AN ENHANCED IMAGE CRYPTOSYSTEM USING BLOWFISH AND DIFFIE-HELLMAN WITH RESIDUE NUMBER SYSTEM

ABDULHAMEED YUNUSA 18/27/MCS022

MARCH, 2021

SCHOOL OF POSTGRADUATE STUDIES (SPGS)

AN ENHANCED IMAGE CRYPTOSYSTEM USING BLOWFISH AND DIFFIE-HELLMAN WITH RESIDUE NUMBER SYSTEM

BY

ABDULHAMEED YUNUSA

18/27/MCS022

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTER OF SCIENCE (M.Sc.) DEGREE IN COMPUTER SCIENCE

A THESIS SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY, KWARA STATE UNIVERSITY, MALETE, NIGERIA.

MARCH, 2021

DECLARATION

I, **ABDULHAMEED, Yunusa** with Matriculation Number **18/27/MCS022** hereby declare that this thesis titled "**An Enhanced Image Cryptosystem Using Blowfish and Diffie-Hellman with Residue Number System**", is a record of my research. It has neither been presented nor accepted in any previous application for higher degree.

ABDULHAMEED Yunusa Signature / Date

APPROVAL PAGE

This is to certify that this thesis was carried out by **ABDULHAMEED Yunusa** with the matriculation number **18/27/MCS022** has been read and approved as meeting the requirements of the Department of Computer Science for the award of the degree of Master of Science (M.Sc.) in Computer Science.

Prof. K. A. Gbolagade — [illegible] 05/05/2021

Main Supervisor — Signature / Date

Dr Ronke S. Babatunde — [illegible] 4/5/21

Co-Supervisor — Signature / Date

Prof. K. A Gbolagade — [illegible] 05/05/2021

Head of Department — Signature / Date

Dr. Lambe Adesina — [illegible] 16/05/2021

Internal Examiner — Signature / Date

External Examiner — Signature / Date

Prof. J. Hamzat Abdul Raheem — [illegible] 17-05-21

RECEIVED

Dean, — Signature / Date

School of Postgraduate Studies(SPGS)

DEDICATION

This thesis is dedicated to Almighty God, who has been my guide throughout the duration of my Post Graduate Programme at Kwara State University, Malete, Nigeria.

ACKNOWLEDGEMENTS

All praise is due to Almighty God for sparing my life up to this moment and granting me wisdom, knowledge and understanding that was required to make the programme a successful one.

I sincerely appreciate the effort of my supervisor, Head of Department and The Dean, Faculty of Information and Communication Technology, Prof. Kazeem Alagbe Gbolagade, for his mentorship and welcoming heart and for taking out of his precious time to guide me through this thesis. Special appreciation to the Dean, School of Post Graduate Studies, Prof. Hamzat I. Abdulraheem and the Secretary, School of Postgraduate Studies, Dn. Micheal O. Dada.

I will love to express my profound gratitude to my parents, Mr. and Mrs. Yunusa, for their parental support financially, mentally, with a lot of encouragement and immeasurable prayer that got me through this. Also, great thanks to Head of Department, Computer Science of Al-Hikmah University, Dr Y.K. Saheed and Mrs. S. Ibrahim of the same department.

Finally, I also convey my regards to the Post Graduate Coordinator; Dr. R.M. Isiaka and all other staff of department; Dr. J.F. Ajao, Dr. R.S. Babatunde, Dr. A.N. Babatunde, Mr. S.O. Abdulsalam, Mrs. S. Yusuf, Mrs. F. Balogun, Mr. D. Popoola and Mr. A. Kadri.

Thank you all, may Almighty Allah bless you all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

Abstract

With the rapid progress in internet and digital imaging technology, there are more and more ways to easily create, publish and distribute images. The major issue for computer networks is to prevent important information from being disclosed to illegal users. However, valuable multimedia content such as digital images, however, is vulnerable to unauthorized access while in storage and during transmission over a network. In this thesis, a new method for encrypting and decrypting images is proposed. The proposed method is implemented by combining the concepts of Diffie Hellman algorithm and Blowfish algorithm together with Residue Number System (RNS). In this technique, a computer user will encrypt an image file using residue number system with respect to the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$, then the cipher residue is re-encrypted using blowfish algorithm that uses a secret key generated by Diffie-Hellman key exchange to produce the cipher image to be sent to the receiver. The proposed scheme produces the cipher image with less computational time and improve the security of the image by encrypting the image twice, which makes it harder to be cracked by intruders.

Keywords: Blowfish; Image; Computational time; Cryptography; Residue number system; Diffie-Hellman.

CHAPTER ONE

INTRODUCTION

1.1 Background to the Study

In order to protect data through the unsecure networks like the Internet, using various types of data protection is necessary. One of the famous ways to protect data through the Internet is Cryptography. Cryptography is more necessary now because of the increasing number of internet users. As digital image plays an important role in multimedia technology, it becomes more important for the user's to maintain privacy. And to provide such security and privacy to the user, image encryption is very important to protect from any unauthorized user access. With the high usage of digital and sophisticated methods for and storing and transfer of images, the issue of protecting the authenticity, integrity and confidentiality of images has become a great concern. Encryption of image is really essential in areas such as confidential transmissions, video surveillance, law enforcement, military operations and medical sciences in order to ensure its authorized access. Encryption refers to encoding of image with some algorithm while decryption is decoding the image with reverse steps. Image encryption not only prevents the encrypted image from being intercepted but also ensures that image can only be decrypted, interpreted and viewed by the intended person. Cryptography is an historical science that began in Egypt around 1900 B.C. with hieroglyphic writing, (Siper, 2005). It uses encryption to scramble the secret information in such a way that only the sender and the intended receiver can reveal it, (Selvaraj, 2017). Encryption is often applied in two different forms, a symmetric key or an asymmetric key. A symmetric key, or secret key, uses one key to both encode and decode the information. This is best used for one to one sharing and smaller data sets. Asymmetric, or public key cryptography, uses two linked keys – one private and one public. The encryption key is public and can be used by anyone to encrypt. The opposite key is kept private and used to decrypt.

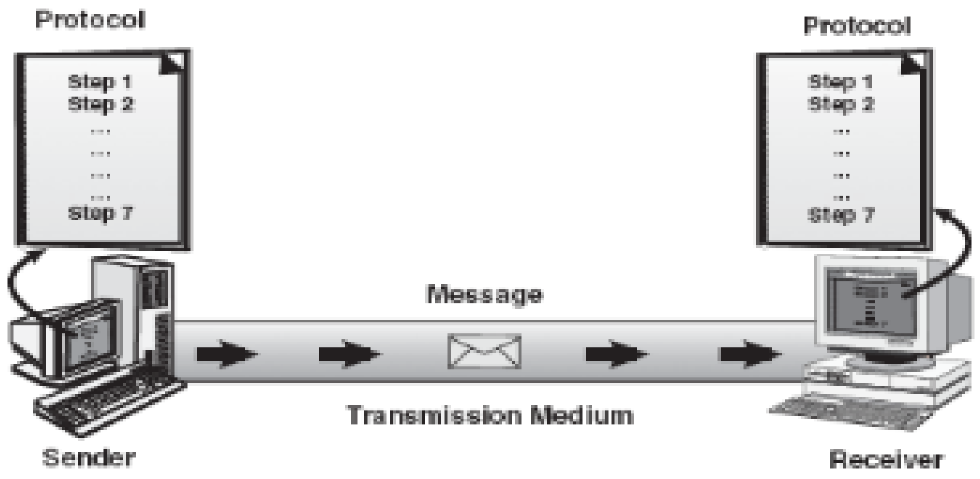


Figure 1.1: Basic Data Communication Process (Oreilly, 2020**)**

Security is the most important factor to be considered when the confidential information is transmitted through the internet. The prevention of unauthorized access to the confidential data is achieved by the two security features known as cryptography. cryptography deals with the encryption and decryption process of the data that is transmitted through an open network, (Meera & Malathi, 2015). Cryptographic algorithm works by combining with a key a word, number, or phrase to encrypt the plaintext.

In cryptography, keys exchange is a method to send a key between a sender and a recipient. The problems of the key exchange are how they send the message so that nobody else can understand the message except for the sender and the recipient. The procedure is one of the first public key cryptographic protocols used to build up a secret key between each other over insecure channel. Diffie-Hellman is suitable to use in information communication and less frequently use for information storage or archived over a long time period, (Wachirapong & Poom, 2018).

Diffie-Hellman key exchange is a mathematical algorithm that permits two PCs to produce an identical shared secret on both systems, despite the fact that those systems might never have communicated with one another. That shared secret can then be utilized to safely exchange a cryptographic encryption key. That key then encrypts traffic between the two systems, (Manku, Saikumar. & Vasanth, 2015).

Blowfish is a type of symmetric block cipher generated by B. Schneier in 1993. Blowfish is fast algorithm, license free, and unpatented (Yahia, Mohamad & Saleh, 2019). It uses a key length in the range of 32–448 and a sixty-four-bit block. The Blowfish algorithm makes use 16 round for the encipherment procedure Fig. 3. Blowfish ordinarily makes use of 4 S-boxes rather than of one S-box. It requires additional processing time because it relies on key length, however it provides strong safety (Faheem, et.al., 2017).

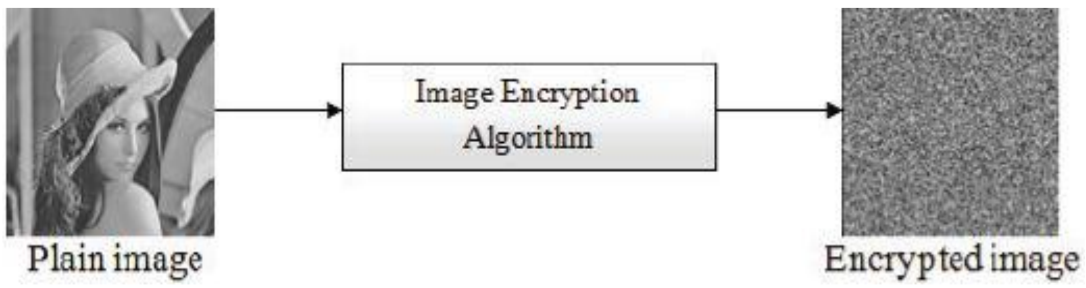


Figure 1.2: Basic Image Encryption (Das, Kumar & Sreenivasulu, 2014**).**

Residue Number System (RNS) is an integer number system with the competency to support parallel, carry-free addition, borrow-free subtraction and single step multiplication without unfinished product. Data Conversion in RNS is usually based on either the Chinese Remainder Theorem (CRT) or the Mixed Radix Conversion (MRC), which can be categorized into forward and reverse conversions. The forward conversion involves converting a binary or decimal number into it RNS equivalent while the reverse conversion is the inverse operation, which involves converting RNS number into binary or decimal. Relatively, reverse conversion is more complex (Aremu & Gbolagade, 2017).

The major issue to design any encryption and decryption algorithm is to improve the encryption and decryption times as well as security level. Therefore, this study aims to propose an enhanced approach to improve the security performance and lower the computational time of the system that will enable a secure transfer of data over an unsecured medium.

1.2 Statement of Problem

Various methods have been proposed and implemented towards the course of ensuring data security, it is to be noted that all information sent across from the sender to the receiver passes through a communication line (mostly unsecured as secured transmission increases cost and setup) which is monitored by the network administrator, a 3rd party or an internet service provider. The information can be intercepted and read. This, therefore, calls for a massive task of proffering a means through which files and images can be sent over insecure medium and decrypted with very minimal delay.

There are many techniques of cryptography that has been proposed which ensures date confidentiality and Integrity. But computational time and one-layer security are the problem with most of these techniques.

Recently, (Hazra, et.al., 2017), proposed an approach that implement hybrid cryptosystem of image and text files using Blowfish and Diffie-Hellman techniques, this approach used blowfish to encrypt and decrypt while the key used in transmitting the data over an unsecured medium is generated using diffie-hellman algorithm, the proposed method was able to achieve it aim by being able to protect data against 'Logjam' attack. But absence of time stamp here made it prone to permutation and computation attacks, and the time consumption and 'Birthday attack' problem of blowfish algorithm still remains and still a major challenge to be addressed. Having understood (Hazra, et.al., 2017) method and identify it problem, it is obvious that a better scheme is required, which will increase reduce the cracking probability also lower the processing time consumption.

1.3 Aim and Objectives

The aim of this research is to enhance blowfish algorithm and Diffie-Hellman key exchange with Residue Number System

The objectives are:

i. Formulate a multilevel model for encryption and decryption process using Blowfish with Diffie-Hellman and residue Number system with respect to $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$
ii. Implementation of the formulated model in (i)
iii. Evaluate and compare the performance of the system.

1.4 Scope of the Study

The proposed scheme is limited to the use of Residue Number System with respect to the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$ and Blowfish algorithm for encryption and decryption of an image while Diffie-Hellman key exchange is used to generate the key use in transmission of the image over an unsecured medium.

1.5 Significance of the study

The proposed scheme will lower the cracking probability of encrypted data to be sent securely over an unsecured network and use residue number system parallelism feature to counter the long computational time of blowfish algorithm.

1.6 Research Layout

This research comprises five chapters.

Chapter one brings an introduction to the research, showcasing the background of the study, the statement of problem to justify the reason behind the proposed scheme, the aim and objectives to solve the identified problem, as well as the scope and significance of the study.

Chapter two deals with review of literature, comprising of previous related works and concepts that are related to the research. Chapter three describes the existing system as well as the mode of operation of the proposed system. Chapter four presents the result, evaluation and comparison of the scheme. Chapter five consists of the summary, conclusion, major contributions to knowledge and recommendation of the proposed scheme as well as relevance for future works on the research area.

CHAPTER TWO

LITERATURE REVIEW

2.1 Related Concept

2.1.1 Data Security

Data security refers to the process of protecting data from unauthorized access and data corruption throughout its lifecycle. Data security includes data encryption, hashing, tokenization, and key management practices that protect data across all applications and platforms. Data security is a set of standards and technologies that protect data from intentional or accidental destruction, modification or disclosure. Data security can be applied using a range of techniques and technologies, including administrative controls, physical security, logical controls, organizational standards, and other safeguarding techniques that limit access to unauthorized or malicious users or processes. Data security technology comes in many shapes and forms and protects data from a growing number of threats. Many of these threats are from external sources, but organizations should also focus their efforts on safeguarding their data from the inside, too. Ways of securing data include: Data encryption, Data masking, Data erasure and Data resilience.

