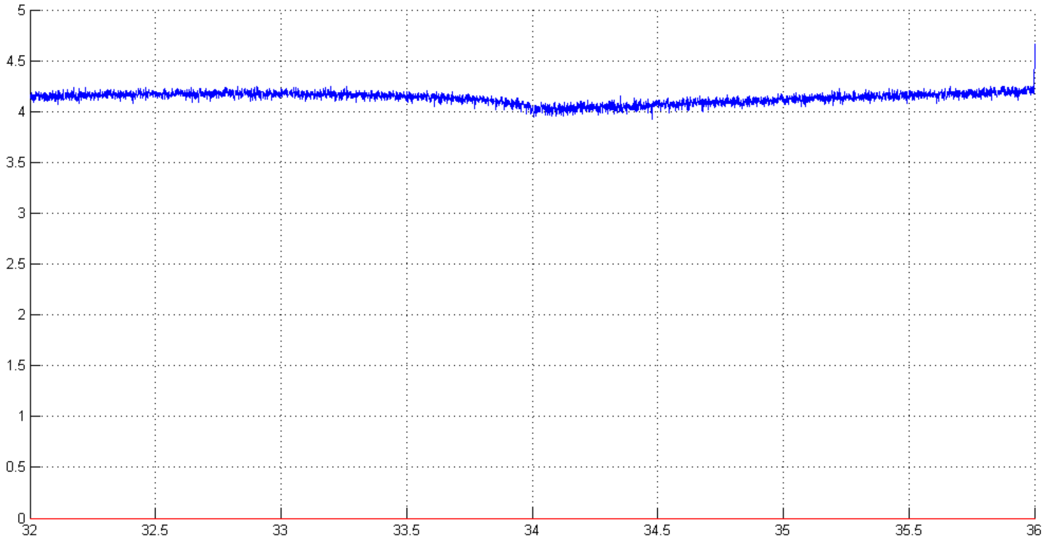


Figure B- 8: Lyapunov exponent of TCM chaotic map with t∈[32, 36]

