DATA ENCRYPTION AND DECRYPTION BY USING HILL CIPHER TECHNIQUE AND SELF REPETITIVE MATRIX

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

in

Electronics & Instrumentation Engineering

By

AMOGH MAHAPATRA

And

RAJBALLAV DASH

Department of Electronics & Instrumentation Engineering

National Institute of Technology

Rourkela

2007

DATA ENCRYPTION AND DECRYPTION BY USING HILL CIPHER TECHNIQUE AND SELF REPETITIVE MATRIX

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

in

Electronics & Instrumentation Engineering

By

AMOGH MAHAPATRA

And

RAJBALLAV DASH

Under the Guidance of

Prof. G.S.Rath

Department of Electronics & Instrumentation Engineering

National Institute of Technology

Rourkela

2007

National Institute of Technology

Rourkela

CERTIFICATE

This is to certify that the thesis entitled, "Data encryption and decryption using Hill Cipher method and Self Repetitive Matrix" submitted by Sri Rajballav Dash and Sri Amogh Mahapatra in partial fulfillments for the requirements for the award of Bachelor of Technology Degree in Electronics & Instrumentation Engineering at National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date:

Prof. G. S. Rath
Dept. of Electronics & Instrumentation Engg
National Institute of Technology
Rourkela - 769008

ACKNOWLEDGEMENT

We would like to articulate our deep gratitude to our project guide Prof. G.S.Rath who has always been our motivation for carrying out the project. It is our pleasure to refer Microsoft Word exclusive of which the compilation of this report would have been impossible. An assemblage of this nature could never have been attempted with out reference to and inspiration from the works of others whose details are mentioned in reference section. We acknowledge out indebtedness to all of them. Last but not the least , our sincere thanks to all of our friends who have patiently extended all sorts of help for accomplishing this undertaking.

RAJBALLAV DASH

AMOGH MAHAPATRA

CONTENTS

Chapter 1

GENERAL INTRODUCTION

Since times immemorial, security of data to maintain its confidentiality, proper access control, integrity and availability has been a major issue in data communication. As soon as a sensitive message was etched on a clay tablet or written on the royal walls, then it must have been foremost in the sender's mind that the information should not get intercepted and read by a rival. Codes, hence, form an important part of our history, starting from the paintings of Da Vinci and Michelangelo to the ancient Roman steganographic practices the necessity of data hiding was obvious.

Today in the e-age, the need to protect communications from prying eyes is greater than ever before. Cryptography, the science of encryption plays a central role in mobile phone communication, e-commerce, Pay-TV, sending private e-mails, transmitting financial information and touches on many aspects of daily lives.

Today's technology can be traced back to earliest ciphers, and have grown as a result of evolution. The initial ciphers were cracked, so new, stronger ciphers emerged. Code breakers set to work on these and eventually found flaws, forcing cryptographers to invent better ciphers and so on. The significance of key is an enduring principle of cryptography.

With the advent of the computer age, the mechanical encryption techniques were replaced with computer ciphers .They operated according to the same principles of substitution and transposition (where the order of letters or bits is altered).

Again each cipher depended on choosing a key, known only by the sender and the receiver which defined how a particular message would be. This meant that there still

was a problem of getting the key to the receiver so that the message could be deciphered. This had to be done in advance, which was an expensive slow and risky process.

For yeas, this key distribution problem haunted code makers i.e. if you want to decipher a scrambled text you have to know the key in advance. But there was revolution in cryptography known as public key cryptography, which destroyed the key distribution problem. This was a technology tailor made for the internet. Customers could send credit card details and send them to retailers on the other side of the planet. It formed the basis of all kinds of modern day communications.

AUTHORS CONTRIBUTION:

The authors suggest an innovation in the age-old conventional cryptographic technique of HILL-CIPHER using the concept of self repetitive matrix. A numerical method has been stated mathematically proved and later implemented ingenerating a random matrix of given periodicity. The method of self repetitive matrix has then been used to simulate a communication channel with proper decompression techniques to facilitate bit saving. An investigation was also carried out to find out a definitive correlation between the Eigen value of a periodic matrix and its periodicity. As a result the age old fault in hill cipher technique which surfaces if the inverse of the multiplicative matrix does not exist has been compromised. This method is easier to implement compared to other techniques like self invertible matrix etc. and if the block size and the modular index is suitably chosen the it becomes extremely tuff to crack it by brute force as it is not very easy to find multiple inverses of a higher square matrix of higher order.

