1. INTRODUCTION

The secrecy of sensitive information against unauthorized access or deceitful changes has been of prime concern throughout the centuries. Before the introduction of digital world, the protection of information to an organization was provided by some physical or administrative means. An example of former is the use of rugged filling cabinets with a combination lock for storing sensitive data. With the introduction of computer, the security of data or information (stored on a computer for a shared system or during their transmission in a distributed system) to maintain its confidentiality, proper access control, integrity and availability has been a major issue [23].

There is only one way to address all these issues, is cryptology. Cryptology is the foundation of all the information security [10]. Cryptology is the science of coding and decoding the secret messages and trying to figure out the secrets is known as cryptanalysis. Modern cryptology is based on ideas from theoretical computer science and increasingly Number Theory (the mathematics of integers). A lot of cryptographic schemes are developed and many of these are broken or some did not work. With the development to the technology today, it is becoming active in cracking the cryptographic system which forces the cryptographers to invent better ciphers.

The cryptographers can make life as difficult as possible for the cryptanalyst by depriving them of some of these things

- Using redundancy reduction before encryption.
- Using an algorithm that is resistant to the cryptanalytic attacks.
- Using a strong enough algorithm that these clues are not really useful.
- Changing keys often and selecting them properly.
- Ensuring that there are not enough computers in the world to do a brute force attack on the algorithm.
- Making sure users of the system understand how to properly use it.

1.1 Introduction and Terminology

Plaintext: This is the original intelligible message or data that is fed into the algorithm as input.

Ciphertext: This is the scrambled text produced as output.

Enciphering or encryption: This is the process by which the plaintext is converted into ciphertext.

Encryption algorithm: The sequence of data processing steps that go into transforming plaintext into ciphertext. In classical cryptography for commercial or other civilian application, the encryption algorithm is made public.

Secret key: The secret key is also input to the encryption algorithm. It is value independent of the plain text, and of the algorithm. The algorithm produces a different output depending on the specific key.

Deciphering or Decryption: This is the process of recovering plaintext from ciphertext.

Decryption algorithm: The sequence of data processing steps that go into transforming ciphertext back into plaintext. In classical cryptography for commercial or other civilian application, the decryption algorithm is made public.

Cryptology: This is the study of techniques of ensuring the secrecy and or authenticity of information. There are main two branches of cryptology.

Cryptography: Cryptography deals with the study of design of the cryptosystems. It is the science of protecting information by encoding it into unreadable format and decoding into readable format in deed.

Cryptanalysis: Cryptanalysis is the study of methods for obtaining the meaning of encrypted information, without access to the secret information that is normally required to do. It is simply defined as “breaking the codes”.

Cryptographic system or cipher: Any single scheme for encryption is referred as cryptographic system or a cipher.

Cryptography is an effective way of protecting sensitive information for storing it on media or transmitting through network communication paths. There are several possibilities of using of the cryptography

- ***Confidentiality***: Confidentiality is the protection of information from unauthorized access or hiding of information from unauthorized individuals. One does not want an eavesdropper to understand the content of transmitted message or stored data.
- ***Authentication***: Authentication is concerned with the assurance to recipient that the message is form the source that it claims to be form. This is equivalent to the signature.
- ***Integrity***: Integrity assures that the messages received as sent have no alteration messages. The alteration of message relates to duplication, insertion, modification, recording, or replay of information.

1.2 Shannon's Conventional Cryptosystem

Let A be finite set of alphabet and |A| denote the cardinality of A. We shall often use $Z_q = \{0, 1, \ldots, q-1\}$ as alphabet, where we work with its elements modulo q. The alphabet Z_{26} can be identified with the set {a, b, ….., z}.

A concatenation of n letters will be called an *n-gram* and denoted by $\mathbf{a} = \{a_0, a_1, a_2, \ldots, a_{n-1}\}$. The special cases are *bi-grams* (n = 2) and *tri-grams* (n = 3). The set of all *n-grams* for A will be denoted by A^n.

A text is an element form $A^* = U_{\geq 0} A^n$.

Let A and B be two finite alphabets. Any one-to-one mapping E of A^* to B^* is called *Cryptographic Transformation*. In most practical situations |A| will be equal to |B| and also often cryptographic transformation E will map *n-grams* into *n-grams*.

Let **m** be the message (a text form A^*) that the sender wants to transmit in secrecy to receiver. This is usually called as *plaintext*. The plaintext will be transform into $\mathbf{c} = E_k(\mathbf{m})$, so called as *ciphertext* and will transmit to sender.

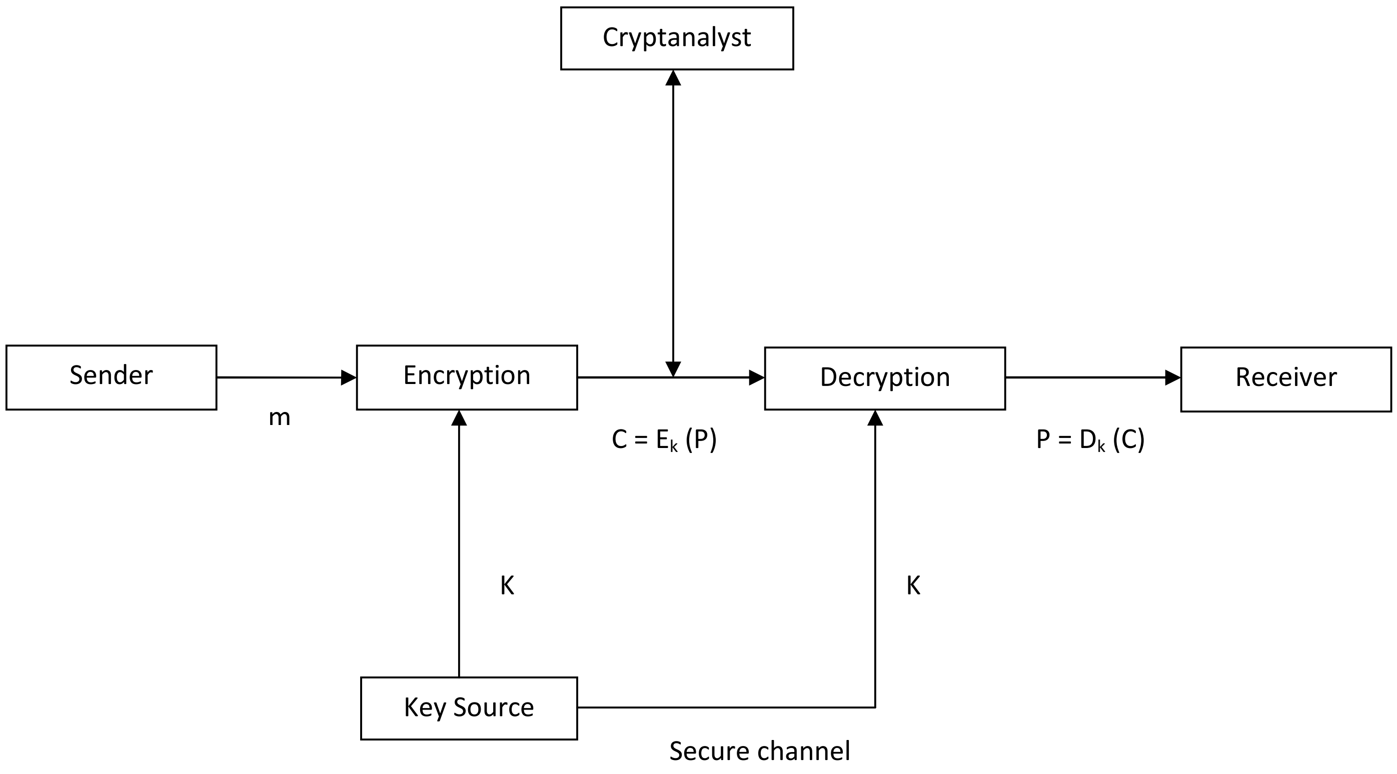

Figure 1: Shannon's Conventional Cryptosystem [10]

Since E_k is one-to-one mapping, its inverse must be exists. We shall denote it D_k. E stands for encryption (or enciphering) and D for decryption (or deciphering). The deciphering is given by $D_k (E_k (\mathbf{m})) = \mathbf{m}$ or $D_k (\mathbf{c}) = D_k (_{Ek} (\mathbf{m}) = \mathbf{m}$.

for all plaintext m € A* and k € K.

Both the sender and receiver must know the particular choice of the key k. They will have agreed on the value of k by means of a so-called, *secure channel*. Normally the cryptosystem will be used for long time, so it is reasonable to assume that this set of cryptographic transformations is also known to cryptanalyst. So, it is required to frequent changes of key to provide the security of data [7, 10].

1.3 Valuation of Secrecy System

There are a number of different criteria that should be applied in estimating or designing the value of a proposed secrecy system.

Amount of secrecy: It is wise to assume that the details of cryptographic algorithm are already available to the attacker in designing the security system. This principle is known as kerckhoffs’s principle – “only secrecy of key provide the security”. Kerckhoffs's principle was reformulated by Claude Shannon as “The enemy knows the system”, known as Shannon’s maxim. Thus, the security of encryption system in most cases is based on the secrecy of the key.

Size of Key

Size of key is very important factor for providing secrecy. The secrecy system may be attacked by using brute force so having large size key system will be strong against brute force approach. On the other hand the key should be transmitted through secure channel. Hence it needs to have to come to some equilibrium point to select size of key.

Complexity

Enciphering and deciphering should be as simple as possible. If they are done manually, complexity leads to loss of time, errors, etc. If done mechanically, complexity leads to large expensive machines.

1.4 Security attacks

The security attacks are classified into passive attacks and active attacks.

Passive attacks

In this attack the attackers try to learn or read the information being stored or transmitted without attempting any change in the message. The goal of the opponent is to obtain the information that is being transmitted. There are two types of passive attacks.

- *Release of message content*: The release of message is concerned with exposure of the confidential information. A telephone conversation, an electronic mail message, and a transferred file may contain sensitive information. We would like to prevent opponent from learning the contents of these transmissions.
- *Traffic analysis*: Traffic analysis is the process of intercepting and examining messages in order to deduce information from patterns in communication. It can be countered by data padding.

Active attacks

Active attacks involve on interruption, modification, and fabrication of the data stream from sender to receiver.

- *Masquerade*: A masquerade is a type of attack where the attacker pretends to be an authorized user of a system in order to gain access to it or to gain greater privileges than they are authorized for. A masquerade may be attempted through the use of stolen logon IDs and passwords, through finding security gaps in programs, or through bypassing the authentication mechanism.
- *Replay*: This involves the passive capture of a data input and its subsequent retransmission to produce unauthorized effect.
- *Modification of message*: This is related to the alteration of some portion of message or that messages are delayed or recorded to produce unauthorized effect.

- *Denial of service*: It prevents or inhibits the normal use or management of communication facilities.

Passive attacks are difficult to detect because there is no alteration of information. So, emphasis of passive attacks is prevention rather than detection. On the other hand, it is quite difficult to prevent active attacks because of the wide variety of potential physical, software and network vulnerabilities [23].

1.5 Motivation

In today's digital world, the invisible ink and paper have been replaced by much more versatile and practical covers for hiding messages like digital documents, images, videos and audio files. Cryptography was mainly used for the security needed for passwords but now it has the important role in protecting the sensitive information passing through the communication channels or stored in the computer. So, it needs a secure system for the security of the information against unauthorized access [19].