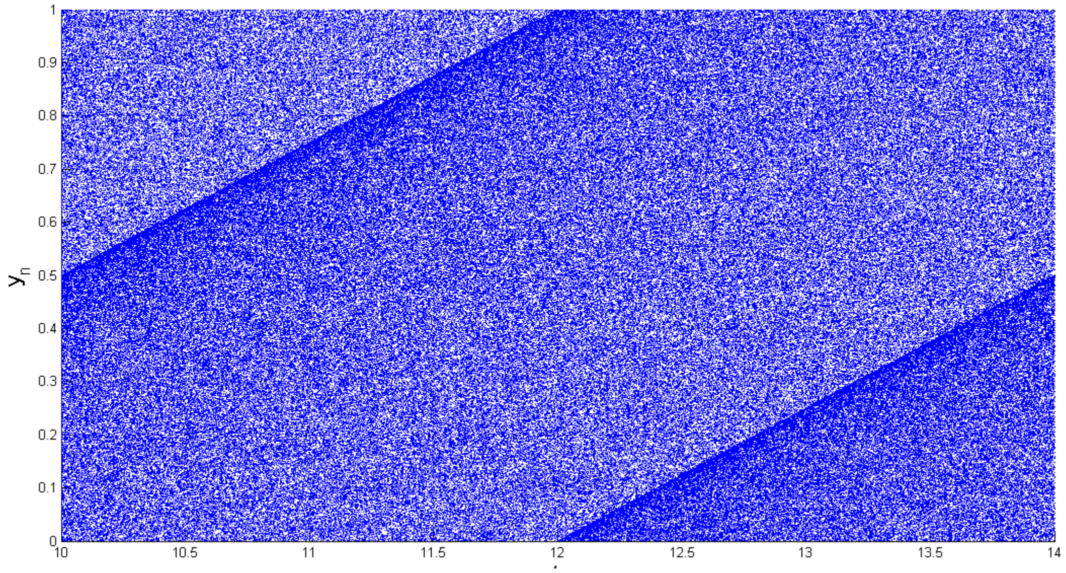


Figure B- 9: TCM chaotic map bifurcation diagram with $t \in [10, 14]$

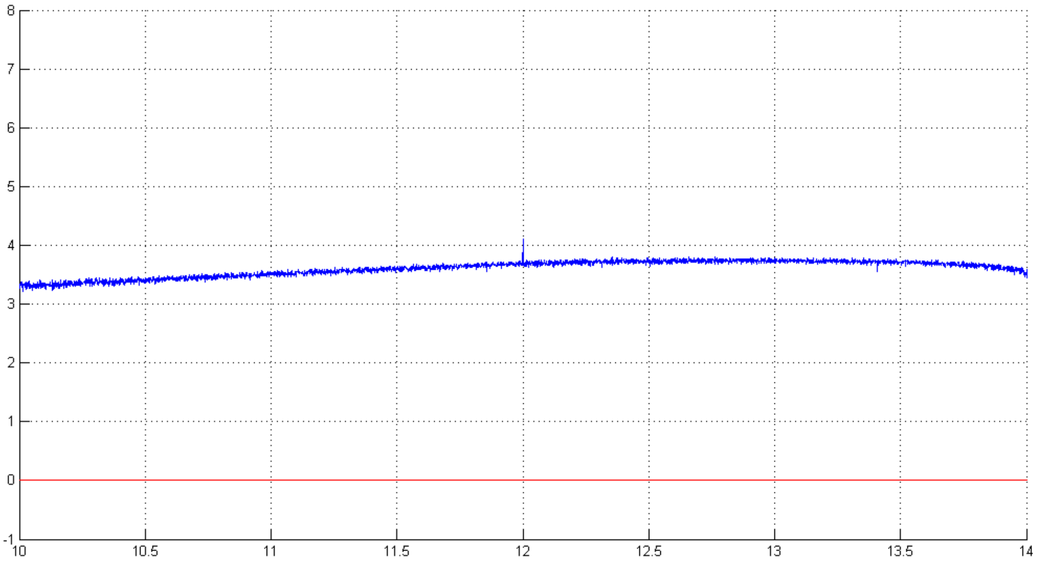


Figure B- 10: Lyapunov exponent of TCM chaotic map with $t \in [10, 14]$

Appendix C

Condition 1: The original message as it is.

Condition 2: Change first character "A" in the original message to "B".

Condition 3: Change the first character in word "cryptographic" from "c" small letter to "C" capital letter.

Condition 4: Remove the full stop from the end of the statement.

Condition 5: Remove the blank space between "A" and "cryptographic".

Condition 6: Change the word "data" in original message to "bata".

Condition 7: Add a blank space at the end of the original message.

Condition 8: Calculate the hash value with K_0 = 0.98974185582137419145 5.

Condition 9: Change the secret key to K_0 = 0.98974185582137419145 6.

Condition 10: Change the secret key to K_0 = 0.98974185582137419145 7.

The correspondence calculated 160-bit hash values in hexadecimal format for each condition as follows:

Condition 1: 44 15 51 1B 3D DF DC 6D 55 B9 B0 23 29 AB E6 20 D4 D4 BF BF

Condition 2: 13 92 7A 4E 75 0D 8D 48 3A 74 D3 9C 96 47 BB 67 04 27 A5 33

Condition 3: AA 4D 87 D7 9D DE 6E D3 E3 58 F0 07 F8 05 8C 0F 79 B3 03 54

Condition 4: FA 08 CE A3 1E BC BB 35 7A 0F B1 1E 37 7D 41 E1 A0 41 FA FD

Condition 5: E6 FE F0 9A C8 4C C6 4E 37 A5 90 13 46 ED FB 13 DE A7 CF 47

Condition 6: 62 80 03 87 10 18 25 5F 7D 70 F4 CB 18 3E FA 4C DC 96 43 87

Condition 7: 3B 7F C0 8D BE 3C F6 38 75 31 51 AE 7A C9 99 78 09 2B 83 66

Condition 8: 74 BB 46 1D C5 BE 6F 9E 12 9C B9 C7 DF BC 2D 78 E9 77 2E 2D

Condition 9: E0 B2 15 03 F1 6C B6 26 29 DF A6 ED 9F 22 09 3B 05 84 5D 02

Condition 10: 20 3C 81 B1 99 5E 80 18 DF 47 30 E8 6E DF 3A 2A A2 87 AC 4E

Appendix D

Scenario 1: The original message as it is.

