

DATA INTEGRITY USING CAESAR CIPHER AND RESIDUE NUMBER SYSTEM

BY

Damilola David POPOOLA

17/27/MCS014

**BEING A DISSERTATION SUBMITTED TO THE DEPARTMENT OF COMPUTER
SCIENCE, COLLEGE OF INFORMATION AND COMMUNICATION TECHNOLOGY,**

KWARA STATE UNIVERSITY, MALETE, NIGERIA.

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

MAY, 2019

CERTIFICATION

This is to certify that this research was carried out by **Damilola David POPOOLA** with Matriculation number **17/27/MCS014**, in the Department of Computer Science, College of Information and Communication Technology, Kwara State University, Malete, Nigeria.

Professor K.A Gbolagade
(Supervisor)

Date

Professor K.A Gbolagade
(Head of Department)

Date

Professor S.K Subair
(Dean, PG School)

Date

External Examiner

Date

DEDICATION

This dissertation is dedicated to Almighty God who has been my guide throughout the duration of my Master Degree programme at Kwara State University, Malete, Nigeria.

ACKNOWLEDGEMENTS

I give thanks to Almighty God for sparing my life up to this moment, for financial empowerment and for granting me wisdom, knowledge and understanding that was required to make the programme a success.

I sincerely appreciate the effort of my Supervisor, Head of Department and The Provost, College of Information and Communication Technology, Prof. Kazeem Alagbe Gbolagade, for his mentorship and welcoming heart and for the clear guide of RNS he has imbibed which was vital to the achievement of the success of this research.

I would also sincerely appreciate the effort of Head of Department; Prof. Kazeem Alagbe Gbolagade, the Post Graduate Coordinator; Dr R.M Isiaka and all other staff of the department; Dr R.S Babatunde, Dr J.F Ajao, Dr A.N Babatunde, Mr S.O Abdulsalam, Mrs Shakirat Yusuff, Mrs Balogun and Mr D.P Oyinloye and Mr T.O Usman.

I also want to appreciate my Course mates (2017/2018 Computer Science Post-Graduate Students), RLFP colleagues within and outside the Department and my friends within and outside the University Community.

I will also love to express my profound gratitude to family members; my father, Mr Moses Popoola and to my late Mother, Mrs. Rachael Popoola for their parental care and support in all areas and my brothers Oluwaseun Popoola and Femi Popoola for their unending support towards my academic pursuit.

TABLE OF CONTENTS

CERTIFICATION	i
DEDICATION	ii
ACKNOWLEDGEMENTS	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	vi
LIST OF TABLES	vii
ABSTRACT	viii
CHAPTER ONE	1
INTRODUCTION	1
1.1 Background of the study	1
1.2 Statement of Problem	4
1.3 Aim and Objectives	5
1.4 Scope of the Study	6
1.5 Significance of the study	6
1.6 Research layout	6
CHAPTER TWO	8
LITERATURE REVIEW	8
2.1 Related Concept	8
2.1.1 Data security	8
2.1.2 Cryptography	8
2.1.3 Residue number system	12

2.1.4 Forward Conversion	13
2.1.5 Reverse Conversion	14
2.1.6 Yoruba Character	15
2.2 Review of related works	17
CHAPTER THREE	21
METHODOLOGY	21
3.1 Existing system	21
3.2 Proposed Methodology	21
3.2.1 English to Yoruba character table formation	23
3.2.2 Yoruba to Numbers character table formation	24
3.2.3 Transposition	24
3.2.4 Encryption process	25
3.2.5 Decryption process	25
3.3 System Requirement	26
3.4 Choice of Programming Language	26
CHAPTER FOUR	27
RESULTS AND DISCUSSION	27
4.1 Text swapping	27
4.2 Encryption of plain text	27
4.3 Decryption of cipher text	32
4.4 Comparison with existing approaches	35
CHAPTER FIVE	37
SUMMARY AND CONCLUSION	37
5.1 Summary	37
5.2 Conclusion	37

5.3	Major contributions to knowledge	37
5.4	Future work	38

LIST OF FIGURES

Figure	Page
1.1: A basic idea for secure communication	1
1.2: Data transmission process	3
3.1: System interface	22
3.2: Samples of captured Handwritten Yoruba Character	23
3.3: Yoruba Character Table	23
3.4: Numeric Character Table	24
4.1: Encryption process	32
4.2: Decryption process	35

LIST OF TABLES

Table	Page
4.1: Yoruba Swapping (First Substitution)	22
4.2: Numeric Swapping (Second substitution)	22
4.3 Residue representation of plain text (cipher text)	31
4.4: Comparison with existing methods	36

ABSTRACT

Data security ensures a secure communication through a digital medium. The data needs to be protected from unauthorized access and transmitted to the intended receiver with confidentiality and integrity. Various schemes have been proposed over the years towards ensuring that data sent over a digital medium is hard to understand by an external body, by transforming the data from plaintext to a gibberish form known as a cipher text. Although the cipher text have been able to make data hard to understand by a third party, the irregular representation attributed to it serves as a means through which eavesdropper's attention is drawn to such data. Therefore, this research presents a method that incorporates both substitutional and transposition cipher in the encryption of data. It also incorporates an additional stage to the traditional Caesar cipher with creation of two character tables, firstly to substitute the plaintext to Yoruba characters and secondly to substitute the Yoruba Characters to Numbers which will allow the implementation of RNS to change the representation of the characters from gibberish form to a matrix. The transposition was done with Residue Number System using the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$. The proposed scheme was able to produce the cipher text as residues in the form of matrix which will make messages sent over a network attain low suspicion from attackers.

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

Digital communication has grown to be a fundamental aspect of interaction between two ends in this present era, with a lot of internet based field. It is of great importance that communication be made as a top secret. Thus, the security of data passed over a network is a primary issue, which stretches to the confidentiality as well as integrity of the data, making it mandatory to protect against unauthorized access and use. In the quest to have a secured communication between two ends, the concept of cryptography came into existence, which is the science of using the mathematics to encrypt and decrypt data. Cryptography enables you to store sensitive information or transmit it across the insecure networks (like the internet), such that it cannot be read by anyone except the intended recipient (Badoni & Jain, 2014). The cryptosystem can therefore be said to be a set of algorithms combined with keys to convert the original message (Plain-text) to encrypted message (Cipher-text) and convert it back in the intended recipient side to the original message (Al-Hazaimah, 2013). This concept was illustrated by Barakat, Eder and Hanke (2018) where the need arises for Alice to send a message securely to Bob in the presence of Eve or Mallory, where Alice denotes the sender, Bob denotes the receiver, Eve denotes the eavesdropper and Mallory denotes the Man-in-the-middle attack as seen in Figure 1.1.

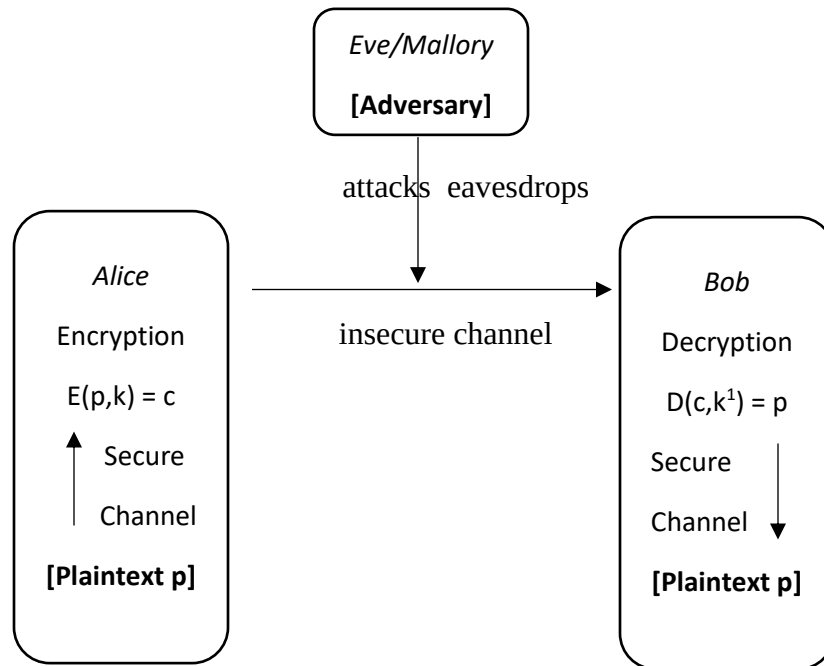


Figure 1.1: A basic idea for secure communication (Source: Barakat, Eder & Hanke, 2018)

The cryptography consists of complex mathematical formulas that dictate the rules of conversion process from plain text to cipher text and vice versa combined with the key. However, some of encryption and decryption algorithms use the same key and in other encryption and decryption algorithms they use different keys but these keys must be related, known as symmetric and asymmetric algorithm respectively (Deepak & Parveen, 2017).

Data that can be seen and understood easily without the need for further understanding is called plain text. The means of changing the representation of such data with the purpose of hiding its substance is referred to as encryption. Encrypting a plaintext leads to an unreadable and difficult to understand representation called cipher text. Encryption is utilized to guarantee that data escapes anybody for whom it isn't planned, even the individuals who can see the encrypted data. The process of reverting cipher text to its original plaintext is called decryption.

There are two techniques of encryption; substitution and transposition technique. In substitution technique, the letters of plaintext are replaced by other letters or any number or by symbols. Example Caesar cipher, hill cipher, mono-alphabetic cipher etc. In transposition technique, some sort of permutation is performed on plaintext in order to change its representation.