Chapter 2

CONVENTIONAL ENCRYPTION

2.1 Introduction

Conventional Encryption is referred to as symmetric encryption or single key encryption. It was the only type of encryption in use prior to the development of public-key encryption. Conventional encryption can further be divided into the categories of classical and modern techniques. The hallmark of the classical technique is that the cipher or the key to the algorithm is shared i.e. known by the parties involved in the secured communication. So there are two types of cryptography: secret key and public key cryptography .In secret key same key is used for both encryption and decryption. In public key cryptography each user has a public key and a private key. In this chapter a very view of conventional cryptography is presented. In the section 2.8 a new method of cipher developed by author has been presented

2.2 What is cryptography?

Cryptography is the study of Secret (crypto-)-Writing (-graphy).It is the science or art of encompassing the principles and methods of transforming an intelligible message into one that is intelligible and then transforming the message back to its original form. As the field of cryptography has advanced; cryptography today is assumed as the study of techniques and applications of securing the integrity and authenticity of transfer of information under difficult circumstances.

Today's cryptography is more than encryption and decryption. Authentication is as fundamentally a part of our lives as privacy. We use authentication throughout our everyday lives when we sign our name to some document and for instance and , as we move to world where our decisions and agreements are communicated electronically, we need to have electronic techniques for providing authentication. Cryptography provides mechanisms for such procedures.

A digital signature binds a document to the possessor of a particular key, while a digital timestamp binds a document to its creation of a particular time. These cryptographic mechanisms can be used to control access to shared disk drive, a high security installation, or a pay-per-view TV channel. The field of cryptography encompasses other uses as well. With just a few basic cryptographic tools, it is possible to build elaborate schemes and protocols that allow us to pay using electronic money, to prove we know certain information without revealing the information itself, and to share quantity in such a way that a subset of the shares can reconstruct the set. While modern cryptography is growing increasingly diverse, cryptography is fundamentally based on problems that are difficult to solve. A problem may be difficult because its solution requires some secret knowledge such as decrypting an encrypted message or signing some digital document.

Cryptographic systems are generally classified along three independent dimensions:

1. Type of operations used for transforming plaintext to cipher text. All encryption algorithms are based on two general principles. Those are substitution, in which each element in the plain text is mapped into another element and transposition in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost. Most systems referred to as product systems, involved multiple stages of substitution and transposition.
2. The number of keys used: If sender and receiver use the same key, the system is referred to as symmetric, single key or secret key conventional encryption. If the sender and the receiver each uses a different key the system is referred to as asymmetric, two key, or public-key encryption.
3. The way in which the plaintext is processed: A block cipher processes the input on block of elements at a time, producing an output block for each input block. A stream cipher processes the input elements continuously, producing output one element at a time, as it goes along.

Steganography, hiding one message inside another, is an old technique that is still in use. For example, a message can be hidden inside a graphics image file by using the low order

bit of each pixel to encode the message. The visual effect of these tiny changes is probably too small to be noticed by the user. The message can be hidden further by compressing it or by encrypting it with a conventional cryptosystems. Unlike conventional cryptosystems, where we assume the attacker knows everything about the cryptosystem except for the secret key, Steganography relies on the secrecy of the method of hiding for its security. If eve does not recognize the message as cipher text, then she is not likely to attempt to decrypt it.

2.3 Conventional encryption model:

A conventional encryption model can be illustrated as assigning Xp to represent the plaintext message to be transmitted by the originator. The parties involved select an encryption algorithm represented by E. The parties agree upon the secret key represented by K. The secret is distributed in a secure manner represented by SC. Conventional encryption's effectiveness rests on keeping the key secret. Keeping the key secret rests in a large on key distribution methods. When E processes Xp and K, Xc is derived. Xc represents the cipher text output, which will be decrypted by the recipient. Upon receipt of Xc, the recipient uses a decryption algorithm represented by D to process Xc and K back to Xp. This is represented in the figure. In conventional encryption, secrecy of the encryption and decryption algorithm is not needed. In fact, the use of an established well known and tested algorithm is desirable over an obscure implementation. This brings us to the topic of key distribution.