Scenario 2: Change the first character “T” in the original message into “U”.

Scenario 3: Change the first character “s” small letter in the word "security" to "S" capital letter.

Scenario 4: Remove the full stop from the end of the statement.

Scenario 5: Remove the first blank space in “The computer”.

Scenario 6: Change the word “to” in the original message into “do”.

Scenario 7: Change the word “programs” in the original message into “programmes”.

Scenario 8: Add a blank space to the end of the original message.

Scenario 9: Add the number zero at the beginning of the original message.

Scenario 10: Remove the comma after the word “data”.

Scenario 11: k = (2F050FE938943ACC45F65567FFFFFFFFF)$_{16}$ is the used key.

Scenario 12: k = (2F050FE938943ACC45F65567FFFFFFFFD)$_{16}$ is the used key.

Correspondence hash values in hexadecimal format for each of the previous scenarios are as follows:

Scenario 1:

1D B4 03 A7 BA 17 5B 51 10 E2 51 FC 53 CE 0B CE AB 55 96 F7 99 B2 E7 1F F7 C0 7D D9 89 6A A0 FD

Scenario 2:

F4 58 9F 14 E0 B3 06 21 6B DE DF 6D 55 6E DC 3A 0D 3A EE A6 01 75 AC 62 61 88 2C 49 46 7B 26 03

Scenario 3:

BA 54 C0 28 33 7C 73 A2 BE 76 67 5B 73 8A 6B 2C 5D CC A8 61 3B 5F 06 8F C0 86 21 49 BF 07 F0 91

Scenario 4:

EB 63 91 2F 05 19 FC 1F F9 AC CC FE E9 9F 4B 48 FF 5B 18 5A 16 E1 DF A1 B0 E3 58 57 46 F5 A1 5E

Scenario 5:

48 E4 7B E0 33 38 D8 35 A0 01 39 0E 48 71 B4 79 BC 36 16 88 EA 8F 4F B1 40 96 ED 53 22 3F 16 3F

Scenario 6:

EC 4B 81 7E 73 22 A0 E1 5A CE 08 4B 8A C8 BD C1 01 B0 D9 EF C5 3C D5 4A E9 B0 A5 61 27 C5 EF 09

Scenario 7:

B3 91 A4 E6 0A 79 81 D3 DC F1 55 BE 7A 5B CE FC 8A B7 59 2E 2E 3E 0C 45 FB 89 C6 41 2A 67 2B 7E

Scenario 8:

63 84 36 51 CF D7 EA 00 2F A6 F2 58 D0 FA 92 4E 45 79 4C 32 2B B4 C7 FF 10 10 B2 FC 25 00 C8 E4

Scenario 9:

EE 6C 00 77 B5 27 43 97 EA DE D9 72 59 80 15 5E EE F0 D9 56 0F 44 E0 AC 5D B6 C7 ED FA 51 86 EF

Scenario 10:

BB 08 75 DC 8D 1B 6A 07 5D 29 C3 70 74 8D 89 D9 C6 29 BF AC 81 1C BD A9 5C 22 74 63 F4 67 44 80

Scenario 11:

8F 29 3F 24 D3 9C 88 CC EF 55 78 F6 48 AC A0 91 B3 A4 0D 88 1D AD 62 D6 84 43 73 D8 31 9B A2 24

Scenario 12:

12 61 EC 23 72 C9 76 A5 FB 8D 3D FB 2C 95 EA EF 89 5D 0C 4A 5E 8F 15 33 FF A0 F7 F5 B3 03 71 C9

Appendix E

Scenario 1: Ayham image with 8-bit greyscale is used.

Scenario 2: Add 1 to the value of the pixel located at the upper left corner.

Scenario 3: Subtract 1 from the value of the pixel located at the upper left corner.

Scenario 4: Add 1 to the value of the pixel located at index x = 250, y = 188.

Scenario 5: Add 1 to the value of the pixel located at the lower right corner.