Hill Cipher is one of the most famous symmetric cryptosystem that can be used to protect information from unauthorized access. Hill Cipher is polygraph substitution cipher and is strong against frequency letter analysis and the cipher text only attack. But there are some disadvantages of the Hill Cipher. It is very weak against the known-plaintext attack and there could be possibility of the brute-force attack also. It needs the inverse of the key matrix for the decryption process. All the key matrices that can be formed form the elements of the field, couldn't have their inverse i.e. only few elements from the field can be used as the key element. So, there is possibility of the brute-force attack in the Hill Cipher. The finding of the inverse of matrix is complex and computationally expensive when the order of key matrix is greater than 2 and become more complex and more expensive in the computational cost as the order of the key matrix increased [18].

There are many approaches by the researchers to overcome the known-plaintext attack in the Hill Cipher but are very costly in terms of execution. Because of the above considerations, it is our interest to suggest a better algorithm for general use in private sector.

1.6 Approach

It will be better to study the weakness of the Hill Cipher and possible attacks to design the supplement encryption algorithm of the Hill Cipher. The possible attacks on the Hill Cipher are known-plaintext attack and the brute-force attack. To overcome these weakness, the core implementation of the scheme going to be proposed is Cipher Block Chaining (CBC) mode and bitwise exclusive-OR operation. The design criteria chosen for the algorithm is discussed under the proposed solution section and the analysis of the proposed algorithm is discussed under the testing and analysis section of this document.

2. LITERATURE REVIEW

Hill cipher is a multi-letter and polygraph substitution cipher. Although the Hill Cipher is resistant to frequency letter analysis and strong against ciphertext only attack, it succumbs to known-plaintext attack. Many researchers designed many approaches to overcome this attack in the Hill Cipher. In the literature of cryptography, it is well known that, confusion and diffusion play a vital role in the development of a cipher. In 2010, Sastry, Shankar and Bhavani introduced interweaving (transposition of the binary bits of the plaintext characters belonging to the neighboring rows and columns) in each step of the iteration. The interweaving of the resulting plaintext, at each stage of iteration, and multiplication with the key matrix leads to diffusion and confusion [22]. In 2009, Hamamreh and Farajallah introduced a new approach to generate the new key in each step of the encryption process to overcome the known-plaintext attack. In their approach they generate new key not every message but every block of plaintext, with steps has many possibilities of generation and have nonlinear algebra steps to make reverse of steps difficult, the first key used send by encrypted data using public RSA key then no key needed to send. The new method depend on the idea that on encryption side each plaintext char convert into two ciphertext chars, and in the decryption side each tow ciphertext chars convert into one plaintext char, in this way we can use any key matrix (invertible or not), also the key generation that always difficult when key not invertible is solved. Some researchers used the involutory, permuted and reiterative key matrix to overcome this attack in the Hill Cipher. In this process the involutory matrices are generated by modifying the values of a column and a row excepting the corresponding diagonal element is outlined. The permutation of the key matrix is carried out by interchanging the positions of the key elements after each step of encryption. The reiterative defines the repetition of the same encryption process many times i.e. there are number of the rounds of the encryption process. All these lead to the diffusion and confusion and make the system harder to break to the attacker [18]. So, to overcome the known-plaintext attack as well as possibility of brute-force attack and also to improve the cost of execution in the decryption process which is high due to the requirement of inverse of matrix in the Hill Cipher, the scheme going to proposed is using the Cipher Block Chaining (CBC) mode and bitwise wise exclusive-OR operation for encryption and decryption. The detail of the scheme is described in section 5 in this document.

2.1 History of cryptography

The history of the cryptography has undergone many changes throughout the centuries. Cryptography has started with hieroglyphics (an art of decorating tombs to tell the story of the life of the deceased, around 2000 B.C. in Egypt. Cryptography started with carving messages into wood or stone (which were then passed to the intended individual who had the necessary means to decipher the messages) to streams of binary code that are stored into digital devices or passes over network wires.

Around 400 B.C., the Spartans used a system of encrypting information by writing a message on a sheet of papyrus, which was wrapped around a staff. (This would look like a piece of paper wrapped around a stick or wooden rod.) The message was only readable if it was around the correct staff, which allowed the letters to properly match up. This is referred to as the ***scytale*** cipher. When the papyrus was removed from the staff, the writing appeared as just a bunch of random characters.

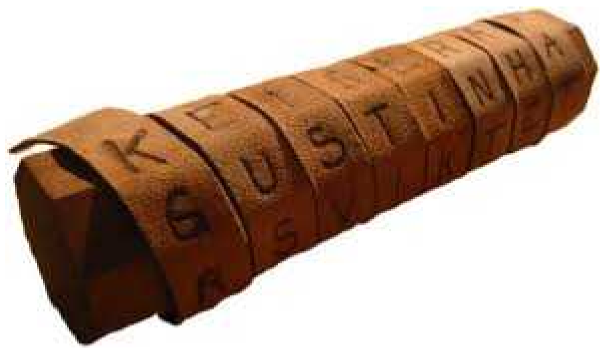

Figure 2: **Scytale used by the Spartans** [24]

The Greek government had carriers run these pieces of papyrus to different groups of soldiers. The soldiers would then wrap the papyrus around a staff of the right diameter and length and all the seemingly random letters would match up and form an understandable message. These could be used to instruct the soldiers on strategic moves and provide them with military directives [21].

The history of cryptography can be broadly divided into three phases.

- From ancient civilizations to the nineteenth century and the first part of the twentieth century, relatively simple algorithms were developed that were designed and implemented by hand.
- Around the period of the World War II, encrypting electro-mechanical machines were used.
- About last fifty years, the use of mathematical algorithms.

In the Bible, there were three encryption techniques, Atbash, it consisted of substituting aleph (the first letter) for tav (the last), beth (the second) for shin (one before last), and so on, reversing the alphabets. Albam, it was similar to Atbash but, instead of considering the whole alphabet, it beforehand divided the alphabet in two and then applied the Atbash rule and Atbah, It used a numerical relation between the original letter (let's call it x) of the alphabet and the one you have to put in place of it (let's call the new letter y):

$x + y = 10$ if x is one of the first 9 letters

$x + y = 45$ if x is one of the last 8 letters

$x + y = 28$ otherwise

There are two kinds of rotation ciphering machines: the Jefferson disk and the Enigma Machine. The Jefferson disk also known as wheel cipher named by third US President Thomas Jefferson, was a cipher using 26 wheels, each with the letters of the alphabet arranged randomly around. Once the order of wheels along the axis has been devised, the user can rotate each wheel up and down until a desired message is spelled out in one row. Then the user can copy a row of text on the wheels other than the one that contains the message. The recipient simply has to put the discs in the agreed-upon order, spell out the encrypted message by rotating the wheels, and then look around the rows until he sees the plaintext message [24].

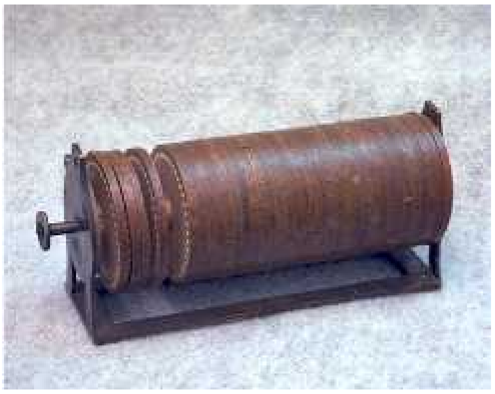

Figure 3: Jefferson Disk [24]

Enigma machine was the ciphering machine used by Germans military force for encryption and decryption of secure messages in the Second World War. Some historians consider the fact that the Allies succeeded in breaking the code so important that they think it was one of the main reasons of the Allies' victory.

Enigma was an electro-mechanical rotor machines. The mechanical mechanism consisted of a keyboard, a set of rotating disks called rotors, arranged adjacently along a spindle, and a stepping mechanism to turn one or more of the rotors with each key press. The mechanical parts acted in such a way as to form a varying electrical circuit: the actual enciphering of a letter was performed electrically [10].

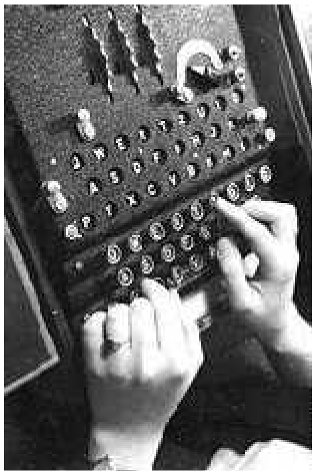

Figure 4: Enigma Machine [24]

2.2 Cipher model

2.2.1 Types of Cipher

There are two basic types of encryption ciphers, substitution and transposition.

2.2.1.1 Substitution Cipher

A substitution technique is one in which the letters of plain text are replaced by other letter or by numbers or symbols [23].

In a simple substitution one choose a fixed permutation ⇔of the alphabets [a, b, …..,z] and applies that to all the letters in the plain text [10].

Let us consider a mathematical function StringReplace

StringReplace [“plaintext” {“a”| ”k”, “e” | ”z”, “I” | ”b”, “l” | ”r”, “n” | ”a”, “p| ”v”, “t” | ”d”}]

Result: vrkbeqzdq

A more formal description of the simple substitution is as follows:

The key space K is the set S_q of all the permutations of {0, 1, …..,q-1} and cryptosystems η is given by

$$\eta = \{ \eta_{\Leftrightarrow} \ \Leftrightarrow \ S_q \}$$

$$\text{Where, } \eta_{\Leftrightarrow} = \Leftrightarrow(m), \qquad 0 \text{ ™} m \text{ ™} q$$

2.2.1.2 Transposition Cipher

Transposition is carried out by changing of character position in the plaintext to generate some ciphertext. In this technique mapping is achieved by performing some sort of permutation on the plaintext letters [23].

This is completely a different way of enciphering. Plaintext is broken into fixed length, say n, and applies a fixed permutation ∃. For instance, with n = 6 and ∃ = 2 5 3 1 6 4,

Plaintext:	thisistranspositionciphertest
Ciphertext:	isihtttonriaiiespoehrsctisiht

The plaintext is written row wise in a matrix of given size, but will be read out column wise in a specific order depending on a keyword [10].

Key	2	5	3	1	6	4
Plaintext	t	h	i	s	i	s
	t	r	a	n	s	p
	o	s	i	t	i	o
	n	c	i	p	h	e
	r	t	e	s	t	

The multistage transposition makes the cipher more secure [1].

2.2.1.3 Running and Concealment Cipher

Running and concealment ciphers are most of the spy-novel-ciphers.

The running key cipher is also known as book cipher. It could use key that does not require an electronic algorithm or bit alteration. The key is usually paragraph, page numbers etc. It uses steps in physical world around us, like books (page number, line number and word count). Each word is described by a sequence of numbers. In the concealment cipher word are taken from fixed interval from a text message and are arranged as a sentence [21].

A message is embedded (hidden) within a seemingly innocuous piece of information. For example

> A treatise to analyze campus knowledge of undergraduate
> tenants portrays outright sympathy totally aimed toward
> negligent idiots notwithstanding elegant amoral mentalities.
>
> **A t**reatise **t**o **a**nalyze **c**ampus **k**nowledge **o**f **u**ndergraduate
> **t**enants **p**ortrays **o**utright **s**ympathy **t**otally **a**imed **t**oward
> **n**egligent **i**diots **n**otwithstanding **e**legant **a**moral **m**entalities.
>
> **Result**: Attack at post at nine am.

