MODIFIED VERTICALLY SCRAMBLED CAESER CIPHER METHOD FOR ENHANCED DATA SECURITY

By

KHADIJA ABDULKADIR

DEPARTMENT OF COMPUTER SCIENCE

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA.

JANUARY, 2017

MODIFIED VERTICALLY SCRAMBLED CAESER CIPHER METHOD FOR ENHANCED DATA SECURITY

By

Khadija, ABDULKADIR

MSC./SCI/ 20832/12-13

A DISSERTATION SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES, AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD

OF

MASTER OF SCIENCE DEGREE IN COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF PHYSICAL SCIENCE

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA

JANUARY, 2017

DECLARATION

I declare that the work in this Dissertation entitled "Modified Vertically Scrambled Caesar Cipher Method for Enhanced Data Security" has been carried out by me in the Department of Computer Science under the supervision of Professor S.B. Junaidu and Dr. S.E. Abdullahi. The information derived from the literature has been duly acknowledged in the text and a list of references provided. No part of this dissertation was previously presented for another degree or diploma at this or any other institution.

Abdulkadir Khadija	_____	_____
Name of Student	Signature	Date

CERTIFICATION

This dissertation entitled "Modified Vertically Scrambled Caesar Cipher Method for Enhanced Data Security" by Abdulkadir Khadija (Msc/Sci/20832/2012-2013) meets the regulations governing the award of the degree of Master of Science of the Ahmadu Bello University, and is approved for its contribution to knowledge and literary presentation.

Prof. Sahalu B. Junaidu	_____	_____
Chairman, Supervisory Committee	Signature	Date
Dr. S.E. Abdullahi	_____	_____
Member, Supervisory Committee	Signature	Date
Prof. A.A. Tijjani	_____	_____
Head of Department	Signature	Date
Prof. K. Bala	_____	_____
Dean, School of Postgraduate Studies	Signature	Date

DEDICATION

This dissertation is dedicated to my lovely parent Alhaji Zakariyyah Abdulkadir and Malama Fadila Aminu for their love for education and inspiration.

ACKNOWLEDGEMENT

I am greatly indebted to many who have in one way or the other contributed to whatever is the success of this dissertation.

First of all, I am very much thankful to Allahu Subhanahu Wa Ta'ala for His blessings and showing me the right direction to reach here.

No words to express my gratitude and pleasure towards the guidance and help I have received from my supervisors, Professor Sahalu Balarabe Junaidu and Dr. S.E Abdullahi whose constant guidance and encouragement made the completion of my MSc. dissertation possible.

My appreciation also goes to all the Lecturers of the Department of Mathematics for instilling in me the basic knowledge about the field that greatly benefitted me while carrying out the project and achieving the goal.

My profound gratitude goes to my parents, Alhaji Zakariyyah Abdulkadir and Malama Fadila Aminu for their moral, material and financial support for which my mere expression of thanks does not suffice.

I feel sincerely appreciative to my siblings in person of Abdulkadir, Ameenah, Fateemah, Maryam, Sulaiman, Musa, Bilkis, Dawood, Ibrahim, Asma'u and my little kids Juwairiyyah (Jumaana), Baby Khadija (Mazeedatul- Khair) and Fateemah (Amrah).

It is with pleasure that I recognize the assistance and help accorded by my brother in-law Abdussami'i O. Sunusi.

This dissertation would not have been possible without the support of Abdulhakeem Ibrahim, I owe my sincerest gratitude to him for his endless support even with his tight schedules.

I remain thankful to my friends, Ocheja Mariam, Habu Saratu, Muhammad Aliyu Kufena, Umar Rabiatu Lawal, Kawu Kabiru for their friendly encouragement.

Finally to my fellow colleagues in Department of Mathematics Ahmadu Bello University, Zaria who are too numerous to mention for their love throughout the course of this program, thank you all.

ABSTRACT

Cryptography is the art and science of converting readable messages into non-readable form. The two techniques for converting data into non-readable form are transposition technique and substitution technique. Caesar cipher is an example of a substitution method. Although Caesar cipher is the simplest type of cipher, it suffers from the limitation of character repetition which makes it prone to plain text or brute force attack. Many researchers have developed techniques to address this problem by improving on key generation or combining two or more algorithms. Despite these efforts, the problem of character repetition still exists. To overcome this limitation, this work proposed a new data security solution using the modified Caesar cipher that eliminates character repetitions. The proposed system is expanded to include alphabets, numbers and symbols. The proposed method is resistant against brute-force attack with 256 keys. It also employs more confusion and diffusion to make the transmission of messages more secure and robust. The algorithm was evaluated based on frequency of character occurrence, character length, running time and security. The proposed system produced cipher texts with entropy value up to 60% higher and index of coincidence 11% lower than that of Vertically Scrambled Caesar Cipher Method. These enhancements are obtained at the expense of acceptable additional computation overhead (running time) of up to 15% in the worst case.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

ABBREVIATIONS AND SYMBOLS

ARPA: Advanced Research Projects Agency

ASCII: American Standard Code for Information Interchange

ATM's: Automatic Teller Machines

GPS: Global Positioning System

H(X): Entropy of 'x'

I.C: Index of Coincidence

IDE: Integrated Development Environment

K1P1: (knit 1, purl 1)

MOD: Modulus

MULTICS: Multiplexed Information and Computing Service

MIT: Massachusetts Institute of Technology

CHAPTER ONE: INTRODUCTION

This chapter discusses the introductory part of this dissertation, which includes background of the study, research motivation, problem statement, research objectives, and finally the dissertation outline.

1.1 Background to the Study

People live in an information society where they rely on the exchange and processing of information. Evidence of this evolution is the rapid growth of communication networks, and a trend towards complex software applications with strong security requirements. Examples of these are the increasing use of portable devices and wireless networks, communication with friends and colleagues via e-mail and chat, the launch of (interactive) digital television and other media platforms (e.g., iTunes), on-line banking and purchase of goods and services, online gaming, GPS navigation, professional and social networks (e.g., LinkedIn and Facebook) and many more (Wyseur, 2009). In one way or another, these new trends affect daily activities in many ways, at home and in our professional life. On the downside however, people have become increasingly dependent on the information infrastructure that empower information society, and hence potentially vulnerable to attacks on them. In recent years, this has been illustrated by attacks on Internet servers, credit card fraud, hacking of banking applications and on-line games, cell phones and TV set-top boxes, phishing, privacy violation, botnet threats, and so forth (Wyseur, 2009).

In order to support information society for the next years and take advantage of the opportunities that it enables, the need for trustworthy information infrastructure is growing. The trend towards complex software applications with strong security requirements

increasingly demands for qualitative protection technologies. One prominent building block to information security is cryptography.

The science of cryptography is not as enigmatic as one might think. A variety of cryptographic techniques are used regularly in everyday life. For example, when you open your newspaper to the entertainment section you will find *Daily Cryptogram*, which is a word puzzle that involves unscrambling letters to find a hidden message. Also, although it is a dying art, many secretaries still use shorthand, or stenography, an abbreviated, symbolic writing method, to take rapid dictation. A form of cryptography is used even in knitting patterns, where directions are written in a coded form, in such patterns as K1P1 (knit 1, purl 1) that only an initiate can understand. These examples illustrate one important application of cryptography; the efficient and rapid transmittal of information, but cryptography also protects and verifies data transmitted via information systems (Micheal and Herbert, 2011).

The science of encryption, known as cryptology, encompasses *cryptography* and *cryptanalysis* as shown in figure 1.1. Cryptography, which comes from the Greek words *kryptos*, meaning "hidden," and *graphein*, meaning "to write," is the process of making and using codes to secure the transmission of information. Cryptanalysis is the process of obtaining the original message (called the plain text) from an encrypted message (called the cipher text) without knowing the algorithms and keys used to perform the encryption. Encryption is the process of converting an original message into a form that is unreadable to unauthorized individuals; that is, to anyone without the tools to convert the encrypted message back to its original format. Decryption is the process of converting the cipher text

message back into plaintext so that it can be readily understood (Micheal and Herbert, 2012).

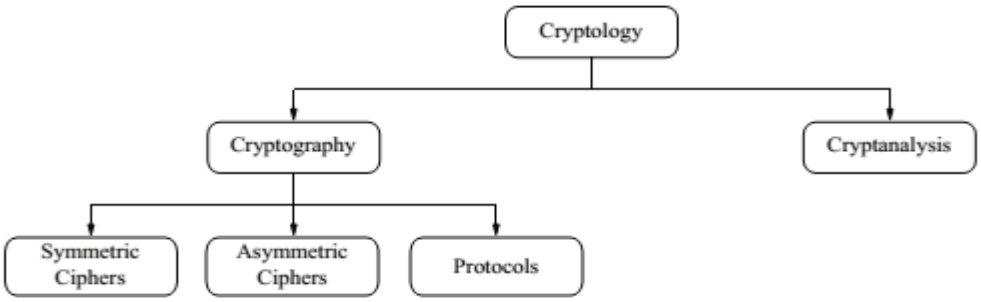

Figure 1.1: Overview of Cryptology (Christof and Pelzi, 2010)

Cryptography is the art and science of protecting information from undesirable individuals by converting it into a form non-recognizable by its attackers while stored and transmitted (Hamdan *et al*., 2010).

The aim of cryptography is to render a message incomprehensible to an unauthorized reader.

In the field of cryptography there exist several techniques for encryption and decryption. These techniques can be generally classified into two major groups, conventional (symmetric) and public key (asymmetric) cryptography as shown in figure 1.2. Conventional encryption is marked by its usage of single key for both the process of encryption and decryption whereas in public key cryptography separate keys are used. Further on conventional techniques are further broken in to Classical and Modern techniques (Anupama, 2014).

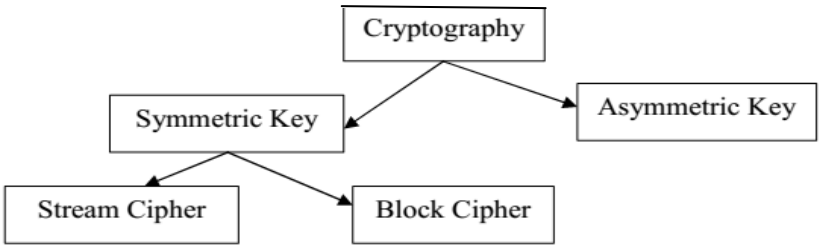

Figure 1.2: Types of Cryptography (Micheal and Herbert, 2011)

The two requirements for secure use of conventional encryption as shown in figure 1.3 are:

a. A strong encryption algorithm in which an opponent should be unable to decrypt cipher text or discover the key even if he or she is in possession of a number of cipher texts together with the plaintext that produced each cipher text.

b. Secret key which both sender and receiver must have obtained copies in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable (William, 2011).

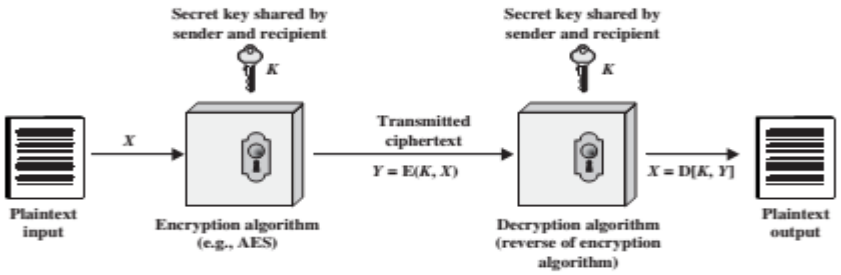

Figure 1.3: Simplified Model of Symmetric Encryption (William, 2011)

A cipher (or cypher) is a pair of algorithms which creates the encryption and the reversing decryption. The detailed operation of a cipher is controlled both by the algorithm and, in each instance, by a key. This is a secret parameter for a specific message exchange context.