Scenario 6: Subtract 1 from the value of the pixel located at the lower right corner.

Scenario 7: Change the last digit in the key from number F to E.

Scenario 8: Change the last digit in the key from number F to 0.

The calculated 256-bit hash values in hexadecimal format for each of the previous scenarios are as follows:

Scenario 1:
E7 90 B2 C1 A8 F1 A5 7E 8B C1 53 0A AE BB 25 71 D3 5 8 03 17 A7 4F F2 D2 A6 B2 51 01 B8 1B 44 DC

Scenario 2:
55 DF F3 19 A0 2F A4 58 5A 37 C3 4C 4D A6 B5 B8 ED BB EF C4 48 4F A1 CF 04 A4 57 7D 38 9D 27 92

Scenario 3:
42 27 72 5B1 90 868 F7 44 5B9 2D0 E6 CE 58 2B F1 48 81 77 B4 09 CB A1 7A D6 24 66 18 79 3A F8

Scenario 4:
53 DB 4D 77 73 15 75 93 56 C2 53 81 96 78 7B 37 0F 2C 4C C1 20 EC 04 5F A3 61 B9 F6 F5 BB FF C9

Scenario 5:

09 EF 06 66 73 93 BB 7C F6 C4 97 AD DE 2C 0E 10 41 A7 16 DD E2 96 98 66 94 00 5D 56 D2 E2 D1 79

Scenario 6:

70 10 87 DB 7D 13 4F B4 02 0E D3 86 E6 5C 35 D7 00 35 C6 54 D2 74 43 12 A8 0A 37 AA 86 3F D9 C6

Scenario 7:

21 83 3C A7 CA 87 0E 32 CB DD 7D C7 F3 ED BB 8F FA 7D 0D A7 C5 76 95 16 EA 1D C2 36 B9 35 EF CA

Scenario 8:

62 08 4E B6 9C 05 83 AE 23 2E 35 EC 19 8E 12 2E E5 A4 A8 7F 13 A5 72 29 8F CB 2A 09 2A 89 B9 2E

Appendix F

NIST Statistical suite test

NIST (National Institute of Standard Technologies) developed a statistical package to test the randomness of generated bit sequences for cryptography applications [211]. It contains 16 tests and each test contains several subtests. The focus of these tests is to confirm the existence of non-randomness in the input bit sequences. A number of tests have chi-square (x^2) and standard normal to refer to distribution. In each test, p-value for each sample in the sequence will be computed, thus m p-values will be computed for each test. A p-value of greater than or equal to α = 0.01 would be considered a random sequence with a confidence level of 99%. A p-value of less than α = 0.01 would be considered a non-random sequence with a confidence level of 99%.

The recommended significance level (α) value is in the range $0.0001 \leq \alpha \leq 0.01$. For the statistical tests in this paper the value of α has been chosen to be 0.01. If the computed p-value is greater than the chosen significance level (α = 0.01), the sequence will be considered a random sequence. In random sequences, if the chosen value of α is 0.01, it would be acceptable to reject only one sequence out of each 100 sequences. The computed p-value$_s$ of p-values should be greater than 0.001 to be considered uniformly distributed. We can broadly classify these tests into two main types: parameterized tests and non-parameterized tests. In the following subsection, we will describe the NIST suite test in high-level description.

1. Non-parameterized tests

1.1 Frequency (Monobit) Test

A frequency test concentrates on the number of ones and zeros in generated bit sequences. This test aims to find out whether the number of zeros and ones in an input bit sequence are roughly equal to the number expected for a truly random input sequence. The proportions of ones and zeros should be very close to each other. Based on the number of zeros and ones in the sequence, the p-value will be computed. The input sequence is concluded to be a random sequence if the computed p-value is greater than or equal to 0.01. Otherwise, the input sequence is concluded to be a non-random sequence. The minimum input sequence length (*n*) in this test is recommended to be 100 bits. All other NIST suite test tests depend on passing this test.