For a message to be sent across from the sender to the receiver, it has to be through a transmission medium, which is coordinated by some set of protocols as illustrated in Figure 1.2. The sender is where the message to be transmitted originates and the receiver is the recipient of the message, the transmission medium is the channel/path through which the message moves from the sender to the receiver and the protocol is the set of rules that govern/coordinates the transmission process.

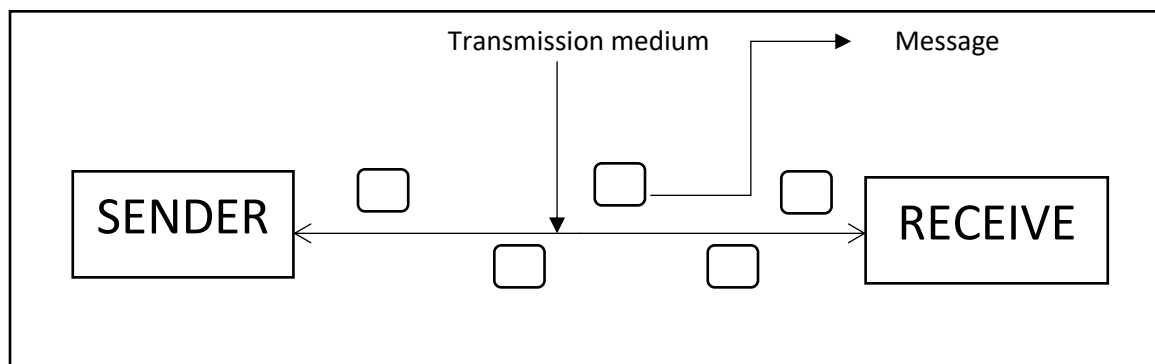


Figure 1.2: Data transmission process

The major issue to design any encryption and decryption algorithm is to improve the security level such that the implementation of such algorithm on data makes it difficult and complex for such data to be understood by an external body.

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 its 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 Caesar cipher algorithm, also known as shift cipher was a method developed by a Roman Ruler known as Julius Caesar. The method represents each alphabet in the message with another alphabet by using the third alphabet to the right of the initiate alphabet. With the ease in implementing a Caesar cipher algorithm, it is considered to be a weak algorithm, as intruders can easily try a shift on each coded message and easily come to conclusion of what the original message is. Subsequently, other approaches came to existence towards ensuring that data sent over a digital medium is hard to understand by an external body and according to analysis carried out by Hercigonja and gimnazija (2016), it was concluded that among symmetric algorithms such as DES, 3DES, CAST-128, BLOWFISH, IDEA, AES, RC6 and RSA being an asymmetric algorithm, symmetric algorithm performs better when it comes to encryption and decryption.

Even though the listed algorithms have achieved a level of success when it comes to encryption and decryption according to Hercigonja and gimnazija (2016), by the various improvement that has been made on them over the years, their encrypted version of the message still comes out in the regular gibberish form. Therefore, in order to ensure messages are sent over a network securely, this research proposes an approach that will keep the identity of the message being sent and also introduce more complexity to the Caesar cipher algorithm which will be used to secure the message combined with residue number system.

1.2 Statement of Problem

Communications in this present era mostly depend on digital medium, which has obviously opened a means through which attackers have the knowledge of what is been communicated. Although this problem might seem to be solved with the various cryptography algorithms in existence, it is still limited to the fact that an encrypted data being in a gibberish form creates an avenue for suspicion with its representation as a result of the substitution or transposition technique.

The Caesar cipher algorithm being a substitution method of encrypting data has the drawback of proneness to attack, due to the fact that the original message can be easily derived once an attacker tries all the possible combination of shifts for all the 26 letters. Recently, Verma, Gaba, and Miglani (2017) proposed a scheme to solve the drawback of the existing schemes by adding symbols and special characters in other to increase the length of combinations an attacker would perform. The new scheme was able to deliver an enhancement through the capability to encrypt symbols, characters as well as digits but still limited to representing the cipher text in a gibberish form, leading to the limitation discovered in the transposition method. In the same vein, despite the success achieved by the proposed scheme, it is still opened to attacker once the attacker can carefully go through all the 82 combinations used in the scheme.

Furthermore, Lubis, Simbolon, Batubara, and Sembiring (2017) proposed a scheme which combines the use of both transposition and substitution method without incorporating the use of symbols but having the result of the encryption in a gibberish form as well

Having understudied both encryption methods, it is obvious that a better scheme is required, which will not only make the cipher-text obvious to attackers but also to make the message difficult to detect by changing the representation of the text completely, as well as increasing the stages involved in the Caesar cipher algorithm.

1.3 Aim and Objectives

The aim of this research is to devise a means of ensuring that encrypted data maintains a high level of invisibility.

The objectives are to;

- i. swap each character of the message to another language for the first level encryption.
- ii. implement forward conversion for each index of the swapped characters using the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$ for the encryption process.
- iii. implement reverse conversion for first level decryption using Chinese Remainder Theorem.
- iv. evaluate and compare the performance of the proposed system

1.4 Scope of the Study

The proposed scheme is limited to the use of the alphanumeric characters and keyboard symbols for the original message, Yoruba alphabets for the first level encryption (substitution method), forward conversion using the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$ for the second level encryption process (transposition method), Chinese remainder theorem for the decryption process and Caesar Cipher for implementation of the character table. The scheme works only with English alphabets as the plaintext which was the bedrock of the character table. The plaintext should be made available in a text file of any desired name and must not exceed a thousand words.

1.5 Significance of the Study

This proposed scheme provides an improvement over the Caesar Cipher towards ensuring that communication of vital information from one end to the other is difficult to detect. The scheme also showcases how residue number system can be used as a means of transposition for encryption.

It also forward validates the co-existence of both transposition and substitution method of encryption as a better means of improving data security.

1.6 Research Layout

This research is comprised of five chapters.

Chapter one brings an introduction to the research, showcasing the background of the study, the statement of the 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, as well as relevant future work based on the outcome of this research.

CHAPTER TWO

LITERATURE REVIEW

2.1 Related Concept

2.1.1 Data security

Data security refers to protective digital privacy measures that are applied to prevent unauthorized access to computers, databases and websites. Data security also protects data from corruption. Data security is an essential aspect of IT for organizations of every size and type. Data security is also known as information security (IS) or computer security. Examples of data security technologies include backups, data masking and data erasure. A key data security technology measure is encryption, where digital data, software/hardware, and hard drives are encrypted and therefore rendered unreadable to unauthorized users and hackers. One of the most commonly encountered methods of practicing data security is the use of authentication. With authentication, users must provide a password, code, biometric data, or some other form of data to verify identity before access to a system or data is granted.

2.1.2 Cryptography

Cryptography is the study of “mathematical” systems involving two kinds of security problems: privacy and authentication. A privacy system prevents the extraction information by unauthorized parties from messages transmitted over a public channel, thus assuring the sender of a message that it is being read only by the intended recipient. An authentication system prevents the unauthorized injection of messages into a public channel, the unauthorized injection of messages into a public channel, assuring the receiver of a message of the legitimacy of its sender.

A channel is considered public if its security is inadequate for the needs of its users. A channel such as a telephone line may therefore be considered private by some users and public by others. Any channel may be threatened with eavesdropping or injection or both, depending on its use. In telephone communication, the threat of injection is paramount, since the called party cannot determine which phone is calling. Eavesdropping, which requires the use of a wiretap, is technically more difficult and legally hazardous. In radio, by comparison, the situation is reversed. Eavesdropping is passive and involves no legal hazard, while injection exposes the illegitimate transmitter to discovery and prosecution. (Delfs & Knebl, 2015). Having divided our problems into those of privacy and authentication we will sometimes further subdivide authentication into message authentication, which is the problem defined above, and user authentication, in which the only task of the system is to verify that an individual is who he claims to be. (He & Zeadally 2015)

For example, the identity of an individual who presents a credit card must be verified, but there is no message which he wishes to transmit. In spite of this apparent absence of a message in user authentication, the two problems are largely equivalent. In user authentication, there is an implicit message. "I AM USER X," while message authentication is just verification of the identity of the party sending the message. Differences in the threat environments and other aspects of these two sub problems, however, sometimes make it convenient to distinguish between them.

A cryptographic system is a single parameter family $\{S_K; z_K \in \{K; z\}\}$ of invertible transformations $SK : \{P\} \rightarrow \{C\}$

(1)

From a space $\{P\}$ of plaintext messages to a space $\{C\}$ of cipher text messages, the parameter K is called the key and is selected from a finite set $\{K\}$ called the key space. If the message

Spaces $\{P\}$ and $\{C\}$ are equal, we will denote them both by $\{M\}$. When discussing individual cryptographic transformations S , we will sometimes omit mention of the system and merely refer to the transformation K .

The goal in designing the cryptosystem $\{S\}$ is to make the enciphering and deciphering operations inexpensive, but to K ensure that any successful cryptanalytic operation is too complex to be economical. There are two approaches to this problem. A system which is secure due to the computational cost of cryptanalysis, but which would succumb to an attack with unlimited computation, is called computationally secure; while a system which can resist any cryptanalytic attack, no matter how much computation is allowed, is called unconditionally secure. Mohammad, (Abdelfatah & Omaima, 2014).