2.2.1.4 Steganography

Steganography, derived from Greek, literally means “Covered Writing” is the art of hiding information in ways that prevent the detection of hidden message. Steganography and Cryptography are cousins. Cryptography scrambles a message that makes it difficult to understand where as steganography hides the message that can’t be seen [16].

The information-hiding process in a steganographic system starts by identifying a cover medium’s redundant bits. The redundant bits are the bits of cover medium that can be modified without destroying the medium’s integrity [15].

The basic model of steganography consists of Carrier, message, embedding algorithm and stego key. The carrier is the object in which the message is embeded by using embedding algorithm along with stego key (a type of password may also be used to hide, then later to decode the message). The covered object with secretly embedded message is known as stego object [16].

History of Steganography

- During the World War II invisible ink was used to write information on piece of paper so that the paper appeared to the average person just being blank piece of paper. On heating with liquids such as vinegar, milk, some fruit juices, urine, get darken and become visible to the human eye.
- In Ancient Greece they used to select messengers and shave their head, they would then write a message on their head. Once the message had been written the hair was allowed to grow back. After the hair grew the messenger was sent to deliver the message, the recipient would shave off the messenger hair to see the secret message [2].

2.3 Methods of Cryptography

2.3.1 Symmetric Cryptosystem

Symmetric cryptosystem also known as conventional encryption or single key encryption was only the encryption in use prior to the development of public-key encryption in 1976. Symmetric cryptography is also known as secret key cryptography because this type of encryption relies on each user to keep the key secret and properly protected.

The symmetric encryption has five ingredients.

Plain text: This is the original intelligible message or data that is fed into the algorithm as input.

Encryption Algorithm: The algorithm performs various substitution and transformations on the plain text.

Secret Key: The secret key is also input to the encryption algorithm. It is value independent of the plain text, and of the algorithm. The algorithm produce a different output depending on the specific key.

Cipher text: This is the scrambled text produced as output.

Decryption Algorithm: It is the algorithm that runs in reverse of the encryption algorithm.

There are two requirements for a symmetric key cryptosystem.

1. The encryption and decryption algorithms need not to be kept secret but it need to keep the secret key secret.
2. Sender and receiver must have obtained copies of secret key in a secure fashion and must keep the key secure [11].

Advantages and Disadvantages

The symmetric cryptosystems are very fast and can be hard to break if the size of key using is large.

As both the users use the same key to encrypt and decrypt message, symmetric cryptosystems can provide confidentiality, but can't provide authentication or non repudiation. It also requires a secure mechanism to deliver the keys properly (key distribution problem) [21].

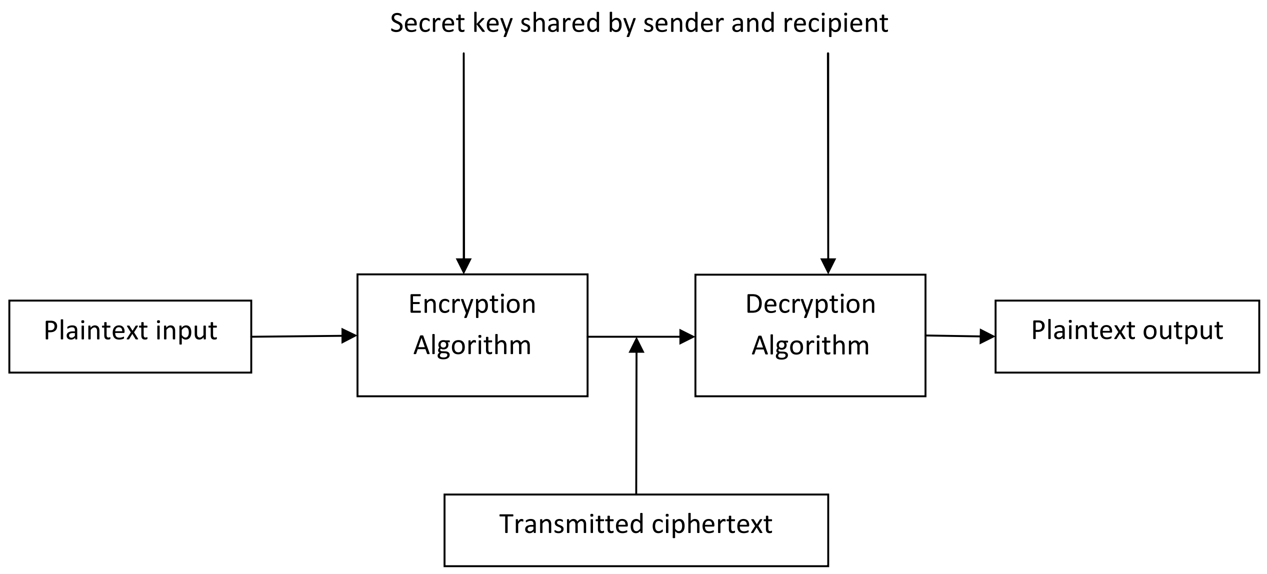

Figure 5: A Simplified Model of Symmetric Encryption [11]

2.3.1.1 Caesar Cipher

The Caesar is a linear cipher which transforms plaintext into ciphertext by applying a set of transformations to each character (or letter) in the plaintext. The particular transformations at any time are controlled by a key used at that time [8].

When we talk cryptography these days, we usually refer to the encryption of digital messages, but encryption actually predates the computer by quite a long period. One of the best examples of early cryptography is Caesar Cipher, named after Julius Caesar [9]. In this cipher each character of a message is replaced by a character three position down in the alphabet.

Plaintext:	are you ready
Ciphertext:	DUH BRX UHDGB

Each letter of the alphabet is represented by an integer that corresponds to its position in the alphabet.

Plaintext	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
Numeric values	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

For each plaintext letter P, the encryption is give by

$$C \quad = \quad E\,(3, P) \quad = \quad (p + 3) \bmod 26$$

A more general version of Caesar Cipher that allows for any degree of shift would be expressed as

$$C \quad = \quad E\,(k, P) \quad = \quad (P + k) \bmod 26$$

Where k is shift value that may have value between 0 and 25.

And decryption is given by

$$P = D(k, C) = D(C - k) \bmod 26$$

Symbols:

P	Plaintext
C	Ciphertext
E	Encryption
D	Decryption

If it is known that a given ciphertext is a Caesar Cipher then, a brute-force cryptanalysis is easily performed by simply trying all the 25 possible keys [23].

2.3.1.2 Monoalphabetic Ciphers

In the Caesar cipher the encryption or decryption is performed by just shifting the characters with only 25 possible keys, which is far from secure.

It can be made more secure by randomizing the alphabet rather than using shift alphabet. This is known as a general monoalphabetic cipher [4].
In monoalphabetic cipher, substitution characters are a random permutation of the 26 letters of the alphabet. The key is the sequence of substitution letters and there are 26! permutations of the alphabets. So there are greater than $4 * 10^{26}$ possible keys.

The large key space of monoalphabetic cipher would rule out a brute-force attack but could be broken with relative frequency of the letters.

2.3.1.3 Block and Stream Ciphers

The symmetric cryptosystem is further divided into block ciphers and stream ciphers. Block ciphers work on blocks of plaintext and ciphertext, whereas stream ciphers work on stream of plaintext and ciphertext, one bit or byte at a time [23].

2.3.1.3.1 Block Ciphers

Block ciphers operates on block of plaintext and produce the block of ciphertext of equal size. The message is divided into blocks of bits. The block size is usually 64 or 128 bits.

It has been said that the properties of a cipher should contain confusion and diffusion. The different unknown key values can cause confusion, because the intruder does not know these values, and diffusion is accomplished by putting the bits within the plain-text through many different functions so that they are dispersed throughout the algorithm.

2.3.1.3.2 Stream Ciphers

Stream cipher operates on streams of plaintext and ciphertext. It does not divide a message into blocks. A stream cipher treats the message as a stream of bits or bytes and performs mathematical functions on them individually. It usually uses 1 bit or 1 byte. The same plaintext bit or byte will be transformed into ciphertext bit or byte each time it is encrypted. It uses a *key stream generator*, which produces a stream of bits that is XORed with the plaintext bits to produce ciphertext.

Example:

Message stream	1001010111
Key stream	0011101010
Ciphertext stream	1010111101

If the crypto system was only dependent upon the key stream generator, an attacker could get a copy of the plaintext and resulting ciphertext, XOR them together, and find the key stream to use in decrypting other messages [21].

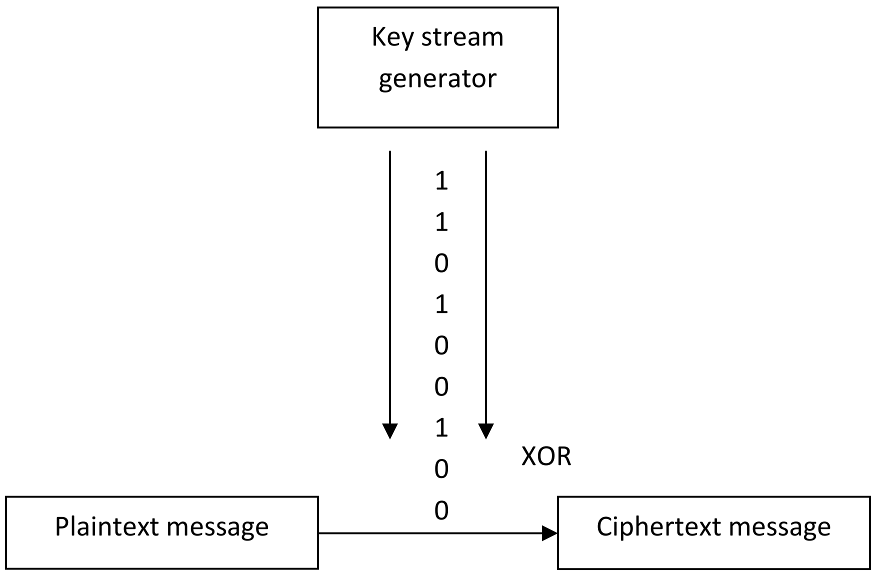

Figure 6: Stream Cipher [21]

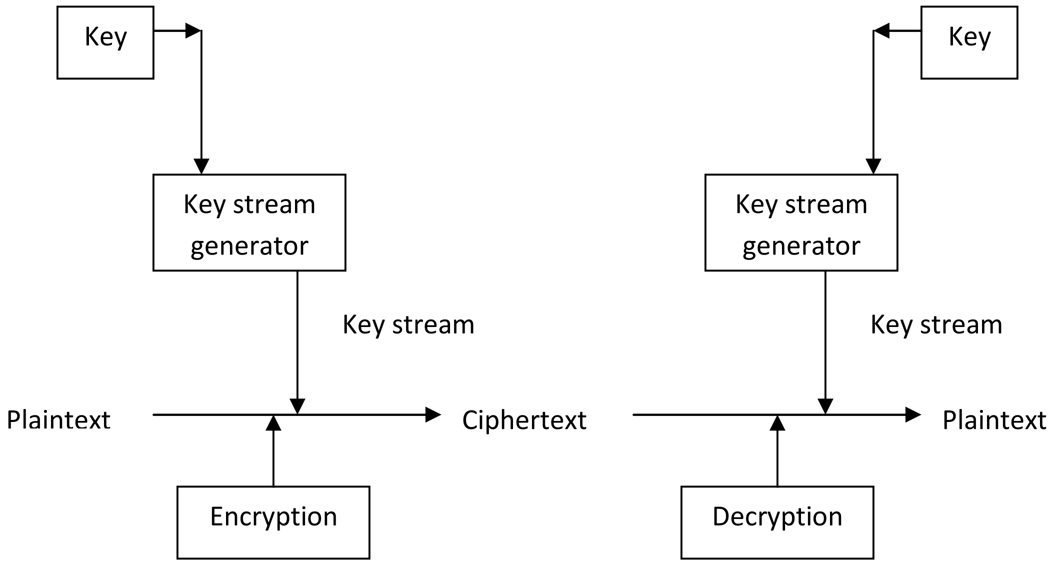

Figure 7: A key stream and key are needed for encryption and decryption [21]

2.3.1.4 Polygraphic Ciphers

A ploygraphic cipher is cipher in which two or more letters are substituted at once. The main purpose of polygraphic ciphers is to reduce the attack by frequency analysis of letters. Often today polygraphic ciphers are called block ciphers because they encrypt blocks of plaintext with block of ciphertext. [5].