Keys are important, as ciphers without variable keys are trivially breakable and therefore less than useful for most purposes.

In Cryptography, the Caesar cipher is one of the most widely known encryption technique. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. The encryption is represented using modular arithmetic by first transforming the letters into numbers, according to the scheme A=0, B=1…..Z=25.

Encryption of a letter `x` by a shift `n` can be described mathematically as `En(x) = (x+n) mod 26`. Decryption is performed as `Dn(x) = (x-n) mod 26`. If it is known that a given cipher text is Caesar cipher, then brute-force cryptanalysis is easily performed (William, 2011).

1.2 Problem Statement

Caesar cipher is an example of a substitution method. Although Caesar cipher is the simplest type of cipher, it suffers from the limitation of character repetition which makes it prone to plain text or brute force attack. Many researchers have developed techniques to address this problem by improving on key generation or combining two or more algorithms. Despite these efforts, the problem of character repetition still exists. This research aims at solving the problem of frequency cryptanalysis which occurs as a result of character repetition by developing a strong encryption method that will exclude the repetitive characters such that there will be no trace of character repetition.

1.3 Research Motivation

Hackers nowadays are always trying to break the cryptographic methods or retrieve keys by different means and one of such methods is the process of inclusion of repetitive texts or characters in a message and then encrypt it to study the behavior of the method and retrieve the key which is needed for decryption (Kumar and Kumar, 2012). Due to the increase in cryptanalysis attack, most existing old methods of security often fail to overcome it. Hence, as an ancient quote says it's true that "Old is Gold" those methods help in developing new ideas to sort out problems, thus leads to discovery of a new horizon (Praloy and Prasenjet, 2013). Most of the powerful encryption techniques are being discovered from the old ones. Little modifications or combinations of two or more algorithms make a lot of progress which leads to improvements.

1.4 Research Aim and Objectives

The research work aims at designing a modified vertically scrambled Caesar cipher method for enhanced data security.

The specific objectives are to:

a. design a modified vertically scrambled Caesar Cipher algorithm that avoids character repetition;
b. implement the algorithm;
c. evaluate performance of the algorithm with respect to frequency of character occurrence, character length, Running Time and Security against the work of Asiya and Ruba, (2015).

1.5 Research Methodology

The following steps were taken in the course of this research.

a. Intensive review and analysis of the vertically scrambled Caesar Cipher algorithm to determine how character repetitions are generated.
b. Modification of the algorithm to eliminate character repetitions by considering ASCII characters and index positioning.
c. Implement the modified algorithm using open source technologies such as Java programming Language and Netbeans Integrated Development Environment (IDE).
d. Evaluate and compare the algorithm with the work of Asiya and Ruba, (2015) based on the following parameters: Frequency of character occurrence, character length, Running Time and Security
e. Presents the results of the performance evaluation using tables and graphs.

1.6 Organization of the Dissertation

The dissertation is structured as follows:

Chapter Two: This chapter reviews the relevant literature that covers information security. An overview of the cryptography is stated in detail. Categories of cryptography algorithms are described. Concepts related to the proposed research work and also review of related works and state-of-art.

Chapter Three: The methodology of the proposed system is demonstrated. The modification aspects of our proposed algorithm are presented in detail.

Chapter Four: The research implementation details of the system with screen shots showing the graphical user interface are displayed in this chapter. This chapter also presents all the experiments conducted to evaluate the proposed system and results of comparative analysis obtained from the research.

Chapter Five: This chapter states the summary of the study and presents the conclusion, recommendation and future work.

CHAPTER TWO: LITERATURE REVIEW

This section reviews several important concepts that are related to the proposed research work and also reviews related works and technologies in our proposed area of study.

2.1 Overview of Information Security

The Advanced Research Projects Agency (ARPA) formed a task force to study the process of securing classified information systems In June, 1967. The Task Force was assembled in October, 1967 and met regularly to formulate recommendations, which ultimately became the contents of the Rand Report R-609 (Hopper, 2010).

The Rand Report R-609 was the first widely recognized published document to identify the role of management and policy issues in computer security. It noted that the wide utilization of networking components in information systems in the military introduced security risks that could not be mitigated by the routine practices then used to secure these systems (Warner and Bernhard, 2002). This paper signaled a pivotal moment in computer security history when the scope of computer security expanded significantly from the safety of physical locations and hardware to include the following:

a. Securing the data.
b. Limiting random and unauthorized access to that data.
c. Involving personnel from multiple levels of the organization in matters pertaining to information security.

Information or data is the wealth of any organization therefore security issues are top priority to an organization dealing with confidential data (Micheal and Herbert, 2011).

Information security evolved from the early field of computer security and it is the protection of information assets that use, store, or transmit information from risk. The critical characteristics of information, among them confidentiality, integrity, and availability (the C.I.A. triangle), must be protected at all times. This protection is implemented by multiple measures through policies, education training, awareness, and technology. Most of the early research on computer security centered on a system called Multiplexed Information and Computing Service (MULTICS). Although it is now obsolete, MULTICS is noteworthy because it was the first operating system to integrate security into its core functions. It was a mainframe, time-sharing operating system developed in the mid-1960s by a consortium of General Electric (GE), Bell Labs, and the Massachusetts Institute of Technology (MIT) (Micheal and Herbert, 2011).

In general, security is "the quality or state of being secure to be free from danger" (Kehoe and Brendan).

In other words, protection against adversaries from those who would do harm, intentionally or otherwise is the objective. National security, for example, is a multi-layered system that protects the sovereignty of a state, its assets, its resources, and its people. Achieving the appropriate level of security for an organization also requires a multifaceted system.

A successful organization should have the following multiple layers of security in place to protect its operations:

a. **Physical Security:** This is to protect physical items, objects, or areas from unauthorized access and misuse.

b. **Personnel Security:** To protect the individual or group of individuals who are authorized to access the organization and its operations.

c. **Operations Security:** To protect the details of a particular operation or series of activities.

d. **Communications Security:** To protect communications media, technology, and content.

e. **Network Security:** To protect networking components, connections, and contents.

f. **Information Security:** To protect the confidentiality, integrity and availability of information assets, whether in storage, processing, or transmission. It is achieved via the application of policy, education, training and awareness, and technology (Micheal and Herbert, 2011).

2.2 Cryptography

Cryptography is the science of writing in secret code and is an ancient art. The first documented use of cryptography in writing dates back to circa 1900 B.C. when an Egyptian scribe used non-standard hieroglyphs in an inscription. Some experts argue that cryptography appeared spontaneously sometime after writing was invented, with applications ranging from diplomatic missives to war-time battle plans. It is no surprise then that new forms of cryptography came soon after the widespread development of computer communications. In data and telecommunications, cryptography is necessary when communicating over any un-trusted medium, which includes just about any network, particularly the Internet.

Cryptography then not only protects data from theft or alteration, but can also be used for user authentication.

2.2.1 Features of Cryptography

Within the context of any application-to-application communication, there are some specific security requirements. Adequate cryptography systems must achieve some goals to ensure high security. These cryptography features are summarized as follows:

a. **Confidentiality:** This is to ensure that the information is encrypted and can only be correctly decrypted by the authorized party. Confidentiality depends on how strong an encryption algorithm is difficult to be broken. It is the assurance that no one other than the intended authorized party can read the information.

b. **Authentication**: Is the assurance that the received information arrived from an authorized person not a false identity.

c. **Integrity:** This is the assurance that the transmitted information is not modified or altered by some unauthorized person. It ensures that the received secret message is not corrupted over communication channel.

d. **Non-Repudiation:** Is making sure that first party or the second party should not be able to deny transmission.

e. **Access control:** This is the last aspect of Cryptography which is used to prevent authorized parties from rejecting actions they performed (Thambiraja *et al*., 2012).

2.3 Cryptographic Scheme

There are in general, three types of cryptographic schemes typically used to accomplish these goals: secret key (or symmetric) cryptography, public-key (or asymmetric) cryptography, and hash functions, each of which is described below. In all cases, the initial

unencrypted data is referred to as *plaintext*. It is encrypted into *cipher text*, which will in turn (usually) be decrypted into usable plaintext (Kessler, 2016).

2.3.1 Symmetric Cryptography

With *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 (or rule set) 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. The biggest difficulty with this approach, of course, is the distribution of the key.

Secret key cryptography schemes are generally categorized as being either *stream ciphers* or *block ciphers*. Stream ciphers operate on a single bit (byte or computer word) at a time and implement some form of feedback mechanism that the key is constantly changing. A block cipher is so-called because the scheme encrypts one block of data at a time using the same key on each block. In general, the same plaintext block will always encrypt to the same cipher text when using the same key in a block cipher whereas the same plaintext will encrypt to different cipher text in a stream cipher (Forouzan, 2007).

2.3.2 Asymmetric Cryptography

In *public key cryptography*, one of the keys is designated as the *public key* and may be advertised as widely as the owner wants. The other key is designated the *private key* and is

never revealed to another party. It is straight forward to send messages under this scheme. Suppose Alice wants to send Bob a message. Alice encrypts some information using Bob's public key; Bob decrypts the cipher text using his private key. This method could also be used to prove who sent a message. Alice, for example, could encrypt some plaintext with her private key, when Bob decrypts using Alice's public key, he knows that Alice sent the message and Alice cannot deny having sent the message (*non-repudiation*) (Micheal and Herbert, 2012).

2.3.3 Hash Function

Hash functions, also called *message digests* and *one-way encryption* are algorithms that in some sense use no key. Instead, a fixed-length hash value is computed based upon the plaintext that makes it impossible for either the contents or length of the plaintext to be recovered. Hash algorithms are typically used to provide a *digital fingerprint* of a file's contents often used to ensure that the file has not been altered by an intruder or virus. Hash functions are also commonly employed by many operating systems to encrypt passwords. Hash functions, then, provide a measure of the integrity of a file (Kessler, 2016).

2.4 Attacks on Crypto Systems

Historically, attempts to gain unauthorized access to secure communications have used brute force attacks, in which the cipher text is repeatedly searched for clues that can lead to the algorithm’s structure. These cipher text attacks involve a hacker searching for a common text structure, wording, or syntax in the encrypted message that can enable him or her to calculate the number of each type of letter used in the message. This process, known as frequency analysis, is used along with published frequency of occurrence patterns of various languages and can allow an experienced attacker to crack almost any code quickly

with a large enough sample of the encoded text. To protect against this, modern algorithms attempt to remove the repetitive and predictable sequences of characters from the cipher text. Attacks on cryptosystems fall into four general categories: man-in-the-middle, correlation, dictionary, and timing.

2.4.1 Man-in-the-Middle Attack

This is an attempt to intercept a public key or even to insert a known key structure in place of the requested public key. Thus, attackers attempt to place themselves between the sender and receiver, and once they have intercepted the request for key exchanges, they send each participant a valid public key, which is known only to them. To the victims of such attacks, encrypted communication appears to be occurring normally, but in fact the attacker is receiving each encrypted message and decoding it (with the key given to the sending party), and then encrypting and sending it to the intended recipient. Establishing public keys with digital signatures can prevent the traditional man-in-the-middle attack, as the attacker cannot duplicate the signatures.