2.1.2 Cryptography

Cryptography is a science that studies how data is converted into a specific design that difficult to be understood. Cryptography aims to maintain the confidentiality of information or data that cannot be identified by unauthorized parties (unauthorized person). Data can be encoded as plaintext or cleartext. Furthermore, the data that has been encrypted called ciphertext. In cryptography required parameters used for the data conversion process that is a set of keys. Encryption and decryption of data are controlled by a key or some keys. Two Greek words ‘Kryptos’ meaning ‘secret’ and ‘Graphein’ meaning ‘writing’ derive the word ‘Cryptography’. So Cryptography means ‘secret writing’, a science of transforming a message into an unintelligible form. The unencrypted message is called ‘plain text’ and after encryption, it is converted into an unintelligible form which is called ‘cipher text’ (Biswas, Gupta, & Haque, 2019). The cipher text is then sent over an insecure channel with the presence of a third party called adversary or intruder and at the receiving end after decrypting the cipher text again the plain text is found. Figure 2.1. illustrates the general concept of cryptography using a block diagram.

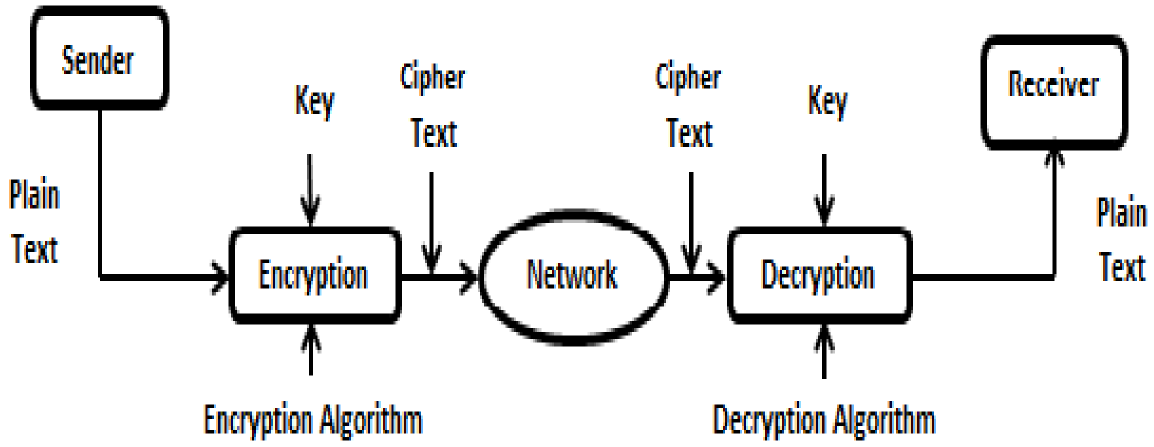


Figure 2.1: General concept of Cryptography (Biswas, Gupta & Haque, 2019).

In data and telecommunications, cryptography is necessary when communicating over any untrusted medium, which includes just about any network, particularly the Internet. There are five primary functions of cryptography; Privacy/confidentiality, Authentication, Integrity, Non-repudiation, Key exchange.

2.1.3 Types of Cryptography

1. **Secret key cryptography:** In secret key cryptography, a single key is for both encryption and decryption. The sender uses the key (or some set of rules) to encrypt the plaintext and sends the cipher text to the receiver. The receiver applies the same key to decrypt the message and recover the plaintext. Because a single key is used for both functions, secret key cryptography is also called symmetric encryption. With this form

cryptography, it is obvious that the key must be known to both the receiver and the sender; that, in fact, is the secret.

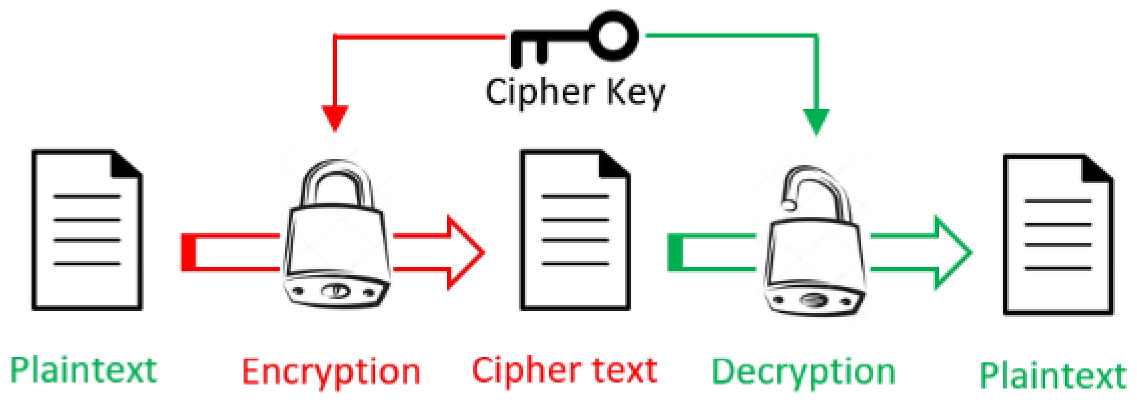


Figure 2.2: Symmetric Encryption (Taha, et.al, 2019).

2. **Public key cryptography:** Public or asymmetric key cryptography involves the use of key pairs: one private key and one public key. Both are required to encrypt and decrypt a message or transmission. The private key, not to be confused with the key utilized in private key cryptography, is just that, private. It is not to be shared with anyone. The owner of the key is responsible for securing it in such a manner that it will not be lost or compromised. On the other hand, the public key is just that, public. Public key cryptography intends for public keys to be accessible to all users. In fact, this is what makes the system strong. If a person can access anyone public key easily, usually via some form of directory service, then the two parties can communicate securely and with little effort, that is, without a prior key distribution arrangement.

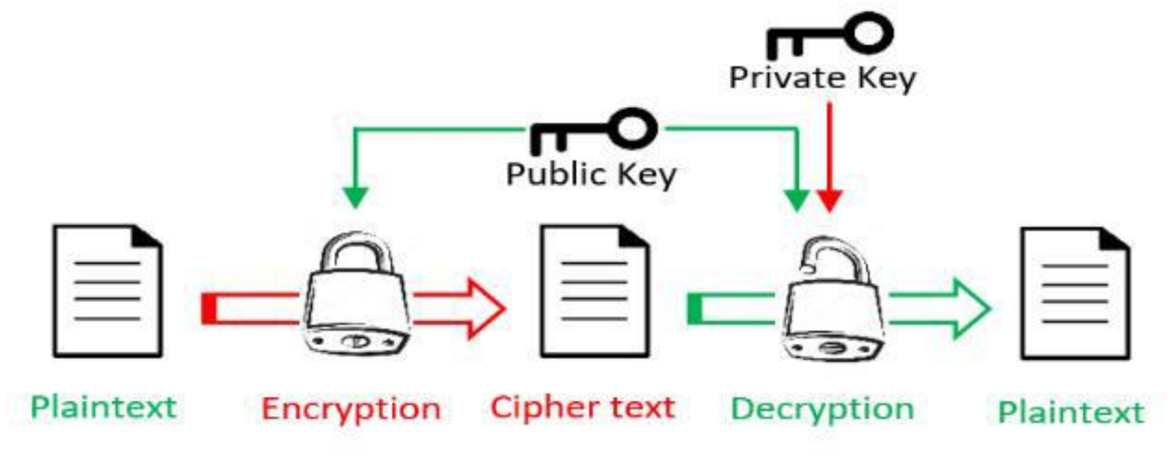


Figure 2.3: Asymmetric Encryption (Taha, et.al, 2019).

2.2 Diffie-Hellman Key Exchange

Diffie-Hellman is a mathematical algorithm that permits two PCs to produce an identical shared secret on both systems, despite the fact that those systems might never have communicated with one another. That shared secret can then be utilized to safely exchange a cryptographic encryption key. That key then encrypts traffic between the two systems. Diffie-Hellman is usually utilized when you encrypt data on the Web utilizing either SSL (Secure Socket Layer) or TLS (Transport Layer Security). The Secure Shell (SSH) protocol also uses Diffie-Hellman. Obviously, in light of the fact that Diffie-Hellman is a piece of the key exchange mechanism for IPSec, any VPN based on that technology uses Diffie-Hellman too.

In cryptography, keys exchange is a method to send a key between a sender and a recipient. The problems of the key exchange are how they send the message so that nobody else can understand the message except for the sender and the recipient. The procedure is one of the first public key cryptographic protocols used to build up a secret key between each other over insecure channel. Diffie-Hellman is suitable to use in information communication and less frequently use for information storage or archived over a long time period, (Wachirapong & Poom, 2018).

2.3 Blowfish Algorithm

Blowfish is a type of symmetric block cipher generated by B. Schneier in 1993. Blowfish is fast algorithm, license free, and unpatented (Yahia, Mohamad & Saleh, 2019). It uses a key length in the range of 32–448 and a sixty-four-bit block. The Blowfish algorithm makes use 16 round for the encipherment procedure Fig. 3. Blowfish ordinarily makes use of 4 S-boxes rather than of one S-box. It requires additional processing time because it relies on key length, however it provides strong safety (Faheem, et.al. 2017). It divides a message up into fixed length blocks during encryption and decryption. The block length for Blowfish is 64 bits; messages that aren't a multiple of eight bytes in size must be padded. Blowfish consists of two parts: key-expansion and data encryption. During the key expansion stage, the in-putted key is converted into several sub key arrays total 4168 bytes. There is the P array, which is eighteen 32-bit boxes, and the S-boxes, which are four 32-bit arrays with 256 entries each. After the string initialization, the first 32 bits of the key are XORed with P1 (the first 32-bit box in the P-array). The second 32 bits of the key are XORed with P2, and so on, until all 448, or fewer, key bits have been XORed Cycle through the key bits by returning to the beginning of the key, until the entire P-array has been XORed with the key. Encrypt the all zero string using the Blowfish algorithm, using the modified P-array above, to get a 64-bit block. Replace P1 with the first 32 bits of output, and P2 with the second 32 bits of output (from the 64-bit block). Use the 64-bit output as input back into the Blowfish cipher, to get a new 64-bit block. Replace the next values in the P-array with the block. Repeat for all the values in the P-array and all the S boxes in order. Encrypt the all zero string using the Blowfish algorithm, using the modified P-array above, to get a 64-bit block. Replace P1 with the first 32 bits of output, and P2 with the second 32 bits of output (from the 64-bit block). Use the 64-bit output as input back into the Blowfish cipher, to get a new 64-bit block. Replace the next values in the P-array with the block. Repeat for all the values in the P-array and all the S boxes in order.

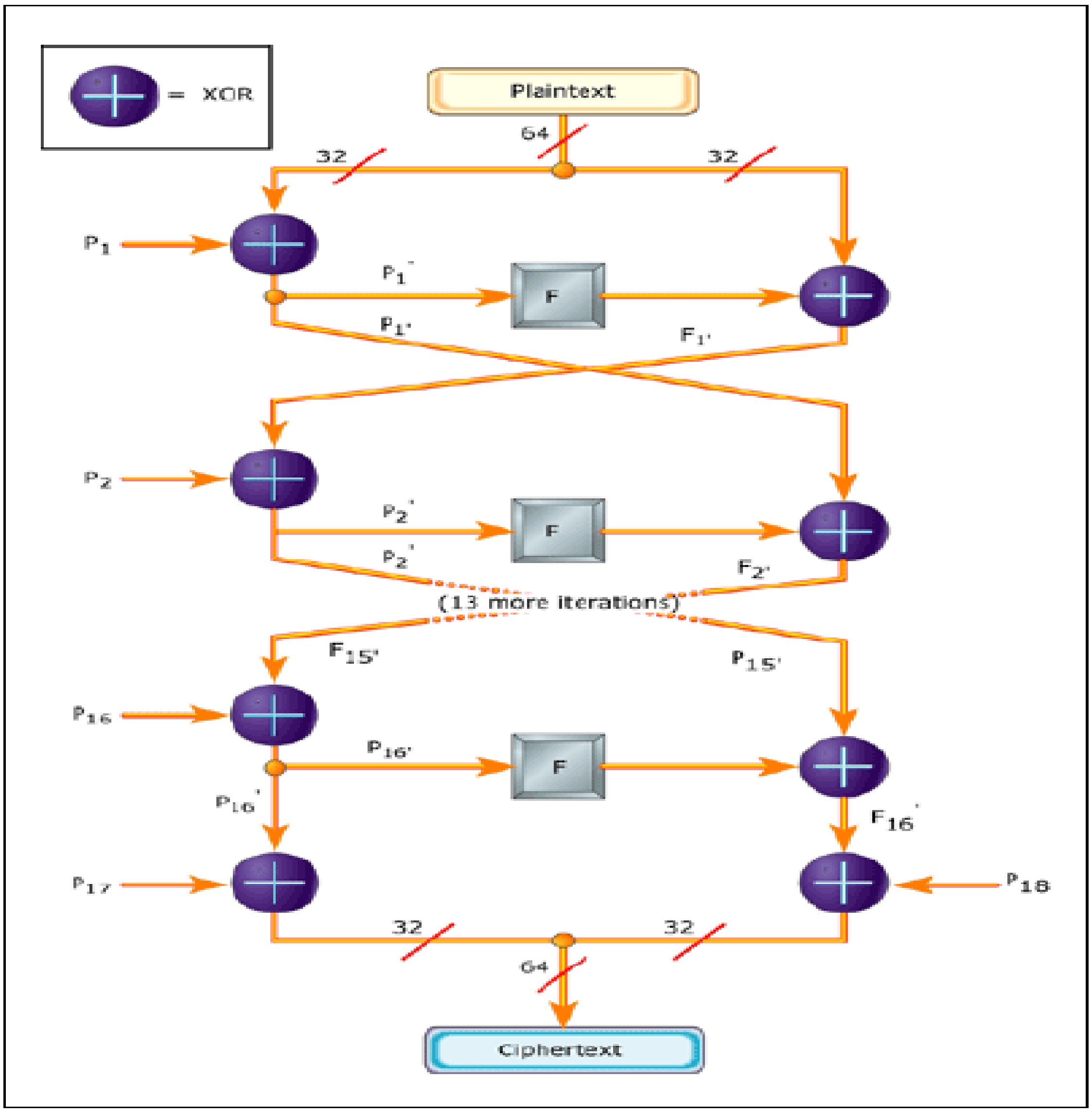


Figure 2.4: Blowfish Algorithm (Valmik & Kshirsagar, 2014).