Unconditional security results from the existence of multiple meaningful solutions to a cryptogram. For example, the simple substitution cryptogram XMD resulting from English text can represent the plaintext messages. A computationally secure cryptogram, in contrast, contains sufficient information to uniquely determine the plaintext and the key. Its security resides solely in the cost of computing them. The only unconditionally secure system in common use is the one time pad, in which the plaintext is combined with a randomly chosen key of the same length. While such a system is provably secure, the large amount of key required makes it impracticable for most applications. When we talk about the need to develop provably secure cryptosystems we exclude those, such as the one time pad, which are un-wieldy to use. Rather, we have in mind systems using only a few hundred bits of key and implementable in either a small amount of digital hardware or a few hundred lines of software. We will call a task computationally infeasible if its cost as measured by either the amount of memory used or the runtime is finite but impossibly large. Much as error correcting codes are divided into convolution and block codes, cryptographic

systems can be divided into two broad classes: stream ciphers and block ciphers. Stream ciphers process the plaintext in small chunks (bits or characters), usually producing a pseudorandom sequence of bits which is added modulo 2 to the bits of the plaintext. Block ciphers act in a purely combinatorial fashion on large blocks of text, in such a way that a small change in the input block produces a major change in the resulting output.

Cryptography is used to achieve three main goals which are confidentiality, data integrity and authentication (Ayushi, 2010). Confidentiality is to ensure data remains private. It is usually achieved using encryption. Encryption algorithms (that use encryption keys) are used to convert plain text into cipher text and the equivalent decryption algorithm is used to convert the cipher text back to plain text. Symmetric encryption algorithms use the same key for encryption and decryption, while asymmetric algorithms use a public/private key pair. Data integrity is to ensure data is protected from accidental or deliberate (malicious) modification. Integrity is usually provided by message authentication code or hashes. A hash value is a fixed length numeric value derived from a sequence of data. Hash values are used to verify the integrity of data sent through insecure channels. The hash value of received data is compared to the hash value of the data as it was sent to determine if the data was altered. Authentication is to assure that data originates from a particular party. Digital certificates are used to provide authentication. Digital signatures are usually applied to hash values as these are significantly smaller than the source data that they represent.

2.1.2.1 Types of cryptography

1. **Secret key cryptography:** In secret key cryptography, a single key is used 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 of cryptography, it is obvious that the key must be known to both the sender and the receiver; that, in fact, is the secret.

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.

2.1.3 Residue Number System

RNS comprises a set of moduli which are independent of each other. An integer is represented by the residue of each of the modulus and arithmetic operations are based on residues individually. The advantage of using the RNS over the conventional system includes “carry-free” operation, fault tolerance, parallelism and modularity. (Gbolagade & Cotofana, 2008). These inherent features make RNS to be widely used in Digital Signal Processing (DSP) applications such as digital filtering, convolution, fast Fourier transform and image processing. (Alhassan & Gbolagade, 2013).

Let $\{m_1, m_2, m_3, \dots, m_n\}$ be a set of positive integers all greater 1. m_i is called a modulus, and then n -tuple set $\{m_1, m_2, m_3, \dots, m_n\}$ is called a moduli set. Consider an integer number Y . For each of the

modulus in $\{ m_1, m_2, m_3, \dots, m_n \}$, we have $y_i = Y \bmod m_i$, (which will be denoted as $|Y|_{m_i}$). Thus the number Y in this system is represented as $Y = (y_1, y_2, y_3, \dots, y_n)$, $0 \leq y_i < m_i$. (Siewobr, Gbolagade & Cotofana, 2014).

Given the moduli set $\{7, 8, 9\}$, the number 150 can be represented in RNS as:

$$y_1 = |Y|_{m_1} = |150|_7 = 3$$

$$y_2 = |Y|_{m_2} = |150|_8 = 6$$

$$y_3 = |Y|_{m_3} = |150|_9 = 6$$

Thus, the RNS representation of 150 is thus; $(3, 6, 6)_{RNS(7, 8, 9)}$

To avoid redundancy, the moduli set must be pair wise relatively prime. Thus $\gcd(m_i, m_j) = 1$ for $i \neq j$, where $\gcd$ means the greatest common divisor of m_i, m_j .

Let $M = \prod_{i=1}^n m_i$, then the RNS representation is unique for any integer $Y \in [0, M - 1]$

M is called the dynamic range. (Siewobr & Gbolagade, 2012)

A Decimal to Residue (D/R) converter (encoder) is needed in order to convert a decimal number to RNS representation.

2.1.4 Forward Conversion

Forward conversion is performed by a forward converter which decomposes a weighted binary number into a residue represented number with regards to a moduli set, that is, 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 collect their remainder. (Gbolagade & Cotofana, 2008).

As an example, given moduli set $\{3, 4, 5\}$, the number 4 can be represented in RNS as:

$$x_i = |X|_{m_i}$$

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

Thus, the RNS representation of 4 is $(1, 0, 4)_{RNS(3,4,5)}$.

2.1.5 Reverse Conversion

The conversion from residue representation to a conventional representation (binary or decimal representation) is known as reverse conversion. Two (2) most widely used techniques of reverse conversion are the Chinese Remainder Theorem (CRT) and Mixed Radix Conversion (MRC). (Gbolagade, 2009).

2.1.5.1 Conversion using the Chinese Remainder Theorem (CRT)

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

The CRT is given by

$$|x|_m = \left| \sum_{j=1}^N \hat{m}_j \left| \frac{r_j}{\hat{m}_j} \right|_{m_j} \right|_m$$

Where $\hat{m}_j = \frac{M}{m_j}$, $M = \prod_{j=1}^N m_j$ and

$$\gcd(m_j, m_k) = 1 \text{ for } j \neq k$$

Example:-

For moduli $m_1 = 13, m_2 = 11, m_3 = 7$ and $m_4 = 9$ find the number whose residue representation is $\{4, 2, 4, 7\}$

Solution:

$$M = \prod_{i=1}^N m_i = 13 \times 11 \times 7 \times 9 = 9009$$

$$|\hat{m}_1|_{m_1} = |11 \times 7 \times 9|_{13} = |693|_{13} = 4 \text{ then } \left| \frac{1}{\hat{m}_1} \right|_{13} = 10$$

$$|\hat{m}_2|_{m_2} = |13 \times 7 \times 9|_{11} = |819|_{11} = 5, \left| \frac{1}{\hat{m}_2} \right|_{11} = 9$$

$$|\hat{m}_3|_{m_3} = |13 \times 11 \times 9|_7 = |1287|_7 = 6, \left| \frac{1}{\hat{m}_3} \right|_7 = 6$$

$$|\hat{m}_4|_{m_4} = |13 \times 11 \times 7|_9 = |1001|_9 = 2, \left| \frac{1}{\hat{m}_4} \right|_9 = 5$$

$$|x|_{9009} = |693|_{10 \times 4|_{13}} + |819|_{9 \times 2|_{11}} + |1286|_{6 \times 4|_7} + |1001|_{5 \times 7|_9|_{9009}}$$

$$= |693 \times 1 + 819 \times 7 + 1287 \times 3 + 1001 \times 8|_{9009}$$

$$= |18295|_{9009} = 277$$

2.1.6 Yoruba Character

There are twenty – five (25) alphabets in Yoruba language. They are:

A B D E Ě F G GB H I J K L M N O Ọ P R S Ș T U W Y.

It is to be noted that letters like, c, q, v, x and z are not among Yoruba alphabets.

How to Pronounce the Alphabets:

The Yoruba Alphabets

Similar Pronunciation In English

A	ah
B	bee
D	dee
E	hay
Ẹ	air
F	fee
G	gee (as in go)
GB	(has no English similarity)
H	hee
I	ee
J	jee
K	key
L	lee
M	mee
N	nee
O	oh
Ọ	or
P	pee
R	ree
S	see
Ẹ	she
T	tee

U	ooh
W	wee
Y	yee

Yoruba Vowels - Awon Faweli Yoruba

Yoruba alphabets contains only seven (7) vowels, namely:

a e ẹ i o ọ u

a (ah) e (hay) ẹ (air) i (ee) o (oh) ọ (or) u (ooh)

In the same vein there are eighteen (18) consonants:

b d f g gb h j k l m n p r s ş t w y

2.2 Review of related works

Combination of Caesar Cipher Modification with Transposition Cipher was proposed by Lubis, Simbolon, Batubara and Sembiring (2017), which involves the combination of substitution cipher with the transposition cipher. it introduces three encryption stages; Caesar modification at first then the generated ciphertext was encrypted with transposition, and last, the result from transposition was encrypted again with the second Caesar modification, similarly at the decryption while the process is reversed. In the modification of Caesar cipher, the shift of letters was not based on the alphabet but based on ASCII table, plaintext gets the addition of characters before encryption and then the new plaintext with the addition of characters will be divided into two, they are plaintext to be encrypted and plaintext are left constantly (no encryption), The third modification is the key that is used dynamically follows the ASCII plaintext value.

A new improved technique of Caesar cipher was proposed by Verma, Gaba and Miglani (2017) in which encryption is not restricted only to alphabets. The diversified Caesar cipher has capability to encrypt symbols, characters as well as digits. A new character value table is also proposed which specifies the positions of symbols, characters and digits in the table. Moreover, matrix concept is used to make the Caesar cipher more complex which creates difficulty for brute force attacker to determine the key value.

The proposed system by Pinheiro and Saraf (2016) involves the recognition of case sensitive character. The encryption and decryption of the plain text is done by making use of the face values and positional values of the corresponding characters as the key. The multistage encryption is imposed on the plain text which indeed improves the security of the plain text and secures it from brute force attack, pattern matching and frequency analysis to an extent.

The scheme proposed by Jain, Dedhia and Patil (2015), contributes in the area of classical cryptography by providing a modified and expanded version for Caesar cipher using knowledge of mathematics and computer science. To increase the strength of this classical encryption technique, the proposed modified algorithm uses the concepts of affine ciphers, transposition ciphers and randomized substitution techniques to create a cipher text which is nearly impossible to decode. It also increases the range of characters which Caesar cipher Algorithm can encrypt by including all ASCII and extended ASCII characters in addition to alphabets. A complex key generation technique which generates two keys from a single key is used to provide enhanced security.