2.4.2 Correlation Attacks

This is a collection of brute-force methods that attempt to deduce statistical relationships between the structure of the unknown key and the cipher text generated by the cryptosystem. The only defense against this attack is the selection of strong cryptosystems that have stood the test of time, thorough key management, and strict adherence to the best practices of cryptography in the frequency of key changes.

Brute force attacks are classified into six categories (cipher text-only, chosen plaintext, adaptive-chosen-plaintext, known-plaintext, chosen-cipher text, adaptive-chosen-cipher

text) that distinguish the kind of information the cryptanalyst has available to mount an attack. The categories of attack are listed here roughly in increasing order of the quality of information available to the cryptanalyst, or, equivalently, in decreasing order of the level of difficulty to the cryptanalyst. The objective of the cryptanalyst in all cases is to be able to decrypt new pieces of cipher text without additional information. The ideal for a cryptanalyst is to extract the secret key.

a. **Cipher text-only:** This attack is one in which the cryptanalyst obtains a sample of cipher text, without the plaintext associated with it. This data is relatively easy to obtain in many scenarios, but a successful cipher text-only attack is generally difficult, and requires a very large cipher text sample. A known-plaintext attack is one in which the cryptanalyst obtains a sample of cipher text and the corresponding plaintext as well.
b. **Chosen plaintext:** This is an attack in which the cryptanalyst is able to choose a quantity of plaintext and then obtain the corresponding encrypted cipher text.
c. **Adaptive-chosen-plaintext:** This is a special case of chosen-plaintext attack in which the cryptanalyst is able to choose plaintext samples dynamically, and alter his or her choices based on the results of previous encryptions.
d. **Chosen-cipher text:** This is a type of attack in which cryptanalyst may choose a piece of cipher text and attempt to obtain the corresponding decrypted plaintext. This type of attack is generally most applicable to public-key cryptosystems.
e. **Adaptive-chosen-cipher text:** This is the adaptive version of the chosen-cipher text attack. Here a cryptanalyst can mount an attack of this type in a scenario in which he

has free use of a piece of decryption hardware, but is unable to extract the decryption key from it (Micheal and Herbert, 2012).

2.4.3 Dictionary Attacks

In a dictionary attack, the attacker encrypts every word in a dictionary using the same cryptosystem as used by the target in an attempt to locate a match between the target cipher text and the list of encrypted words. Dictionary attacks can be successful when the cipher text consists of relatively few characters, for example files which contain encrypted usernames and passwords. An attacker who acquires a system password file can run hundreds of thousands of potential passwords from the dictionary he or she has prepared against the stolen list. Most computer systems use a well-known one-way hash function to store passwords in such files, but an attacker can almost always find at least a few matches in any stolen password file. After a match is found, the attacker has essentially identified a potential valid password for the system.

2.4.4 Timing Attacks

In a timing attack, the attacker eavesdrops on the victim's session and uses statistical analysis of patterns and inter-keystroke timings to discern sensitive session information. While timing analysis may not directly result in the decryption of sensitive data, it can be used to gain information about the encryption key and perhaps the cryptosystem. It may also eliminate some algorithms, thus narrowing the attacker's search and increasing the odds of eventual success. Having broken an encryption, the attacker may launch a replay attack, which is an attempt to resubmit a recording of the deciphered authentication to gain entry into a secure source.

2.5 Cipher Method

A cipher (or cypher) is a pair of algorithms which creates the encryption and the reversing decryption. The detailed operation of a cipher is controlled both by the algorithm and, in each instance, by a key. This is a secret parameter for a specific message exchange context. Keys are important, as ciphers without variable keys are trivially breakable and therefore less than useful for most purposes (Kester, 2013).

The two basic building blocks of all encryption techniques are substitution and transposition method.

2.5.1 Substitution Method

Substitution cipher is one of the bases of cryptography which has been around for a long time and is still in use. It involves replacing the plaintext characters with a different character to give the cipher text character (Serge, 2006). This substitution is carried out according to a particular scheme usually known as the key. The types of substitution cipher that exist are mono alphabetic and poly alphabetic substitution ciphers.

2.5.2 Transposition Method

In cryptography, transposition cipher is a process of encryption which involves the shifting of positions held by plaintext according to a regular pattern such that the cipher text constitutes a permutation of the plaintext (Kester, 2013). Transposition cipher, although, weak on its own and may sometimes be broken by cryptanalysis, can be combined with other classical ciphers to make it difficult to break.

2.6 Classical Cryptography

This is the oldest branch of cryptography which has been in existence for over 4000 years. They are ciphers that operate on an alphabet of letters (such as “A-Z”) and are implemented by hand or with simple mechanical devices (Katz and Lindell, 2008). They are the earliest and most basic type of ciphers which makes them not very reliable with the recent development in technology. Classical ciphers were mostly used in the ancient days for military and diplomatic purposes. The two basic components of classical cryptography are substitution and transposition (Serge, 2006). In substitution, letters are replaced by other letters, while in transposition ciphers, the letters are rearranged in a different order to give the cipher text. These ciphers can be:

a. Mono-alphabetic (Only one substitution/transposition is used) or
b. Poly-alphabetic (Several substitutions/transpositions are used)

2.6.1 Caesar Cipher

In cryptography, a Caesar cipher, also known as Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on. The method is named after Julius Caesar, who used it in his private correspondence.

The encryption step performed by a Caesar cipher is often incorporated as part of more complex schemes, such as the Vigenère cipher, and still has modern application in the

ROT13 system. As with all single-alphabet substitution ciphers, the Caesar cipher is easily broken and in modern practice offers essentially no communication security.

The transformation can be represented by aligning two alphabets; the cipher alphabet is the plain alphabet rotated left or right by some number of positions. For instance, here is a Caesar cipher using a left rotation of three places, equivalent to a right shift of 3 (the shift parameter is used as the key):

Plain: ABCDEFGHIJKLMNOPQRSTUVWXYZ

Cipher: XYZABCDEFGHIJKLMNOPQRSTUVW

When encrypting, a person looks up each letter of the message in the "plain" line and writes down the corresponding letter in the "cipher" line.

Plaintext: THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Cipher text: QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD

Deciphering is done in reverse, with a right shift of 3.

The encryption can also be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, `A = 0, B = 1,..., Z = 25`. Encryption of a letter `x` by a shift n can be described mathematically as,

$$En(x) = (x+n) \bmod 26$$

Decryption is performed similarly,

$$Dn(x) = (x-n) \bmod 26$$

2.6.2 Code Breaking Methods

Some of the known methods of breaking cryptography codes are the Friedman test and the Kasiski test (Khalid *et al.*, 2012).

a. **The Friedman Test**

This is a statistical probabilistic method that can be used to determine the likelihood that the cipher text message produced comes from a mono alphabetic or poly alphabetic cipher and for determining the length of the keyword if the cipher is poly alphabetic (Chris, 2006). Friedman test is sometimes called the Kappa Test. This technique of cryptanalysis was developed in 1925 by William Friedman. His method was based upon the probability of randomly selecting two letters from an alphabet and having them be the same. Consider for example, an event of drawing two spades from a standard deck of cards of 52. Determining the probability of both cards drawn, there are 13 spades in the deck of 52. Therefore, the probability that the first card selected is a spade is 13/52.

Now, only 12 spades remain among the remaining 51 cards, so the probability that the second card drawn is 12/51.

Therefore, the total probability that both cards drawn are spades is: (13/52) * (12/51) ≈ 0.0588

Applying the above concept to determine the probability of drawing two of the same letters from a cipher text, the probability of random selection for drawing two characters to be the same is considered.

Assuming n represents the number of letters in a given cipher text and na denotes the number of a's. The probability of selecting two a's would be:

$$\frac{na}{n} * \frac{na-1}{n-1} \quad (2.1)$$

Where na is the number of a's and n is the number of letters in the given ciphertext.

The probability of choosing two letters which are the same (i.e., two a's, two b's… two z's), would be represented as:

$$\frac{na}{n} * \frac{na-1}{n-1} + \frac{nb}{n} * \frac{nb-1}{n-1} + \ldots + \frac{nz}{n} * \frac{nz-1}{n-1} \quad (2.2)$$

Where na is the number of $a's$, nb is number of $b's$, and nz number of $z's$ in the cipher text.

The resultant probability is called the Index of Coincidence of the cipher text, and is denoted by *I.*

The index of coincidence is a measure of how similar a frequency distribution is to the uniform distribution. The index of coincidence of a piece of text does not change if the text is enciphered with a substitution cipher. It is the probability that two randomly selected letters are the same, and is defined by the formula (Chris, 2006):

$$I.C = \frac{\sum_{i=1}^{i=n} fi(fi-1)}{N(N-1)} \quad \ldots(2.3)$$

Where fi represents the count (frequency) of letter i (where $i = A, B, C, \ldots Z$) in the ciphertext, and N is the total number of letters in the ciphertext (Chris, 2006).

The probability of the frequencies of the letters in the English language alphabet is represented in table 2.1.

Table 2.1: Probabilities of the Letter frequency of English Alphabets **(**Forouzan, 2007)

Letter	A	B	C	D	E	F	G	H	I	J	K	L	M
Frequency	0.0817	0.015	0.0278	0.0425	0.127	0.0223	0.0202	0.0609	0.0697	0.0015	0.0077	0.0403	0.0241
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	0.0675	0.0751	0.0193	0.001	0.0599	0.0633	0.0906	0.0276	0.0098	0.0236	0.0015	0.0197	0.0007

According to Table 2.1, if a text were encrypted using a single alphabet, the probability of "drawing" two letters that are the same is:

$$AA(0.0817*0.0817)+BB(0.015*0.015)+CC(0.0278*0.0278)+...+ZZ(0.0007*0.0007) \quad \ldots(2.4)$$

The Index of Coincidence I approximately equals to 0.06560

If more than one alphabet were used to encrypt, the frequencies of the letters should be more nearly uniform. If they were uniform, the probability of "drawing" two letters that were the same would be:

$$I=\left(\frac{1}{26}*\frac{1}{26}\right)+\left(\frac{1}{26}*\frac{1}{26}\right)+\left(\frac{1}{26}*\frac{1}{26}\right)+\cdots\left(\frac{1}{26}*\frac{1}{26}\right)=\frac{1}{26}\approx 0.038 \quad \ldots(2.5)$$

The idea of the test is to determine if the cipher text generated was generated by a mono alphabetic cipher or poly alphabetic cipher. If the cipher text were generated by a mono alphabetic cipher, the index of coincidence *I* should be close to 0.065, because a mono alphabetic cipher is just a permutation of the letters of a single alphabet. If on the other hand, the cipher were generated by a poly alphabetic cipher, the frequencies of the letters would become more nearly uniform and the index of coincidence *I* would be close to 0.038. The cipher text generated is tested by calculating *I* based on the cipher text frequencies. The closer *I* is to 0.065, the more likely it is that it is a mono alphabetic cipher, and the closer *I* is to 0.038, the more likely it is to be a poly alphabetic cipher.

b. **Kasiski Test**

Friedrich Kasiski was the first to publish a successful attack where he suggested that one may look for repeated fragments in the cipher text which comprises of repetitions when the keyword encrypts the same fragment of text several times. The presence of these repetitions is used to compile a list of the distances that separate the repetitions. This distance is likely to be the keyword length, which is used for further cryptanalysis. Once the length of the keyword is discovered, the message could be grouped in columns and then mono alphabetic substitution is performed on each column to determine the keyword. Kasiski observed the following (Katz and Lindell, 2008).