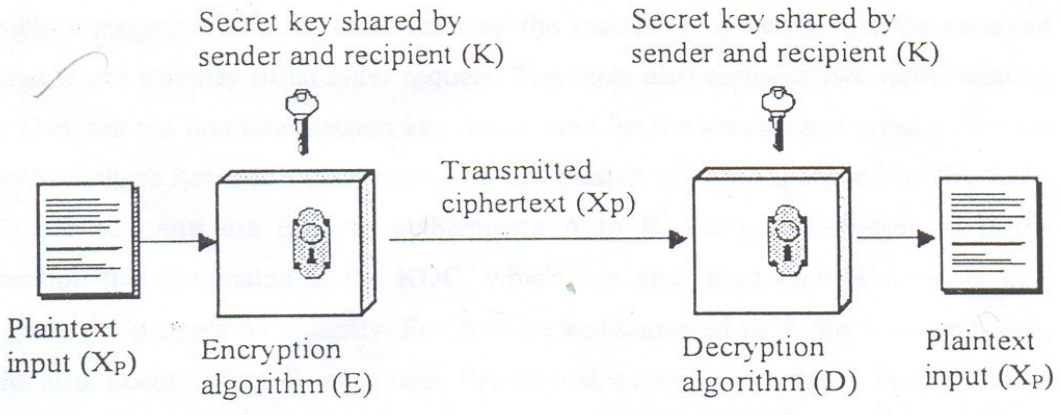


Figure 2.1: Simplified Model of Conventional Encryption

2.4 Key Distribution-Conventional Cryptography:

The strength of any cryptographic system rests on the key distribution technique. Conceivably for two parties: A could select a key and hand delivers it to B, or A and could rely on a trusted courier. If A and B have an established secure connection, they could exchange a key via encrypted messaging, or if A and B have a established secure connection via a third party C,C could provide this trusted courier service. The first two situations could provide a logistical nightmare if the number of communicating parties increase, the number of discrete keys needed also increases exponentially. In option, weaknesses are revealed, that if an attacker ever is able to compromise any one key it can compromise all the keys i.e. attacker can masquerade as the third party. The fourth option demonstrates a Key Distribution Center (KDC) which is largely adopted. A KDC is responsible for securely delivering unique key pair to its clients. It is also responsible for key management. A key management center uses a hierarchy of keys to provide authentication, integrity, no repudiation, and confidentiality to its users. The hierarchy of keys consists of session keys, which are used for logical connection between end users. The session keys are encrypted by a master key which is shared by the KDC and an end user. Consider this classic key distribution scenario: A uses his secret key to request a session key from KDC to establish a logical connection with B. The request includes both the identities of A and B and a unique identifier for the transaction. A is a contrivance invented or used for this particular, singular occasion. The replies to A with encrypted message which contains the requested one time session key and the original message with the nonce. The original message is used to verify the reply's integrity. The nonce is used by the requestor to verify that the returned message is not a replay of an older request.

The reply also contains two items relating to B. They are the one time session key, to be used for by session and an identifier for A. Both these items are encrypted with the master key shared by KDC and B. The items are used to authenticate A and B. Party A forwards to B the information that originated at KDC, that was encrypted with B's master key. This gives the process its integrity.

For B to be authenticated to A the following steps should occur: Party B now uses the logical connection created by the shared session key to send a half defined nonce to A.

Upon receipt of B defined nonce, A performs a mathematical function on the nonce; A performs a mathematical function on the nonce and returns result to B through the logical connection. The keys have to manage across KDC domains. Keys issued to an entity by one KDC have to validate by the issuer before they can be accepted by an entity serviced by a different issuer. The issuers have to collaborate on acceptable method of authenticating inter-domain transaction. Currently, KDCs use a hierarchy for key sharing. Each local KDC negotiates keys for its subscribers through a global KDC. Session generated must have a finite lifetime. Keys are exchanged frequently to prevent an opponent from having a large amount of data encoded with the same key. This brings us to the subject of cryptanalysis.