Diagrams and Trigrams

The pairs of adjacent characters are referred as to diagrams, and triples of characters are referred to as trigrams. The messages are divided into groups of words with two or three characters and are substituted into cipher text [21].

2.3.1.4.1 Playfair Ciphers

Playfair cipher is a famous digraphic cipher. It treats digrams in the plaintext as single unit and translates these units into ciphertext. The Playfair cipher was first invented by Charles Wheatstone in 18th century but it has heavily used and popular by Lord Playfair. This cipher mainly relied on polyalphabetic cipher [17].

The Playfair cipher is based on the use of 5 * 5 matrix of letters constructed using a key. The key is entered in the cells of a matrix in left to right fashion starting with the first cell at the top corner and rest of cells are filled with the remaining letters. The letters I and J are assigned the same cell. For example the key is "mynepal".

M	Y	N	E	P
A	L	B	C	D
F	G	H	I / J	K
O	Q	R	S	T
U	V	W	X	Z

Figure 8: A Playfair Cipher Matrix Box

In Playfair cipher the plaintext are encrypted two letters at a time, according to the following rules.

- Two plaintext letters that fall in the same row of the 5*5 matrix are replaced by letters to the right of each in the row. The “rightness” property is to be interpreted circularly in each row (i.e. the first entry in each row is to the right of the last entry). Therefore, the pair of letters “ld” will get replaced by “BA”.
- Two plaintext letters that fall in the same column are replaced by the letters just below them in column. The “belowness” property is to be considered circular (i.e. the topmost entry in a column is below the bottom-most entry). Therefore, the pair of letters “lq” will get replaced by “GV”.
- Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Therefore, the pair of letters “ls” will get replaced by “CQ” and “oe” will get replaced by “SM”.
- The duplicates in key must be dropped.
- For the repeating letter in the plaintext, it must be insert a chosen “filler” letter (let say it is ‘x’). So a plaintext word such as “letter” becomes “letxter”.

There are only 26 letters; there is 26 X 26 = 676 diagrams will be produced so it was very difficult to identify the particular structure [17]. The Playfair cipher is a great advance over simple monoalphabetic ciphers and was thought to be unbreakable to many years. It was used as the encryption system by the British Army in World War I. It was also used by the U.S. Army and other Allied forces in World War II.

Despite this level of confidence in its security, it can be easily cracked if there is enough text. Calculating the key stream can be very easy if plaintext and ciphertext are known [17, 23].

2.3.2 Asymmetric Model

Asymmetric cryptography also known as public-key cryptography, is cryptography in which a pair of keys is used to encrypt and decrypt a message in which one key is used to encrypt the plaintext and the other is used to decrypt the ciphertext. One of these keys is published or made known to everyone, called as public-key and the other is kept secret by owner, called as private key [23].

The ingredients of the asymmetric model are as follows:

Plain text: This is the original intelligible message or data that is fed into the algorithm as input.

Encryption algorithm: The algorithm performs various substitution and transformations on the plain text.

Encryption key: The encryption key is also input to the encryption algorithm. It may be value dependent of the plain text, and of the algorithm. The algorithm produces a different output depending on the specific key.

Cipher text: This is the scrambled text produced as output after the encryption process.

Decryption algorithm: It is the algorithm that runs in reverse of the encryption algorithm.

Decryption key: The decryption key is input to the decryption algorithm along with the cipher text. The algorithm recovers the plain text from the cipher.

The public key and private key are mathematically related and the keys can’t be derived from each other. Messages are encrypted with recipient’s public key and can be decrypted with only the corresponding private key [21].

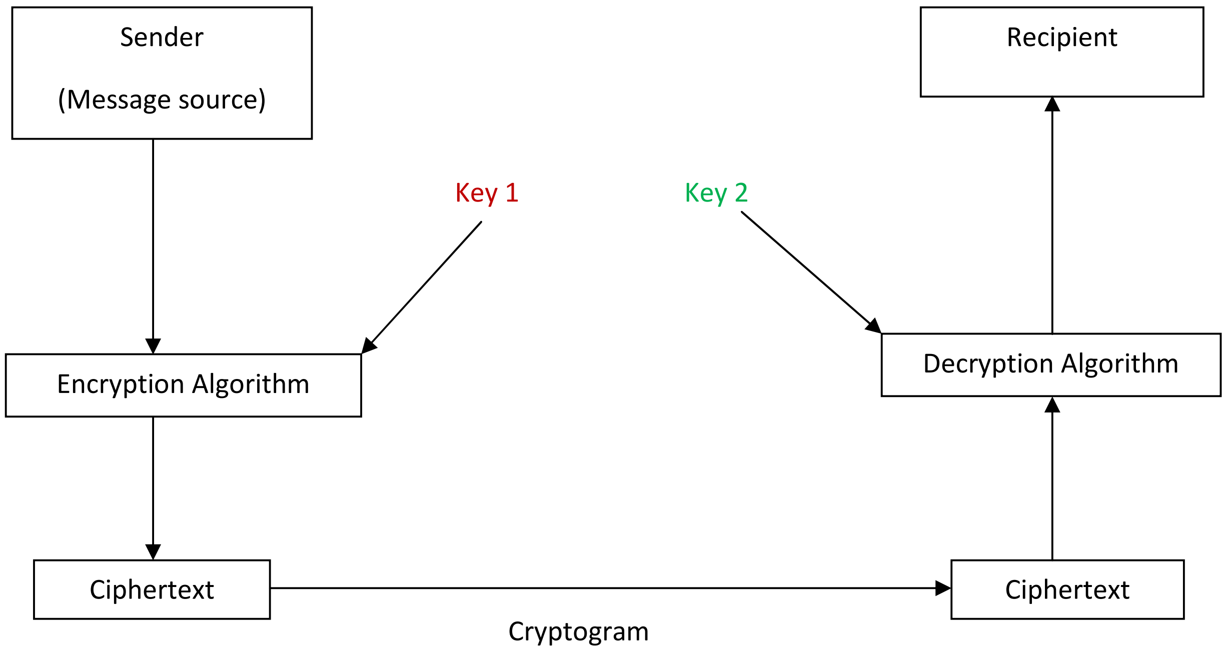

Figure 9: Asymmetric Cryptogram [21]

2.4 Threat Models

Application security has become a major concern in recent years. Attackers are using new techniques to gain access to sensitive data, crack the application. They try to get the key or plaintext structure from ciphertext applying these techniques. Threat modeling is one of the methods which is used to implement in design process of an application [20].
The attacking techniques may vary in different schemes; but, by the study of historical background of cryptography, the possible attacks can be broadly classified into following models [12].

Black-box model

In the traditional black-box model, the attacker is restricted to observe input and output of the algorithm, without any side-channels of information. A secret key of a cryptographic algorithm is hidden in the black-box and is never exposed. The security depends on the strength of the cryptographic algorithm.

Grey-box model

Another model is the grey-box model where an attacker is also able to monitor side effects of the program execution. For example, an attacker can monitor the execution time, power consumption, and electromagnetic radiation.

White-box model

In the white-box model, the attacker also has total visibility into software implementation and execution. *White-box cryptography* is aimed at protecting secret keys from being disclosed in a *software* implementation. To prevent an attacker from finding the key, the key needs to be hidden in the implementation.

2.5 Cryptanalytic Attacks

Cryptanalysis is the study of methods for obtaining the meaning of encrypted information, without access to the secret information that is normally required to do [6]. The main goal of the attacker is to obtain the key used in the scheme. The precise methods used for cryptanalysis depend on whether the "attacker" has just a piece of ciphertext, or pairs of plaintext and ciphertext, how much structure is possessed by the plaintext, and how much of that structure is known to the attacker [1].

The types of legitimate attacks can be classified in three categories [13].

2.5.1 Ciphertext-onlyAttack

This type of attack involves the cipher text and the producing algorithm only. The cryptanalyst intercepts one or more messages all encoded with the same encryption key i.e. try hit and trial method to obtain the key.

2.5.2 Known Plaintext Attack

The attacker has knowledge of encryption algorithm and tries to obtain the key with one or more plaintext-ciphertext pairs known.

2.5.2 Chosen Plaintext Attack

This type of attack involves the algorithm, ciphertext generated by the encryption algorithm and some plaintext chosen by the attacker.

2.6 Brute-force Attack

A brute-force attack or exhaustive key search involves systematically checking all the possible keys until the correct key is found when encryption and decryption algorithms are publically available [1]. It is very costly and on average, half of all possible key must be tried to achieve the success. To prevent the encryption scheme from brute force attack, the possible numbers of key should be large, which is possible with the sufficiently large key space.

2.7 Cryptographic Principles

Designing the cryptographic scheme is notably difficult. There are various principles, regarding the design of the cryptographic scheme.

2.7.1 Kerkhoff’s Principle

A fundamental assumption in cryptanalysis was first stated by A. Kerkhoff in the nineteenth century [10]. It states that the adversary knows all the details of the cryptosystem, including algorithms and their implementations i.e. the security of a cryptosystem must be entirely based on the secret keys.

2.7.2 Unconditionally Secure Scheme

An encryption scheme is said to be unconditionally secure if the cipher text generated by the scheme does not contain enough information to determine the plain text uniquely [23].

2.7.3 Computationally Secure Scheme

An encryption scheme is said to be computationally secure if at least one of the following holds:

- The cost of breaking the cipher text exceeds the value of encrypted information.
- The time required to break the cipher text exceeds the useful life time of the information [23].

2.8 Galois Field

A Field {F, +, ×}, is a set of elements with two binary operations called addition and multiplication such that F is an integral domain and multiplicative inverse of the elements must exist in F. A finite field with order as power of prime p^n, where n is positive integer and p is prime, is denoted by GF (p^n) and is known as Galois Field. Hill Cryptosystem is based on Galois Filed in which matrix multiplication and modular arithmetic operations are used to obtain the scrambled text [23].