If a repeated substring in a plaintext is encrypted by the same substring in the keyword, then the cipher text contains a repeated substring and the distance of the two occurrences is a multiple of the length of the keyword.

Not every repeated string in the cipher text arises in this way; but the probability of a repetition by chance is very small.

2.7 Confusion and Diffusion

Shannon (1949), proposed a theory to assess the secrecy of symmetric cryptosystems. This theory was based on his earlier theory of information and entropy in 1948 which involved examining the amount of information about a random message (drawn from probability distribution) an attacker gains after being given an encrypted text of that message. An encryption scheme is however said to be perfect if an attacker gains no information about the message or the used key from its encryption. However, for perfect secrecy to be achieved, it is suggested that every bit of information in the message must be encrypted

using a bit of information in the key. This development led to the introduction of the concepts of diffusion and confusion by Shannon in order to strengthen encryption. According to Shannon, diffusion is defined as a cryptographic technique that seeks to obscure the statistical structure of the plaintext by spreading out the influence of each individual plaintext character over many cipher text characters. Confusion on the other hand is a cryptographic technique that seeks to make the relationship between the cipher text character and the value of the encryption key as complex as possible, by the use of a complex scrambling algorithm that depends on the key and the input (William, 2011).

2.8 Performance Metrics

2.8.1 Frequency Analysis

The first known recorded explanation of frequency analysis (indeed of any kind of cryptanalysis) was given in the 9^{th} century by Al-Kindi, an Arab polymath in a manuscript on deciphering cryptographic messages (Al-Kindi, 1992).

Frequency Analysis is based on the fact that in any given stretch of written language, certain letters or combinations of letters occur with varying frequencies. Here the cryptanalyst counts the frequency of cipher text letters and then associate guessed plain text letters with them.

Friedrich Kasiski was the first to suggest that one may look for repeated fragments in the cipher text which comprises of repetitions when the keyword encrypts the same fragment of text several times. The presence of these repetitions is used to compile a list of the distances that separate the repetitions. This distance is likely to be the keyword length, which is used for further cryptanalysis (Katz and Lindell, 2008).

Suppose a cryptanalyst has intercepted a given cryptogram which is known to have been encrypted using simple substitution cipher as follows:

a. For this example, *UPPERCASE LETTERS* are used to denote cipher text, *lowercase letters* denote plaintext (or guesses) and ***X→ t*** is used to express a guess that cipher text ***X*** represents the plaintext letter ***t***

   LIVITCSWPIYVEWHEVSRIQMXLEYVEOIE

   WHRXEXIPFEMVEWHKVSTYLXZIXLIKIIX

   PIJVSZEYPERRGERIM

b. Frequency analysis can help solve the message along the following lines:

   i. Counts of the letter in the cryptogram show that ***I*** is the most common single letter, ***XL*** most common *bigram*, and ***XLI*** is the most common *trigram*.

   ii. ***e*** is the most common letter in the English language, ***th*** is the common *bigram*, and ***the*** is the most common *trigram*.

   iii. This strongly suggests that:

   X → t , L → h and ***I → e***

c. The second most common letter in the cryptogram is ***E***; since the first and second most frequent letters in English language, ***e*** and ***a*** are accounted for, the cryptanalyst can guess that ***E → a***

d. Tentatively making these assumptions, the following partial decrypted message is obtained.

   heVeTCSWPeYVaWHaVSReQMthaYVaOea

WHRtatePFaMVaWHKVSTYhtZetheKeet

PeJVSZaYPaRRGaReM

e. Using these initial guesses, the cryptanalyst can spot patterns that confirm his choices, such as **heVe**, **Rtate** e.t.c

f. Moreover, patterns can suggest further guesses **"heVe"** might be **"here"**; **"Rtate"** could be guessed as **"state"** giving $V \rightarrow r$ *and* $R \rightarrow s$

iv. Filling in these guesses. The cryptanalyst gets:

hereTCSWPeYraWHarSseQithaYraOea

WHstatePFairaWHKrSTYhtmetheKeet

PeJrSmaYPassGasei

v. It is relatively straightforward to deduce the rest of the letters, eventually yielding the plaintext.

vi. Lastly, the cryptanalyst insert spaces and punctuations.

NOTE: In cases where the initial guesses are incorrect, the cryptanalyst backtracks incorrect guesses or analyzes the available statistics in much more depth than the somewhat simplified justifications given in the above example.

2.9 Review of Related Works

To give more prospective about the performance of the encryption algorithms, this part describes and examines previous work done in field of data encryption.

Vinod *et al*., (2012), presented a perspective on various techniques which are currently used for cryptanalysis purpose. The paper focused on various types of attacks such as

Boomerang attack, Brute force attack, Davies' attack, Birthday attack, Side-Channel attack e.t.c which are mainly used in Symmetric cipher, Asymmetric cipher and Hash system. This research produced a strong algorithm that is devoid of being attacked by brute force or frequency analysis attack.

Srikantaswamy and Phaneendra (2012), proposed a hybrid encryption algorithm of an extended Caesar cipher technique (by increasing the range of characters used to 94) and the columnar transposition cipher using a random number generation technique to generate the key used in the algorithm. However they only made used of 94 characters, whereas our proposed system considered 256 characters to solve the problem of character repetition.

Khashish and Supriya (2013), proposed an algorithm where key generation play a crucial role in designing the cipher. The author modified the traditional Caesar cipher and fixed the key size as one. The algorithm used alphabet position as index which is applied in the algorithm for security purpose. This was achieved by increasing the value of an even index by one and reducing the value of an odd index by one. The fundamental problem of this work is that the security provided is not strong enough since there are only 26 possible keys, and one can try them all using brute force attack. The reliability of this system however has a limitation such that with careful study of the cipher text, the pattern will easily be noticed by cryptanalysts. The security of the system can however be further enhanced if more than one algorithm is applied to the data. Figure 2.1(a) and (b) shows the encryption and decryption process of the work. Though in our own proposed system, we made use of index positioning but in conjunction with ASCII values.

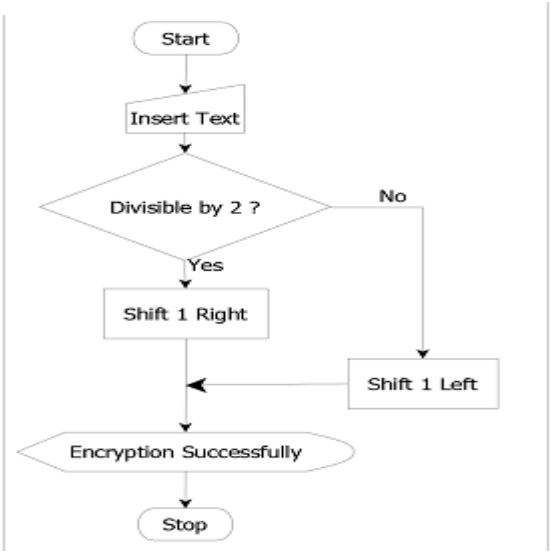

Figure 2.1(a): Encryption Process

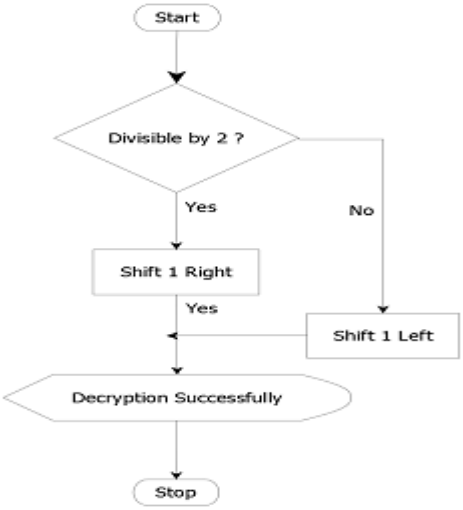

Figure 2.1(b): Decryption Process

Anupama (2014), induced some strength to the classical encryption. For that purpose the researcher blended Caesar Cipher with play fair cipher. One of the limitations of this work is that the algorithm can be easily exploited with the aid of frequency analysis if the language of the plain text is known.

Omolara *et al.,* (2014), developed a modified Caesar cipher by combining both traditional Caesar cipher with Vigenere cipher. Lettered key and numbered key were used for the encryption process. A Caesar cipher was performed on the lettered key using the shift of the numbered key. Vigenere cipher is then performed on the plaintext using the new key. The encryption was only based on the 26 characters of the English language alphabet which makes the system prone to brute force attack.

However they only made used of 26 characters, whereas our proposed system considered 256 characters to solve the problem of character repetition.

Fahad (2015), combined Caesar cipher with rail fence. The algorithm uses idea of dynamic key to exclude repetitive terms from a message which is encrypted using Rail fence cipher.

The reliability of this system however has a limitation because it can easily be exploited since rail fence cipher offers essentially no communication.

Asiya and Ruba (2015), proposed a new technique for protecting and locking messages. This approach uses improved version of ciphering with the combination of double phase encryption. The author applied a simple technique of vertically selecting the text for ciphering using a 6 x 6 matrix based on the alphabets used in the text message. Once the text is scrambled vertically, Caesar cipher method is applied on its outcome with the shift key. The merit of this algorithm is the combination of both substitution and transposition method, transformation of results from vertically scrambled text and simplicity of implementation. However, there is still problem of character repetition because there are only 26 possible keys, and one can try them all using brute force attack. Figure 2.2 (a) and (b), shows the encryption and decryption process of their work.

Hence, the need to design an improved vertically scrambled Caesar Cipher method that will solve the problem of character repetition and enhance data security.