2.4 Residue Number System

Residue Number System (RNS) is a non-weighted number system that provides carry-free, parallel, high speed, protected and fault tolerant arithmetic operations (Barati, Movaghar, Modiri & Sabaei, 2012; Barati, Movaghar, Sabaei, 2014). This number system has received extraordinary attention in the past decades. The interest in RNS arithmetic has begun since 1950's (Mohan, 2002; Omondi, Premkumar, 2007). The earliest hardware based on the RNS was built in 1967. Similarly, work in this field continued and many improvements in all areas of the RNS have arisen, in order to enhance its features, resolve its related problems and find suitable applications that benefit from RNS’s features. Most of the early designs of RNS were based on read-only memories (ROM). However, the great advance in VLSI (Very Large Scale Integration) technology paved the way for new approaches in designing RNS systems. New trends to design non-ROM based RNSs have appeared. RNS also has parallel operations that reduce power consumption and delay simultaneously (Gbolagade, Cotofana, 2009; Barati, Movaghar, Sabaei, 2014). In addition, in RNS, residues of the original number with respect to moduli set are represented instead of the original number itself. Thus, the number will be split into some smaller numbers which are autonomous and operations can be performed on them separately and concurrently which makes the computations simpler and much faster. Also, RNSs offer great potential for high-speed computer arithmetic due to their inherent features, such as parallelism, modularity, fault tolerance, and carry-free operations. It also supports carry-limited, high-speed arithmetic and error detection as well as error correction applications (Bankas, Gbolagade, 2014). Because of all of these unique features, its usage in variety of fields is increasing speedily. Some applications of RNS are Digital Signal Processing (DSP), Digital Image Processing, Digital Filters, Fast Fourier Transform (FFT) computations as well as Digital Communications. RNS is very helpful in error detection and correction because an error in one number could not affect any other digits (Khademolhosseini, Hosseinzadeh, 2011; Bankas & Gbolagade, 2014). In the RNS, a set of moduli which are independent of each other is given. An integer is represented by the residue of each modulus and the arithmetic operations are based on the residues individually. The arithmetic operations based on RNS can be performed on various moduli autonomously to avoid the carry in addition, subtraction and multiplication, which is usually time consuming.

Additionally, RNS is a non-weighted number system, unlike weighted number system such as binary or decimal number systems. Residue arithmetic operations like addition, subtraction, and multiplication are inherently carry-free, that is, each digit of the result is a function of only one digit from each operand, thereby independent of all other digits. RNS represents a large integer using a set of smaller integers, so that computation may be performed more efficiently. It relies on the CRT of modular arithmetic for its operation, a mathematical idea from Sun Tsu Suan-Ching (Master Sun's Arithmetic Manual) in the 4th century AD. RNS is described by a set of N integer constants, $\{m1, m2, m3, ..., mN\}$, referred to as the moduli. Let M be the least common multiple of all the mi. Any arbitrary integer X smaller than M can be represented in the defined residue numeral system as a set of N smaller integers $\{x1, x2, x3, ... , xN\}$ with $xi =$ *X modulo m* representing the residue class of X to that modulus.

However, for representational efficiency, the moduli should be pairwise coprime; that is, no modulus should have a common factor with any other. M is then the product of all the mi. For illustration purpose, RNS(4|2) has non-coprime moduli, with a LCM of 4, and a product of 8, resulting in the same representation for different values smaller than the product.

$$(3)\text{ decimal} = (3|1)\text{RNS}(4|2)$$

$$(7)\text{ decimal} = (3|1)\text{RNS}(4|2)$$

Equally, in RNS representation, the moduli are required to have the ability of efficient representation and balance. The most significant issues that must be taking into account when designing an RNS system are:

i. a proper moduli set selection,

ii. forward conversion,

iii. residue arithmetic units and

iv. reverse conversion.

Selection of moduli and the number of moduli determine both the dynamic range and the complexity of the resulting hardware. However, there are certain points that must be considered for the selection of moduli. These points are:

i. moduli must be mutually prime,

ii. the magnitude of the largest modulus dictates the speed of arithmetic operations, and

iii. all other moduli should be made comparable in size to the largest one.

2.4.1 Forward Conversion

Forward conversion is done by a forward converter which decomposes a weighted binary number into a residue represented number with regards to a moduli set. It is the conversion from a conventional representation to a residue representation by dividing the number X by each of the given moduli and then collecting their remainder (Gbolagade & Cotofana, 2009). Taking the example, moduli set {3, 4, 5}, 4 is depicted in RNS as: Therefore, the depiction of 4 in RNS as:

$$x_i = |X|_{m_i} \qquad (2.1)$$

$$x_1 = |X|_{m_1} = |4|_3 = 1;\ x_2 = |X|_{m_2} = |4|_4 = 0;\ x_3 = |X|_{m_3} = |4|_5 = 4$$

therefore, the RNS representation of 4 is $(1, 0, 4)_{RNS\ (3|4|5)}$

2.4.2 Reverse Conversion

The conversion from residue to a conventional representation (either binary or decimal representation) is known as reverse conversion. The two most widely used techniques of reverse conversion are the Mixed Radix Conversion (MRC) and Chinese Remainder Theorem (CRT) and Gbolagade, 2009; Gbolagade 2009).

2.4.2.1 Conversion using Chinese Remainder Theorem (CRT)

CRT is a hypothesis of numeric theory, which expresses that in case one knows the remnants of the division of an entire number n by a couple of numbers, one can choose curiously whatever is left of the division of n by the consequence of these numbers, under the condition that the divisors are pairwise coprime (Gbolagade & Cotofana, 2009; Arutselvan & Maheswari, 2013; Gbolagade, 2013). This theorem is used in converting RNS to binary/decimal. Given the residue representation{m_1, m_2, …, m_k} with gcd(m_i,m_j) = 1 for i≠j and a dynamic range M = $\prod_{i=1}^{n} mi$, the residue number {x_1, x_2, …, x_k} can be converted into the decimal number X if the moduli set are co-prime (Salifu, Gbolagade, 2016).

$$X = \left| \sum_{i=1}^{k} m_i \left| m_i^{-1} x_i \right|_{mi} \right|_m \tag{2.2}$$

$M = \prod_{i=1}^{n} m_i$ and $M_i = \frac{M}{mi}$ and M_i^{-1} is the multiplicative inverse of M_i with respect to m_i.

Example:

Finding the decimal equivalent of the RNS number (0, 0, 4) with respect to the moduli set {3,4,5}.

Solution:

$$M = \prod_{i=1}^{N} mi = 3 \text{ x } 4 \text{ x } 5 = 60$$

$M_1 = M/m_1 = 60/3 = 20$ then $= M_1^{-1} = 2$

$M_2 = M/m_2 = 60/4 = 15$ then $= M_2^{-1} = 3$

$M_3 = M/m_3 = 60/3 = 12$ then $= M_3^{-1} = 3$

$$X = |0 \text{ x } 20 \text{ x } 2 + 0 \text{ x } 15 \text{ x } 3 + 4 \text{ x } 12 \text{ x } 3|_{60}$$

$$= |0 + 0 + 144|_{60}$$

$$= 24$$

2.5 Review of Related Works

Patro K.A.K., et.al. (2020), proposed a Text-to-Image Encryption and Decryption Using Piece Wise Linear Chaotic Maps. The merit of this method is that it is entirely secure and hard to break. Also, PWLCM Maps is the simplest among chaotic maps. The main disadvantage of conservative flows is that they are not good models of most processes in nature. Mechanical systems have friction, and electrical systems have resistance, etc.

Albahrani E.A., Maryoosh A.A. & Lafta, S.H. (2020), proposed a block image encryption based on modified playfair and chaotic system. This method proposed an adaptive playfair cypher algorithm using Radix 64 conversion. The proposed algorithm used a matrix of 8 × 8 for encryption and decryption. This matrix is padding with the given keyword called "HACKER". After that the rest of the felids are padding with the remaining uppercase alphabets, lowercase alphabets, numbers (0-9) and the only two special characters (+ and /). In this system, the matrix is static and based on a known keyword, and only the lower, upper alphabet and numbers can be handled. Matrix conversions are complex and computationally expensive to execute.

Biswas C., Gupta U.D. & Haque Md.M. (2019), proposed a hybrid cryptography scheme using AES and RSA. In this hybrid cryptography, the symmetric key used for message encryption is also encrypted, which ensures a better security. Digital signature is created by encrypting the hash value of message. At the receiving side this digital signature is used for integrity checking. Then the encrypted message, encrypted symmetric key and encrypted digest are combined together to form a complete message. This complete message again has been secured using the steganography method, LSB.

Yahia A., Mohamad A.M. & Saleh A. (2019), carried out a research on various cryptography techniques. The research analysed the want to improve a combine encipherment algorithm that mixes various encipherment algorithms on the basis of all appropriate factors that are used to increase the overall safety and security of encipherment methods.

Hazra T.K., et.al., (2017), proposed A hybrid cryptosystem of image and text files using blowfish and Diffie-Hellman techniques, this approach used blowfish to encrypt and decrypt while transmitting over an unsecured medium using Diffie-Hellman algorithm, absence of time stamp here made it prone to permutation and computation attacks; also decryption time here is slow.

Algorithm 2.1: Blowfish and Diffie Hellman

Step 1: First, both users agree upon a prime number **p** and another number **g** that has no factor in common. Note that **g** is also known as the generator and **p** is known as prime modulus.

Both are public keys. So, a third person sitting in between and listening to this communication also gets to know p and g.

Step 2: Now user **A** takes a private key **x**. So, user **A** calculates a key **R1**= (g^x) mod p.

Step 3: User **A** generates a secret key and cipher and then encrypts the file using the secret key and cipher generated by blowfish algorithm.

[

```
keyGenerator = KeyGenerator.getInstance("Blowfish");
secretKey = keyGenerator.generateKey(); cipher = Cipher.getInstance("Blowfish");
```

Step 4: Now User **B** wants to decrypt the file. User **B** takes a private key **y**.

So, user **B** calculates a key **R2**= (g^y) mod p.

]

Step 5: Now user **B** wants permission for decryption from user **A**. So, both of them share **R1** and **R2** with each other through the insecure channel of communication. So, these values become public.

Step 6: User **A** calculates final key k1= (R2^x) mod p. User **B** calculates k2=(R1^y) mod p.

Step 7: If the values of k1 and k2 match then only user **B** gets the permission of decryption. So, user **A** sends the secret key for Blowfish to user **B**. Then user **A** sends the encrypted file to user **B**.

Step 8: Now user **B** decrypts the file using Blowfish algorithm.

Saputra I., Hasibuan N.A., & Rahim R. (2017), suggested a Vigenere cypher algorithm with grayscale image key generator for a secure text file. This approach works by using a key generated from the 8-bit grayscale image that has been converted into a character pixel value using ASCII table it will produce a series of the main characters were very random and the length determined by the size or dimensions of the image are used as the key. Because of the imagery used at the time of encryption, there are differences with the image pixel value used at the time decryption of the cipher-text will not go back into the original plaintext. Pixel conversion is slow.

CHAPTER THREE

METHODOLOGY

3.1 Existing System

Today, important information sent over internet is not limited to text data, but also contain multimedia information like image, video, and audio data etc. Some conventional cryptographic systems directly convert image data into encrypted form of data. But this is not advisable due to very huge data size of image data and a real-time constraint of such type of data. Conventional cryptosystem requires huge amount of time to directly convert thousands of image pixel values into encrypted form and in image data, a decrypted image is normally acceptable even if it contain small level of distortion in image data.

From the review of literature, it is clear that noticeable drawbacks are in the previous schemes proposed. The drawback peculiar to most is the simplicity attached to the algorithms, making them easy to crack and at the same time making the encrypted image easily detected. It is to be noted that the main aim of any encrypting algorithms is to prevent confidential data from unauthorized access. But when such purpose is not achieved effectively, it is essential to provide a means of ensuring that the main goal is acquired through provision of a better scheme to cater for the drawback.

3.2 Proposed Methodology

In this Chapter, an improved image cryptosystem using blowfish algorithm with diffie-hellman key exchange and residue number system is proposed. It is basically the combination of hybrid technique to encrypt the image. Plain and secret images are first encrypted using residue number system with respect to the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$ through the computation of forward conversion to form cipher residues. Then Diffie-hellman algorithm is used to generate the encryption and decryption key after both the sender and receiver must have agreed upon a prime number **p** and another number **g** that has no factor in common. Where g is also known as the generator and p is known as prime modulus.