3. ANALYSIS OF ALGORITHM

3.1 Background

Additive cipher: The encryption is carried out by adding key value to the plaintext value and decryption is carried out by adding the additive inverse of key to the cipher text.

$$c = p + (key) \bmod n$$

$$p = c + (-key) \bmod n$$

Multiplicative cipher: The encryption is carried out by multiplying the key with the plaintext and decryption is carried out by multiplying with multiplicative inverse of the key.

$$c = p * (key) \bmod n$$

$$p = c * (key)^{-1} \bmod n$$

Affine cipher: The encryption is carried out by combining multiplicative and additive cipher.

$$c = p * (multiplicative\ key) + (additive\ key) \bmod n$$

Each of these can be easily attacked by frequency analysis. It is easy to spot high frequency letter (e, t, a, o, i, n, s). One way to destroy the frequency analysis is to encrypt a string of letters as a block.

3.2 Hill Cipher

Hill cipher is a multi-letter cipher developed by Lester Hill in 1929. It is a polygraph substitution cipher using multiple substitution of alphabets in which the same plaintext letters are replaced with different ciphertext letters which makes it difficult to crack [23].

The encryption is carried out by taking m successive plaintext letters and substituted with m ciphertext letters. The substitution is determined by m linear equations in which each character assigned a numeric value a = 0, b = 1, …….., z = 25.

For m = 3, the system can be described as

$C_1 = (K_{11}P_1 + K_{12}P_2 + K_{13}P_3) \bmod 26$

$C_2 = (K_{21}P_1 + K_{22}P_2 + K_{23}P_3) \bmod 26$

$C_3 = (K_{31}P_1 + K_{32}P_2 + K_{33}P_3) \bmod 26$

This can be represented in the column vector and matrices form as

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} \begin{matrix} = \\ = \\ = \end{matrix} \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} \bmod 26$$

$C = E(K, P) \bmod 26 = KP \bmod 26$

Where C and P is column vector of length 3 represents the ciphertext and plaintext respectively and K is the key, is a square matrix of size 3.

Decryption: The decryption of the Hill Cipher is obtained by multiplying the inverse of the key matrix with the column vector of the ciphertext.

$P = D(K, C) \bmod 26 = K^{-1}C \bmod 26 = K^{-1}KC = P$

$K^{-1}K = I$, where I is Identity matrix.

3.3 Cryptanalysis of Hill Cipher

1. Hill Cipher completely hides the single-letter frequency analysis if the plaintext block contains two characters at a time and use of large matrix hides more frequency information [23].

2. Hill cipher is a polygraph substitution cipher using multiple substitution alphabets in which the same plaintext letters are replaced with different ciphertext letters, it is strong against the cipher-text only attack [14].

3. In the literature of the cryptography, it is well known that the confusion and diffusion makes the block cipher more strong. The change of just one or two characters in plaintext block can completely change the corresponding ciphertext and this dispersal of characters defines the diffusion in Hill Cipher and makes the cipher strong [22].

4. In the Hill cipher, the decryption needs the inverse of the key matrix and not all of the elements of the integers modulo 26 (i.e. 1, 2, ….., 26) could be used as the key elements. Only the elements which are relatively primes to 26 and have multiplicative inverses modulo 26 can be used. There are only 12 elements that are relatively prime to 26 have multiplicative inverses modulo 26 (1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25,). Thus, there is possibility of the **brute-force attack**.

5. Hill cipher is easily broken with **known plaintext attack**. For an m * m Hill cipher, let us take m plaintext-ciphertext pairs, each of length m and unknown key matrix K.

   $Y = KX$

   Where Y and X are ciphertext and plaintext column vectors respectively.

   If X has an inverse, then it can be determined $K = YX^{-1}$.

   Suppose the plaintext "FRIDAY" is encrypted using a 2 * 2 Hill cipher to yield the ciphertext PQCFKU. Thus, we know that [23]

$$K \begin{pmatrix} 5 \\ 15 \end{pmatrix} \bmod 26 = \begin{pmatrix} 15 \\ 16 \end{pmatrix}$$

$$K \begin{pmatrix} 8 \\ 3 \end{pmatrix} \bmod 26 = \begin{pmatrix} 2 \\ 5 \end{pmatrix}$$

$$K \begin{pmatrix} 0 \\ 24 \end{pmatrix} \bmod 26 = \begin{pmatrix} 10 \\ 20 \end{pmatrix}$$

Using the first two plaintext-ciphertext pairs, we have

$$\begin{pmatrix} 15 & 2 \\ 16 & 5 \end{pmatrix} = K \begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix} \bmod 26$$

The inverse of X can be computed as

$$\begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix}^{-1} = \begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix}$$

So,

$$K = \begin{pmatrix} 15 & 2 \\ 16 & 5 \end{pmatrix} \begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix} = \begin{pmatrix} 137 & 60 \\ 149 & 107 \end{pmatrix} \bmod 26 = \begin{pmatrix} 7 & 8 \\ 19 & 3 \end{pmatrix}$$

Thus, in Hill cipher, there are possibilities of brute-force attack and known plaintext attack.

4. PROBLEM DEFINITION

Computers today have become inseparable part of our lives. People are extensively depended on computer in storing or transferring information through network. It is the prime concern to maintain the secrecy of the private or sensitive data against unauthorized access or deceitful changes as the possibilities of attacks are rising at alarming rate every year.

The secrecy system is mainly concern with the confidentiality, authentication, integrity, non-repudiation. The issues of the secrecy systems furnish an interesting application of communication theory and can be addressed by cryptography. The main goal of the cryptography is to keep the plaintext secret from eaves-droppers trying to get the some information of the plaintext. One of the most difficult and least understood problems at the communication is the enhancement of security. Though many algorithms are present, some better algorithms in terms of security are very costly in terms of execution. Asymmetric encryption techniques, in which different keys are used for encryption and decryption are almost thousands times slower than symmetric techniques because they require more computational processing power. Even if the algorithm is very complex and thorough, there are other issues that can weaken the strength of encryption scheme. An extremely strong algorithm can be used, using a large key space and a large and random key value.

Hill cipher is a multi-letter and polygraph substitution cipher using multiple substitution alphabets. It uses the matrix multiplication. It addresses the diffusion in the encryption scheme. With the change of characters position in the plaintext block, changes the pattern of the ciphertext. Although, the scheme is strong on frequency analysis, ciphertext-only attack, it is very weak against known plaintext attack. For the decryption, it needs inverse of the key and not all the elements from the key space can be used in the generation of the key. So, there is possibility of the brute-force attack. To come over the brute-force attack, the key space can be made large by generating larger sized key matrices. With increase in the matrix size again increase the computational cost.

By understanding the problems of the secrecy system, problem in confidentiality and other various attacking techniques, it is worthy and important to introduce the new systems or modify the existing schemes. The main goal of this research is to design a computationally secure and efficient symmetric scheme for maintaining the secrecy in data storage or in communication system based on Hill cipher matrix manipulation and XOR operation with Cipher Block Chaining mode of operation.

5. PROPOSED SOLUTION

5.1 Cipher Block Chaining (CBC)

In the Cipher Block Chaining (CBC) mode of operation, the results of the previous blocks are fed back into the encryption of the current block. The plaintext is XORed with the previous ciphertext block before it is encrypted.

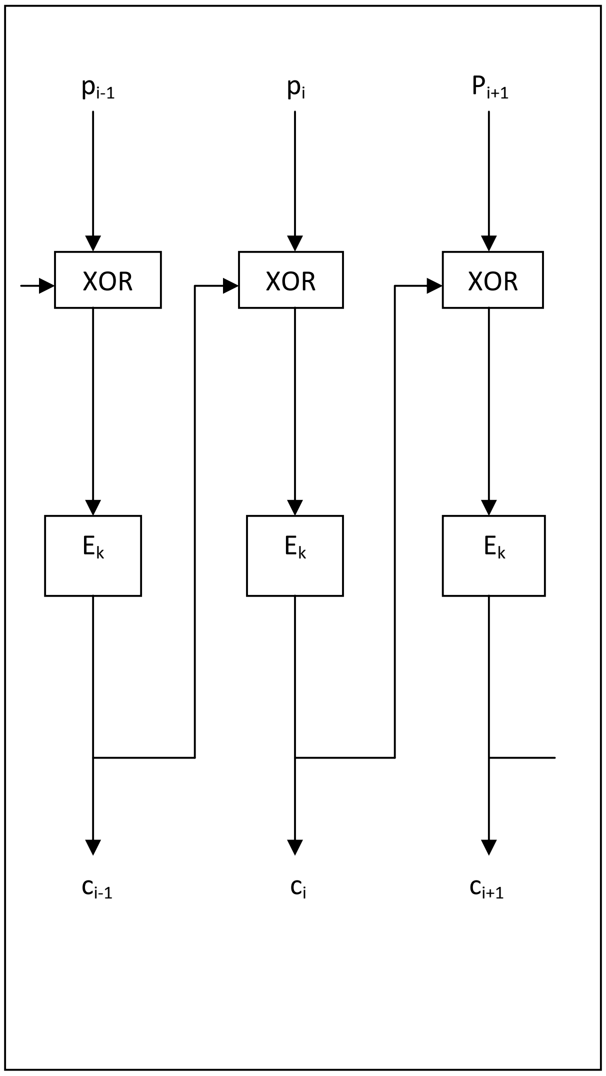

Figure 10: CBC Encryption [21]

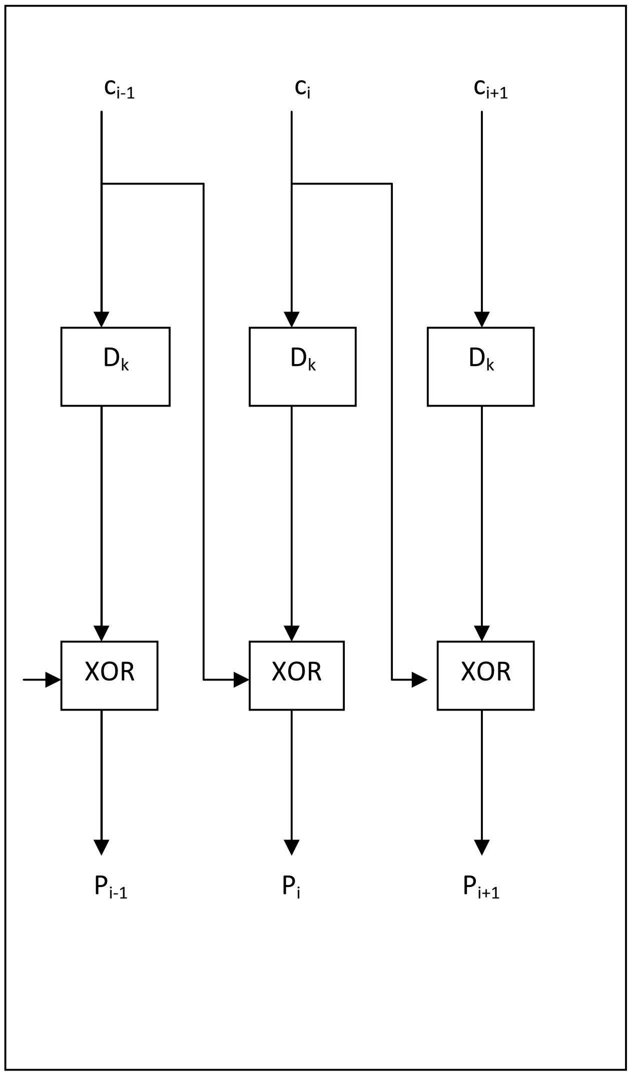

Figure 11: CBC Decryption [21]

5.2 Development of Cipher

In the proposed scheme, the encryption algorithm takes m successive plaintext letters and substitutes into m successive ciphertext letters.