2.5 Cryptanalysis:

Code making involves the creation of encryption products that provide protection of confidentiality. Code breaking involves defeating this protection by some mens other than the standard decryption process used by a intended recipient. Five scenarios for which code breaking is used. They are ensure accessibility, spying on opponents, selling cracking products and services, pursuing the intellectual aspects of code breaking and testing whether one's codes are strong enough. Cryptanalysis is the process of attempting to discover either the plaintext Xp or the key K. Discovery of the encryption is the most desired one as with its discovery all the subsequent messages can be deciphered. Therefore, the length of encryption key, and the volume of the computational work necessary provides for its strength i.e. resistance to breakage. The longer the key, the stronger the protection, the more brute force is needed. Neither conventional encryption nor public key encryption is more resistant to cryptanalysis than the other. All that the user of an encryption algorithm can strive for is an algorithm that meets one or both of the following criteria: the cost of breaking the cipher exceeds the value of the encrypted information, the time required to break to exceeds the normal lifetime of the information.

2.6 Classical encryption techniques:

The technique enables us to illustrate the basic approaches to conventional encryption today. The two basic components of classical ciphers are substitution and transposition. Then other systems described that combines both substitution and transposition.

2.6.1 Substitution techniques

In this technique letters of plaintext are replaced by or by numbers and symbols.If plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with cipher text bit patterns.

2.6.1.1 Caesar Cipher

Caesar Cipher replaces each letter of the message by a fixed letter a fixed distance away e.g. uses the third letter on and repeatedly used by Julius Caesar.

For example:
Plaintext: I CAME I SAW I CONQUERED
Cipher text: L FDPH L VDZ L FRQTXHUHG
Mapping is:
ABCDEFGHIJKLMNOPQRSTUVWXYZ
DEFGHIJKLMNOPQRSTUVWXYZABC

Can describe the Cipher as:
Encryption: C=E (P)=(P+3)mod 26-------(2.1)
Decryption: P=D(C) =(C-3) mod26-------- (2.2)

2.6.1.2 Mono-alphabetic Cipher:

The Caesar cipher uses only 26 rotations out of the 26 permutations on the alphabet. The mono-alphabetic cipher uses them all. A key k is an arbitrary permutation of the alphabet. Ek(m) replaces each letter of a of m by k(a) to yield c. To decrypt Dk(c) replaces each letter b of c by k-inverse (b). The size of the key space is 26!> 2 pow(74), which is too large for a successful brute force attack. However, mono-alphabetic ciphers can be easily broken down using letter frequency analysis, given a long enough message. Because each occurrence of a letter a in the message is replaced by the same letter k(a), the most frequently occurring letter of m will correspond with the most frequently occurring letter of c. while jack might know what the most frequently occurring letter of m is, if the message is long enough and she knows that it is English, then it is quite likely that the

most frequently occurring letter in m is one of the most frequently occurring letters in English i.e. ‘e’ or maybe ‘t’. She can assume then that the most frequent letter ‘b1’ in c is ‘e’, the next most frequent letter b2 in ‘t’., and so forth. Of course, not all these guesses will be correct, but the number of likely candidates for each cipher text letter is greatly reduced. Moreover, many wrong guesses can quickly discarded eve without constructing the entire trial key because they lead to unlikely letter combinations.

2.6.1.3 Playfair Cipher

The Playfair is a substitution cipher bearing the name of the man who popularized but not created it. The method was invented by Sir Charles Wheatstone, in around 1854; however he named it after his friend Baron Playfair. The Playfair Cipher was developed for telegraph secrecy and it was the first literal digraph substitution cipher. This method is quite easy to understand and learn but not easy to break, because you would need to know the “keyword” to decipher the code. The system functions on how letters are positioned in a 5*5 alphabet matrix. A “KEYWORD” sets the pattern of letters with the other letters the cells of the matrix in alphabetical order(I and j are usually combined in one cell). For instance, suppose we use a keyword Charles then matrix would look like this:

c h a r l

e s b d f

g i/j k m n

o p q t u

v w x y z

Now supposing the message to be enciphered here is “the scheme really works”. First of all the plaintext is divide into two letter groups. If there are double letters occurring, in the message, either an ‘x’ will be used to separate the double letters or an ’x’ will be added to make a two letter group combination. In our example, the phrase becomes:

Enciphered text: th es c hem er ea lx ly wo rk sx

Ecah of the above two letter combinations will have 3 possible relationships with each other in the matrix: they can be in the same column, same row, or neither. The following rules for replacement should be used:

If two letters are in the same column of the matrix, use letter below it as the cipher text.(columns are cyclical).