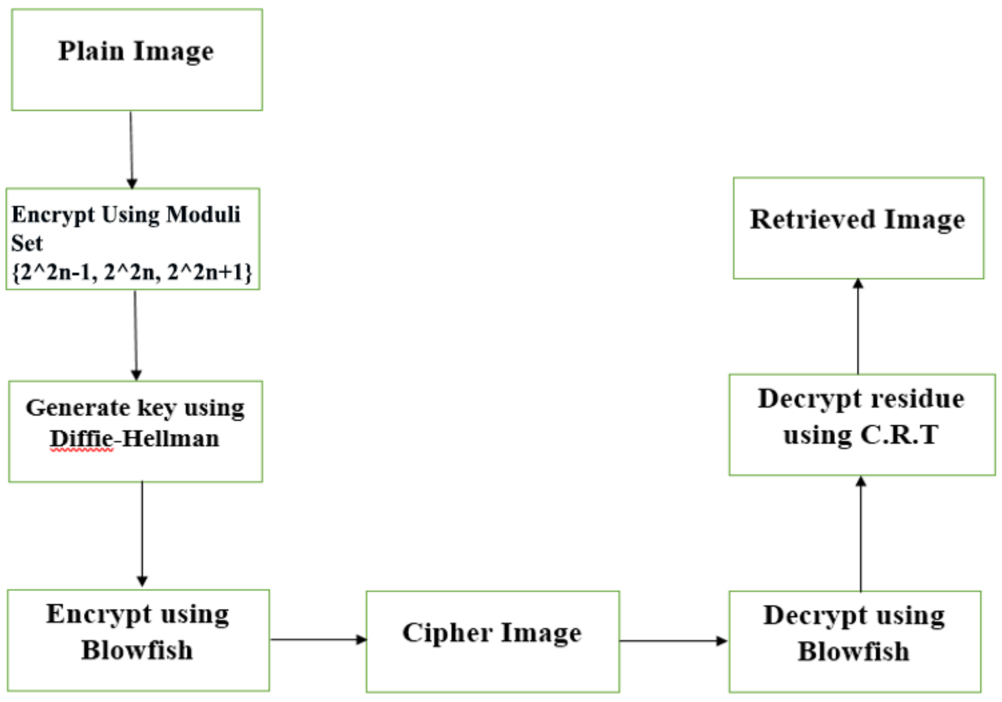

Plain Image
Encrypt Using Moduli Set {2^2n-1, 2^2n, 2^2n+1}
Generate key using Diffie-Hellman
Encrypt using Blowfish
Cipher Image
Decrypt using Blowfish
Decrypt residue using C.R.T
Retrieved Image

Figure 3.1: System Architecture

Thereafter, the concept of blowfish encryption using the key generated by Diffie-hellman algorithm is adopted on the cipher residues to generate a cipher image that will be sent to the receiver at the other end.

The receiver decrypts the image by firstly decrypting blowfish algorithm using the key generated by Diffie-hellman to get the cipher residue. The cipher residue is then decrypted using Chinese remainder theorem (reverse conversion) which then return the cipher residue back to its original pixel value, which is then displays the original image.

This will be achieved using the developed system shown in figure 3.1.

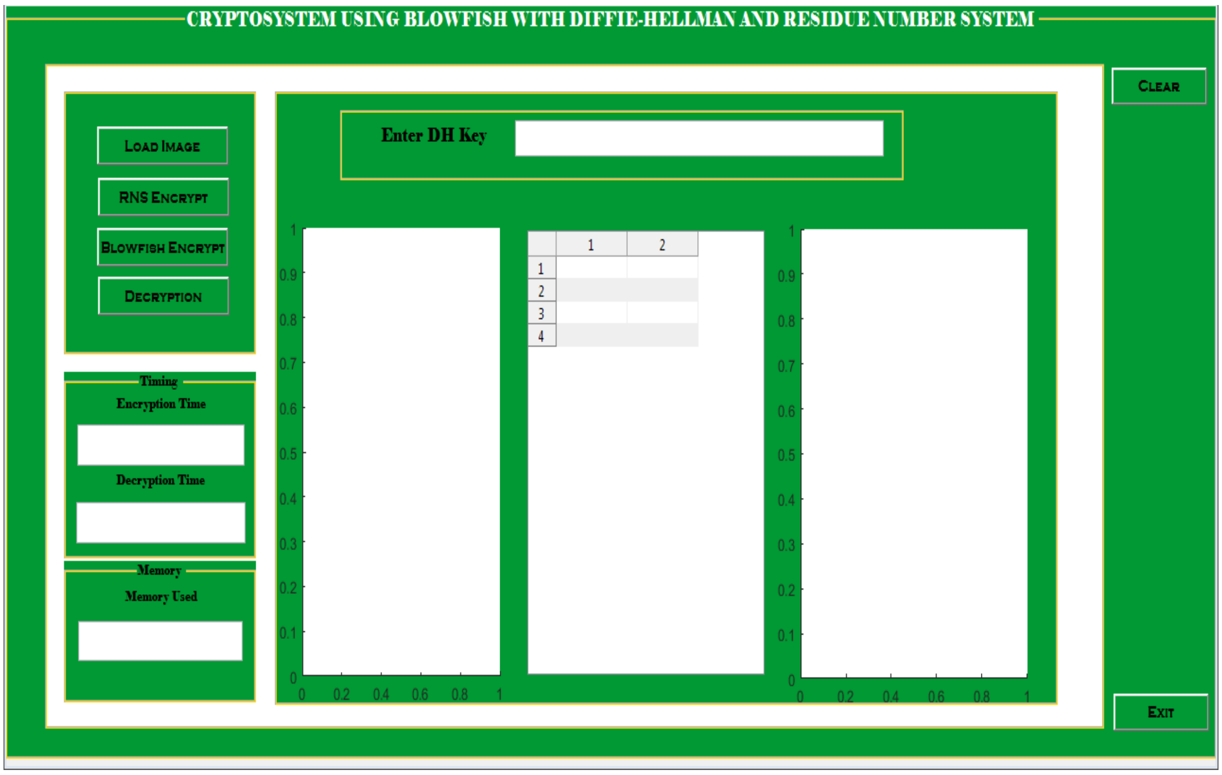


Figure 3.2: System Interface

The system accepts image files which can be fetched from the local disk. A dialog box will come up once the LOAD IMAGE button has been clicked, which enables the user to select the image file to be read by the system. Once the image is successfully read, the content is displayed.

Thereafter, the user will be required to click on the RNS ENCRYPT button, which will encrypt the image using the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$ into a cipher residue. Once the image is converted into residues, the user will be residue to enter a numeric value to generate a key using Diffie-hellman algorithm. After the value is entered, the final encryption process will commence after the BLOWFISH ENCRYPT button has been clicked, which will encrypt the cipher residue using blowfish algorithm.

To decrypt the cipher image, the user only needs to click the DECRYPTION button, which makes the system to reverse the process to produce the original image.

Algorithm 3.2.1: Blowfish with Diffie Hellman and RNS

Step 1: First, both users agree upon a prime number **p** and another number **g** that has no factor in common. Note that **g** is also known as the generator and **p** is known as prime modulus. Both are public keys. So, a third person sitting in between and listening to this communication also gets to know p and g.

Step 2: Forward conversion is implemented on an image using the moduli set {2^{2n}-1, 2^{2n}, 2^{2n}+1}

Step 3: Now user **A** takes a private key **x**. So, user **A** calculates a key **R1**= (g^x) mod p.

Step 4: User **A** generates a secret key and cipher and then encrypts the residue using the secret key and cipher generated by blowfish algorithm.

[

keyGenerator = KeyGenerator.getInstance("Blowfish");

secretKey = keyGenerator.generateKey(); cipher = Cipher.getInstance("Blowfish");

Step 5: Now User **B** wants to decrypt the file. User **B** takes a private key **y**.

So, user **B** calculates a key **R2**= (g^y) mod p.

]

Step 6: Now user **B** wants permission for decryption from user **A**. So, both of them share **R1** and **R2** with each other through the insecure channel of communication. So, these values become public.

Step 7: User **A** calculates final key k1= (R2^x) mod p. User **B** calculates k2=(R1^y) mod p.

Step 8: If the values of k1 and k2 match then only user **B** gets the permission of decryption. So, user **A** sends the secret key for Blowfish to user **B**. Then user **A** sends the encrypted residue to user **B**.

Step 9: Now user **B** first decrypts the file using Blowfish algorithm and reverse conversion on the residue is carried out before getting the original image.

3.2.1 Transposition

The first stage of the transposition involves performing the conversion of the selected image into a grayscale image if it is an RGB image. The forward conversion takes place on the grayscale image using the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$. A value n is assigned to the moduli set which serves as a secret key. Using the moduli set it is expected that for each bit of the image, three residue numbers will be computed. From the computed residue numbers, a matrix is formed and the matrix is encrypted using blowfish algorithm that uses Diffie-hellman algorithm for the key exchange. The encrypted output will form a cipher image which will appeared scrambled

The transposition is achieved using;

$$x = m_i[x/m_i] + |x|_{mi} \tag{3.1}$$

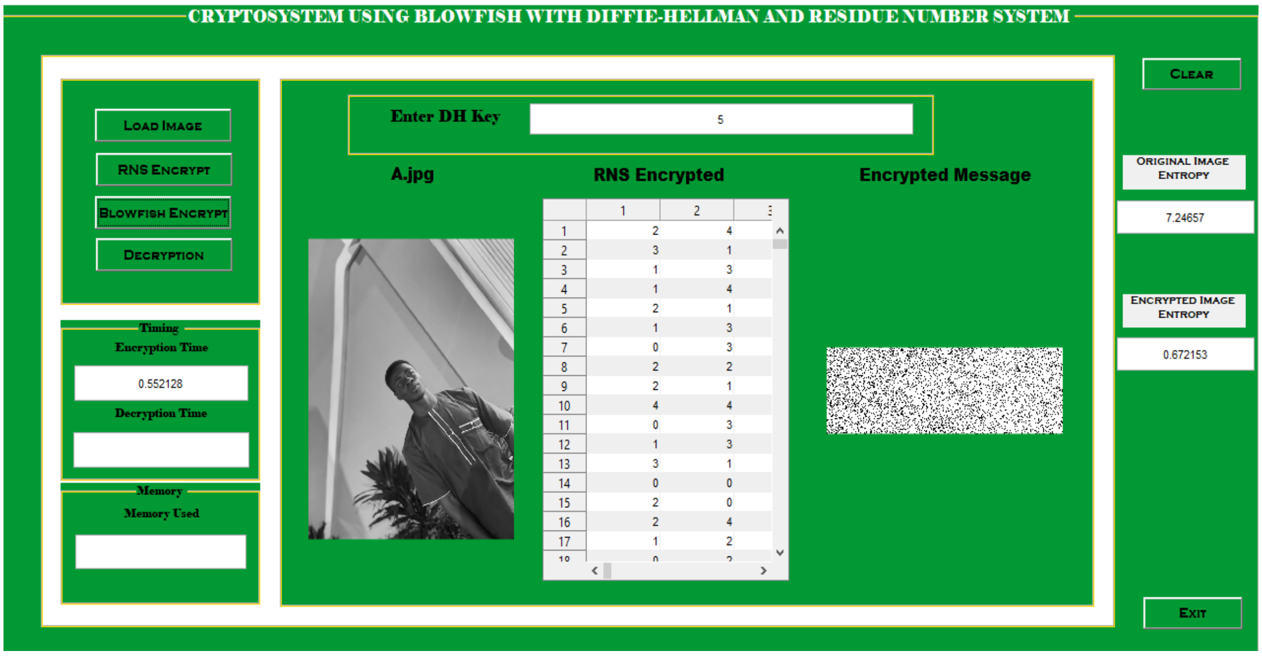
where *m* is the moduli and *x* are the numbers to be transposed.

3.2.2 Encryption process

The encryption stage which is comprised of both the transposition and the encryption will be achieved using the procedure below.

1. Read the image and convert it into grayscale if it is in RGB
2. Transpose each of the image bit using forward conversion based on the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$, to generate residue of each of the bits.
3. Generate an exchange key using Diffie-hellman algorithm where both the sender and receiver must have agreed upon a prime number **p** and another number **g** that has no factor in common.
4. Encrypt the cipher residue using blowfish algorithm using the generate key for the exchange.

Figure 3.3: Testing Image



Enter DH Key: 5

Encryption Time: 0.552128

Decryption Time:

Memory Used:

Original Image Entropy: 7.24657

Encrypted Image Entropy: 0.672153

	1	2
1	2	4
2	3	1
3	1	3
4	1	4
5	2	1
6	1	3
7	0	3
8	2	2
9	2	1
10	4	4
11	0	3
12	1	3
13	3	1
14	0	0
15	2	0
16	2	4
17	1	2
18	0	2

Figure 3.4: Encryption Process

3.2.3 Decryption process

For the decryption process, the key generated using Diffie-hellman will be required to decrypt the cipher image using blowfish algorithm. Once the cipher image is decrypted to get the cipher residue, the residues generated will undergo backward conversion using Chinese Remainder Theorem (CRT).

This theorem is used in converting residue number system to decimal. Given the residue representation $\{r_1, r_2, \ldots, r_n\}$ of c, the CRT makes it possible to determine $|c|_m$, provided the gcd of any pair of moduli is 1. Such moduli are called pairwise relatively prime.