A plaintext block P consists of 2n characters. P can be represented in matrix form as

$$P = [p_{ij}], \qquad I = 1 \text{ to } n, \qquad j = 1 \text{ to } 2.$$

For n =3, the plaintext block can be represented as

$$P = \begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \\ p_{31} & p_{32} \end{pmatrix}$$

Where, p_{11}, p_{12}, …., represents alphabetic characters of plaintext block.

Each element $[p_{ij}]$ of plaintext P is then replaced by the corresponding numeric value.

$$\begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \\ p_{31} & p_{32} \end{pmatrix} = \begin{pmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \\ N_{31} & N_{32} \end{pmatrix}$$

Where, N_{11}, N_{12}, ….., represents the numeric values in the plaintext block.

The numeric values of elements of the plaintext block are then converted into binary forms. Let m be the length of the binary string of each element in plaintext block.

$$\begin{pmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \\ N_{31} & N_{32} \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix}$$

Where, b_{11}, b_{12}, ….., represents the binary elements in the plaintext block.

Now, the elements of the plaintext block are concatenated row wise. This results the plaintext column vector. Here, the length of the binary string of each element is 2m.

$$\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} b_{11}b_{12} \\ b_{21}b_{22} \\ b_{31}b_{32} \end{pmatrix} = \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix}$$

Where, P_1, P_2, P_3 represents the binary elements in the plaintext column vector.

The key is the square matrix. Let us consider the key matrix of order 3. The elements of key matrix are represented by the binary form of length 2m.

$$K = \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix}$$

The exclusive-OR operation is carried out between the key matrix and the plaintext block vector which can be described as

$$C_1 = ((K_{11} \oplus P_1) \oplus K_{12}) \oplus K_{13}$$

$$C_2 = ((K_{21} \oplus P_2) \oplus K_{22}) \oplus K_{23}$$

$$C_3 = ((K_{31} \oplus P_3) \oplus K_{32}) \oplus K_{33}$$

The XOR operation results the ciphertext column vector and is represented by

$$C = \begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix}$$

Where, C_1, C_2, C_3 represents the binary elements of length 2m in the ciphertext column vector.

Now, the ciphertext column vector elements are further processed to get the 2n successive ciphertext characters.

The binary strings of each element in the ciphertext column vector are split into two half. This results the ciphertext block with the 2n successive ciphertext elements of binary string of length m.

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} b_{11}b_{12} \\ b_{21}b_{22} \\ b_{31}b_{32} \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix}$$

The binary value of each element in the ciphertext block is converted into corresponding numeric values.

$$\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \\ N_{31} & N_{32} \end{pmatrix}$$

The permutation of elements is done here in the ciphertext block. The permutation is carried out by interchanging the position of numeric elements in the ciphertext block. After applying permutation, each element in the ciphertext block is replaced with the corresponding character value.

$$\begin{pmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \\ N_{31} & N_{32} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{pmatrix}$$

Sub key generation

The binary elements of the ciphertext column vector are used as the elements of the key to perform the Cipher Block Chaining (CBC) mode operation in the next encryption process. The CBC operation is carried out by replacing any row or column of the key matrix by these ciphertext column vector elements. This generates the new key which is used for the next encryption operation. In this way after each encryption step a new key is generated which address the properties of confusion and makes the cipher more strong against known-plain text attack.

Decryption

Here, the ciphertext block can be represented as

$$C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{pmatrix}$$

Where, c_{11}, c_{12}, …., represents alphabetic characters of ciphertext block.

Each element $[c_{ij}]$ of ciphertext C is then replaced by the corresponding numeric value.

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{pmatrix} = \begin{pmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \\ N_{31} & N_{32} \end{pmatrix}$$

Where, N_{11}, N_{12}, ….., represents the numeric values in the ciphertext block.

The permutation of elements is done here in the ciphertext block. The pattern of permutation is same which is applied during the encryption process.

The numeric values of elements of the ciphertext block are then converted into binary forms. Here, the length of the binary string of each element in ciphertext block is m.

$$\begin{pmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \\ N_{31} & N_{32} \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix}$$

Where, b_{11}, b_{12}, ….., represents the binary elements in the ciphertext block.

Now, the elements of the ciphertext block are concatenated row wise. This results the ciphertext column vector. Here, the length of binary the string of each element is 2m.

$$\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} b_{11}b_{12} \\ b_{21}b_{22} \\ b_{31}b_{32} \end{pmatrix} = \begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix}$$

Where, C_1, C_2, C_3 represents the binary elements in the ciphertext column vector.

The exclusive-OR operation is carried out between the key matrix and the plaintext block vector which can be described as

$$P_1 = ((K_{11} \oplus C_1) \oplus K_{12}) \oplus K_{13}$$

$$P_2 = ((K_{21} \oplus C_2) \oplus K_{22}) \oplus K_{23}$$

$$P_3 = ((K_{31} \oplus C_3) \oplus K_{32}) \oplus K_{33}$$

The XOR operation results the plaintext column vector and is represented by

$$P = \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix}$$

Where, P_1, P_2, P_3 represents the binary elements of length 2m in the plaintext column vector.

Now, the plaintext column vector elements are further processed to get the 2n successive plaintext characters.

The binary string of each element in the plaintext column vector are split into two half. This results the plaintext block with the 2n successive plaintext elements of binary string of length m.

$$\begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} = \begin{pmatrix} b_{11}b_{12} \\ b_{21}b_{22} \\ b_{31}b_{32} \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix}$$

The binary value of each element in the ciphertext block is converted into corresponding numeric values.

$$\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \\ N_{31} & N_{32} \end{pmatrix}$$

The numeric value of each element in the ciphertext block is then converted into corresponding character value. This, results the original plaintext characters.

$$\begin{pmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \\ N_{31} & N_{32} \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \\ p_{31} & p_{32} \end{pmatrix}$$

Sub key generation

The binary elements of the ciphertext column vector are used as the elements of the key to perform the Cipher Block Chaining (CBC) mode operation in the next decryption process. The CBC operation is carried out by replacing any row or column of the key matrix same as in the encryption process.

5.3 Algorithms

5.3.1 Algorithm for Encryption

```
1.  Read P, K, n, m

2.  Calculate Plaintext length, N

3.  While(N >= m)
    {
            for (I = 0; I < n; i++)
            {
                    XORed operation
            }
    }
4.  Write C
```

5.3.2 Algorithm for Decryption

```
1.  Read C, K, n, m

2.  Calculate Ciphertext length, N

3.  While(N >= m)
    {
            for (I = 0; I < n; i++)
            {
                    XORed operation
            }
    }
4.  Write P
```

5.4 Implementation

5.4.1 Tools

5.4.1.1 JAVA

Java is an Object oriented programming language developed by James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Sheridan at Sun Microsystems in 1991. The language was initially called “Oak” and later renamed “Java” in 1995. Java applications are typically compiled to bytecode (class file) that can run on any Java Virtual Machine (JVM) regardless of the computer architecture. Java Development Kit jdk-6u24-windows-i586 is used to compile and run the codes written in Java. The encryption and decryption codes are included in the Appendix section of this document.

6. TESTING AND ANALYSIS

6.1 Testing of proposed scheme

In Symmetric cipher, the secret key is shared between two communicating parties. Here, for the testing purpose, the key is defined by default.

The proposed system is tested with some sample inputs and their outputs.

Input:	Nepal is a landlocked sovereign state located in South Asia
Output:	peqNNZ~sKeKhdnvSjodejincrethFks~BnhFl~zhUmdetTjboSiRXlsABGS)
Input:	This is Cryptographic test
Output:	ihK!TeC~deWdgoBvadciiYpiXtQdx)
Input:	Cryptography plays important role in security system
Output:	yriOCcarCTtulpdL)Emiv%gDnaKSrDeltQsbesp?IDytjx(uetQgL)
Input:	Tribhuvan University
Output:	irCMTyna!DHbevj<KeXyQeM)

6.2 Analysis of proposed scheme

6.2.1 Known-plaintext attack

In the Hill Cipher, there is direct relation between the plaintext P and ciphertext C, where P and C are plaintext and ciphertext column vectors respectively. The relation is given by

$$C = KP \bmod 26$$

Taking the known plaintext and ciphertext pairs, the equation can be written in the form

$$X = KY \bmod 26$$

Where, Y is plaintext matrix and X is ciphertext matrix. The key can be obtained by the equation

$$K = XY^{-1} \bmod 26$$

Thus, the Hill Cipher is broken by known plaintext attack.

In the proposed scheme, as the sub keys are generated with the application of Cipher Block Chain (CBC) operation in each step of encryption process and encryption in each next step is carried out by the new encryption key, the relation between plaintext and ciphertext is hindered. The CBC operation and permutation in the ciphertext matrix elements address the properties of the confusion and hides the relation between ciphertext and the encryption key. So, the proposed scheme can’t be broken by the known plaintext attack.

6.2.2 Brute-force attack

Based on keys

For a matrix of order n x n of the field length m, there could be m^{nxn} number of possible keys.

In the Hill Cipher, all the elements from the field can’t be used the key element. For the decryption process, it needs the multiplicative inverse of the key. Thus, only the elements in field, which are relatively prime to filed length and have multiplicative inverse modulo field length, can be used as the key element. So, the numbers of possible keys that can be form are less and the key space also becomes small.

Let us consider, the size of key matrix 4 x 4 and field length m = 26, then the number of possible keys are

$$26^{16} = 43{,}608{,}742{,}899{,}428{,}874{,}059{,}776$$

In case of Hill Cipher, for key matrix 4 x 4 and field length m = 26, the number of possible keys are

$$12{,}303{,}585{,}972{,}327{,}392{,}870{,}400$$

This shows that there is possibility of the brute force attack in the Hill Cipher.

In the proposed scheme, as it doesn’t need the inverse of the key for the decryption process, all the elements from the field can be used as the key element and have large number of possible keys and large key space than the Hill Cipher.

It can be made an attacker hard to find the key by brute force approach by increasing the size of key matrix. On increase in the matrix size, the key space becomes large and prohibits the identifying the key matrix. In the Hill Cipher, as it require the inverse of the key for decryption and it need to calculate the determinant of the key to find the inverse of the key. As the order of the matrix is increased, it becomes more difficult to calculate its determinant. Thus, computational cost of the decryption process becomes more and more expensive and impractical to implement.

In the proposed scheme, the computational cost remains same for the encryption and decryption process and practically possible in the view of implementation for higher order matrices. Thus, the proposed scheme can’t be broken by this type of attack.

Ciphertext only attack

In the proposed scheme, the length of the plaintext block is 6 characters i.e. 36 binary bits. The space of the plaintext is 2^{36}. The space of the plaintext can be increased by two ways. One way, to increase the space of plaintext is by increasing the number of columns in the plaintext block.

If we take 3n characters in plaintext block then it contains 9 characters i.e. 54 binary bits and the space of the plaintext is 2^{54}.

If we take 4n characters in plaintext block then it contains 12 characters i.e. 72 binary bits and the space of the plaintext is 2^{72}.

The other way, to increase in the space of plaintext is by increasing the number of rows in the plaintext block. This can be carried out when the size of matrix is increased. This shows that the space of plaintext can be increased and makes the cipher strong against the ciphertext only attack.