1.2 Run Test

A run test concentrates on the entire number of runs in the input sequences. This test aims to find out whether the number of runs of zeros and ones is as expected for an input random sequence of different length. A run is a continuous sequence of matching bits. A run of size *l*-bit consists of *l* matching bits. This test helps to determine whether the fluctuation between ones and zeros is very slow or very fast. The input sequence is concluded to be a random sequence if the computed p-value is greater than or equal to 0.01. Otherwise, the input sequence is concluded to be a non-random sequence. The minimum length of input sequence (*n*) in this test is recommended to be 100 bits.

1.3 Longest Run Test

A frequency test concentrates on the longest run of ones within a block of *l* bits size. This test aims to find out whether the length of the longest run of ones of the input sequences is steady with the length of the longest run of ones expected in input random sequences. The main focus of this test is on longest run of ones, but it implies the longest run of zeros. P-values are used to determine whether the input sequences are random or not. The input sequence is concluded to be a random sequence if the computed p-value is greater than or equal to 0.01. Otherwise, the input sequence is concludes to be a non-random sequence. The minimum input sequence length (*n*) in this test depends on sub-block size.

1.4 Binary Matrix Rank Test

The rank test concentrates on the rank of disjointed sub-matrices of complete input sequences. This test aims to check for linear dependence between fixed-length streams of input sequence. The input sequence is concluded to be a random sequence if the computed p-value is greater than or equal to 0.01. Otherwise, the input sequence is concluded to be a non-random sequence. The minimum input sequence length (*n*) in this test is recommended to be *38MQ*, where *M* and Q are the number of rows and columns in each matrix, respectively.

1.5 Discrete Fourier Transform (DFT) Test

The DFT test concentrates on testing the peak highest in the Discrete Fourier Transform of the input bits sequence. This test aims to find out the nearby repetitive patterns in the input sequence that could deviate from the random sequence. This test detects whether the number of peaks that are greater than the threshold (95%) is considerably different from 5% (1.00-0.95) (see equation F-1). If the calculated test value exceeds the threshold value (95%), then we conclude the sequence is not random. Otherwise, we conclude the sequence is random. The input sequence is concluded to be a random sequence if the computed p-value is greater than or equal to 0.01. Otherwise, the input sequence is concluded to be a non-random sequence. The minimum input sequence length (*n*) in this test is recommended to be 1000 bits.

$$T = \sqrt{\left(\log \frac{1}{0.05}\right) n} \tag{9-1}$$

1.6 Cumulative Sum Test

Cumulative sum (Cusum) test concentrates on the random walk highest excursion (from zero) in the sequence. This test aims to conclude whether the cumulative sum of sub-sequences occurring in the input sequence is very small or very large compared with the expected behaviour of the random sequence, which may be considered as random walk. Random walk excursions of random sequences should be close to zero for random tested sequences. The input sequence is concluded to be a random sequence if the computed p-value is greater than or equal to 0.01. Otherwise, the input sequence is concluded to be a non-random sequence. The minimum input sequence length (*n*) in this test is recommended to be 100 bits.

1.7 Random Excursions Test

Random Excursions test concentrates on the number of cycles that have exactly t visits in Cusum random walk. This test aims to find out whether number of visits to specific state within cycle deviates from what we expect for random sequence. Cycle random walk consists of unit length steps in the sequence starting from one point and returning to the same starting point (ex. 0, 1, 2, 1, 2, 1, 2, 0). This test consists of eight sub-tests and each one has one conclusion for each case: +4, +3, +2, +1 and -1, -2, -3, -4. The input sequence is concluded to be a random sequence if the computed p-value is greater than or equal to 0.01. Otherwise, the input sequence is concluded to be a non-random sequence. The minimum input sequence length (*n*) in this test is recommended to be 10^6 bits.

1.8 Random Excursions Variant Test

Random Excursions Variant test concentrates on the total number of occurrences of a specific state in Cusum random walk. This test aims to calculate the deviation from the expected number of visits for different states in random walk. This test consists of eighteen sub-tests and each one has one conclusion for each case: +9, +8, ..., +2, +1 and -1, -2, ..., -8, -9. The input sequence is concluded to be a random sequence if the computed p-value is greater than or equal to 0.01. Otherwise, the input sequence is concluded to be a non-random sequence. The minimum input sequence length (*n*) in this test is recommended to be 10^6 bits.