In the course of looking for a better means of improving the Caesar cipher algorithm, Purnama and Rohayani (2015) proposed a scheme which modify the Caesar cipher method to produce cipher-

text that can be read. With the cipher text that can be read, then cryptanalysis not suspicious of the cipher text. Caesar cipher modification is done by replacing the alphabet into two parts, the vocals were replaced with the alphabet vocal too, and the consonant alphabet was replaced with a consonantal alphabet. However, there are some alphabet consonants which are not replaced, and this is because the frequency of the alphabet is rarely used in an Indonesian text. From the test results obtained cipher-text that can be read.

Imran and Abdulkareem (2014) in their scheme, proposed a method that encryption will be implemented on information to make it hard to be readable and secure. The encryption uses three methods in which each letter in the plaintext is replaced by some fixed number of position down the alphabet. The result from this project is a data which is encrypted and be decrypted to its readable form.

Srikantaswamy and Phaneendra (2012), proposed a method to improve Caesar cipher with random number generation technique for key generation operations. The Caesar cipher has been expanded so as to include alphabets, numbers and symbols. The original Caesar cipher was restricted only for alphabets. The key used for Caesar Substitution has been derived using a key Matrix Trace value restricted to Modulo 94. The Matrix elements are generated using recursive random number generation equation, the output of which solely depends on the value of seed selected.

Abraham and Shefiu (2012), introduced some novel improvements to the traditional Caesar cipher algorithm, which completely eliminates its fundamental weaknesses. Firstly, it eliminates spaces from the cipher-text and secondly the process of creating the Cipher-text by the Improved Caesar Cipher (ICC) involves two steps; encryption and scrambling of the letters in the cipher-text, so even if it is decrypted the result would be gibberish.

An algorithm was proposed by Deepak and Parveen (2017) based on ASCII values and binary operations for both encryption and decryption to enhance the security so that the attackers might not easily crack the logic to reach the original message sent via network.

Dey (2016), proposed a new cryptographic technique to exclude the repetitive terms in a message, when it is to be encrypted, so that it becomes almost impossible for a person to retrieve or predict the original message from the encrypted message. The scheme applied SD-AREE cryptographic method to exclude repetitive terms from a message, which is to be encrypted. In SD-AREE method the repetitive bits / characters are removed and there is no trace of any repetition in the message.

A scheme proposed by Al-Hazaimah (2013), was focused on how the security goals can be enhanced via "A New Approach for Complex Encrypting and Decrypting Data" which maintains the security on the communication channels by making it difficult for attackers to predicate a pattern as well as speed of the encryption / decryption scheme. The algorithm for the scheme consists of combination of public key infrastructure for hybrid system and RC6 algorithm for confusion and diffusion operations.

CHAPTER THREE

METHODOLOGY

3.1 Existing system

From the review of literature it can be seen that there are noticeable drawbacks in the previously proposed schemes. 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 text easily detected. It is to be noted that the main purpose of encryption is to make sure that the text is not retrieved by a third party. But when such purpose has not been effective, 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

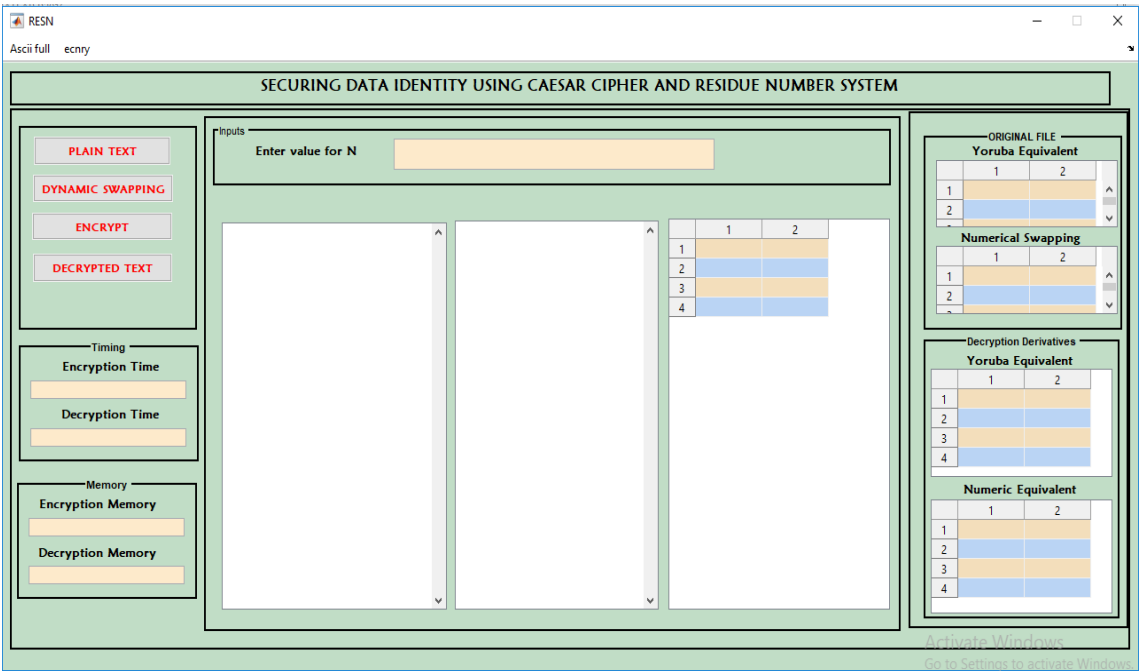
In order to improve the existing system towards providing a better means through which secure communication can be ensured from one end to another, and also to improve on the limitations of the Caesar cipher algorithm, a new scheme was proposed. The scheme comprises of two character tables, the first character table is used for substitution while the second is used to prepare the characters for transposition. In the first character table, the indexes which are expected to be numbers have been changed to Yoruba characters, while the second table is used to assign a numeric value to the Yoruba characters derived after substitution made from the first table.

Thereafter, the concept of residue number system was adopted on the numbers corresponding to each position of the available characters, through computation of forward conversion on the characters for encryption. Hence the message is represented with residues in a matrix form.

The Reverse conversion, which is the decryption process makes use of the Chinese remainder method which first returns each of the numbers to the corresponding initial values before the

system makes use of the character table to map the numbers back to Yoruba character and the Yoruba character eventually getting mapped to the English characters of the original message.

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



Figure

3.1: System interface

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

Thereafter, the user will be required to click the *DYNAMIC SWAPPING* button, which will trigger the system to perform both the first and second substitution. After the substitution, the user will be required to insert a value of N in other to compute the moduli. Once a value has been inserted for N, the final encryption will commence after the *ENCRYPT* button has been clicked.

To decrypt the cipher text, the user only needs to click the *DECRYPT TEXT* button, enabling the system to reverse the process to produce the plain text.

3.2.1 Formation of English to Yoruba character table

The expected mode of communication involves the use of the English alphabets. For the purpose of hiding the existence of the message, a table was formed.

It is understood that there are nineteen Yoruba characters, an handwritten form of these characters was made available by (Falilat, Olufemi, and Odetunji, 2016) in Figure 3.1, through a sample of written Yoruba character that was captured which shows that the Yoruba language has 19 distinct alphabets, although some of these alphabets can take various forms based on their context of use while forming a Yoruba word.

a	à	á	b	d	e	è
é	ẹ	è	f	g	gb	h
í	ì	m	n	o	o'	o'
ò	ó	p	r	s	ş	t
ú	ù	w	y			

Figure 3.2: Samples of captured Handwritten Yoruba Character(Source: Falilat, Olufemi, and Odetunji, 2016)

After representation each of the characters in their various forms, a total number of 40 valid Yoruba characters was generated which is enough to cater for the substitution stage as shown in Table 3.1. The upper row is for the Yoruba characters while the lower row is for the English characters and numbers. The last column in the upper row is used to represent spaces within the text, as z is not a Yoruba character.

a	à	á	b	d	e	è	ẹ	é	è	é	f	g	h	i	ì	í	j	k
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
l	m	n	o	ò	ó	o	p	r	s	ş	t	u	ú	ù	w	y	z	
t	u	v	w	x	y	z		0	1	2	3	4	5	6	7	8	9	

Figure 3.3 : Yoruba Character Table

3.2.2 Formation of Yoruba to Numbers character table

The second character table incorporates the 40 Yoruba characters from the first character table, combined with special key board characters.

a	à	á	b	d	e	è	ẹ	é	è	é	f	g	h	i	ì	í	j	k	l	m	n	o
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
ò	ó	o	p	r	s	ş	t	u	ú	ù	w	y	!	@	#	\$	%	^	&	*	(	)
24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
_	-	+	=	[	]	{	}	\		;	:	,	"	<	>	.	/	?	*	z		
47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67		

Figure 3.4 : Numeric character table

3.2.3 Transposition

The first stage of the transposition involves performing the substitution of each of the English alphabets present in the message to be encrypted, followed by the assignment of numeric value to each of the swapped characters. The forward conversion takes place for each of the characters using the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$. A value n is assigned to the moduli set which serves as the secret key. Using the moduli set, it is expected that for each character, three residue numbers will be computed. From the computed residue numbers, a matrix is formed and this in turn becomes the cipher-text with each character of the original message occupying a row of the matrix.

The transposition is achieved using;

$$x = m_i[x/m_i] + |x|_{m_i}$$

where m is the moduli and x is the number to be transposed.