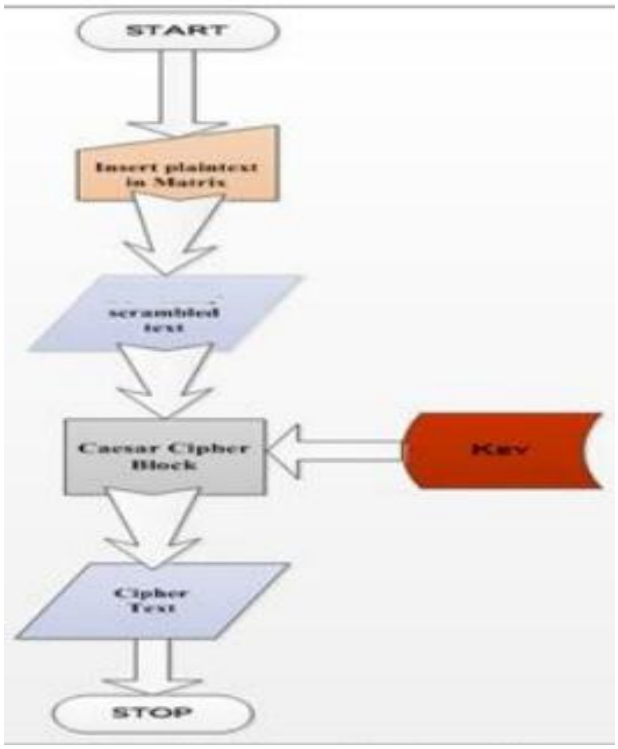

Figure 2.2(a): Encryption Process

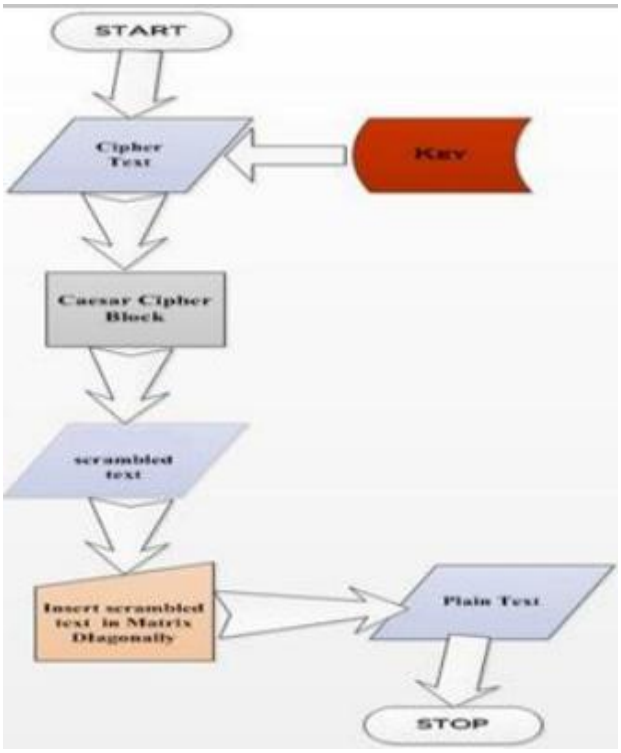

Figure 2.2(b): Decryption Process

2.9.1 Gap in Literature

Though each of the literature reviewed modified the Caesar cipher, it is not sufficient to say the modification is not correct but the modification could not handle some of the problems associated with the security and robustness of message transmission.

Hence, this research work extends Asiya and Ruba (2015) by increasing the number of input alphabets and applying index positioning as well as ASCII characters so as to prevent repetition of characters which makes the algorithm more secure and robust.

CHAPTER THREE: THE PROPOSED SYSTEM

This chapter presents the architecture of the proposed system, based on the existing system of Asiya and Ruba (2015). The modified encryption and decryption algorithms are also presented.

3.1 Proposed System

The main weakness of the Caesar cipher is the repetition of characters for the encryption process. This repetition leaves the algorithm vulnerable to frequency analysis when repeating characters encrypt the same set of characters leading to similar fragments in the cipher text which could be solved using mono alphabetic method to determine the length of the key, and subsequently determining the encryption key. To curb this problem, the system builds on the work of Asiya and Ruba, (2015) by improving on the number of inputs as well as applying index positioning and ASCII characters so as to prevent repetition of characters thereby making the algorithm more secure and robust.

3.2 Description of the Proposed System

To encrypt a message, the system requires plaintext and encryption key. The encryption key is an integer that determines the alphabet to be used for substitution. It is based on modulus 256 arithmetic to ensure that the integer value wrap round in case encryption key supplied is more than 256. Decryption follows reverse operations performed during the process of encryption. It requires decryption key, and encrypted text. The decryption key should be a complement to the encryption key so that reverse character substitution can be achieved. As stated earlier, Caesar cipher simply shifts encrypted character by number of positions.

The algorithm is a symmetric key cryptographic method which combines both substitution and transposition cryptographic techniques. In this method, plaintext is first scrambled vertically in a matrix of size 6 x 6 so as to include more confusion and diffusion into the algorithm which will eliminate the problem of character repetition. If the message is lengthy and contains more letters not fit in the defined dimension then more than one matrices of same size will be used. Once the text is scrambled vertically, the modified Caesar cipher is applied on it. The designed system can be applied to encrypt plaintext messages composed of alphabets (UPPERCASE and LOWERCASE), numbers and symbols.

3.3 Architecture of the Proposed System

The pseudo code for encryption and decryption process of the proposed system as shown in figure 3.1(a) and (b) is as follows:

3.3.1 Encryption Process

Stage 1

a. Construct a 6 X 6 matrix by filling the characters of the plaintext from left to right and top to bottom.
b. Fill the remaining cells of the matrix with alphabets from A to Z.
c. Get the elements of the matrix

Stage 2

Encrypt the resulting concatenated texts using the modified Caesar cipher as shown in figure 3.2.

a. Get the index of the plaintext (concatenated text) characters.

b. Get the ASCII representation of each character.

c. Check if index and ASCII representation are both odd or even (move forward by key) else, move backward by key.

d. Get the encrypted text.

3.3.2 Decryption Process

Stage 1

a. Get the cipher text

b. Using modified Caesar cipher

   i. Get the index of the encrypted text

   ii. Get the ASCII representation of the cipher text

Stage 2

a. Fill in the characters of the concatenated cipher text into a 6 x 6 matrix vertically

b. Get the original plain text from the matrix.

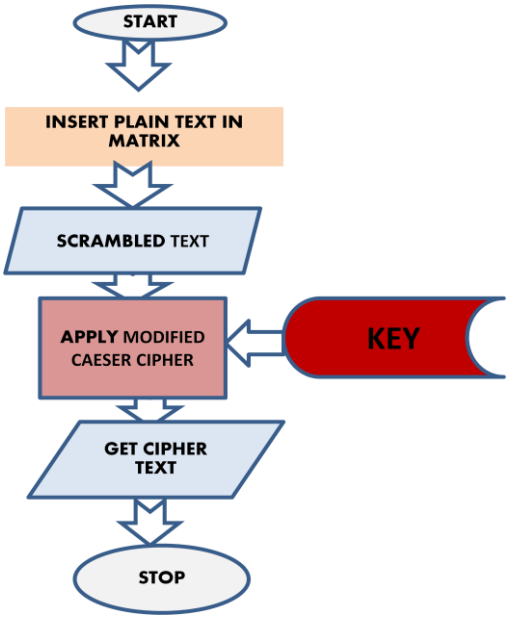

Figure 3.1(a): Encryption Process

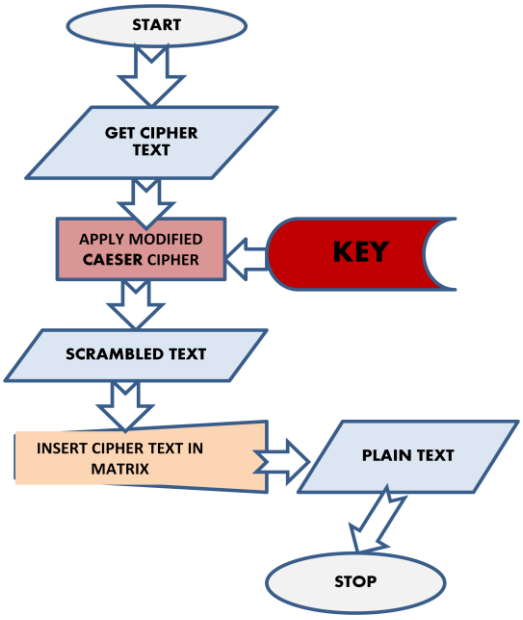

Figure 3.1(b): Decryption Process

3.4 Flow Diagram of the Stages involved in Modified Caesar Cipher Algorithm

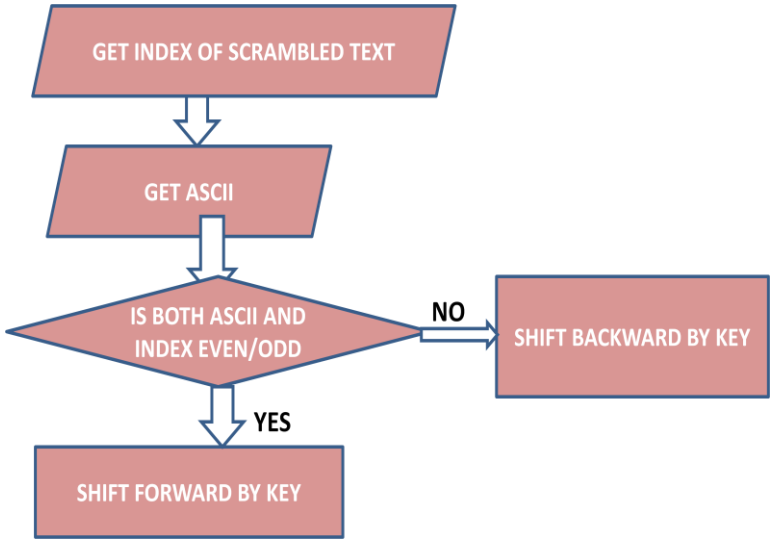

Figure 3.2: Stages of modified Caesar cipher Algorithm

3.5 Proposed System Algorithm

The encryption and decryption algorithm are given in the following sub sections.

3.5.1 Encryption Algorithm

1. Input key (key) and plaintext (ptext) to encrypt
2. Size = the divisor of length of ptext by 36
3. combineMatrix[]=null
4. If size <= 1, then
    generate 6 x 6 matrix
  else
    while size >1
      matrix = generate 6 x 6 matrix
      add matrix to list of matrix in combineMatrix
    end while
 end if
5. Let x = " "
6. For each row in combineMatrix
    select odd columns
    concatenate to x
  end for each
7. For each row in combineMatrix
    select even columns
    concatenate to x
 end for each
8. *Display modifyCaesarCipher (x,key)*

9. *Subalgorithm modifyCaesarCipher (x,key)*

10. *For each char in x*

11. *i = index of char*

12. *ascii = ASCII value of i*

13. *If i and ascii are even*

ascii+= key

result = ascii MOD 256

ctext = ASCII code of result

else

asci -= key

result = ascii MOD 256

ctext = get ASCII code of result

end if

14. *Return ctext*

15. *End sub algorithm*

3.5.2 Decryption Algorithm

1. Input key (key) and ciphertext (ctext) to decrypt

2. Size = the divisor of length of ptext by 36

3. combineMatrix[]=null

4. If size <= 1, then

generate 6 x 6 matrix

else

while size >1

matrix = generate 6 x 6 matrix

add matrix to list of matrix in combineMatrix

end while

end if

5. Let x = " "

6. For each row in combineMatrix

select odd columns

concatenate to x

end for each

7. For each row in combineMatrix

select even columns

concatenate to x

end for each

8. *Display modifyCaesarCipher (x,key)*

9. *Subalgorithm modifyCaesarCipher (x,key)*

10. *For each char in x*

11. *i = index of char*

12. *ascii = ASCII value of i*

13. *If i and ascii are even*

ascii+= key

result = ascii MOD 256

ptext = ASCII code of result

else

asci -= key

result = ascii MOD 256

ptext = get ASCII code of result

end if

14. *Return ptext*

15. *End sub algorithm*

CHAPTER FOUR: IMPLEMENTATION AND RESULT ANALYSIS

This chapter discusses the implementation details of the system with screen shot showing the graphical user interface. In addition, the section also presents all the experiments conducted to evaluate the proposed system and results of comparative analysis obtained from the research.

4.1 System Requirement

The software requirement for the effective development and implementation of this system are as follows:

a. Java Programming

b. Netbeans IDE

The hardware requirements are:

a. A Pentium IV CPU, 1.8 GHz processor with 512 MB memory

b. A 4 GB hard drive capacity

c. A graphic adaptor with screen resolution of 800 * 600 pixels and 32 bit quality

d. A CD/DVD rewritable drive

e. Laser jet printer for draft copies

4.2 Implementation Details

This section discusses the requirement needed for the implementation of the system.