6.2.3 Chosen ciphertext attack and chosen plaintext attack

In the proposed scheme, the changing the position of the plaintext change the pattern of the ciphertext i.e. it satisfy the properties of the diffusion. The scheme is also the polygraph substitution in which the same character in the plaintext is substituted with the different characters in the ciphertext. As it can be seen that the scheme contains many characters in the plaintext block, it is strong against the frequency analysis. Thus, the scheme is also strong against these types of attacks.

Thus the proposed scheme is strong one and can’t be broken by any type of cryptanalytic attack.

7. CONCLUSION AND FUTURE WORK

Hill Cipher is a classical symmetric and multi-letter encryption algorithm. Although it is strong against frequency analysis and ciphertext only attack, it succumbs to the known-plaintext attack.

In the proposed scheme, a block cipher is introduced that is secure against the known-plaintext attack and is strong against other types of attack. The scheme uses the Cipher Block Chaining (CBC) mode of operation and permutation of elements in cipher text block in each step of the encryption such that in each step of encryption the plaintext block is encrypted with the different new sub keys that address the properties of confusion of block cipher and make the scheme strong against the known-plaintext attack. The scheme is based on multi-letter cipher and polygraph substitution cipher. So, it is strong against frequency letter analysis and other types of attack like chosen ciphertext attack and chosen plaintext attack. The scheme is also strong against the brute-force attack and is carried out by the increase in the key matrix size or the number of characters in the plaintext block. The increase in key matrix size doesn't have any effect in the decryption process as the computational cost of decryption and encryption is same as it doesn't need the inverse of the key matrix unlike in Hill Cipher. In the Hill Cipher, it needs the inverse of the key matrix for decryption and it need to calculate determinant of the key matrix, which is computationally expensive and difficult for higher order of matrices.

Since the encryption is done in bit level so it is quite easy to understand and can be implemented for any kind of application like text applications as well as images, video, audio encryption. It can be also used to store the confidential information on end user devices as to protect information from unauthorized access.

In the proposed scheme, the discussion focused on the increase in security to overcome the different types of attacks. It is needed to study the computational complexity to the proposed scheme. Also the proposed scheme is implemented with the single round of operation. It can be used a number of round functions that will made the scheme more secure over attacks by creating more diffusion and confusion.

The encryption and decryption of the scheme are implemented in Java language and the implementation codes are included in appendix section of this document.

8. REFERENCES

[1] Avinash Kak "*Classical Encryption Techniques* (Lecture Notes on Computer and Network Security)" Prudue University, 2009.

[2] Aelphaeis Mangarae "*Steganography FAQ*" [Zone-H-Org], Zone-h unrestricted information, 18 March 2006.

[3] Amogh Mahapatra & Rajballav Dash "*Data Encryption and Decryption by Using Hill Cipher Technique and Self Repetitive Matrix*", Department of Electronics & Instrumentation Engineering National Institute of Technology Rourkela, 2007.

[4] Chris Brew "*Monoalphabetic Ciphers*", January 9 2008.

[5] Chris Christensen "*Polygraphic Ciphers*", MAT/CSC 483, 2006.

[6] Craig Smith "*Basic Cryptanalysis Techniques*", November 17 2001.

[7] C. E. Shannon "*Communication Theory of Secrecy Systems*", Bell System Technical Journal, 1949.

[8] Dennis Luciano & Gordon Prichett "*Cryptology From Caesar Ciphers to Public Key Cryptosystems*", The College of Mathematics Journal, January 1987.

[9] Heijltjes & Wadler "*The Caesar Cipher*", Informatics – Functional Programming: Tutorial 3, 2008.

[10] Henk C.A. van Tilborg, "*Fundamentals of Cryptology*", Eindhoven University of Technology The Netherlands, Kluwer Academic Publishers, London.

[11] Kartik Krishnan "*Computer Networks and Computer Security*", SFWR 4C03, Mar 8-11 2004.

[12] Marc Joye "*White box Cryptography*" Published in A. Elci, S.B. Ors, and B. Preneel, Eds, *Security of Information and Networks*, pp. 7-12, Trafford Publishing, 2008.

[13] Mohd Zaid Waqiyuddin & Mohd Zulkifli "*Attack on Cryptography*", April 2008.

[14] Murray Eisenberg "*Hill Ciphers and Modular Linear Algebra*", November 3, 1999.

[15] Niels Provos and Peter Honeyman “*An introduction to Steganography*”, University of Michigan, 2003.

[16] Neil F. Johnson & Sushil Jajodia, “*Exploring Steganography*” George Mason University, 1998.

[17] Packirisamy Murali and Gandhidoss Senthilkumar “*Modified Version of Playfair Cipher using Linear Feedback Shift Register*”, IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.12, December 2008.

[18] Rushdi A. Hamamreh and Mousa Farajallah, “*Design of Robust Cryoptosystem Algorithm for Non-Invertible Matrices Based on Hill Cipher*”, IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.5, May 2009.

[19] Ramchandra S. Mangrulkar, Pallavi V. Chavan, “*Encrypting Informative Image by Key Image using Hill Cipher Technique*”, International Journal of Recent Trends in Engineering, Vol. 1, No. 1, May 2009.

[20] Steven F Burns “*Threat Modeling: A Process To Ensure Application Security*”, GIAC Security Essentials Certification (GSEC) Practical Assignment Version 1.4c, January 5 2005.

[21] Shawn Harris “*Cryptography All in One Certification Exam Guide*”, CISSP, 2001.

[22] V. Umakanta Sastry, N. Ravi Shank2, and S. Durga Bhavani “*A Modifed Hill Cipher Involving Interweaving and Iteration*”, International Journal of Network Security, Vol.11, No.1, PP.11-16, July 2010.

[23] William Stallings, “*Cryptography and Network Security”*, Fourth Edition, Prentice Hall of India; ISBN-978-81-203-3018-4.

[24] William L. Steffan “Cryptography Fundamentals”, CS-340 Legal and Ethical Issues in Computing, University of Alabama 11 Novemeber 2010.

APPENDIX

```
import java.lang.*;
import java.io.*;

// ---------------------------------- Character Set -------------------------------------
class CharacterSet
{
        int num;
        int charCount = 64;
        char[] ch = new char[charCount];

        CharacterSet()
        {
                ch[0] = 'a';
                ch[1] = 'b';
                ch[2] = 'c';
                ch[3] = 'd';
                ch[4] = 'e';
                ch[5] = 'f';
                ch[6] = 'g';
                ch[7] = 'h';
                ch[8] = 'i';
                ch[9] = 'j';
                ch[10] = 'k';
                ch[11] = 'l';
                ch[12] = 'm';
                ch[13] = 'n';
                ch[14] = 'o';
                ch[15] = 'p';
                ch[16] = 'q';
                ch[17] = 'r';
                ch[18] = 's';
                ch[19] = 't';
                ch[20] = 'u';
                ch[21] = 'v';
                ch[22] = 'w';
                ch[23] = 'x';
                ch[24] = 'y';
                ch[25] = 'z';
                ch[26] = 'A';
                ch[27] = 'B';
                ch[28] = 'C';
                ch[29] = 'D';
                ch[30] = 'E';
                ch[31] = 'F';
                ch[32] = 'G';
                ch[33] = 'H';
                ch[34] = 'I';
                ch[35] = 'J';
                ch[36] = 'K';
                ch[37] = 'L';
                ch[38] = 'M';
                ch[39] = 'N';
                ch[40] = 'O';
                ch[41] = 'P';
                ch[42] = 'Q';
                ch[43] = 'R';
                ch[44] = 'S';
                ch[45] = 'T';
                ch[46] = 'U';
                ch[47] = 'V';
                ch[48] = 'W';
                ch[49] = 'X';
                ch[50] = 'Y';
                ch[51] = 'Z';
                ch[52] = '#';
                ch[53] = '!';
                ch[54] = '$';
                ch[55] = '%';
                ch[56] = '<';
                ch[57] = '>';
                ch[58] = '?';
                ch[59] = '@';
                ch[60] = '(';
                ch[61] = ')';
                ch[62] = '[';
                ch[63] = '~';
        }
```

```
        // Get a character of corresponding numeric value
        char getCharacter(int index)
        {
                return ch[index];
        }

        // Get numeric value of corresponding character
        int getNumericValue(char charParam)
        {
                for(int i = 0; i < charCount; i++)
                {
                        char temp = ch[i];
                        if(temp == charParam)
                        {
                                num = i;
                                break;
                        }
                }
                return num;
        }
}

// ----------------------------------- Code for Encryption ------------------------------------
class Encryption
{
        int plaintextLength;
        String input = "";
        String ciphertext = "";
        String plaintext = "";
        String subPlaintext = "";

        int key[] = new int[9];
        int keyBlock[] = new int[9];
        int bitLength = 6;
        int maxBitLength = 12;
        int charCount = 6;
        int matrixSize = 3;

        char inputArray[] = new char[6];
        String outputArray[] = new String[6];
```

```
	int inputNumericArray[] = new int[6];
	int outputNumericArray[] = new int[6];
	int plainTextBlock[] = new int[3];
	int cipherTextBlock[] = new int[3];

	// Temporary arrays
	String temp[] = new String[6];
	String temp1[] = new String[3];


	// For permutation purpose
	int pIndex;
	int pFlag = 0;
	int pInit;
	int pInitStore;

	// To append Extra String
	String append = "X";

	Encryption()
	{
		key[0] = 12;
		key[1] = 22;
		key[2] = 3;
		key[3] = 11;
		key[4] = 5;
		key[5] = 6;
		key[6] = 32;
		key[7] = 8;
		key[8] = 9;
	}

	// Formation of palintext block with three numeric value after concatenation of bits and
convertion into numeric form

	void createPlainTextBlock()
	{
		// Convert the character numeric values into bit string of Bit Length of each
character
```

```
for(int i = 0; i < charCount; i++)
{
	temp[i] = Integer.toBinaryString(inputNumericArray[i]);
	String s = temp[i];
	int length = s.length();

	if(length < bitLength)
	{
		while(length != bitLength)
		{
			s = '0' + s;
			length = s.length();
		}

		temp[i] = s;
	}
}

// Bit pattern of plaintext of each character
System.out.println("");
System.out.println("Bit pattern of palintext of each character: ");
System.out.println("");

for(int i = 0; i < charCount; i++)
{
	System.out.println(temp[i]);
}

// Concatenating the plaintext bits row wise
for(int i = 0; i < matrixSize; i ++)
{
	temp1[i] = temp[i] + temp[i+matrixSize];
}

// Bit pattern of plaintext column vector
System.out.println("");
System.out.println("Bit pattern of palintext column vector: ");
System.out.println("");
```

```
		for(int i = 0; i < matrixSize; i++)
		{
			System.out.println(temp1[i]);
		}

		// Convert the plaintext column vector elements into numeric form
		for(int i = 0; i < matrixSize; i ++)
		{
			plainTextBlock[i] = Integer.parseInt(temp1[i], 2);
		}

		// Dispaly the final plaintext in numeric form
		/*for(int i = 0; i < matrixSize; i++)
		{
			System.out.println(plainTextBlock[i]);
		}*/

	}

	// Rowise XOR operation between key elements and plaintext column vector elements
	void doEncryption()
	{
		for(int i = 0; i < matrixSize; i++)
		{
			cipherTextBlock[i] = ((keyBlock[i] ^ plainTextBlock[i]) ^ keyBlock[i+3])
^ keyBlock[i+6];
			temp1[i] = Integer.toBinaryString(cipherTextBlock[i]);
			String s = temp1[i];
			int length = s.length();

			if(length < maxBitLength)
			{
				while(length != maxBitLength)
				{
					s = '0' + s;
					length = s.length();
					temp1[i] = s;
				}
			}
```

```
                String s1 = s.substring(0, bitLength);
                String s2 = s.substring(bitLength);

                outputArray[i] = s1;
                outputArray[i+3] = s2;

                outputNumericArray[i] = Integer.parseInt(s1, 2);
                outputNumericArray[i+3] = Integer.parseInt(s2, 2);
            }

            // Bit pattern of ciphertext column vector after XOR operation
            System.out.println("");
            System.out.println("Bit pattern of ciphertext column vector: ");
            System.out.println("");

            for(int i = 0; i < matrixSize; i++)
            {
                System.out.println(temp1[i]);
            }

            // Bit pattern of ciphertext length of each character
            System.out.println("");
            System.out.println("Bit pattern of each ciphertext characters: ");
            System.out.println("");
            for(int i = 0; i < charCount; i++)
            {
                System.out.println(outputArray[i]);
            }

            // Dispaly the final ciphertext in numeric form
            /*for(int i = 0; i < charCount; i++)
            {
                System.out.println(outputNumericArray[i]);
            }*/
        }

        // Replacing the first row of key values with the ciphertext produced to perform Cipher
Block Chain(CBC)
```

```
	void keyCBC()
	{
		keyBlock[0] = cipherTextBlock[0];
		keyBlock[3] = cipherTextBlock[1];
		keyBlock[6] = cipherTextBlock[2];
	}

	// Main function starts here
	public static void main(String args[]) throws IOException
	{
		// Creating objects of Characterset and EncryptionTest
		CharacterSet cs = new CharacterSet();
		Encryption en = new Encryption();

		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		System.out.println(" ");
		System.out.print("Enter Plaintext: ");
		en.plaintext = br.readLine();
		en.plaintextLength = en.plaintext.length();

		// Checking the plaintext length multiplicity with Character count in each step
processing
		int rem = en.plaintextLength % en.charCount;

		if(rem != 0)
		{
			while(rem != 0)
			{
				en.plaintext += en.append;

				en.plaintextLength = en.plaintext.length();
				rem = en.plaintextLength % en.charCount;
			}
		}

		en.plaintextLength = en.plaintext.length();
		en.subPlaintext = en.plaintext;

		int matrixElements = en.matrixSize * en.matrixSize;
```

```
for(int i = 0; i < matrixElements; i++)
{
        en.keyBlock[i] = en.key[i];
}

while(en.plaintextLength >= en.charCount)
{
        en.input = en.subPlaintext.substring(0, en.charCount);
        en.subPlaintext = en.subPlaintext.substring(en.charCount);
        en.plaintextLength = en.subPlaintext.length();

        int length = en.input.length();
        for(int i = 0; i < length; i++)
        {
                char ch = en.input.charAt(i);
                Character c = ch;

                if(c.hashCode() == 32)
                {
                        ch = '~';
                }
                en.inputArray[i] = ch;
        }

// Forming the numeric array of input characters
for(int i = 0; i < en.charCount; i++)
{
        en.inputNumericArray[i] = cs.getNumericValue(en.inputArray[i]);
}

// Creating plaintext block
en.createPlainTextBlock();
en.doEncryption();
en.keyCBC();

// Permutaion of array elements
int len = en.outputNumericArray.length;
for(int i = 0; i < len; i++)
{
        if((i % 2) == 0)
```

```
                {
                        if(en.pFlag == 0)
                        {
                                en.pInit = i;
                                en.pInitStore = en.outputNumericArray[i];
                                en.pIndex = i;
                                en.pFlag = 1;
                        }
                        else
                        {
                                en.outputNumericArray[en.pIndex]                    =
en.outputNumericArray[i];;
                                en.pIndex = i;
                        }
                }

                        if(i == len - 1)
                        {
                                en.outputNumericArray[en.pIndex] = en.pInitStore;
                                en.pFlag = 0;
                        }
                }

                for(int i = 0; i < en.charCount; i++)
                {
                        en.ciphertext += cs.getCharacter(en.outputNumericArray[i]);
                }

        } // End of while of encryption method

        System.out.println("");
        System.out.print("Ciphertext: " + en.ciphertext);
        System.out.println("");

    } // End of main function
}
```

```
// --------------------------- Code for Decryption --------------------------------------

class Decryption
{
        int ciphertextLength;
        String input = "";
        String ciphertext = "";
        String plaintext = "";
        String subCiphertext = "";

        int key[] = new int[9];
        int keyBlock[] = new int[9];
        int bitLength = 6;
        int maxBitLength = 12;
        int charCount = 6;
        int matrixSize = 3;

        char inputArray[] = new char[6];
        String outputArray[] = new String[6];
        int inputNumericArray[] = new int[6];
        int outputNumericArray[] = new int[6];
        int plainTextBlock[] = new int[3];
        int cipherTextBlock[] = new int[3];

        // Temporary arrays
        String temp[] = new String[6];
        String temp1[] = new String[3];

        // For permutation purpose
        int pIndex;
        int pFlag = 0;
        int pInit;
        int pInitStore;

        Decryption()
        {
                key[0] = 12;
                key[1] = 22;
                key[2] = 3;
                key[3] = 11;
```

```
		key[4] = 5;
		key[5] = 6;
		key[6] = 32;
		key[7] = 8;
		key[8] = 9;
	}
```