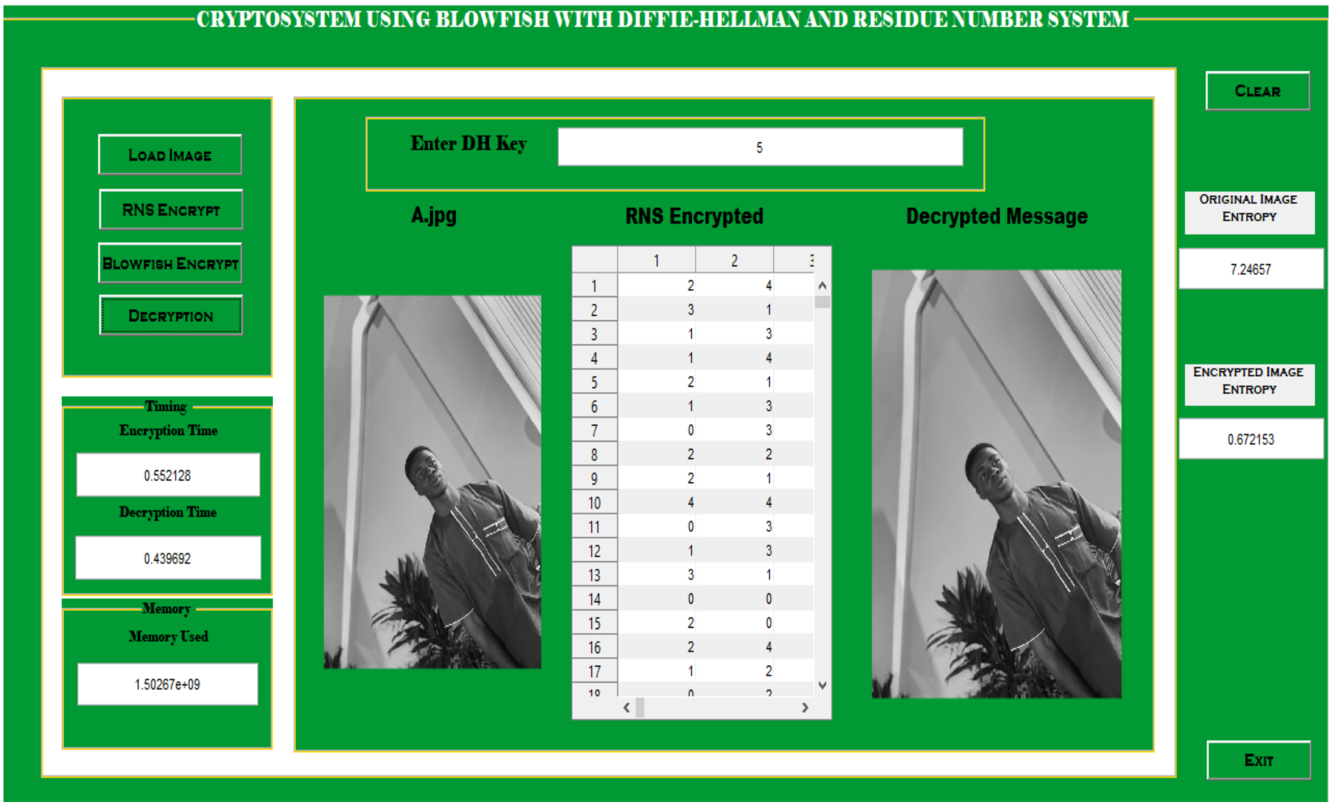
The CRT is given by

$$X = \left| \sum_{i=1}^{k} m_i \left| m_i^{-1} x_i \right|_{mi} \right|_m \tag{3.2}$$

$M = \prod_{i=1}^{n} m_i$ and $M_i = \frac{M}{mi}$ and M_i^{-1} is the multiplicative inverse of M_i with respect to m_i.

The procedure for the decryption is as follows;

1. Enter the decryption key generated using Diffie-hellman algorithm.
2. Decrypt the cipher image using blowfish algorithm.
3. Using Chinese remainder theorem, perform a reverse conversion on the residues based on the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$ to generate image bits.
4. Generate the original image from the image bits.



Enter DH Key: 5

Encryption Time: 0.552128

Decryption Time: 0.439692

Memory Used: 1.50267e+09

ORIGINAL IMAGE ENTROPY: 7.24657

ENCRYPTED IMAGE ENTROPY: 0.672153

	1	2
1	2	4
2	3	1
3	1	3
4	1	4
5	2	1
6	1	3
7	0	3
8	2	2
9	2	1
10	4	4
11	0	3
12	1	3
13	3	1
14	0	0
15	2	0
16	2	4
17	1	2
18	0	2

Figure 3.5: Decryption Process

3.3 System Requirement

For better effectiveness and performance, the developed system requires a computer system with minimum intel core 2 processor running at 2.0 GHZ, 4 GB ram and 120 GB free hard disk space. Also, it requires operating system of minimum Window 7 with MATLAB installed on it.

3.4 Choice of Programming Language

The entire system will be developed using MATLAB. MATLAB is a high-level programming language and interactive environment for numerical computation, visualization, and programming. Using MATLAB, you can analyze data, develop algorithms, and create models and standalone applications. The language, tools, and in-built math functions enable you to explore multiple approaches and reach a solution faster than with spreadsheets or traditional programming languages, such as C/C++ or Java. MATLAB programming language is also object based programming language and it is integrated with MATLAB Compiler Runtime for the purpose of creating applications that will work outside of MATLAB integrated environment, thus making room for applications to run without installing MATLAB.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Analysis of the proposed system

The performance of the implemented system was evaluated based on 3 performance metrics which are: encryption time, decryption time and entropy. The results obtained where then recorded, visualized, interpreted and compared with the existing system.

4.2 Entropy Analysis

Entropy analysis is carried out to check the randomness and security of cipher or encrypted image. entropy is one of the statistical parameter that is used to measure the uncertainly and randomness of a large image data. According to theory of Shannon for cryptography, image entropy is defined as the number of bits that is necessary to encode every pixel of the image data. The entropy shows the random pattern of the image data and texture of pixels in the cipher image. The entropy values of original images and the encrypted images given table 4.1.

Table 4.1: Entropy Analysis

File Name	**File Size (KB)**	**Hazra, et.al. (2017) (e)**	**PROPOSED SCHEME (e)**
A	**40.1**	**6.361**	**0.672**
B	**52.6**	**7.351**	**0.651**
C	**110**	**7.686**	**0.658**
D	**37.7**	**6.926**	**0.664**
E	**17.6**	**5.444**	**0.552**

4.3 Encryption and Decryption Time Analysis

The breaking of large pixel integer into smaller pixel integer sets using Residue Number System makes the computation much easier and faster. Table 4.2 and Table 4.3 shows the time it takes to encrypt and decrypt five different images of different sizes on both the proposed scheme and previous scheme.

Table 4.2: Encryption Time

File Name	File Size (KB)	Hazra, et.al. (2017) (ms)	PROPOSED SCHEME (ms)
A	40.1	7.261	0.586
B	52.6	7.980	0.565
C	110	5.871	0.566
D	37.7	4.595	0.489
E	17.6	4.896	0.517

Table 4.3: Decryption Time

File Name	File Size (KB)	Hazra, et.al. (2017) (ms)	PROPOSED SCHEME (ms)
A	40.1	0.350	0.302
B	52.6	0.322	0.303
C	110	0.314	0.300
D	37.7	0.323	0.300
E	17.6	0.295	0.294

4.4 Comparison with existing approaches

After the evaluation of the proposed scheme, as shown in Table 4.1, Table 4.2 and Table 4.3, the scheme was compared with the existing scheme by Hazra T.K. et.al. (2017) titled "A Hybrid Cryptosystem of Image and Text Files Using Blowfish and Diffie-Hellman Techniques"

In term of encryption and decryption time, the proposed scheme is faster than the scheme proposed by Hazra T.K. et.al. (2017) by breaking of large pixel integer into smaller pixel integer sets using Residue Number System which makes the computation much easier and faster.

Y axis – Time in seconds, X axis – Existing system vs Enhanced system

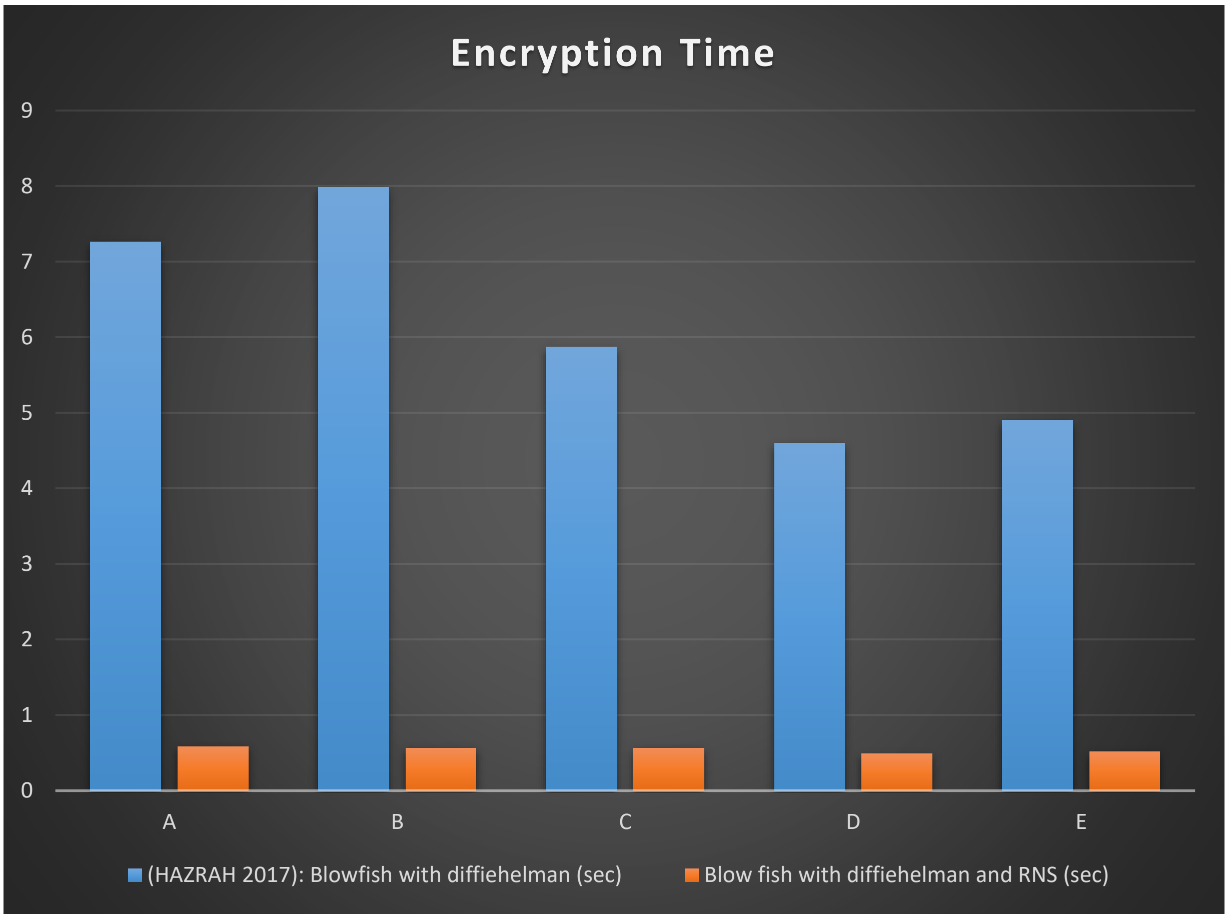


Figure 4.1: Comparison in term of Encryption Time

Y axis – Time in seconds, X axis – Existing system vs Enhanced system

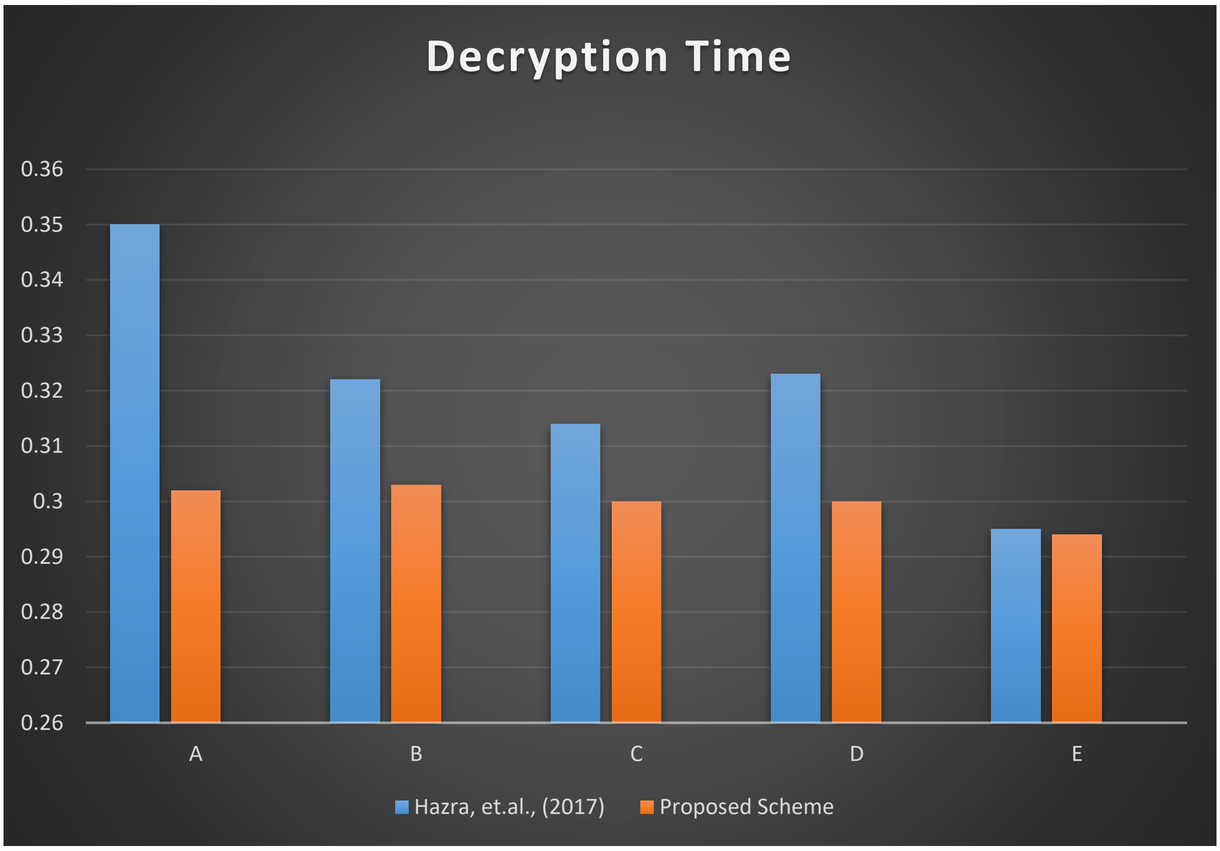


Figure 4.2: Comparison in term of Decryption Time

In term of security, the scheme proposed by Hazra T.K. et.al. (2017) was able to improve the original blowfish algorithm which can't provide authentication and non-repudiation, by using Diffie-hellman key exchange to take care of the authentication and non-repudiation. But the scheme was still prone to middle man attack and replay attack, So the proposed scheme neutralized these problems by encrypting the image before it was re-encrypted using blowfish algorithm and using Diffie-hellman algorithm for the key exchange making both the middle man attack and the replay attack to be ineffective against the scheme.

Also in term of cracking probability, the proposed scheme cracking probability is lower compared to Hazra T.K. et.al. (2017) due to it being a double level encryption scheme by encrypting the image using both residue number system and blowfish algorithm while Hazra T.K. et.al. (2017) encrypt the image using blowfish algorithm alone.