The system “Modified Vertically Scrambled Caesar Cipher Method” is considered to be a standalone application where it does not require using neither a database nor web server. It is expected to perform two main operations: encrypting and decrypting messages.

The system was implemented on Windows 7 Ultimate Operating System. It was implemented on a machine that has processor of 2.27 GHz Intel Core i3 with memory 2 GB 350 MHz DDR3. The system was written in java language and implemented in Netbeans environment.

4.3 Graphical User Interface of the Proposed System

The Modified Vertically Scrambled Caesar Cipher Method has the following functions. Shift Button, Message Report Area, Plaintext Area, Cipher Text Area, Encryption and Decryption Buttons, Clear Button, E-Mail Button and the Quit Button. Figure 4.1 shows the graphical user interface of the designed system.

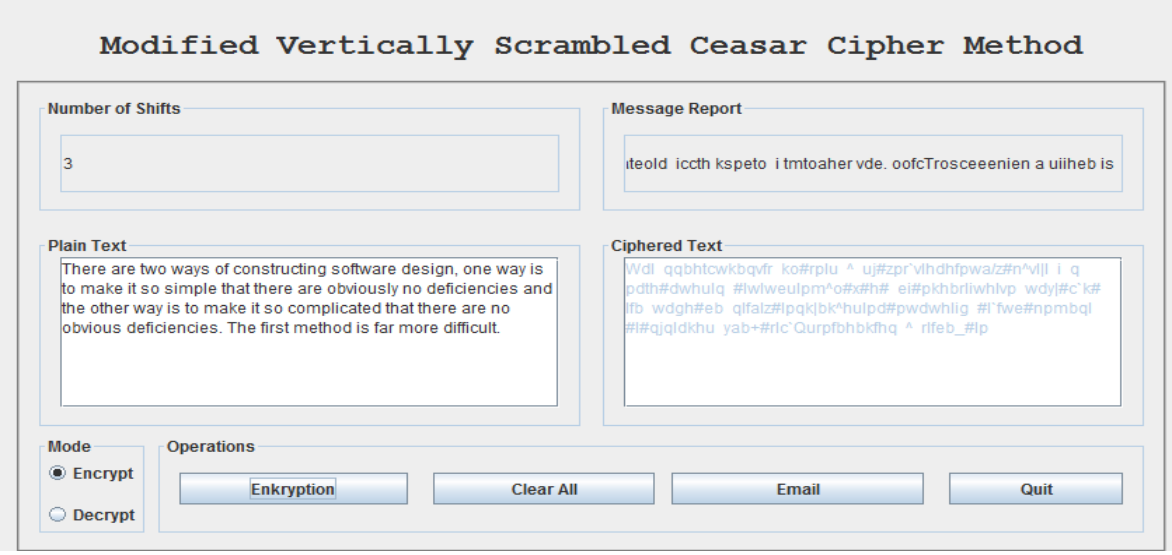

Figure 4.1: Graphical User interface of Designed System

4.3.1 Plain Text Encryption

During the encryption stage, the user enters the plain text (which can include combination of numbers, letters, and special characters) in the plain text area, then he proceeds to

specify the number of shifts which can only take integer values and finally choose the mode of operation (*encryption*) in order to encrypt the plain text message to get the cipher text.

4.3.2 Cipher Text Decryption

During the decryption stage, the user enters the cipher text (result from the encrypted plain text) in the cipher text area, then he proceeds to specify the number of shifts which can only take integer values and finally choose the mode of operation (in this case, decryption) in order to decrypt the cipher text message to get the original plain text.

4.4 Experimental Results and Analysis

During this stage, the performance of the system was tested with messages of various lengths as shown in table 4.1 and compared with the work of Asiya and Ruba, (2015) in line with the stated objectives of our research. The comparative performance and security of the proposed system was based on the following parameters: Frequency of character occurrence, character length, Running Time, Integrity and Security. The results are presented in graphs and tables.

4.4.1 Dataset

This sub-section describes the dataset of plaintexts that were used for the experiment. The plaintexts were taken from Hoare's quote (Friedman, 1996) and the work of (Asiya and Ruba, 2015).

4.4.2 Frequency Analysis

One of the classical cryptanalysis methods used is by detecting the frequency of characters (character repetition) in the encrypted text (message). So to test the effectiveness of the proposed system, the frequencies of characters were closely observed. The index of

coincidence (IC) is one of the cryptanalysis techniques for studying the probability of repeated letters in a ciphered text (Chris, 2006).

$$\mathbf{IC} = \sum_{i=1}^{i=n} \frac{Fi(Fi-1)}{N(N-1)} \quad \ldots(4.1)$$

With `Fi`, the number of occurrences of the letter `i` in the text and `N` the total number of letters of the cipher text.

The lower the Index of Coincidence the lesser the possibility of cryptanalyst to break the encrypted text and the higher the Index of Coincidence, the higher the possibility (Aliyu and Olaniyan, 2016).

Table 4.1 shows the encryption result and the frequency occurrences of character patterns from both the proposed system and the existing system.

Table 4.1: Encryption Result from Existing System and Proposed System

Plaintext	**Encrypted Text**	
Sample 1: (Hoare's Quote)	**Existing System**	**Proposed System**
There are two ways of constructing software design, one way is to make it so simple that there are obviously no deficiencies and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult.	WUDRFR**HWV**VLWHRIUJDKHCQWIUAR WQADAFXVUKHXGHDUEOIFGDLQFHKH RVHQQHYCLLWURRLVHKHLIFHURXFHL DE**G****HWV**HQRLLIHRVLVUUYHQKWD**JPV CHFLRXDJH**NTAFLEKQWDIGMSYEKIOU BGMD**JPVCHFLRXDJH**NTAFLEKQWDIG MSYEKIOUBGM	Wdlqqbhtcwkbqvfrko#rplu^uj#zpr`vlhdhfpwa/z#n^ vl|liqpdth#dw**hu**lq#lwlweulpm^o#x#h#ei#pkhbrliw hlvpwdy|#c`k#lfb**wd**gh#ebqlfalz#lpqk|bk^hulpd#p **wd**whlig#l`fwe#npmbql#l#qjqldk**hu**yab+#rlc`Qurpf bhbkfhq^rlfeb_#lp
Sample 2: (Asiya and Ruba, 2015) ENCRYPT ME PLEASE	HWDGMSFHHLOUBOEKQWQPVHNTUSDJ PVSHFLRX	BQoddP@P>FFRVBhHTQB?KkuhpAMMSm1CO O

It is worthy of note that the resultant cipher text obtained from the proposed system contains characters that cannot easily be deciphered using ordinary symmetric means. The cipher text becomes a mixture of characters from the ASCII table which contains both the control characters and the printable characters. The mixture of these characters ensures the increased strength of the new system, which in turn makes brute force cryptanalysis difficult or impossible. It also ensures that the range of keys that could be tested by brute force increased from 26 to 256. This method has introduced complexity into the system such that any adversary trying to understand the cipher text will find it confusing. The frequency analysis of the cryptosystem cannot be determined because only the alphabets of the English language have frequencies of occurrence. Numbers, symbols and other characters have no specific frequency occurrence, thereby causing the frequency analysis based cryptanalysis to fail.

Figure 4.2 and 4.3 shows the frequency of cipher text characters obtained from both the existing and proposed system respectively.

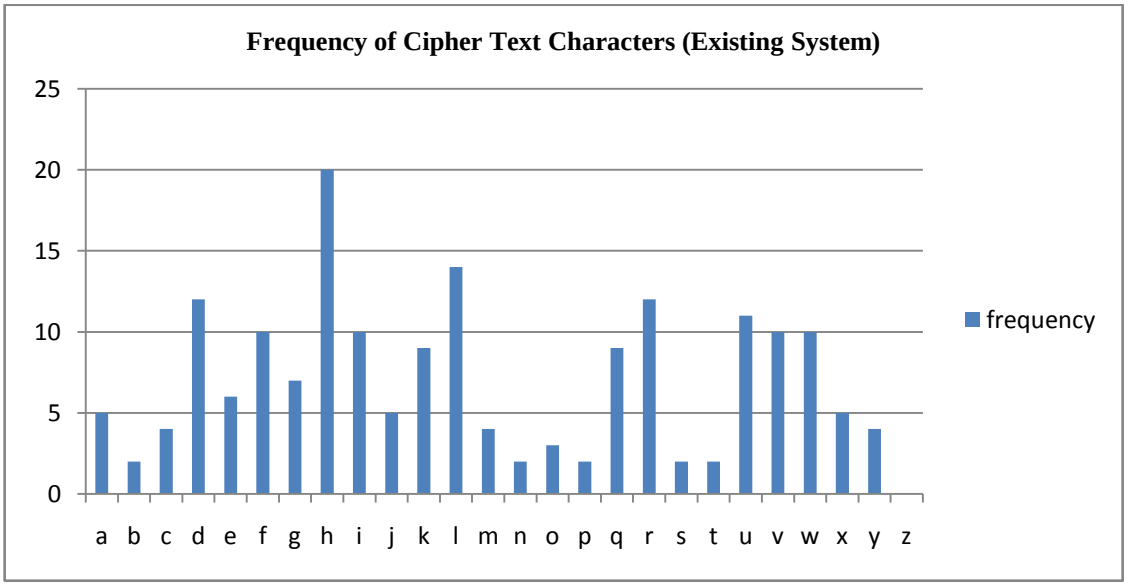

Figure 4.2: Frequency of Cipher text characters (Existing System)

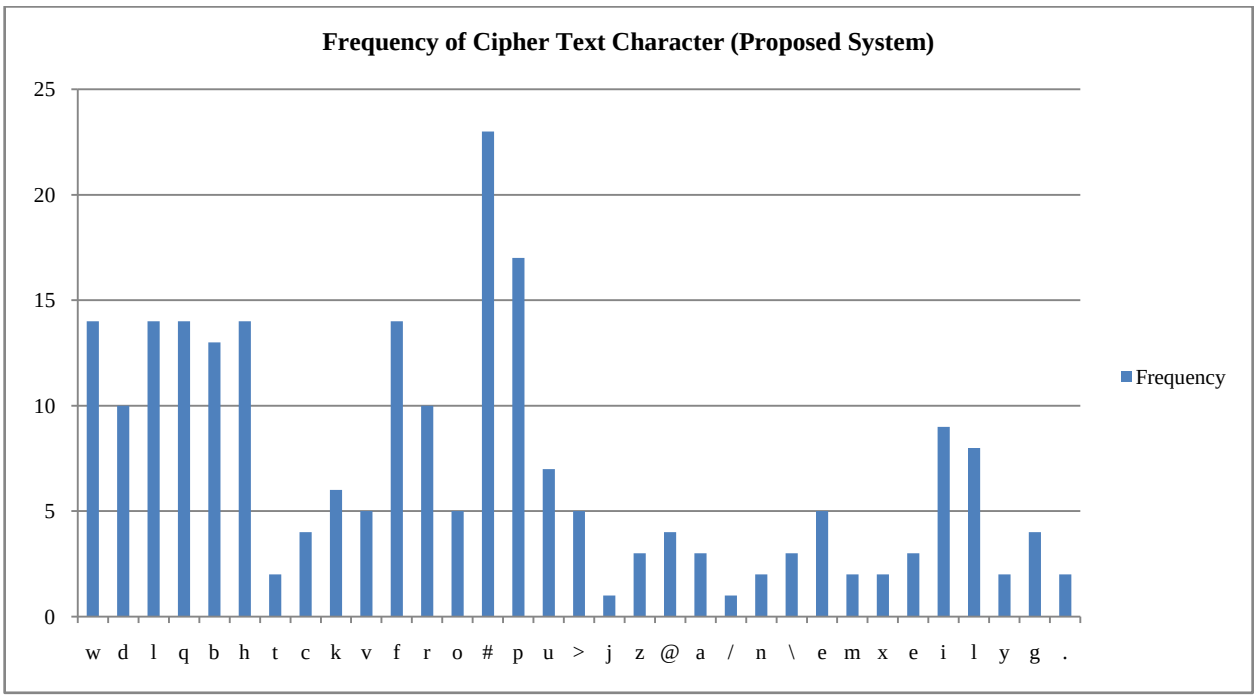

Figure 4.3: Frequency of Cipher text characters (Proposed System)

The results obtained from calculating Index of Coincidence of the two systems is shown in table 4.2 and the graph is represented in figure 4.4.