3.2.4 Encryption process

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

1. Read all English Alphabets
2. Substitute each of the characters with a Yoruba character based on the Yoruba character table
3. Substitute each of the Yoruba characters with a number based on Numeric character table
4. Transpose each of the numbers using forward conversion based on the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$, to generate residues for each of the numbers.

3.2.5 Decryption process

For the decryption process, the residues generated for the encryption process will undergo backward conversion using Chinese Remainder Theorem.

This theorem is used in converting RNS to decimal. Given the residue representation $\{r_1, r_2, \dots, r_n\}$ 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

$$|x|_m = \left| \sum_{j=1}^N \hat{m}_j \left| \frac{r_j}{\hat{m}_j} \right|_{m_j} \right|_m$$

where

$$\hat{m}_j = \frac{M}{m_j}, \quad M = \prod_{j=1}^N m_j$$

and

$$\gcd(m_j, m_k) = 1 \text{ for } j \neq k$$

$$X = |x_1|_{m_1}|\hat{m}_1|_{m_1} + |x_2|_{m_2}|\hat{m}_2|_{m_2} + |x_3|_{m_3}|\hat{m}_3|_{m_3}$$

The procedure for the decryption is stated below;

1. Read all residues, one row at a time.
2. Using the Chinese Remainder Theorem, perform a reverse conversion on the residues based on the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$.
3. Substitute each of the numbers generated from the reverse conversion to Yoruba character based on the numeric character table.
4. Substitute each of the Yoruba characters to English alphabets using the Yoruba character table.

3.3 System Requirement

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

3.4 Choice of Programming Language

The entire application will be developed in Matlab. MATLAB is a high-level language and interactive environment for numerical computation, visualization, and programming. Using MATLAB, you can analyse data, develop algorithms, and create models and standalone applications. The language, tools, and built-in 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 its integrated with Matlab Compiler Runtime for the purpose of Creating applications that will work outside of Matlab integrated environment, thus making room for applications run without installing Matlab.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Text swapping

The first stage of the encryption process is the swapping of each character of the message to a Yoruba using the given character table. The character table comprises of the English alphabet a-z and numbers 0-9, representing the content of the original message.

Furthermore, the Yoruba language consists of mainly 19 alphabets but with variants of the vowel letters. These variants for the purpose of making up the content of the original message were included and the double character “gb” was excluded from the Yoruba characters being used in this research, making the character table having a total number of 36 Yoruba characters. An addition z character which is not a valid Yoruba character was included in the Yoruba section to cater for spaces.

4.2 Encryption of plain text

Given *computer* as the plain text, the first substitution using the Yoruba character table generates the characters Table 4.1.

Table 4.1 Yoruba Swapping (First Substitution)

á	í	g	ì	m	l	d	J
---	---	---	---	---	---	---	---

Thereafter, each of the characters shown in Table 4.1 undergoes another substitution in order to acquire a numeric value as shown in Table 4.2.

Table 4.2 Numeric Swapping (Second substitution)

3	15	13	16	21	20	5	18
---	----	----	----	----	----	---	----

Transposition of the numbers takes place thereafter, using forward conversion based on the chosen moduli set.

Given the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$ the value of n was set to 1 and the computation of the moduli set generates the 3, 4, 5 as our moduli

Such that $m_1 = 3, m_2 = 4$ and $m_3 = 5$, which will be used to determine the RNS representation of each of the numbers in Table 4.2.

For the first character where numeric value is 3,

$$m_1 = 3, \quad [3/3] = 1$$

$$m_2 = 4, \quad [3/4] = 0$$

$$m_3 = 5, \quad [3/5] = 0$$

$$|x|_{m_1} = r_1 = x - m_1[x/m_1] = 3 - 1 \times 3 = 0$$

$$|x|_{m_2} = r_2 = x - m_2[x/m_2] = 3 - 0 \times 4 = 3$$

$$|x|_{m_3} = r_3 = x - m_3[x/m_3] = 3 - 0 \times 5 = 3$$

Therefore the RNS representation of 22 is $\{0, 3, 3\}$

For the next character where numeric value is 15

$$m_1 = 3, \quad [15/3] = 5$$

$$m_2 = 4, \quad [15/4] = 3$$

$$m_3 = 5, \quad [15/5] = 3$$

$$|x|_{m_1} = r_1 = x - m_1[x/m_1] = 15 - 5 \times 3 = 0$$

$$|x|_{m_2} = r_2 = x - m_2[x/m_2] = 15 - 3 \times 4 = 3$$

$$|x|_{m_3} = r_3 = x - m_3[x/m_3] = 15 - 3 \times 5 = 0$$

Therefore the RNS representation of 22 is $\{0, 3, 0\}$

For the next character where numeric value is 13

$$m_1 = 3, \quad [13/3] = 4$$

$$m_2 = 4, \quad [13/4] = 3$$

$$m_3 = 5, \quad [13/5] = 2$$

$$|x|_{m_1} = r_1 = x - m_1[x/m_1] = 13 - 4 \times 3 = 1$$

$$|x|_{m_2} = r_2 = x - m_2[x/m_2] = 13 - 3 \times 4 = 1$$

$$|x|_{m_3} = r_3 = x - m_3[x/m_3] = 13 - 2 \times 5 = 3$$

Therefore the RNS representation of 22 is $\{1, 1, 3\}$

For the next character where numeric value is 16

$$m_1 = 3, \quad [16/3] = 5$$

$$m_2 = 4, \quad [16/4] = 4$$

$$m_3 = 5, \quad [16/5] = 3$$

$$|x|_{m_1} = r_1 = x - m_1[x/m_1] = 16 - 5 \times 3 = 1$$

$$|x|_{m_2} = r_2 = x - m_2[x/m_2] = 16 - 4 \times 4 = 0$$

$$|x|_{m_3} = r_3 = x - m_3[x/m_3] = 16 - 3 \times 5 = 1$$

Therefore the RNS representation of 22 is $\{1, 0, 1\}$

For the next character where numeric value is 21

$$m_1 = 3, \quad [21/3] = 7$$

$$m_2 = 4, \quad [21/4] = 5$$

$$m_3 = 5, \quad [21/5] = 4$$

$$|x|_{m_1} = r_1 = x - m_1[x/m_1] = 21 - 7 \times 3 = 0$$

$$|x|_{m_2} = r_2 = x - m_2[x/m_2] = 21 - 5 \times 4 = 1$$

$$|x|_{m_3} = r_3 = x - m_3[x/m_3] = 21 - 4 \times 5 = 1$$

Therefore the RNS representation of 22 is $\{0, 1, 1\}$

For the next character where numeric value is 20

$$m_1 = 3, \quad [20/3] = 6$$

$$m_2 = 4, \quad [20/4] = 5$$

$$m_3 = 5, \quad [20/5] = 4$$

$$|x|_{m_1} = r_1 = x - m_1[x/m_1] = 20 - 6 \times 3 = 2$$

$$|x|_{m_2} = r_2 = x - m_2[x/m_2] = 20 - 5 \times 4 = 0$$

$$|x|_{m_3} = r_3 = x - m_3[x/m_3] = 20 - 4 \times 5 = 0$$

Therefore the RNS representation of 22 is $\{2, 0, 0\}$

For the next character where numeric value is 5

$$m_1 = 3, \quad [5/3] = 1$$

$$m_2 = 4, \quad [5/4] = 1$$

$$m_3 = 5, \quad [5/5] = 1$$

$$|x|_{m_1} = r_1 = x - m_1[x/m_1] = 5 - 1 \times 3 = 2$$

$$|x|_{m_2} = r_2 = x - m_2[x/m_2] = 5 - 1 \times 4 = 1$$

$$|x|_{m_3} = r_3 = x - m_3[x/m_3] = 5 - 1 \times 5 = 0$$

Therefore the RNS representation of 22 is $\{2, 1, 0\}$

For the next character where numeric value is 18

$$m_1 = 3, \quad [18/3] = 6$$

$$m_2 = 4, \quad [18/4] = 4$$

$$m_3 = 5, \quad [18/5] = 3$$

$$|x|_{m_1} = r_1 = x - m_1[x/m_1] = 18 - 6 \times 3 = 0$$

$$|x|_{m_2} = r_2 = x - m_2[x/m_2] = 18 - 4 \times 4 = 2$$

$$|x|_{m_3} = r_3 = x - m_3[x/m_3] = 18 - 3 \times 5 = 3$$

Therefore the RNS representation of 22 is $\{0, 2, 3\}$

At the end forward conversion, the residue for each character is represented as a row matrix as shown in Table 4.3.

Table 4.3 Residue representation of plain text (cipher text)

0	3	3
0	3	0
1	1	3
1	0	1
0	1	1
2	0	0
2	1	0
0	2	3

The content of Table 4.3 is the cipher text which is to be communicated to the second party.

The encryption process can be seen in Figure 4.1.