Y axis – Time in seconds, X axis – Existing system vs Enhanced system

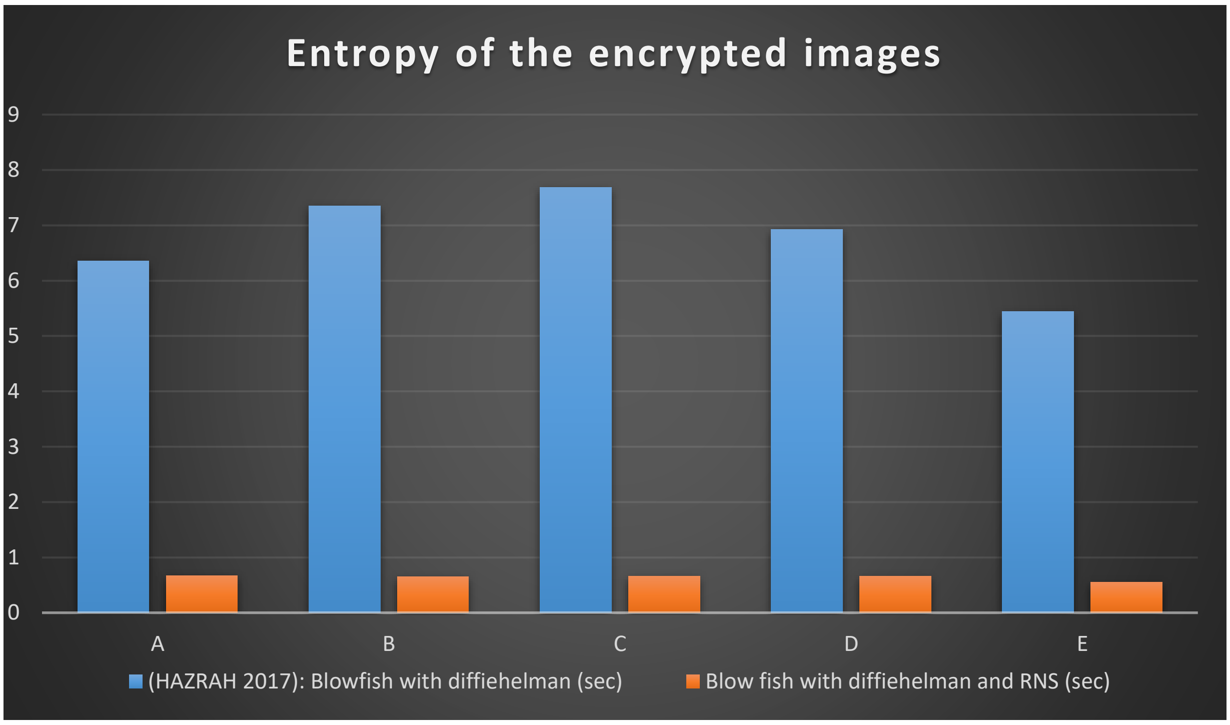


Figure 4.3: Comparison in term of Encrypted Image Entropy

Table 4.4: Comparison with existing method

PARAMETERS	**Hazra T.K. et.al. (2017)**	**PROPOSED SCHEME**
ENCRYPTION AND DECRYPTION TIME	HIGH	LOW
SECURITY	LOW	HIGH
CRACKING PROBABILITY	HIGH	LOW
ENCRYPTION STAGE	1	2

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATIONS

5.1 Summary

The research proposed the use of Blowfish algorithm and Residue number system to encrypt and decrypt an image where Diffie-hellman algorithm is used for the key exchange. The image is first converted into grayscale if it is a RGB image before it is encrypted using Residue number system by carrying out forward conversion on the image with respect to the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$. The residue is then encryption using blowfish algorithm where Diffie-hellman algorithm generate the encryption and decryption key. Both users must have must have agreed upon a prime number **p** and another number **g** that has no factor in common for Diffie-hellman algorithm to generate the exchange key, and but users must have earlier known the moduli set used and the value of n in the moduli set to be able to decrypt the image.

5.2 Conclusion

The proposed scheme has provided a better means of sending images securely over an unsecure network while the image maintains a high level of privacy, proving to outperform its predecessor in term of security of the image, computational time was also less compare to its predecessor. The cipher image produced by the proposed scheme is very complex and difficult to crack due to having so many probabilities to determine the moduli set used, determine the value of n for the moduli set, the key generated by Diffie-hellman algorithm.

5.3 Major contribution to knowledge

The major contributions of this dissertation are listed below:

i. Provides a better security with the implementation of a double layer encryption thereby reducing the cracking probability.

ii. Improved computational time of the system in-terms of encryption and decryption time

5.4 Future work

For the purpose of further improvement and providing a more efficient way of securing image using this approach, the following are expected future works;

i. Implementing this scheme with an RGB image

ii. Use of Mixed Radix conversion with Blowfish algorithm to evaluate the efficiency of this scheme in term of computational speed in the reverse process.

iii. Combination of Residue number system with Blowfish algorithm using other key exchange algorithm.

REFERENCES

Agrawal M. & Mishra P. (2012). A modified approach for symmetric key cryptography based on blowfish algorithm. *International Journal of Engineering and Advanced Technology*. volume 1 no 6 pp 79-83.

Ahmed M., et.al. (2012). Diffie-Hellman and its application in security protocols. *International Journal of Engineering and Science Innovation Technology*, 1(2), pp. 69-73.

Albahrani E.A., Maryoosh A.A., & Lafta S.H. (2020). Block image encryption based on modified Playfair and chaotic system. *Journal of Information Security and Applications*, 51, 102445.

Aremu I.A. & Gbolagade K.A. (2017). An overview of Residue Number System. *International Journal of Advanced Research in Computer Engineering & Technology*, 6(10), 1618 1623.

Babaei M. (2013). A novel text and image encryption method based on chaos theory and DNA computing. *Natural Computing*, 12(1), 101-107.

Bankas E. K., & Gbolagade, K. A. (2014). A new efficient RNS reverse converter for the 4 Moduli Set $\{2n, 2n+ 1, 2n - 1, 22n+1 – 1\}$. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 8(2),328 - 332.

Barati, A., Movaghar, A., & Sabaei, M. (2014). Energy efficient and high-speed error control scheme for real time wireless sensor networks. *International Journal of Distributed Sensor Networks*, 2014, 1-9.

Barati, A., Movaghar, A., Modiri, S., & Sabaei, M. (2012). A reliable and energy-efficient scheme for real time wireless sensor networks applications. *Journal of Basic and Applied Scientific Research*, 2(10),10150-10157.

Biswas C., Gupta U.D. & Haque Md.M. (2019). An Efficient Algorithm for Confidentiality, Integrity and Authentication Using Hybrid Cryptography and Steganography.

International Conference on Electrical, Computer and Communication Engineering (ECCE).

Das P.K., Kumar P. & Sreenivasulu M. (2014). Image Cryptography: A Survey towards its Growth. *Advance in Electronic and Electric Engineering*. ISSN 2231-1297, Volume 4, Number 2 (2014), pp. 179-184.

Devi A., Sharma A. & Rangra A. (2015). A review on DES, AES and Blowfish for image encryption and decryption. *International Journal of Computer Science. And Information Technology*. vol 6 no 3. pp 3034-6 ISSN:0975-9646.

Faheem M.M., Jamel S., Disina A.H., Pindar Z.A., Shakir N.S.A., & Deris M.M. (2017). A survey on the cryptographic encryption algorithms, *International Journal of Applied Engineering Research. Computer Science Applications*, 8(11), pp. 333-344.

Gbolagade, K. A., & Cotofana, D. S. (2009). A reverse converter for the new 4-moduli set. *Proceedings of 2009 16th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 113-116.

Godfrey L.D., Bobby D.G., & Ruji P.M. (2019). A unique message encryption technique based on enhanced blowfish algorithm. *The International Conference on Information Technology and Digital Applications*. doi:10.1088/1757-899X/482/1/012001.

Basic Data Communication Process. (2020). Retrieved May 1, 2020, from https://www.oreilly.com/

Hazra T.K., Mahato A., Mandal A., & Chakraborty A.K. (2017). A hybrid cryptosystem of image and text files using blowfish and Diffie-Hellman techniques. *8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON).*

Hazra T.K. & Bhattacharyya S. (2016). Image encryption by blockwise pixel shuffling using Modified Fisher Yates shuffle and pseudorandom permutations. *IEEE 7th Annual*

Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 1-6. doi: 10.1109/IEMCON.2016.7746312

Karule K.P. & Nagrale N.V. (2016). Comparative analysis of encryption algorithms for various types of data files for data security. *International Journal of Scientific Engineering and Applied Science*, 2(2), pp. 495-498.

Khademolhosseini, H., & Hosseinzadeh, M. (2011). A robust redundant residue representation in residue number system with moduli set (rn-2, rn-1, rn). *International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering*, 5(8), 1170-1175.

Manku K., Saikumar V. & Vasanth K. (2015). Blowfish encryption algorithm for information security. *ARPN Journal of Engineering and Applied Sciences* 10.10 4717-4719.

Meera, M. & Malathi, P. (2015). An improved embedding scheme in compressed domain image steganography. *International Journal of Applied Engineering Research* 2015; 10 (55):1933-1937.

Omondi, A., & Premkumar, B. (2007). *Residue number system: Theory and implementation.* Imperial College Press, London.

Patro K.A.K., Soni S., Sharma V.K., & Acharya B. (2020). Text-to-Image Encryption and Decryption Using Piece Wise Linear Chaotic Maps. *In Advances in Data and Information Sciences* (pp. 23-33). Springer, Singapore.

Popoola D. (2019). Data Integrity Using Caesar Cipher and Residue Number System. Kwara State University, Malete.

Priyanka D. & Kulkarni R.N. (2019). Secret Image Sharing using 2-Pixel Visual Cryptography Encryption. *International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878*, Volume-8 Issue-4.

Saputra I., Hasibuan N.A., & Rahim R. (2017). Vigenere cypher algorithm with grayscale image key generator for secure text file. *International Journal of Engineering Research & Technology (IJERT)*, 6(1), 266-269.

Selvaraj D. (2017). Development of a secure communication system based on steganography for mobile devices. p 3.

Sharma M. & Sharma V. (2016). A hybrid cryptosystem approach for file security by using merging mechanism. *2nd International Conference on Applied and Theoretical Computing and Communication Technology. (ICATCCT) (IEEE)* 978-1-5090 2399-8 pp 713-7.

Singh S. & Kumar S. (2018). Analysis of various cryptographic algorithms, *Int. J. Advanced Research in Science, Engineering and Technology*, 5(3), pp. 5341-5348.

Siper A. (2005). The rise of steganography *Proceedings of Student/Faculty Research Day CSIS*, Pace University.

Srinivas B.L., Shanbhag A. & Souza A.S.D. (2014). A comparative performance analysis of DES and BLOWFISH symmetric algorithm. *International Journal of Innovative Research in Computer and Communication Engineering*, 2(5), pp. 77-88.

Valmik N. & Kshirsagar V.K. (2014). Blowfish Algorithm. *IOSR Journal of Computer Engineering (IOSR-JCE,)* e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, PP 80-83

Wachirapong J. & Poom K. (2018). The Generalized Diffie-Hellman Key Exchange Protocol on Groups. *Econometrics for Financial Applications, Studies in Computational Intelligence 760*, https://doi.org/10.1007/978-3-319-73150-6_8

Yahia A., Mohamad A.M. & Saleh A. (2019). Research on Various Cryptography Techniques. *International Journal of Recent Technology and Engineering (IJRTE)* ISSN: 2277