Table 4.2: Result obtained from calculating Index of Coincidence

	Existing System	**Proposed System**
Index of Coincidence	0.0504	0.0447

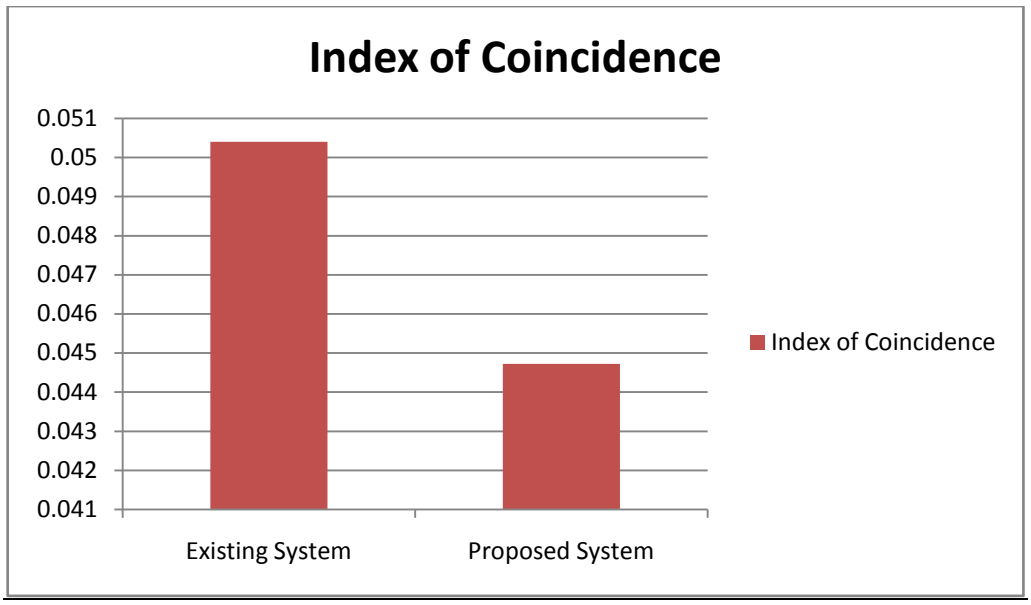

Figure 4.4: Bar chart showing results obtained from calculating index of coincidence

From the analysis it is seen that the proposed system has a lower Index of coincidence of 0.0447 when compared with the existing system which has 0.0504. Thus, making the new system less prone to frequency analysis attack.

4.4.3 Character Length

Character length should be very large in order to increase the complexity and security of algorithms. The proposed system was able to solve the problem of frequency analysis and brute force attack by increasing the number of input characters from 26 to 256 (which include letters (both upper and lower case), numbers and other special characters).

4.4.4 Security Analysis

In order to test the security level of the designed algorithm, a set of tests and analysis were performed on the algorithm. Some of these tests are taken from different cryptanalysis

papers, the following analysis methods are performed on the algorithm: Information Entropy and index of Coincidence.

To calculate the entropy `H(X)`, we have:

$$H(x) = -P\sum_{i=0}^{n} \log_2 P \qquad \ldots(4.2)$$

Where P is the probability of occurrence of each character

$H(x)$ is the entropy measured in bits or Shannon (Aliyu and Olaniyan, 2016).

The results obtained from calculating entropy of the two systems is shown in table 4.3 and the graph is represented in figure 4.5.

Table 4.3: Results obtained from calculating Entropy

	Existing System	**Proposed System**
Entropy	0.4355	0.6964

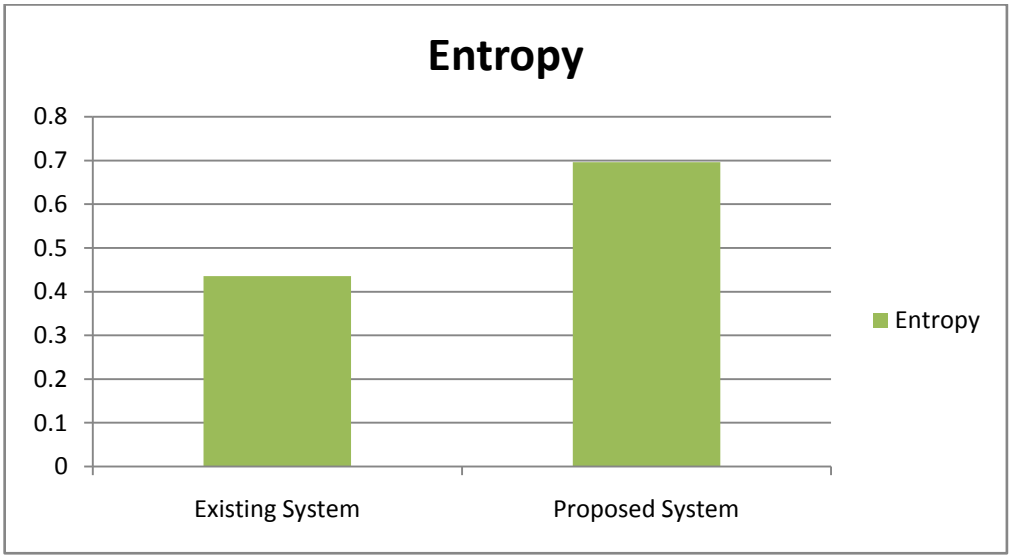

Figure 4.5: Bar chart of results obtained from calculating entropy

The result shows the entropy values H (X) for the proposed system as 0.6964 bits which is higher than that of the existing system having 0.4355 bits. This indicates that the proposed algorithm is secure upon the entropy attack because of its use of index positioning and ASCII codes that prevents character repetition. Thus, solving the problem of frequency cryptanalysis attack.

4.4.5 Time

The time needed for encryption and decryption process is higher at the expense of acceptable additional computation overheads of up to 15% in the worst case when compared with that of Asiya and Ruba (2015) which will make it difficult for a cryptanalyst to decrypt a message. Thus, solving the problem of brute force attack.

CHAPTER FIVE: SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

This dissertation introduced a new approach for complex encrypting and decrypting data. Although there have been many researches on cryptography, most of the existing algorithms have several weaknesses either caused by low security or repetition of characters due the designs of the algorithms themselves. Our proposed algorithm was implemented and tested against frequency analysis attack and proved more secure and robust than the base algorithm we extended. Therefore, it can be considered as a good alternative to some applications such as ATM's and electronic commerce.

5.2 Conclusion

In line with the objectives of the research, the following have been achieved:

a. A new and better secure confusion and diffusion has been introduced into Caesar Cipher which produced a strong algorithm that is devoid of being attacked by brute force or frequency analysis.
b. An algorithm where the frequency of the letters and symbols cannot be used to determine the plaintext unlike the existing system that could be broken using frequencies of the letters have been produced.
c. The algorithm was successfully implemented in java and Netbeans IDE.
d. A comparison analysis carried out between the existing system and the proposed system showed that the proposed system gives a better security.

5.3 Recommendation

Further studies can be conducted as a complement to this dissertation or as a follow-up by improving on the algorithm by reducing the time complexity.

5.4 Research Contribution to Knowledge

In line with the objectives of the research:

a. A new and better secure confusion and diffusion has been introduced into Caesar Cipher which produced a strong algorithm that is devoid of being attacked by brute force or frequency analysis.
b. An algorithm where the frequency of the letters and symbols cannot be used to determine the plaintext unlike the existing system that could be broken using frequencies of the letters have been produced with the addition of 256 characters, thus, leading to natural encryption and decryption.
c. Implemented and evaluated the system with various lengths of messages using Index of coincidence and entropy as test bed to further buttress the security of the proposed algorithm.

REFERENCES

Aliyu, M. A. and Olaniyan, A. (2016). Vigenere Cipher: Trends, Review and Possible Modifications. *International Journal of Computer Applications*, *135*(11), 46 - 50.

Al-Kindi. (1992). The origins of cryptology: The Arab contributiion. *Journal of Cryptologia*, *16*(2) 97-126.

Anupama, M. (2014). Enhancing Security of Caesar Cipher using Different Methods. *International Journal of Research in Engineering and Technology*, *2*(9), 327 - 332.

Asiya, A. and Ruba, M. (2015). Vertically Scrambled Caesar Cipher Method. *International Journal of Computer Applications*, *118* (21), 9 - 13.

Chris, C. (2006). *Friedman Test*. Retrieved August 24, 2015, from www.nku.edu/chrisensen.

Christof, P. and Pelzi, J. (2010). *Understanding Cryptography*. Germany: Springer Heidelberg.

Fahad, N.N. (2015). A New Modified Caeser cipher method with rail fence to encrypt message. *International Journal of Advanced Research, 3* (2), 1-5.

Forouzan, B. (2007). *Cryptography and Network Security. (Special Indian Edition).*New York: McGraw Hills Company Incorporation.

Friedman, W. (1996). Frequency Analysis. Retrieved June 26, 2016, from http://www.cs.mtu.edu/-shene/NSF-4/tutorial/Vig/Vig-IOC.html

Hamdan, O., Zaidan, B. and Zaidan, A. (2010). New Comparative study between DES 3DES and AES within nine factors. *International Journal of Computer Applications, 2*(3), 152-157.

Hopper, D. (2010). Mafiaboy Faces up to 3 Years in Prison. Retrieved June 26, 2016, from http://books.google.com.ng/books?135176731

Kashish, G. and Supriya, K. (2013). Modified Caesar Cipher for Better Security Enhancement. *International Journal of Computer Applications, 73*(3), 26-31.

Katz, J. and Lindell, Y. (2008). Introduction to Modern Cryptography. North West, London: Chapman, Hall, Taylor and Francis Group.

Kehoe, and Brendan, P. (1992). *the Art of the Internet.* Masachusetts: MIT Press, Cambridge.

Kessler, G. (2016). Overview of Cryptography. Retrieved June 26, 2016, from http://www.garyKessler.net/library/crypto.html.

Kester, Q. (2013). A Hybrid Cryptosystem based on Vigenere cipher and Columnar Transposition Cipher. *International Journal of Advanced Technology and Engineering*, *3*(1), 141-147.