If two letters are in the same row of the matrix, use letter to the right as the cipher text.(columns are cyclical).

If neither the same column or row, then each are exchanged with the letter at the intersection of its own row and the other column.

From our example:
Plaintext: th es ch em er lx ly wo rk sx
Cipher text: pr sb ha dg bc az rz vp am bw

For deciphering, the rules are exact opposite.

2.6.1.4 HILL CIPHER

The core of Hill-cipher is matrix manipulations. It is a multi-letter cipher, developed by the mathematician Lester Hill in 1929.

For encryption, algorithm takes m successive plaintext letters and instead of that substitutes m cipher letters. In Hill cipher each character is assigned a numerical value like:
a=0,
b=1,
…..
…..
z=25.
The substitution of cipher text letters in place of plaintext leads to m linear equations. For m=3, the system can be described as follows:

$C1=(K_{11}P1+K_{12}P2+K_{13}P3)MOD26$---------(2.3)

$C1=(K_{21}P1+K_{22}P2+K_{23}P3)MOD26$---------(2.4)

$C1=(K_{31}P1+K_{32}P2+K_{33}P3)MOD26$---------(2.5)

This can be expressed in terms of column vectors and matrices:

C=KP

Where C and P are column vectors of length 3, representing the plaintext and the cipher text and K is a 3*3 matrix, which is the encryption key. All operations are performed mod 26 here. Decryption requires the inverse of matrix K. The inverse K^{-1} of a matrix K is defined by the equation.

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

NOTE: The inverse of a matrix doesn't always exist, but when it does it satisfies the proceeding equation.

K^{-1} is applied to the cipher text, and then the plain text is recovered. In general terms we can write as follows:

For encryption: C=Ek(P)=Kp

For decryption: P=Dk(C) =K^{-1} C= K^{-1}Kp=P

2.7 Novel Modification to the Algorithm

As we have seen in Hill cipher decryption, it requires the inverse of a matrix. So while one problem arises that is:

Inverse of the matrix doesn't always exist. Then if the matrix is not invertible then encrypted text cannot be decrypted.

In order to overcome this problem author suggests the use of self repetitive matrix.

This matrix if multiplied with itself for a given mod value (i.e. mod value of the matrix is taken after every multiplication) will eventually result in an identity matrix after N multiplications. So, after N+ 1 multiplication the matrix will repeat itself.

Hence, it derives its name i.e. self repetitive matrix. It should be non singular square matrix.

Note: Historically, there are already many available methods, like self invertible matrix etc. But this method as suggested by the author is both easy to compute and simpler to implement.

2.7.1 Modular Arithmetic: A brief Analysis

The analysis presented here for generation of self repetitive matrix is valid for matrix of positive integers that are the residues of modulo arithmetic on a prime number. So in analysis the arithmetic operations presented here are addition, subtraction, Unary operation, Multiplication and division.

The Modulo operator have the following properties:

1. a=b mod p if n| (a-b)
2. (a mod p)=(b mod p)=>a=b mod p
3. a=b mod p=>b=a mod p
4. a=b mod p and b=a mod p=>a=c mod p

The modulo arithmetic have the following properties:

Let z= [0,1,…..,p-1], the set residues modulo p. If modular arithmetic is performed within the set z_n, the following equations present the arithmetic identities:

1. Addition: (a+b) mod p=[(a mod p)+(b mod p)]mod p
2. Subtraction: (a-b) mod p=[(a mod p)-(b mod p)]mod p
3. Multiplication: (a*b) mod p=[(a mod p)*(b mod p)]mod p
4. Negation: -a mod p=p-(a mod p)
5. Division: (a/b) mod P=c when a=(c*b) mod p
6. Multiplicative inverse: (a^{-1})=c if there exists (c*z) mod p=1