APPENDIX: SOURCE CODE

BFDHRNS.m:

```
nction varargout = BFDHRNS(varargin)
% BFDHRNS MATLAB code for BFDHRNS.fig
%      BFDHRNS, by itself, creates a new BFDHRNS or raises the existing
%      singleton*.
%
%      H = BFDHRNS returns the handle to a new BFDHRNS or the handle to
%      the existing singleton*.
%
%      BFDHRNS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in BFDHRNS.M with the given input arguments.
%
%      BFDHRNS('Property','Value',...) creates a new BFDHRNS or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before BFDHRNS_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to BFDHRNS_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help BFDHRNS

% Last Modified by GUIDE v2.5 09-Jul-2020 23:12:36

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @BFDHRNS_OpeningFcn, ...
                   'gui_OutputFcn',  @BFDHRNS_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before BFDHRNS is made visible.
function BFDHRNS_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to BFDHRNS (see VARARGIN)

% Choose default command line output for BFDHRNS
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes BFDHRNS wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = BFDHRNS_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Img X
user=getenv('username');
cd(strcat(['/Users/abdulhameed/Downloads/hameed/']));
[X,pathname] = uigetfile('*.jpg;*.tiff;*.bmp;*.ppm;*.pgm;*.png','pick a
jpge file');
    Img = imread((strcat(pathname,X)));
    if size(Img,3)>1
    Img=rgb2gray(Img);
end
    axes(handles.axes1)
    imshow(Img);
    [~,nam,ext] = fileparts(X);
    namy = [nam,ext];
    e = entropy(Img);
set(handles.text4,'string', namy);
set(handles.edit7,'string', e);



% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global dec1 t ha g enc Img xx ff t1
tic
 f= entropy(ff);
 g = get(handles.edit2,'string');
    if isempty(g)| isletter(g);
        errordlg('Please enter a Numeric key');
        set (handles.edit2,'BackgroundColor', 'r');
    return
    end
            set (handles.edit2,'BackgroundColor', 'w');
axes(handles.axes2)
    imshow(ff);
t3=toc;
t4= t3+t1;
set(handles.edit3,'string', t4);
set(handles.edit8,'string', f);
set(handles.text5,'string', 'Encrypted Message');
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ff t1  t ui enc Img xx m n Ind y Gb Gb1 sha256hasher X g
tic
if isempty(X);
        errordlg('Please load the Image to encrypted');
        set (handles.listbox2,'BackgroundColor', 'r');
    return
    end
[m,n]=size(Img);
Gb=uint8([30, 14, 36, 213, 65, 49, 153, 42, 140, 13, 254,  116,  179,
201,...
    185, 204, 15,  147,  210,  147,  229,  148,  117,  100,  128,   26, ...
    249, 48, 63, 65, 47, 130]);
Gb1=Gb;
sha256hasher = System.Security.Cryptography.SHA256Managed;

y=uint8(zeros(m,n));
k=1;




P = g;
XL=y;
XL = XL.*P;
XR= xx;



sh=rand(1,512*512);
[t, Ind]=sort(sh);
x1=Img(:);
x2=x1(Ind);
xx=reshape(x2,512,512);



ui = double(xx);
        P =(ui);
        f1 = mod(P,5);
    f2 = mod(P,6);
    f3 = mod(P,7);
    ff = [f1; f2; f3];
    ff = ff';
        set(handles.uitable1,'Data',ff);

      t1=toc;
```

```
set(handles.text8,'string', 'RNS Encrypted');


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ff  dec1 t  ui m n xx Ind y Gb Gb1 sha256hasher Img
if isempty(ui);
        errordlg('Please encrypt the message using RNS first');
        set (handles.uitable1,'BackgroundColor', 'r');
    return
    end
tic
 am=memory;
      % RNS Decryption
     modmul = 5*6*7;
    dmod1 = modmul/5;
    dmod2 = modmul/6;
    dmod3 = modmul/7;
  x=1;
  for i = 1:100
      dody1= (dmod1*x);
      remy1 = rem(dody1,5);
       if (remy1==1)
           break
       end
       x=x+1;
  end
  divx1=x;

  x=1;
  for i = 1:100
      dody2 = (dmod2*x);
      remy2 = rem(dody2,6);
       if (remy2==1)
           break
       end
       x=x+1;
  end
  divx2=x;

  x=1;
  for i = 1:100
      dody3 = (dmod3*x);
      remy3 = rem(dody3,7);
       if (remy3==1)
           break
       end
       x=x+1;
  end
  divx3=x;

  ff1=ff(:,1);
  ff2=ff(:,2);
  ff3=ff(:,3);
  nf1 = (ff1.*dmod1.*divx1);
  nf2 = (ff2.*dmod2.*divx2);
  nf3 = (ff3.*dmod3.*divx3);
  capff =[nf1,nf2,nf3];
  cat = sum(capff');
  revd = mod(cat,modmul);
   pp= char(revd);


Gb1(1)=Gb(1)+10;
y(1,1)=y(1,1)+7;
z=uint8(zeros(m,n));
k=1;


z1=Img(:);

Z=reshape(Img,m,n);
isequal(Img,Z)



axes(handles.axes2)
    imshow(Z);
 tm=toc;
    memor3=am.MemUsedMATLAB;

set(handles.edit4,'string',tm);
set(handles.edit5,'string',memor3);
set(handles.text5,'string', 'Decrypted Message');



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double
global ha g pg pp



 p = 7;
xa = randi([1 p-1]);%Calculate value of Xa
xb = randi([1 p-1]);%Calculate value of Xb




%Calculate value of Ya and Yb
ya = power(g,xa);
ya = mod(ya,p);
yb = power(g,xb);
yb = mod(yb,p);
```

```
%Calculate shared key
ha = power(yb,xa);
ha = mod(ha,p);
hb = power(ya,xb);
hb = mod(hb,p);

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, ~, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cla(handles.axes1);
cla(handles.axes2);
set(handles.edit3,'string','')
set(handles.edit2,'string','')
set(handles.edit4,'string','')
set(handles.edit5,'string','')
set(handles.uitable1, 'Data', {});




% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double



% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a
double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a
double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a
double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a
double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Blowfish keys.m:

```
function blowfish_keys(P,S)
P = [
    0x243f6a88, 0x85a308d3, 0x13198a2e, 0x03707344, 0xa4093822, 0x299f31d0,
    0x082efa98, 0xec4e6c89, 0x452821e6, 0x38d01377, 0xbe5466cf, 0x34e90c6c,
    0xc0ac29b7, 0xc97c50dd, 0x3f84d5b5, 0xb5470917, 0x9216d5d9, 0x8979fb1b
]
```

```
S = [
    [
        0xd1310ba6, 0x98dfb5ac, 0x2ffd72db, 0xd01adfb7, 0xb8e1afed,
0x6a267e96,
        0xba7c9045, 0xf12c7f99, 0x24a19947, 0xb3916cf7, 0x0801f2e2,
0x858efc16,
        0x636920d8, 0x71574e69, 0xa458fea3, 0xf4933d7e, 0x0d95748f,
0x728eb658,
        0x718bcd58, 0x82154aee, 0x7b54a41d, 0xc25a59b5, 0x9c30d539,
0x2af26013,
        0xc5d1b023, 0x286085f0, 0xca417918, 0xb8db38ef, 0x8e79dcb0,
0x603a180e,
        0x6c9e0e8b, 0xb01e8a3e, 0xd71577c1, 0xbd314b27, 0x78af2fda,
0x55605c60,
        0xe65525f3, 0xaa55ab94, 0x57489862, 0x63e81440, 0x55ca396a,
0x2aab10b6,
        0xb4cc5c34, 0x1141e8ce, 0xa15486af, 0x7c72e993, 0xb3ee1411,
0x636fbc2a,
        0x2ba9c55d, 0x741831f6, 0xce5c3e16, 0x9b87931e, 0xafd6ba33,
0x6c24cf5c,
        0x7a325381, 0x28958677, 0x3b8f4898, 0x6b4bb9af, 0xc4bfe81b,
0x66282193,
        0x61d809cc, 0xfb21a991, 0x487cac60, 0x5dec8032, 0xef845d5d,
0xe98575b1,
        0xdc262302, 0xeb651b88, 0x23893e81, 0xd396acc5, 0x0f6d6ff3,
0x83f44239,
        0x2e0b4482, 0xa4842004, 0x69c8f04a, 0x9e1f9b5e, 0x21c66842,
0xf6e96c9a,
        0x670c9c61, 0xabd388f0, 0x6a51a0d2, 0xd8542f68, 0x960fa728,
0xab5133a3,
        0x6eef0b6c, 0x137a3be4, 0xba3bf050, 0x7efb2a98, 0xa1f1651d,
0x39af0176,
        0x66ca593e, 0x82430e88, 0x8cee8619, 0x456f9fb4, 0x7d84a5c3,
0x3b8b5ebe,
        0xe06f75d8, 0x85c12073, 0x401a449f, 0x56c16aa6, 0x4ed3aa62,
0x363f7706,
        0x1bfedf72, 0x429b023d, 0x37d0d724, 0xd00a1248, 0xdb0fead3,
0x49f1c09b,
        0x075372c9, 0x80991b7b, 0x25d479d8, 0xf6e8def7, 0xe3fe501a,
0xb6794c3b,
        0x976ce0bd, 0x04c006ba, 0xc1a94fb6, 0x409f60c4, 0x5e5c9ec2,
0x196a2463,
        0x68fb6faf, 0x3e6c53b5, 0x1339b2eb, 0x3b52ec6f, 0x6dfc511f,
0x9b30952c,
        0xcc814544, 0xaf5ebd09, 0xbee3d004, 0xde334afd, 0x660f2807,
0x192e4bb3,
        0xc0cba857, 0x45c8740f, 0xd20b5f39, 0xb9d3fbdb, 0x5579c0bd,
0x1a60320a,
        0xd6a100c6, 0x402c7279, 0x679f25fe, 0xfb1fa3cc, 0x8ea5e9f8,
0xdb3222f8,
        0x3c7516df, 0xfd616b15, 0x2f501ec8, 0xad0552ab, 0x323db5fa,
0xfd238760,
        0x53317b48, 0x3e00df82, 0x9e5c57bb, 0xca6f8ca0, 0x1a87562e,
0xdf1769db,
        0xd542a8f6, 0x287effc3, 0xac6732c6, 0x8c4f5573, 0x695b27b0,
0xbbca58c8,
        0xe1ffa35d, 0xb8f011a0, 0x10fa3d98, 0xfd2183b8, 0x4afcb56c,
0x2dd1d35b,
        0x9a53e479, 0xb6f84565, 0xd28e49bc, 0x4bfb9790, 0xe1ddf2da,
0xa4cb7e33,
        0x62fb1341, 0xcee4c6e8, 0xef20cada, 0x36774c01, 0xd07e9efe,
0x2bf11fb4,
        0x95dbda4d, 0xae909198, 0xeaad8e71, 0x6b93d5a0, 0xd08ed1d0,
0xafc725e0,
        0x8e3c5b2f, 0x8e7594b7, 0x8ff6e2fb, 0xf2122b64, 0x8888b812,
0x900df01c,
        0x4fad5ea0, 0x688fc31c, 0xd1cff191, 0xb3a8c1ad, 0x2f2f2218,
0xbe0e1777,
        0xea752dfe, 0x8b021fa1, 0xe5a0cc0f, 0xb56f74e8, 0x18acf3d6,
0xce89e299,
        0xb4a84fe0, 0xfd13e0b7, 0x7cc43b81, 0xd2ada8d9, 0x165fa266,
0x80957705,
        0x93cc7314, 0x211a1477, 0xe6ad2065, 0x77b5fa86, 0xc75442f5,
0xfb9d35cf,
        0xebcdaf0c, 0x7b3e89a0, 0xd6411bd3, 0xae1e7e49, 0x00250e2d,
0x2071b35e,
        0x226800bb, 0x57b8e0af, 0x2464369b, 0xf009b91e, 0x5563911d,
0x59dfa6aa,
        0x78c14389, 0xd95a537f, 0x207d5ba2, 0x02e5b9c5, 0x83260376,
0x6295cfa9,
        0x11c81968, 0x4e734a41, 0xb3472dca, 0x7b14a94a, 0x1b510052,
0x9a532915,
        0xd60f573f, 0xbc9bc6e4, 0x2b60a476, 0x81e67400, 0x08ba6fb5,
0x571be91f,
        0xf296ec6b, 0x2a0dd915, 0xb6636521, 0xe7b9f9b6, 0xff34052e,
0xc5855664,
        0x53b02d5d, 0xa99f8fa1, 0x08ba4799, 0x6e85076a
    ],

    [
        0x4b7a70e9, 0xb5b32944, 0xdb75092e, 0xc4192623, 0xad6ea6b0,
0x49a7df7d,
        0x9cee60b8, 0x8fedb266, 0xecaa8c71, 0x699a17ff, 0x5664526c,
0xc2b19ee1,
        0x193602a5, 0x75094c29, 0xa0591340, 0xe4183a3e, 0x3f54989a,
0x5b429d65,
        0x6b8fe4d6, 0x99f73fd6, 0xa1d29c07, 0xefe830f5, 0x4d2d38e6,
0xf0255dc1,
        0x4cdd2086, 0x8470eb26, 0x6382e9c6, 0x021ecc5e, 0x09686b3f,
0x3ebaefc9,
        0x3c971814, 0x6b6a70a1, 0x687f3584, 0x52a0e286, 0xb79c5305,
0xaa500737,
        0x3e07841c, 0x7fdeae5c, 0x8e7d44ec, 0x5716f2b8, 0xb03ada37,
0xf0500c0d,
        0xf01c1f04, 0x0200b3ff, 0xae0cf51a, 0x3cb574b2, 0x25837a58,
0xdc0921bd,
        0xd19113f9, 0x7ca92ff6, 0x94324773, 0x22f54701, 0x3ae5e581,
0x37c2dadc,
        0xc8b57634, 0x9af3dda7, 0xa9446146, 0x0fd0030e, 0xecc8c73e,
0xa4751e41,
        0xe238cd99, 0x3bea0e2f, 0x3280bba1, 0x183eb331, 0x4e548b38,
0x4f6db908,
        0x6f420d03, 0xf60a04bf, 0x2cb81290, 0x24977c79, 0x5679b072,
0xbcaf89af,
        0xde9a771f, 0xd9930810, 0xb38bae12, 0xdccf3f2e, 0x5512721f,
0x2e6b7124,
        0x501adde6, 0x9f84cd87, 0x7a584718, 0x7408da17, 0xbc9f9abc,
0xe94b7d8c,
        0xec7aec3a, 0xdb851dfa, 0x63094366, 0xc464c3d2, 0xef1c1847,
0x3215d908,
        0xdd433b37, 0x24c2ba16, 0x12a14d43, 0x2a65c451, 0x50940002,
0x133ae4dd,
        0x71dff89e, 0x10314e55, 0x81ac77d6, 0x5f11199b, 0x043556f1,
0xd7a3c76b,
        0x3c11183b, 0x5924a509, 0xf28fe6ed, 0x97f1fbfa, 0x9ebabf2c,
0x1e153c6e,
        0x86e34570, 0xeae96fb1, 0x860e5e0a, 0x5a3e2ab3, 0x771fe71c,
0x4e3d06fa,
        0x2965dcb9, 0x99e71d0f, 0x803e89d6, 0x5266c825, 0x2e4cc978,
0x9c10b36a,
        0xc6150eba, 0x94e2ea78, 0xa5fc3c53, 0x1e0a2df4, 0xf2f74ea7,
0x361d2b3d,
        0x1939260f, 0x19c27960, 0x5223a708, 0xf71312b6, 0xebadfe6e,
0xeac31f66,
        0xe3bc4595, 0xa67bc883, 0xb17f37d1, 0x018cff28, 0xc332ddef,
0xbe6c5aa5,
        0x65582185, 0x68ab9802, 0xeecea50f, 0xdb2f953b, 0x2aef7dad,
0x5b6e2f84,
        0x1521b628, 0x29076170, 0xecdd4775, 0x619f1510, 0x13cca830,
0xeb61bd96,
        0x0334fe1e, 0xaa0363cf, 0xb5735c90, 0x4c70a239, 0xd59e9e0b,
0xcbaade14,
        0xeecc86bc, 0x60622ca7, 0x9cab5cab, 0xb2f3846e, 0x648b1eaf,
0x19bdf0ca,
        0xa02369b9, 0x655abb50, 0x40685a32, 0x3c2ab4b3, 0x319ee9d5,
0xc021b8f7,
        0x9b540b19, 0x875fa099, 0x95f7997e, 0x623d7da8, 0xf837889a,
0x97e32d77,
        0x11ed935f, 0x16681281, 0x0e358829, 0xc7e61fd6, 0x96dedfa1,
0x7858ba99,
        0x57f584a5, 0x1b227263, 0x9b83c3ff, 0x1ac24696, 0xcdb30aeb,
0x532e3054,
        0x8fd948e4, 0x6dbc3128, 0x58ebf2ef, 0x34c6ffea, 0xfe28ed61,
0xee7c3c73,
        0x5d4a14d9, 0xe864b7e3, 0x42105d14, 0x203e13e0, 0x45eee2b6,
0xa3aaabea,
        0xdb6c4f15, 0xfacb4fd0, 0xc742f442, 0xef6abbb5, 0x654f3b1d,
0x41cd2105,
        0xd81e799e, 0x86854dc7, 0xe44b476a, 0x3d816250, 0xcf62a1f2,
0x5b8d2646,
        0xfc8883a0, 0xc1c7b6a3, 0x7f1524c3, 0x69cb7492, 0x47848a0b,
0x5692b285,
        0x095bbf00, 0xad19489d, 0x1462b174, 0x23820e00, 0x58428d2a,
0x0c55f5ea,
        0x1dadf43e, 0x233f7061, 0x3372f092, 0x8d937e41, 0xd65fecf1,
0x6c223bdb,
        0x7cde3759, 0xcbee7460, 0x4085f2a7, 0xce77326e, 0xa6078084,
0x19f8509e,
        0xe8efd855, 0x61d99735, 0xa969a7aa, 0xc50c06c2, 0x5a04abfc,
0x800bcadc,
        0x9e447a2e, 0xc3453484, 0xfdd56705, 0x0e1e9ec9, 0xdb73dbd3,
0x105588cd,
        0x675fda79, 0xe3674340, 0xc5c43465, 0x713e38d8, 0x3d28f89e,
0xf16dff20,
        0x153e21e7, 0x8fb03d4a, 0xe6e39f2b, 0xdb83adf7
    ],