2. Parameterized tests

1.9 Frequency Test within Block

Frequency test within block concentrates on the fraction of ones within a certain block size. This test aims to find out the number of ones in a block of size *m*-bit. If the number of ones is very close to *m/2,* the block is assumed to be random. Otherwise, it is assumed to be non-random. P-value is computed based on the number of ones in the blocks of each sequence. The input sequence is concluded to be a random sequence if the computed p-value is greater than or equal to 0.01.

Otherwise, the input sequence is concluded to be a non-random sequence. A small p-value indicates a big variation from the fraction of ones and zeros in one or more blocks. The minimum input sequence length (n) in this test is recommended to be 100 bits.

1.10 Non-overlapping Template Matching Test

Non-overlapping Template Matching test concentrates on the number of predefined target strings in the input sequence. This test aims to find out whether sequence generators produce sequences with too many given non-periodic patterns. The search for patterns of b-bit size will be in a window of the same size. The window will shift one bit position until the pattern is found. If the pattern is in the current window, the search will be restarted from the bit immediately after the found pattern. The input sequence is concluded to be a random sequence if the computed p-value is greater than or equal to 0.01. Otherwise, the input sequence is concluded to be a non-random sequence. The minimum targeted string (template) length (*n*) in this test is recommended to be 9 or 10 bits.

1.11 Overlapping Template Matching Test

Overlapping Template Matching test concentrates on the number of predefined target strings in the input sequence. The search for patterns of b-bit size will be in a window of the same size. The window will shift one bit position until the pattern is found. The only difference between overlapping and non-overlapping template matching tests is that, in the non-overlapping matching template test, after the pattern is found, the sliding will be one bit only before the search is resumed. The input sequence is concluded to be a random sequence if the computed p-value is greater than or equal to 0.01. Otherwise, the input sequence is concluded to be a non-random sequence. The minimum-targeted string (template) length (*n*) in this test is recommended to be 9 or 10 bits.

1.12 Universal Statistical (Maurer's) Test

The Universal Statistical test concentrates on the number of bits between similar patterns in the input sequences. This test aims to find out the possibility of compressing the input sequence without loss of the information. If the sequence can be significantly compressible, then we conclude that the sequence is non-random. The input sequence is concluded to be a random sequence if the computed p-value is greater than or equal to 0.01. Otherwise, the input sequence is concluded to be a non-random sequence.

1.13 Linear Complexity Test

The Linear Complexity test concentrates on size of a linear feedback shift register. This test aims to find out whether the input sequence is complex enough. If the size of a linear feedback shift register is too short, the sequence will not be complex and will be considered non-random. Otherwise, the sequence will be considered complex and random. The input sequence is concluded to be a random sequence if the computed p-value is greater than or equal to 0.01. Otherwise, the input sequence is concluded to be a non-random sequence. The minimum input sequence length (*n*) in this test is recommended to be 10^6 bits and the length of each block is greater than or equal to 500 and less than or equal to 5000.

1.14 Serial Test

The Serial test concentrates on the frequency of every possible overlapping b-bit patterns over the entire input sequence. This test aims to find out whether the incidence of the 2^s b-bit overlapping patterns is roughly same as that expected for a random bit sequence. The chance of every b-bit occurring in the sequence is equal to any other bit of the same size, which is called uniformity of random sequence. The input sequence is concluded to be a random sequence if the computed p-value is greater than or equal to 0.01. Otherwise, the input sequence is concluded to be a non-random sequence. The input sequence length (*n*) and block length (*b*) in this test is recommended to be such that $b < (\log_2 n)-2$.

1.15 Approximate Entropy Test

Approximate Entropy test concentrates on the incidence of every possible overlapping b-bit pattern over the complete input sequence. This test aims to compare the incidence of two adjacent lengths (*b* and *b+1*) of overlapping blocks with the expected result for random input sequence. The input sequence is concluded to be a random sequence iff the computed p-value is greater than or equal to 0.01. Otherwise, the input sequence is concluded to be a non-random sequence. The input sequence length (*n*) and block length (*b*) in this test is recommended to be such that $b < (\log_2 n)-5$.