Table exhibits the properties of modulo arithmetic:

SL. No.	Property	Expression
1.	Commutative Law	(w + x) mod p=(x + w) mod p (w*x) mod p= (x*w) mod p
2.	Associative Law	[(w + x)+y] mod p = [w+ (x + y)] mod p
3.	Distributive Law	[w*(x + y)] mod p=[w*x +w*y] mod p [w*(x * y)] mod p=[{w*x mod p}*{(w*y)mod p}] mod p
4.	Identities	(0+a) mod p = a mod p and (1*a) mod p= a mod p

5.	Inverse	For each X belongs to z_p , there exists y such that (x + y) mod p =0 then y= -x For each X belongs to z_p, there exists y such that (x*y) mod p=1

2.7.2 Complex modular numbers:

Used while calculating negative or non existent square roots.

Say, for example, for mod 7

sqrt(1)=1;

sqrt(2)=4;

sqrt(4)=2;

But square root of 3, 5, 6 doesn't exist

So sqrt(6)=sqrt(-1)sqrt(-6)=j where j stand for square of -1 in modular arithmetic

Similarly, sqrt(5)=4j ; sqrt(3)=2j

Extensive no of simulation exercises were carried out to understand the properties of modular arithmetic.

2.7.3 Generation of a self repetitive Matrix A for a Given N:

The initial conditions for the existence of a self repetitive matrix are:

1. **The matrix should be square.**
2. **It should be non-singular.**

But trying to find out the value of N (the value where the matrix becomes a identity matrix) through the method of brute force may not be the best idea always; because the matrix is of dimension greater than 5*5 and with mod index (i.e.) greater than 91 then the brute force technique might take very long time and N value may be in the range of millions. A normal Pentium 4 machine might hang if asked to do the computations for 15*15 matrixes or more.

Hence, it would be comfortable to know the value of N and then generate a random matrix accordingly.

This can be done as follows:

1. First a diagonal matrix A is chosen, and then the values powers of each individual element when they reach unity is calculated and denoted as n1, n2, n3…. Now LCM of these values is taken to given the value of N.
2. Now the next step is generate a random square matrix whose N value is same as the N calculated in the previous step.
3. Pick up any random invertible square matrix B
4. Generate $C=B^{-1}AB$
5. The N value of C is also N

Mathematical proof:

$(B^{-1}AB)^N = (B^{-1})^N * (A)^N * (B)^N$

$A^N = I$ as calculated before as it is a diagonal matrix and N is the LCM of all elements

$(B^{-1}B)*(B^{-1}*B)$…….n times=I

2.7.4 Another Line of Investigation:

In pursuit of finding a many one relationship of N value with matrices led us towards investigating the correlation of N values with l (lambda) Eigen values.

A relationship established like this could easily solve the key distribution problem.

The relationship worked out but not conclusively implemented is like this:

NOTE: ALL operations are modular.

Say A is any square matrix and B be the diagonal matrix containing the Eigen values of A. Let P(l) be the characteristic Eigen equation of the A.

Then, l^N-1 will be divisible by P (l). (N is the n value of A) only real l values

NOTE: These methods shown and proved here have been used to simulate a transmitter and a receiver. Care has been taken to properly compress the code as well. The algorithm and the code include in the implementation section.

2.8 Poly-alphabetic Cipher:

A poly-alphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. Mono-alphabetic Cipher can be broken. The reason: Same plain letters are encoded to same cipher letters; the underlying letter frequencies remain unchanged.

Cryptographers have tried to overcome this dilemma simply by assigning various cipher letters or symbols to same plain letters. Such ciphers are called Poly-alphabetic Ciphers. The most popular of such ciphers is the "Vigenere Cipher". The Vigenere Cipher is an improvement of the Caesar Cipher key is multiple letters long K=k1, K2, k3….k_d. To encrypt a message, a key is needed that is long as message. Usually, the key is a repeating keyword. For example, if the keyword is deceptive, the message "we are discovered save youself" is encrypted as follows:

Key: deceptivedeceptivedeceptive

Plaintext: wearediscoveredsaveyourself

Ciphertext: ZICVTWQNGRZGVTWAVZHCQYGLMGJ

Decryption is equally simple. The key letter again identifies the row. The position of the cipher text letter in that row determines the column, and the plaintext letter is at the top of the column.