Khalid, I., Neela, W. and Varibhav, M. (2012). Alpha-Qwerty cipher: An extended viginere cipher. *Advanced Computing:An International Journal*, *3*(3)107-117.

Kumar, A. and Kumar, P. (2012). PA Substitution Cipher. *International Journal of Engineering Research and Technology*, *1*(10), 45-52.

Michael, W. E. and Herbert, M. J. (2011). *Principles of Information Security, 4th Ed.* Kennsaw University: Cengage Learning Press.

Micheal, E. and Herbert, J. (2012). *Principles of information System.* Kennsaw University, Cengage Learning Press.

Omolara, O.E, Oludare, A.I. and Abdullahi, S.E (2014). Developing a modified Hybrid Caesar cipher and Vigenere cipher for secure Data Communication. *Journal of Computer Engineering and Intelligent System*, 5(5), 34 - 46.

Praloy, S. and Prasenjit, M. (2013). DEDD Symmetric-Key Cryptosystem. *International Journal of Advanced Computer Research*, *3*(8), 171-176.

Serge. (2006). *A Classical Introduction to Cryptography Applications for Communication Security*. Germany: Springer Science Business Media, Incorporation.

Srikantaswamy, S. G. and Phaneendra, H. D. (2012). Improved Caesar Cipher with Random Number Generation Technique and Multistage Encryption. *International Journal on Cryptography and Information Security*, *2*(4), 39-49.

Thambiraja, E., Ramesh, G. and Umarani, D. R. (2012). A survey on various most common encryption techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(7), 226-233.

Vinod, S., Suman, M. and Jyoti, M. (2012). A review of various techniques of cryptanalysis). *International Journal of Advanced Research in Computer Science and Software Engineering*, *2*(10), 89-92.

Warner and Bernhard, (2002). Internet Firm Hacked Out of Business. Retrieved from www.zdnet.com/article/internet-firm-hacked-out-of-business.html.

William, S. (2011). *Cryptography and Network Security-Principle and Practice.* United State of America: Pearson Education.

Wyseur, B. (2009). *White-Box Cryptography.* Retrieved from www.white-box cryptography.com.html.

APPENDIX 1: SAMPLE PROGRAM CODE

```
package dijekode;


import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import javax.swing.JOptionPane;
public class Main extends javax.swing.JFrame {


    int shiftkey = 0;
    String[] matrixHouse;
    String[][] smallmatrix;
    String mydictionary = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    /**
     * Creates new form Main
     */
    public Main() {
        initComponents();
        enkrypt.setText("Enkryption");
        enkrypt.setEnabled(false);
    }
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {


        jPanel1 = new javax.swing.JPanel();
        jPanel2 = new javax.swing.JPanel();
```

```
        shift = new javax.swing.JTextField();
        jPanel3 = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        ptext = new javax.swing.JTextArea();
        jPanel4 = new javax.swing.JPanel();
        jScrollPane2 = new javax.swing.JScrollPane();
        stext = new javax.swing.JTextArea();
        jPanel5 = new javax.swing.JPanel();
        msgreport = new javax.swing.JTextField();
        jPanel6 = new javax.swing.JPanel();
        enkrypt = new javax.swing.JButton();
        klear = new javax.swing.JButton();
        quit = new javax.swing.JButton();
        jButton1 = new javax.swing.JButton();
        eventmode = new javax.swing.JPanel();
        en = new javax.swing.JRadioButton();
        de = new javax.swing.JRadioButton();
        jLabel1 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jPanel1.setBorder(javax.swing.BorderFactory.createEtchedBorder(java.awt.Color.lightGray, java.awt.Color.gray));

        jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("Number of Shifts"));

        shift.setText("3");
```

```
        shift.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                shiftActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(shift)
                .addContainerGap())
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(shift, javax.swing.GroupLayout.DEFAULT_SIZE, 47,
Short.MAX_VALUE)
                .addContainerGap())
        );

        jPanel3.setBorder(javax.swing.BorderFactory.createTitledBorder("Plain Text"));

        ptext.setColumns(20);
```

```
        ptext.setLineWrap(true);
        ptext.setRows(5);
        ptext.setText("KHADIJa ABDULKADiR");
        ptext.setWrapStyleWord(true);
        jScrollPane1.setViewportView(ptext);

        javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel3Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 356,
Short.MAX_VALUE)
                .addContainerGap())
        );
        jPanel3Layout.setVerticalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 123,
Short.MAX_VALUE)
                .addContainerGap())
        );

        jPanel4.setBorder(javax.swing.BorderFactory.createTitledBorder("Ciphered Text"));
```

```
        stext.setColumns(20);
        stext.setLineWrap(true);
        stext.setRows(5);
        stext.setWrapStyleWord(true);
        jScrollPane2.setViewportView(stext);

        javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
        jPanel4.setLayout(jPanel4Layout);
        jPanel4Layout.setHorizontalGroup(

jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel4Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE, 356,
Short.MAX_VALUE)
                .addContainerGap())
        );
        jPanel4Layout.setVerticalGroup(

jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel4Layout.createSequentialGroup()
                .addComponent(jScrollPane2)
                .addContainerGap())
        );

        jPanel5.setBorder(javax.swing.BorderFactory.createTitledBorder("Message Report"));

        msgreport.setEditable(false);
```

```
        msgreport.setColumns(2);
        msgreport.setHorizontalAlignment(javax.swing.JTextField.LEFT);

        javax.swing.GroupLayout jPanel5Layout = new javax.swing.GroupLayout(jPanel5);
        jPanel5.setLayout(jPanel5Layout);
        jPanel5Layout.setHorizontalGroup(

jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel5Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(msgreport)
                .addContainerGap())
        );
        jPanel5Layout.setVerticalGroup(

jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel5Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(msgreport)
                .addContainerGap())
        );

        jPanel6.setBorder(javax.swing.BorderFactory.createTitledBorder("Operations"));

        enkrypt.setText("Encryption");
        enkrypt.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                enkryptActionPerformed(evt);
```

```
    }
});

klear.setText("Clear All");
klear.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        klearActionPerformed(evt);
    }
});

quit.setText("Quit");
quit.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        quitActionPerformed(evt);
    }
});

jButton1.setText("Email");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

javax.swing.GroupLayout jPanel6Layout = new javax.swing.GroupLayout(jPanel6);
jPanel6.setLayout(jPanel6Layout);
jPanel6Layout.setHorizontalGroup(
```

```
jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel6Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(enkrypt, javax.swing.GroupLayout.PREFERRED_SIZE, 163,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(klear, javax.swing.GroupLayout.PREFERRED_SIZE, 158,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGap(18, 18, 18)
                .addComponent(quit, javax.swing.GroupLayout.PREFERRED_SIZE, 124,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );
        jPanel6Layout.setVerticalGroup(

jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel6Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASE
LINE)
                    .addComponent(enkrypt)
                    .addComponent(klear)
                    .addComponent(quit, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jButton1))
```

```
            .addGap(20, 20, 20))
        );

        eventmode.setBorder(javax.swing.BorderFactory.createTitledBorder("Mode"));

        en.setText("Encrypt");
        en.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                enActionPerformed(evt);
            }
        });

        de.setText("Decrypt");
        de.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                deActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout eventmodeLayout = new
javax.swing.GroupLayout(eventmode);
        eventmode.setLayout(eventmodeLayout);
        eventmodeLayout.setHorizontalGroup(

eventmodeLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
eventmodeLayout.createSequentialGroup()
                .addGap(0, 0, Short.MAX_VALUE)
```

```
.addGroup(eventmodeLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(de)
                .addComponent(en)))
        );
        eventmodeLayout.setVerticalGroup(
eventmodeLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(eventmodeLayout.createSequentialGroup()
                .addComponent(en)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(de))
        );

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
```

```
                    .addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                    .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))


.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)


.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING, false)

                    .addComponent(jPanel4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                    .addComponent(jPanel5, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

                .addGap(0, 0, Short.MAX_VALUE))

            .addGroup(jPanel1Layout.createSequentialGroup()

                .addComponent(eventmode, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)


.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

                .addComponent(jPanel6, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))

        .addContainerGap())

    );

    jPanel1Layout.setVerticalGroup(


jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(jPanel1Layout.createSequentialGroup()

            .addContainerGap()


.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING, false)

                .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
                .addComponent(jPanel5, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

            .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING, false)

                .addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                .addComponent(jPanel4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING, false)

                .addComponent(eventmode, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                .addComponent(jPanel6, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))

        );


        jLabel1.setFont(new java.awt.Font("DialogInput", 1, 24)); // NOI18N

        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

        jLabel1.setText("Enhanced Vertically Scrambled Ceasar Cipher Method");


        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

        getContentPane().setLayout(layout);

        layout.setHorizontalGroup(

            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(layout.createSequentialGroup()
```

```
            .addContainerGap()


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)

                .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))

        );

        layout.setVerticalGroup(

           layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

           .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

            .addContainerGap()

            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 50,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

            .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))

        );


        pack();

    }// </editor-fold>


    private void shiftActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
```

```
    }

    private void quitActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        int showConfirmDialog = JOptionPane.showConfirmDialog(this, "Sure to Exit", "Exit the App", 0);
        //System.out.println(showConfirmDialog);
        if(showConfirmDialog == 0){
            this.dispose();
            System.exit(0);
        }
    }

    private void klearActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        shift.setText("");
        msgreport.setText("");
        ptext.setText("");
        stext.setText("");
        shift.setEnabled(true);
        shift.setEditable(true);

    }

    private void enkryptActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        if(en.isSelected()){
            msgreport.setText("");
```

```
        try{
            shiftkey = Integer.parseInt(shift.getText());
            if(ptext.getText().length() > 0){
                String plaintext = ptext.getText(); //get plaintext
                msgreport.setText("");
                int num_of_matrix = plaintext.length()/36;
                int textpointer = 0; //position of the pointer on the plaintext string
                StringBuilder buffer = new StringBuilder(); //holds the concatenated Text
                if(num_of_matrix < 1){
                    smallmatrix = new String[6][6];//create the small matrix of 6x6


                        int hashValue = 0; //use this fill remaining empty spaces
                        for(int i = 0; i < 6; i++){//filling the matrix with the plaintext
                            for(int j = 0;j < 6;j++){
                                if(textpointer < plaintext.length()){
                                    String charAt = ""+plaintext.charAt(textpointer); //begin to take
character from plaintext
                                    smallmatrix[i][j] = charAt;
                                    textpointer++;
                                }else{
                                    if(hashValue <= 25){
                                        smallmatrix[i][j] = ""+mydictionary.charAt(hashValue);
                                        hashValue++;
                                    }else{
                                        hashValue = 0;
                                        smallmatrix[i][j] = ""+mydictionary.charAt(hashValue);
                                        hashValue++;
                                    }
```

```
        }
    }
}

for(int i = 0; i<6; i++){    //printing the Matrix
    for(int j = 0; j < 6; j++){
        System.out.print(smallmatrix[i][j]+"\t");
    }
    System.out.println("");
}
System.out.println("-----------------------------------------");

int column = 0;
//pointer = 0;
while(column < 6){  //pick a column
    if(column%2 == 0){//gets the 1st, 3rd and 5th  columns
        for(int i=0; i < 6; i++){//pick a row
            String value = smallmatrix[i][column];
            buffer.append(value);
            //pointer++;
        }
```