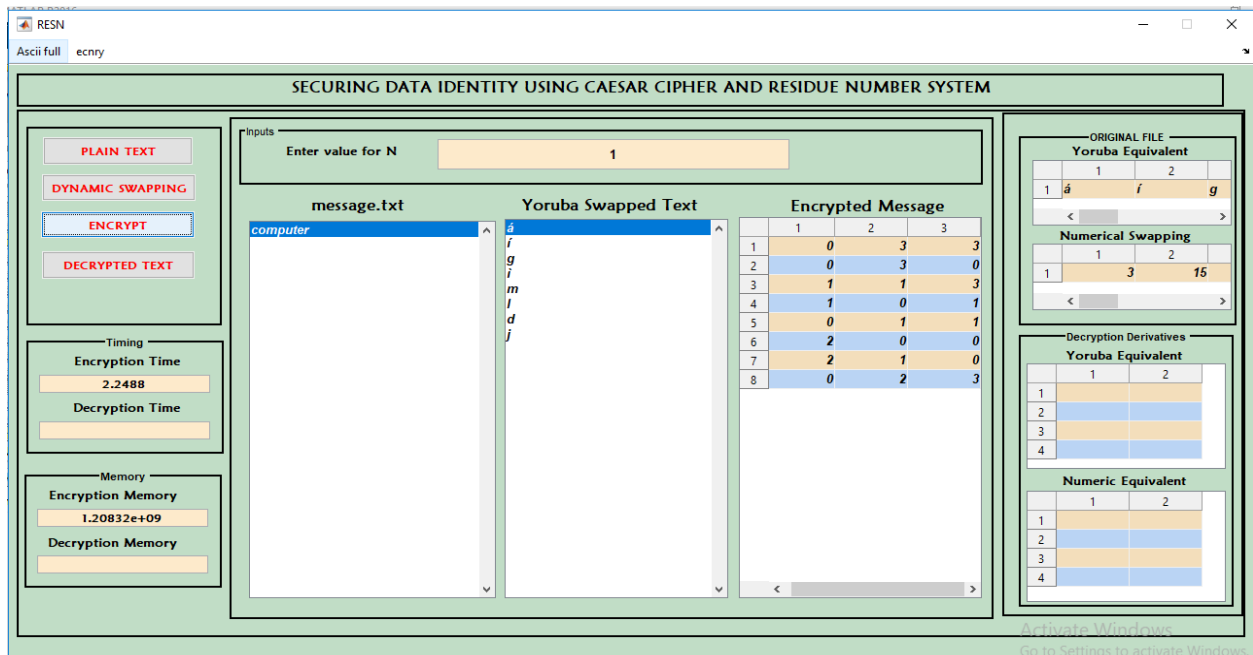


Figure 4.1: Encryption process

4.3 Decryption of cipher text

For the message to be retrieved by the second party, he/she must be aware of the moduli set and the value of n to compute the moduli.

For the reversal of the transposition which was used in the final stage of encryption, the Chinese Remainder Theorem is applied

The CRT is given by

$$|x|_m = \left| \sum_{j=1}^N \hat{m}_j \left| \frac{r_j}{\hat{m}_j} \right|_{m_j} \right|_m$$

Where $\hat{m}_j = \frac{M}{m_j}$, $M = \prod_{j=1}^N m_j$ and

$$\gcd(m_j, m_k) = 1 \text{ for } j \neq k$$

Using the moduli $m_1 = 3, m_2 = 4$ and $m_3 = 5$, the dynamic range is;

$$M = \prod_{i=1}^N m_i = 3 \times 4 \times 5 = 60$$

$$|\hat{m}_1|_{m_1} = |4 \times 5|_3 = |20|_3 = 2 \text{ then } \left| \frac{1}{\hat{m}_1} \right|_3 = 2$$

$$|\hat{m}_2|_{m_2} = |3 \times 5|_4 = |15|_4 = 3, \left| \frac{1}{\hat{m}_2} \right|_4 = 3$$

$$|\hat{m}_3|_{m_3} = |3 \times 4|_5 = |12|_5 = 2, \left| \frac{1}{\hat{m}_3} \right|_5 = 3$$

For the first row of the cipher text with the residue $\{0,3,3\}$ the reverse conversion is;

$$|x|_{60} = |0|_{20} \times 2|_3 + 3|_{15} \times 3|_4 + 3|_{12} \times 3|_5|_{60}$$

$$= |0 + 135 + 108|_{60}$$

$$= |243|_{60} = 3$$

For the next row of the cipher text with the residue $\{0, 3, 0\}$ the reverse conversion is;

$$\begin{aligned} |x|_{60} &= |0|_{20} \times 2|_3 + 3|_{15} \times 3|_4 + 0|_{12} \times 3|_5|_{60} \\ &= |0 + 135 + 0|_{60} \\ &= |135|_{60} = 15 \end{aligned}$$

For the next row of the cipher text with the residue $\{1, 1, 3\}$ the reverse conversion is;

$$\begin{aligned} |x|_{60} &= |1|_{20} \times 2|_3 + 1|_{15} \times 3|_4 + 3|_{12} \times 3|_5|_{60} \\ &= |40 + 45 + 108|_{60} \\ &= |193|_{60} = 13 \end{aligned}$$

For the next row of the cipher text with the residue $\{1, 0, 1\}$ the reverse conversion is;

$$\begin{aligned} |x|_{60} &= |1|_{20} \times 2|_3 + 0|_{15} \times 3|_4 + 1|_{12} \times 3|_5|_{60} \\ &= |40 + 0 + 36|_{60} \\ &= |76|_{60} = 16 \end{aligned}$$

For the next row of the cipher text with the residue $\{0, 1, 1\}$ the reverse conversion is;

$$\begin{aligned} |x|_{60} &= |0|_{20} \times 2|_3 + 1|_{15} \times 3|_4 + 1|_{12} \times 3|_5|_{60} \\ &= |0 + 45 + 36|_{60} \\ &= |81|_{60} = 21 \end{aligned}$$

For the next row of the cipher text with the residue $\{2, 0, 0\}$ the reverse conversion is;

$$|x|_{60} = |2|_{20} \times 2|_3 + 0|_{15} \times 3|_4 + 0|_{12} \times 3|_5|_{60}$$

$$= |80 + 0 + 0|_{60}$$

$$= |80|_{60} = 20$$

For the next row of the cipher text with the residue $\{2, 1, 0\}$ the reverse conversion is;

$$|x|_{60} = |2|20 \times 2|_3 + 1|15 \times 3|_4 + 0|12 \times 3|_5|_{60}$$

$$= |80 + 45 + 0|_{60}$$

$$= |125|_{60} = 5$$

For the next row of the cipher text with the residue $\{0, 2, 3\}$ the reverse conversion is;

$$|x|_{60} = |0|20 \times 2|_3 + 2|15 \times 3|_4 + 3|12 \times 3|_5|_{60}$$

$$= |0 + 90 + 108|_{60}$$

$$= |198|_{60} = 18$$

The computation carried out on each of the residue produces the initial numeric value as shown in Table 4.2.

Thereafter, each number is substituted with corresponding Yoruba characters, before eventually returning it back to the respectively english alphabet for the original message.

The encryption process can be seen in Figure 4.2.

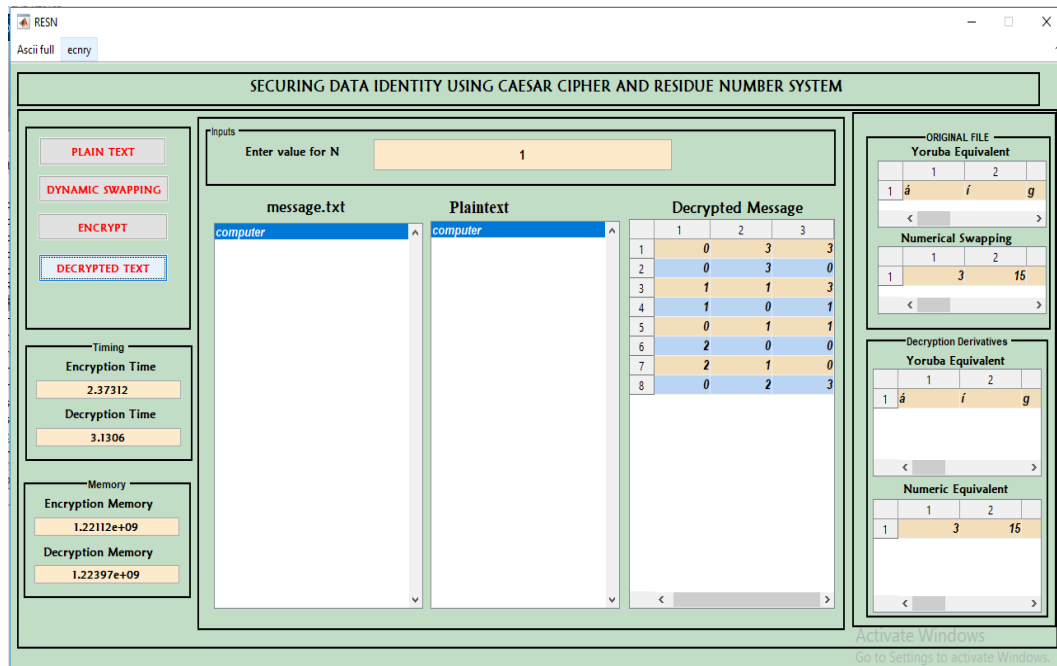


Figure 4.2: Decryption process

4.4 Comparison with existing approaches

After the evaluation of the proposed scheme, as show in Table 4.3, the scheme was compared with the existing scheme to check the success attained using the following;

- i. Diversified Caesar cipher for impeccable security
- ii. Combination of Caesar cipher modification with transposition cipher

In terms of complexity, (i) and (ii) incorporates more complexity compared to the proposed scheme. (i) makes use of five stages in other to produce a cipher text while (ii) makes use of four stages to produce a cipher text compared to the proposed scheme where three stages are involved. In terms of substitution stages, (i) incorporates only one substitution stage with (ii) and proposed scheme but making use of a double substitution stage.

Lastly, when it comes to possibility of attracting the attention of attackers, (i) and (ii) have a very high possibility rate due to the representation of their cipher text in a gibberish form while the

proposed scheme has a low possibility of attack as it is difficult for an attacker to believe that a message will be a mathematical matrix since a matrix is a valid mathematical representation.