    [
        0xe93d5a68, 0x948140f7, 0xf64c261c, 0x94692934, 0x411520f7,
0x7602d4f7,
        0xbcf46b2e, 0xd4a20068, 0xd4082471, 0x3320f46a, 0x43b7d4b7,
0x500061af,
        0x1e39f62e, 0x97244546, 0x14214f74, 0xbf8b8840, 0x4d95fc1d,
0x96b591af,
        0x70f4ddd3, 0x66a02f45, 0xbfbc09ec, 0x03bd9785, 0x7fac6dd0,
0x31cb8504,
        0x96eb27b3, 0x55fd3941, 0xda2547e6, 0xabca0a9a, 0x28507825,
0x530429f4,
        0x0a2c86da, 0xe9b66dfb, 0x68dc1462, 0xd7486900, 0x680ec0a4,
0x27a18dee,
        0x4f3ffea2, 0xe887ad8c, 0xb58ce006, 0x7af4d6b6, 0xaace1e7c,
0xd3375fec,
        0xce78a399, 0x406b2a42, 0x20fe9e35, 0xd9f385b9, 0xee39d7ab,
0x3b124e8b,
        0x1dc9faf7, 0x4b6d1856, 0x26a36631, 0xeae397b2, 0x3a6efa74,
0xdd5b4332,
        0x6841e7f7, 0xca7820fb, 0xfb0af54e, 0xd8feb397, 0x454056ac,
0xba489527,
        0x55533a3a, 0x20838d87, 0xfe6ba9b7, 0xd096954b, 0x55a867bc,
0xa1159a58,
        0xcca92963, 0x99e1db33, 0xa62a4a56, 0x3f3125f9, 0x5ef47e1c,
0x9029317c,
        0xfdf8e802, 0x04272f70, 0x80bb155c, 0x05282ce3, 0x95c11548,
0xe4c66d22,
        0x48c1133f, 0xc70f86dc, 0x07f9c9ee, 0x41041f0f, 0x404779a4,
0x5d886e17,
        0x325f51eb, 0xd59bc0d1, 0xf2bcc18f, 0x41113564, 0x257b7834,
0x602a9c60,
        0xdff8e8a3, 0x1f636c1b, 0x0e12b4c2, 0x02e1329e, 0xaf664fd1,
0xcad18115,
        0x6b2395e0, 0x333e92e1, 0x3b240b62, 0xeebeb922, 0x85b2a20e,
0xe6ba0d99,
        0xde720c8c, 0x2da2f728, 0xd0127845, 0x95b794fd, 0x647d0862,
0xe7ccf5f0,
        0x5449a36f, 0x877d48fa, 0xc39dfd27, 0xf33e8d1e, 0x0a476341,
0x992eff74,
        0x3a6f6eab, 0xf4f8fd37, 0xa812dc60, 0xa1ebddf8, 0x991be14c,
0xdb6e6b0d,
        0xc67b5510, 0x6d672c37, 0x2765d43b, 0xdcd0e804, 0xf1290dc7,
0xcc00ffa3,
        0xb5390f92, 0x690fed0b, 0x667b9ffb, 0xcedb7d9c, 0xa091cf0b,
0xd9155ea3,
        0xbb132f88, 0x515bad24, 0x7b9479bf, 0x763bd6eb, 0x37392eb3,
0xcc115979,
        0x8026e297, 0xf42e312d, 0x6842ada7, 0xc66a2b3b, 0x12754ccc,
0x782ef11c,
        0x6a124237, 0xb79251e7, 0x06a1bbe6, 0x4bfb6350, 0x1a6b1018,
0x11caedfa,
        0x3d25bdd8, 0xe2e1c3c9, 0x44421659, 0x0a121386, 0xd90cec6e,
0xd5abea2a,
        0x64af674e, 0xda86a85f, 0xbebfe988, 0x64e4c3fe, 0x9dbc8057,
0xf0f7c086,
        0x60787bf8, 0x6003604d, 0xd1fd8346, 0xf6381fb0, 0x7745ae04,
0xd736fccc,
        0x83426b33, 0xf01eab71, 0xb0804187, 0x3c005e5f, 0x77a057be,
0xbde8ae24,
        0x55464299, 0xbf582e61, 0x4e58f48f, 0xf2ddfda2, 0xf474ef38,
0x8789bdc2,
        0x5366f9c3, 0xc8b38e74, 0xb475f255, 0x46fcd9b9, 0x7aeb2661,
0x8b1ddf84,
        0x846a0e79, 0x915f95e2, 0x466e598e, 0x20b45770, 0x8cd55591,
0xc902de4c,
        0xb90bace1, 0xbb8205d0, 0x11a86248, 0x7574a99e, 0xb77f19b6,
0xe0a9dc09,
        0x662d09a1, 0xc4324633, 0xe85a1f02, 0x09f0be8c, 0x4a99a025,
0x1d6efe10,
        0x1ab93d1d, 0x0ba5a4df, 0xa186f20f, 0x2868f169, 0xdcb7da83,
0x573906fe,
        0xa1e2ce9b, 0x4fcd7f52, 0x50115e01, 0xa70683fa, 0xa002b5c4,
0x0de6d027,
        0x9af88c27, 0x773f8641, 0xc3604c06, 0x61a806b5, 0xf0177a28,
0xc0f586e0,
        0x006058aa, 0x30dc7d62, 0x11e69ed7, 0x2338ea63, 0x53c2dd94,
0xc2c21634,
        0xbbcbee56, 0x90bcb6de, 0xebfc7da1, 0xce591d76, 0x6f05e409,
0x4b7c0188,
        0x39720a3d, 0x7c927c24, 0x86e3725f, 0x724d9db9, 0x1ac15bb4,
0xd39eb8fc,
        0xed545578, 0x08fca5b5, 0xd83d7cd3, 0x4dad0fc4, 0x1e50ef5e,
0xb161e6f8,
        0xa28514d9, 0x6c51133c, 0x6fd5c7e7, 0x56e14ec4, 0x362abfce,
0xddc6c837,
        0xd79a3234, 0x92638212, 0x670efa8e, 0x406000e0
    ],

    [
        0x3a39ce37, 0xd3faf5cf, 0xabc27737, 0x5ac52d1b, 0x5cb0679e,
0x4fa33742,
        0xd3822740, 0x99bc9bbe, 0xd5118e9d, 0xbf0f7315, 0xd62d1c7e,
0xc700c47b,
        0xb78c1b6b, 0x21a19045, 0xb26eb1be, 0x6a366eb4, 0x5748ab2f,
0xbc946e79,
        0xc6a376d2, 0x6549c2c8, 0x530ff8ee, 0x468dde7d, 0xd5730a1d,
0x4cd04dc6,
        0x2939bbdb, 0xa9ba4650, 0xac9526e8, 0xbe5ee304, 0xa1fad5f0,
0x6a2d519a,
        0x63ef8ce2, 0x9a86ee22, 0xc089c2b8, 0x43242ef6, 0xa51e03aa,
0x9cf2d0a4,
        0x83c061ba, 0x9be96a4d, 0x8fe51550, 0xba645bd6, 0x2826a2f9,
0xa73a3ae1,
        0x4ba99586, 0xef5562e9, 0xc72fefd3, 0xf752f7da, 0x3f046f69,
0x77fa0a59,
        0x80e4a915, 0x87b08601, 0x9b09e6ad, 0x3b3ee593, 0xe990fd5a,
0x9e34d797,
        0x2cf0b7d9, 0x022b8b51, 0x96d5ac3a, 0x017da67d, 0xd1cf3ed6,
0x7c7d2d28,
        0x1f9f25cf, 0xadf2b89b, 0x5ad6b472, 0x5a88f54c, 0xe029ac71,
0xe019a5e6,
        0x47b0acfd, 0xed93fa9b, 0xe8d3c48d, 0x283b57cc, 0xf8d56629,
0x79132e28,
        0x785f0191, 0xed756055, 0xf7960e44, 0xe3d35e8c, 0x15056dd4,
0x88f46dba,
        0x03a16125, 0x0564f0bd, 0xc3eb9e15, 0x3c9057a2, 0x97271aec,
0xa93a072a,
        0x1b3f6d9b, 0x1e6321f5, 0xf59c66fb, 0x26dcf319, 0x7533d928,
0xb155fdf5,
        0x03563482, 0x8aba3cbb, 0x28517711, 0xc20ad9f8, 0xabcc5167,
0xccad925f,
        0x4de81751, 0x3830dc8e, 0x379d5862, 0x9320f991, 0xea7a90c2,
0xfb3e7bce,
        0x5121ce64, 0x774fbe32, 0xa8b6e37e, 0xc3293d46, 0x48de5369,
0x6413e680,
        0xa2ae0810, 0xdd6db224, 0x69852dfd, 0x09072166, 0xb39a460a,
0x6445c0dd,
        0x586cdecf, 0x1c20c8ae, 0x5bbef7dd, 0x1b588d40, 0xccd2017f,
0x6bb4e3bb,
        0xdda26a7e, 0x3a59ff45, 0x3e350a44, 0xbcb4cdd5, 0x72eacea8,
0xfa6484bb,
        0x8d6612ae, 0xbf3c6f47, 0xd29be463, 0x542f5d9e, 0xaec2771b,
0xf64e6370,
        0x740e0d8d, 0xe75b1357, 0xf8721671, 0xaf537d5d, 0x4040cb08,
0x4eb4e2cc,
        0x34d2466a, 0x0115af84, 0xe1b00428, 0x95983a1d, 0x06b89fb4,
0xce6ea048,
        0x6f3f3b82, 0x3520ab82, 0x011a1d4b, 0x277227f8, 0x611560b1,
0xe7933fdc,
        0xbb3a792b, 0x344525bd, 0xa08839e1, 0x51ce794b, 0x2f32c9b7,
0xa01fbac9,
        0xe01cc87e, 0xbcc7d1f6, 0xcf0111c3, 0xa1e8aac7, 0x1a908749,
0xd44fbd9a,
        0xd0dadecb, 0xd50ada38, 0x0339c32a, 0xc6913667, 0x8df9317c,
0xe0b12b4f,
        0xf79e59b7, 0x43f5bb3a, 0xf2d519ff, 0x27d9459c, 0xbf97222c,
0x15e6fc2a,
        0x0f91fc71, 0x9b941525, 0xfae59361, 0xceb69ceb, 0xc2a86459,
0x12baa8d1,
        0xb6c1075e, 0xe3056a0c, 0x10d25065, 0xcb03a442, 0xe0ec6e0e,
0x1698db3b,
        0x4c98a0be, 0x3278e964, 0x9f1f9532, 0xe0d392df, 0xd3a0342b,
0x8971f21e,
        0x1b0a7441, 0x4ba3348c, 0xc5be7120, 0xc37632d8, 0xdf359f8d,
0x9b992f2e,
        0xe60b6f47, 0x0fe3f11d, 0xe54cda54, 0x1edad891, 0xce6279cf,
0xcd3e7e6f,
        0x1618b166, 0xfd2c1d05, 0x848fd2c5, 0xf6fb2299, 0xf523f357,
0xa6327623,
        0x93a83531, 0x56cccd02, 0xacf08162, 0x5a75ebb5, 0x6e163697,
0x88d273cc,
        0xde966292, 0x81b949d0, 0x4c50901b, 0x71c65614, 0xe6c6c7bd,
0x327a140a,
        0x45e1d006, 0xc3f27b9a, 0xc9aa53fd, 0x62a80f00, 0xbb25bfe2,
0x35bdd2f6,
        0x71126905, 0xb2040222, 0xb6cbcf7c, 0xcd769c2b, 0x53113ec0,
0x1640e3d3,
        0x38abbd60, 0x2547adf0, 0xba38209c, 0xf746ce76, 0x77afa1c5,
0x20756060,
        0x85cbfe4e, 0x8ae88dd8, 0x7aaaf9b0, 0x4cf9aa7e, 0x1948c25c,
0x02fb8a8c,
        0x01c36ae4, 0xd6ebe1f9, 0x90d4f869, 0xa65cdea0, 0x3f09252d,
0xc208e69f,
        0xb74e6132, 0xce77e25b, 0x578fdfe3, 0x3ac372e6
    ]
]
```