// Formation of ciphertext block with three numeric value after concatenation of bits and convertion into numeric form

```
	void createCipherTextBlock()
	{
```

// Convert the character numeric values into bit string of Bit Length of each character

```
		for(int i = 0; i < charCount; i++)
		{
			temp[i] = Integer.toBinaryString(inputNumericArray[i]);
			String s = temp[i];
			int length = s.length();

			if(length < bitLength)
			{
				while(length != bitLength)
				{
					s = '0' + s;
					length = s.length();
				}
				temp[i] = s;
			}
		}

		// Bit pattern of ciphertext of each character
		System.out.println("");
		System.out.println("Bit pattern of ciphertext of each character: ");
		System.out.println("");

		for(int i = 0; i < charCount; i++)
		{
			System.out.println(temp[i]);
		}
		// Concatenating the ciphertext bits row wise
		for(int i = 0; i < matrixSize; i ++)
		{
			temp1[i] = temp[i] + temp[i+matrixSize];
		}

		// Bit pattern of ciphertext column vector
		System.out.println("");
		System.out.println("Bit pattern of ciphertext column vector: ");
		System.out.println("");

		for(int i = 0; i < matrixSize; i++)
		{
			System.out.println(temp1[i]);
		}

		// Convert the ciphertext column vector elements into numeric form
		for(int i = 0; i < matrixSize; i ++)
		{
			cipherTextBlock[i] = Integer.parseInt(temp1[i], 2);
		}

		// Dispaly the final ciphertext in numeric form
		/*for(int i = 0; i < matrixSize; i++)
		{
			System.out.println(cipherTextBlock[i]);
		}*/
	}

	// Rowise XOR operation between key elements and ciphertext column vector elements
	void doDecryption()
	{
		for(int i = 0; i < matrixSize; i++)
		{
			plainTextBlock[i] = ((keyBlock[i] ^ cipherTextBlock[i]) ^ keyBlock[i+3])
^ keyBlock[i+6];

			temp1[i] = Integer.toBinaryString(plainTextBlock[i]);
			String s = temp1[i];
			int length = s.length();
```

```
        if(length < maxBitLength)
        {
                while(length != maxBitLength)
                {
                        s = '0' + s;
                        length = s.length();
                        temp1[i] = s;
                }
        }
        String s1 = s.substring(0, bitLength);
        String s2 = s.substring(bitLength);

        outputArray[i] = s1;
        outputArray[i+3] = s2;

        outputNumericArray[i] = Integer.parseInt(s1, 2);
        outputNumericArray[i+3] = Integer.parseInt(s2, 2);
}

// Bit pattern of plaintext column vector after XOR operation
System.out.println("");
System.out.println("Bit pattern of plaintext column vector: ");
System.out.println("");

for(int i = 0; i < matrixSize; i++)
{
        System.out.println(temp1[i]);
}

// Bit pattern of plaintext length of each character
System.out.println("");
System.out.println("Bit pattern of each plaintext characters: ");
System.out.println("");

for(int i = 0; i < charCount; i++)
{
        System.out.println(outputArray[i]);
}
```

```
		// Dispaly the final plaintext in numeric form
		/*for(int i = 0; i < charCount; i++)
		{
			System.out.println(outputNumericArray[i]);
		}*/
	}

	// Replacing the first row of key values with the ciphertext recently used as input to
perform Cipher Block Chain(CBC)
	void keyCBC()
	{
		keyBlock[0] = cipherTextBlock[0];
		keyBlock[3] = cipherTextBlock[1];
		keyBlock[6] = cipherTextBlock[2];
	}

	// Main function starts here
	public static void main(String args[]) throws IOException
	{
		// Creating objects of Characterset and DecryptionTest
		CharacterSet cs = new CharacterSet();
		Decryption dn = new Decryption();

		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		System.out.println(" ");
		System.out.print("Enter ciphertext: ");
		dn.ciphertext = br.readLine();

		dn.ciphertextLength = dn.ciphertext.length();
		dn.subCiphertext = dn.ciphertext;

		int matrixElements = dn.matrixSize * dn.matrixSize;
		for(int i = 0; i < matrixElements; i++)
		{
			dn.keyBlock[i] = dn.key[i];
		}

		while(dn.ciphertextLength >= dn.charCount)
		{
```

```
            dn.input = dn.subCiphertext.substring(0, dn.charCount);
            dn.subCiphertext = dn.subCiphertext.substring(dn.charCount);
            dn.ciphertextLength = dn.subCiphertext.length();

            int length = dn.input.length();
            for(int i = 0; i < length; i++)
            {
                dn.inputArray[i] = dn.input.charAt(i);
            }

            // Forming the numeric array of input characters
            for(int i = 0; i < dn.charCount; i++)
            {
                dn.inputNumericArray[i] = cs.getNumericValue(dn.inputArray[i]);
            }

            // Permutaion of array elements
            int len = dn.inputNumericArray.length;
            for(int i = len - 1 ; i >= 0; i--)
            {
                if((i % 2) == 0)
                {
                    if(dn.pFlag == 0)
                    {
                        dn.pInit = i;
                        dn.pInitStore = dn.inputNumericArray[i];
                        dn.pIndex = i;
                        dn.pFlag = 1;
                    }
                    else
                    {
                        dn.inputNumericArray[dn.pIndex] = dn.inputNumericArray[i];
                        dn.pIndex = i;
                    }
                }

                if(i == 0)
                {
                    dn.inputNumericArray[dn.pIndex] = dn.pInitStore;
                    dn.pFlag = 0;
                }
            }

            // creating ciphertext block
            dn.createCipherTextBlock();
            dn.doDecryption();
            dn.keyCBC();

            for(int i = 0; i < dn.charCount; i++)
            {
                char ch = cs.getCharacter(dn.outputNumericArray[i]);

                if(ch == '~')
                {
                    ch = ' ';
                }
                dn.plaintext += ch;
            }

        } // End of while of encryption

        System.out.println("");
        System.out.println("Plaintext: " + dn.plaintext);
        System.out.println("");

    } // End of main function
}
```