Table 4.4: Comparison with existing methods

PARAMETERS	DIVERSIFIED CAESAR CIPHER FOR IMPECCABLE SECURITY	COMBINATION OF CAESAR CIPHER MODIFICATION WITH TRANSPOSITION CIPHER	PROPOSED SCHEME
COMPLEXITY	MORE	MORE	LESS
STAGES OF SUBSTITUTION	1	2	2
POSSIBILITY OF ATTACK	HIGH	HIGH	LOW
OUTPUT FILE	GIBBERISH	GIBBERISH	MATRIX

CHAPTER FIVE

SUMMARY AND CONCLUSION

5.1 Summary

This research proposed the use of Caesar cipher algorithm and Residue number system to encrypt and decrypt the characters based on a character table and RNS (Forward and Reverse Conversion). The characters are first swapped into Yoruba language before assigning numeric value to each of the number, which later comes a residue which is transmitted to the receiver. The receiver on must have earlier received the moduli set being used, the value of n , as well as the position of spaces within the characters to be able to decrypt the message accurately.

5.2 Conclusion

The proposed scheme has provided a better means of sending messages securely over a network while the message maintains a high level of secrecy, proving to outperform its predecessors in terms of secrecy attached to the message, the stages involved in the encryption process and the lesser number of complexity while attaining a maximal result. The result of cipher-text produced in the proposed scheme is very complex and to solve it there are so many possibilities that should be tried, such as: finding the sequence of algorithms used, determining the moduli set, determining the value of n for the moduli set and also determining the first and second character table.

5.3 Major contributions to knowledge

The major contributions of this thesis are presented below. The scheme produced a system that;

- i. provides a better security with the implementation of a double layer substitution.
- ii. makes the cipher text maintain a decent representation, making it hard for an attacker to suspect such data has been encrypted.

5.4 Future work

For the purpose of further enhancement and providing a more efficient way of sending message securely using this approach, the following are the expected future works;

- i. An increase in the length of the moduli set, to make the encrypted text harder to crack by an attacker. This will increase the number of possible combinations an attacker will have to make in order to decrypt the message.
- ii. Use of Mixed Radix conversion with Caesar cipher to evaluate the efficiency of this scheme in terms of speed of computation in the reverse process.
- iii. Use of New CRT with Caesar cipher which is expected to have an improved computational speed over the traditional CRT
- iv. Combination of Caesar cipher and other ciphers for the first and second character table with the aim of increasing the level of security at the substitution stage.

REFERENCES

- Abraham, O., & Shefiu, G. O. (2012). An improved Caesar Cipher (ICC) algorithm.
International Journal Of Engineering Science & Advanced Technology, 2(5), 1198–1202.
- Alhassan S & Gbolagade K.A (2013). Enhancement of the Security of a Digital Image using the Moduli Set $2n-1, 2n, 2n+1$. *International Journal of Advanced Research in Computer Engineering & Technology*, 2(7).
- Al-Hazaimeh, O. M. (2013). A new approach for complex encrypting and decrypting data.
International Journal of Computer Networks & Communications, 5(2), 95-103.
- 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.
- Ayushi. (2010). A Symmetric Key Cryptographic Algorithm. *International Journal of Computer Applications*, 1(15), 1-4.
- Badoni, V. D., & Jain, M. A. (2014). Chip Implementation of Text Encryption and Decryption Algorithms. *Journal of Computer Engineering*, 56-61.
- Barakat, M., Eder, C., & Hanke, T. (2018). An Introduction to Cryptography.
- Deepak, & Parveen. (2017). Modern Encryption and Decryption Algorithm based on ASCII Value and Binary Operations. *International Journal of Computer Applications*, 172(1), 30-34.

Delfs .H and Knebl .H (2015). Introduction to Cryptography. *Springer Berlin Heidelberg*, 49
106.

Dey, S. (2016). A New Modified Caesar Cipher Cryptographic Method Along with Bit-
Manipulation to Exclude Repetition from a Message to be Encrypted. *Global Journal of
Computer Science and Technology Network, Web & Security*, 90-94.

Doshi, R., Jain, P., & Gupta, L. (2012). Steganography and Its Applications in Security .
International Journal of Modern Engineering Research, 4634-4638.

Ajao. J., Olufemi, O. D., & Odetunji, O. O. (2018). Offline Yoruba Handwritten Character
Recognition using Freeman Chain Code and K-Nearest Neighbor Classifier. *Jurnal
Teknologi dan Sistem Komputer*, 6(10), 129-134

Gbolagade K.A (2009). A Shorter Algorithm for Efficient Residue to Decimal Conversion,
Advances in Computer Science and Engineering, 3(2) ,147-156.

Gbolagade K.A. and Cotofana S.D (2009). An efficient rns to binary converter using the moduli
set $\{2n+1, 2n, 2n-1\}$. *Conference on Design of Circuits and Integrated Systems*.

He .D and Zeadally S. (2015). An analysis of RFID authentication schemes for internet of things
in healthcare environment using elliptic curve cryptography. *IEEE internet of things
journal*, 2(1), 72-83.

Hercigonja, Z., & gimnazija, D. (2016). Comparative analysis of cryptographic algorithms.
International Journal of digital technology & economy, 127-134.

Imran, E. I., & abdulkareem, F. a. (2014). Enhancement Caesar Cipher for better security.
Journal of Computer Engineering, 1 - 5 .

Jain, A., Dedhia, R., & Patil, A. (2015). Enhancing the security of Caesar cipher substitution method using randomized approach for more secure communication. *International Journal of Computer Applications*, 129(13), 6 - 11.

Lubis, F. I., Simbolon, H. F., Batubara, T. P., & Sembiring, R. W. (2017). Combination of Caesar Cipher Modification with Transposition Cipher. *Advances in Science, Technology and Engineering Systems Journal*, 2(5), 22-25.

Mohammad A. A, Abdelfatah A.T and Omaira N.A (2014). Cryptography Based Authentication Methods. *World Congress on Engineering and Computer Science*, 1(1)Vol, 6-8.

Pinheiro, G., & Saraf, S. (2016). Improved Caesar Cipher Algorithm using multistage encryption. *International Journal of Computer Science and Technology*, 117 - 119.

Purnama, B., & Rohayani, H. (2015). A new modified Caesar cipher cryptography method with legible Cipher-text from a message to be encrypted. *International Conference on Computer Science and Computational Intelligence*, 195 - 204).

Siewobr H, Gbolagade K.A. (2012). An Area Efficient RNS-to-Binary Converter for the Moduli Set $\{2^n, 2^{2n+1}-1, 2^{2n}-1\}$. *International Conference on Adaptive Science and Technology*, 104-107.

Siewobr H, Gbolagade K.A, and Cotofana S.D (2014). An Efficient Residue-to-Binary Converter for the New Moduli Set $\{2^{n/2} \pm 1, 2^{2n+1}, 2^{n+1}\}$. *International Symposium on Integrated Circuits*.

Srikantaswamy, S. G., & Phaneendra, D. H. (2012). Improved Caesar Cipher with random number generation technique and multistage encryption. *International Journal on Cryptography and Information Security*, 294), 39 - 49.

Verma, P., Gaba, G. S., & Miglani, R. (2017). Diversified Caesar Cipher for Impeccable Security. *International Journal of Security and Its Application*, 11(3), 33 - 40.

APPENDIX

```
function varargout = RESN(varargin)
% RESN MATLAB code for RESN.fig
%     RESN, by itself, creates a new RESN or raises the existing
%     singleton*.
%
%     H = RESN returns the handle to a new RESN or the handle to
%     the existing singleton*.
%
%     RESN('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in RESN.M with the given input arguments.
%
%     RESN('Property','Value',...) creates a new RESN or raises the
%     existing sin
gleton*. Starting from the left, property value pairs are
%     applied to the GUI before RESN_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to RESN_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 RESN

% Last Modified by GUIDE v2.5 19-Mar-2019 05:37:54

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @RESN_OpeningFcn, ...
                  'gui_OutputFcn',  @RESN_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 RESN is made visible.
function RESN_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 RESN (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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

% --- Outputs from this function are returned to the command line.
function varargout = RESN_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 filename2 file_size A user
user=getenv('username');
cd(strcat(['C:\Users\'',user,'\Documents\MATLAB\RSNN']));
[filename2, pathname] = uigetfile({
    '*.txt','text files (*.txt)'; ...
    '*.*', 'All Files (*.*)'}, ...
    'Pick a file');
%set(handles.edit1,'string',filename2);
set(handles.text4,'string',filename2);
fileID = fopen(filename2,'r');
formatSpec = '%s';
A = fread(fileID, '*char').';
A=lower(A);
sizer=length(A);
set(handles.listbox2,'string',A);
get_file_lenght;
kb=file_size/1024;
% set(handles.edit4,'string',kb);

% --- 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 searchyl mod1 mod2 mod3 ff
h = waitbar(0,'Peforming Decryption,plz wait...');

```

```

steps = 1000;
for step = 1:steps
    % computations take place here
    waitbar(step / steps)
end
modmul=mod1*mod2*mod3;
dmod1=modmul/mod1;
dmod2=modmul/mod2;
dmod3=modmul/mod3;
%% find rem with 1 set 1
x=1;
for i = 1:100
    dody1=(dmod1*x);
    remy1=rem(dody1,mod1);
    if(remy1==1)
        break
    end
    x=x+1;
end
divx1=x;

%% find rem with 1 set 2
x=1;
for i = 1:100
    dody2=(dmod2*x);
    remy2=rem(dody2,mod2);
    if(remy2==1)
        break
    end
    x=x+1;
end
divx2=x;

%% find rem with 1 set 3
x=1;
for i = 1:100
    dody3=(dmod3*x);
    remy3=rem(dody3,mod3);
    if(remy3==1)
        break
    end
    x=x+1;
end
divx3=x;
%% backward conversion
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);
%% pick up equivalent yoruba
[my,ny]=size(revd);

```

```

count=1
for k = 1 : ny
    test=revd(count);
    [ny,ty,ry]=xlsread('a.xlsx','b');

    yr{k}=ty(test);
    % counter=counter+1;
    count=count+1;
end
showyr=cell2table(yr)
showyr=table2array(showyr);
set(handles.uitable4,'Data',showyr);
%%select equivalent English
nsize=size(yr);
nsize=nsize(2)
counted=1
for i = 1 : nsize
    pickrev=yr(counted);
    [no,to,ro]=xlsread('a.xlsx','a');
    yoru=to(1,:);
    enge=to(2,:);
    searchyo(:,counted)=find(strcmp(yoru,pickrev{1,1})==1)
    valuesyo{i}=yoru(strcmp(yoru,:),pickrev{1,1})==1);
    counted=counted+1;
end
set(handles.uitable5,'Data',searchyo);
%% finial
tim=tic;
am=memory;
nsizer=size(searchyo);
nsizer=nsizer(2)
counte=1
for i = 1 : nsizer
    pickeng=searchyo(counte);
    [no,to,ro]=xlsread('a.xlsx','a');
    eng=to(2,:);
    equv{i}=eng(pickeng);
    counte=counte+1;
end
eshow=cell2table(equv)
eshow=table2array(eshow);
eshow=char(eshow);
timy=toc(tim);
memor=am.MemUsedMATLAB
set(handles.edit9,'string',timy);
set(handles.edit17,'string',memor);
% set(handles.uitable1,'Data',g);
set(handles.listbox3,'string',eshow)
set(handles.text19,'string','Decrypted Message')
close(h);
% --- 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 imgf filename A nn searchyl
set(handles.listbox3,'string','');

```

```

sizer=length(A);
count=1;
for i = 1 : sizer
    test=A(count);
    [n,t,r]=xlsread('a.xlsx','a');
    English=t(2,:);
    search{i}=find(strcmp(english,test)==1);
    values=English(strcmp(English(:),test)==1);
    count=count+1;
end
h = waitbar(0,'Swapping yoruba index');
steps = 1000;
for step = 1:steps
    % computations take place here
    waitbar(step / steps)
end
%%selection of yoruba index
tal=t(1,:);
[m,n]=size(search);
counter=1;
nn={' '};
for j = 1 : n
    yind= search(counter);
    yind=cell2mat(yind);
    if isempty(yind);

yind=37;
end
pickv=tal(yind);
nn{j}=pickv;
counter=counter+1;
end
%set(handles.listbox2,'string',char(nn));
nshow=cell2table(nn)
g=table2array(nshow);
set(handles.uitable1,'Data',g);
set(handles.listbox3,'string',g');
set(handles.text18,'string','Yoruba Swapped Text');
close(h);
%% select numerical wsapping
h1 = waitbar(0,'Swapping numerical equivalent');
steps = 1000;
for step = 1:steps
    % computations take place here
    waitbar(step / steps)
end
[mmm,nnn]=size(nn);
county=1;
for k = 1: nnn
    [n1, t1, r1]=xlsread('a.xlsx','b');
    picky=nn(county);
    searchy{k}=find(strcmp(t1,picky{1,1})==1);
    county=county+1;
end
ff=67;
searchy1=cell2mat(searchy);
searchy1(searchy1==ff) = [];

```

```

set(handles.uitable2,'Data',searchy1);
close(h1)
% timer=toc(timy);
% close(h);
%msgbox('Encryption Finished');
% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject      handle to listbox1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
% Hints: contents = cellstr(get(hObject,'String')) returns listbox1 contents
as cell array
%      contents{get(hObject,'Value')} returns selected item from listbox1
% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject      handle to listbox1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called
% Hint: listbox 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 edit1_Callback(hObject, eventdata, handles)
% hObject      handle to edit1 (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 edit1 as text
%      str2double(get(hObject,'String')) returns contents of edit1 as a
double
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit1 (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 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
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, 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
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
% --- Executes on selection change in listbox2.
function listbox2_Callback(hObject, eventdata, handles)
% hObject      handle to listbox2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
% Hints: contents = cellstr(get(hObject,'String')) returns listbox2 contents
as cell array
%           contents{get(hObject,'Value')} returns selected item from listbox2
% --- Executes during object creation, after setting all properties.
function listbox2_CreateFcn(hObject, eventdata, handles)
% hObject      handle to listbox2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called
% Hint: listbox 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 edit6_Callback(hObject, eventdata, handles)
% hObject      handle to edit6 (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 edit6 as text
%           str2double(get(hObject,'String')) returns contents of edit6 as a
double
% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit6 (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

```

```
end
```

```
% --- Executes on selection change in popupmenu1.  
function popupmenu1_Callback(hObject, eventdata, handles)  
% hObject    handle to popupmenu1 (see GCBO)  
% eventdata  reserved - to be defined in a future version of MATLAB  
% handles    structure with handles and user data (see GUIDATA)  
% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1  
contents as cell array  
%          contents{get(hObject,'Value')} returns selected item from popupmenu1
```

```
global dt;  
val=get(hObject,'Value');
```

```
switch val
```

```
case 1
```

```
dt=0
```

```
case 2
```

```
dt=1
```

```
case 3
```

```
dt=2
```

```
case 4
```

```
dt=3
```

```
    otherwise
```

```
        dt=0;
```

```
end
```

```
% --- Executes during object creation, after setting all properties.
```

```
function popupmenu1_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to popupmenu1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: popupmenu 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 checkbox1.
```

```
function checkbox1_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to checkbox1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of checkbox1
```

```
global crt
```

```
if (get(hObject,'Value') == get(hObject,'Max'))
```

```
    crt=1;
```

```
else
```

```
    crt='';
```

```
end
```

```
function edit9_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit9 (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 edit9 as text
%         str2double(get(hObject,'String')) returns contents of edit9 as a
double
% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (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
% --- Executes on selection change in listbox3.
function listbox3_Callback(hObject, eventdata, handles)
% hObject    handle to listbox3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = cellstr(get(hObject,'String')) returns listbox3 contents
as cell array
%         contents{get(hObject,'Value')} returns selected item from listbox3
% --- Executes during object creation, after setting all properties.
function listbox3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: listbox 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 edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (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 edit11 as text

```

```

%         str2double(get(hObject,'String')) returns contents of edit11 as a
double
% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (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 edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (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 edit12 as text
%         str2double(get(hObject,'String')) returns contents of edit12 as a
double
% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (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 edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (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 edit14 as text
%         str2double(get(hObject,'String')) returns contents of edit14 as a
double
% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (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 edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (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 edit15 as text
%         str2double(get(hObject,'String')) returns contents of edit15 as a
double

```

```

% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (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 when selected object is changed in uibuttongroup2.
function uibuttongroup2_SelectionChangedFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uibuttongroup2
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global model;
value= get(eventdata.NewValue,'Tag')
switch value
    case 'seq'
        model=1
    case 'radiobutton3'
        model=2
    case 'radiobutton4'
        model=3
end
% -----
function Ascii_full_Callback(hObject, eventdata, handles)
% hObject    handle to Ascii_full (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
binterface;
% -----
function ecnry_Callback(hObject, eventdata, handles)
% hObject    handle to ecnry (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
binterface2;
% --- Executes on button press in pushbutton16.
function pushbutton16_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global searchy1 mod1 mod2 mod3 ff
p=get(handles.edit11,'string');
p=str2num(p);
if isempty(p);
    errordlg('Please select value for n');
    set(handles.edit11,'BackgroundColor','r');
    return
end
text=get(handles.listbox2,'string');
if isempty(text);
    errordlg('Please load in a plain text');
    set(handles.listbox2,'BackgroundColor','r');
    return
end

```

```

end
set(handles.listbox2,'BackgroundColor','w');
timy=tic;
am=memory;
h = waitbar(0,'Peforming Encryption with the inserted range and modulli
set');
steps = 1000;
for step = 1:steps
    % computations take place here
    waitbar(step / steps)
end
na=p;
mod1=(2^(2*na))-1;
mod2=(2^(2*na));
mod3=(2^(2*na)+1);
% forward conversion
f1=mod(searchy1,mod1);
f2=mod(searchy1,mod2);
f3=mod(searchy1,mod3);
ff=[f1;f2;f3];
ff=ff';
tim=toc(timy);
memor=am.MemUsedMATLAB
set(handles.edit8,'string',tim);
set(handles.edit16,'string',memor);
close(h);
msgbox('Encryption Finished');
set(handles.uitable3,'Data',ff);
set(handles.text19,'String','Encrypted Message');
% --- Executes on selection change in listbox4.
function listbox4_Callback(hObject, eventdata, handles)
% hObject    handle to listbox4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% Hints: contents = cellstr(get(hObject,'String')) returns listbox4 contents
as cell array
%         contents{get(hObject,'Value')} returns selected item from listbox4
% --- Executes during object creation, after setting all properties.
function listbox4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns called
% Hint: listbox 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 edit16_Callback(hObject, eventdata, handles)
% hObject    handle to edit16 (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 edit16 as text
%         str2double(get(hObject,'String')) returns contents of edit16 as a
double
% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)

```

```

% hObject      handle to edit16 (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 edit17_Callback(hObject, eventdata, handles)
% hObject      handle to edit17 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% h
andles        structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit17 as text
%        str2double(get(hObject,'String')) returns contents of edit17 as a
double
% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit17 (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

```