2.9 Transposition Scheme:

Transposition (or permutation) does not alter any of the bits in the plaintext, but instant moves the position around within it. If thr resultant Cipher text is then put through more transpositions, the end result has increasing security.

2.9.1 Rail Fence Technique:

The simplest transposition ciphering technique is the rail fence cipher, in which we write message with letters, alternate rows and read off Cipher row by row. For example, to encipher the message "I CAME I SAW I CONQUERED" with a rail fence of depth 2, we write the following:

Plain: I A E S W C N U R D

Cipher: C M I A I O Q E E

Cipher-Text: IAESWCNURDCMIAIOQEE

Amore complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of columns. The order of the columns then becomes the key to the algorithm.

2.10 Rotor Machines:

Rotor machine use multiple stages. Each symbol is substituted several times. A rotor machine consists of multiple elements. Each has 26 inputs and 26 outputs. In this way, every character is translated to a new character. This output is again used as an input to the next rotor. After several steps, the encrypted character is produced. After each character is typed, the fastest rotor rotates one step. As soon as it has rotated n-steps, the next fastest cylinder rotates. The initial set up position is the key.

2.11.1 Data Encryption Standard:

Data encryption standard is the most widely used method of data encryption using a secret key. There are 72,000,000,000,000,000 (72 quadrillion) or more possible encryption keys that can be used. For each given message, the key is chosen at random from among this enormous number of keys. Like other private key cryptographic methods, both the sender and the receiver must know and use the same private key.

It was developed in the 1970s by the National Bureau of Standards with the help of the National Security Agency. Its purpose is to provide a standard method for sensitive commercial and unclassified data. IBM created the first draft of the algorithm, calling it LUCIFER. DES officially became a standard in November of 1976.
Data encryption algorithm has a 64-bit block size and uses a 56bit key during execution (8 parity bits are stripped of from the full 64-bit key). The DEA can also be used for single user encryption, such as to store files in a hard in encrypted form. NIST re-certifies DES every 5 years. DES has been in world wide use for over 20 years, and due to the fact that it is a defined standard that any system implementing DES can communicate with any other system using is it.

2.11.1.1 DES Encryption:

DES is a symmetric, block-cipher algorithm with a key length of 64 bits, and the algorithm operates on successive 64 bit blocks of plain text. Due to symmetric, the same key is used for encryption and decryption, and also uses the same algorithm for encryption and decryption.

Initially a transposition is carried out according to a set table (the initial permutation), the 64-bit plaintext block is then split into two 32-bit block, and 16 identical operations called rounds are carried out on each half. The two halves are joined back together , and the reverse of the initial permutation is carried out. The purpose of the first transposition is clear; it does not affect the security of the algorithm, but is through for the purpose of allowing plaintext and cipher text to be loaded into 8-bit chip in byte-sized pieces. In any round, only one half of the original 64-bit block is operated on. The rounds alternate between the two halves. One round in DES consists of the following:

2.11.1.2 Key Transformation:

The keys reduced from 64-bit to 56-bit by removing every eighth bit, which are sometimes used for error checking. 16 different 48 bit sub-keys are then generated i.e. one for each round. This is achieved by splitting the 56-bit key into the two halves, and then circularly shifting them left by one or two bits, depending on the round .After this, 48 of the bits are selected. Because they are shifted, different groups of key bits are used in each sub key. This process is called compression permutation due to transposition of bits & reduction of the overall size.

2.11.1.3 Expansion Permutation:

Whichever half of the block is being operated on undergoes a permutation after key transformation. In this operation, the expansion & the transposition are achieved simultaneously by allowing the first and fourth bits in each block for bit block to appear twice in the output that is the fourth input bit becomes the fifth and the seventh output bit.

The expansion permutation achieves 3 things. Those are described below:

1) It increases the size of the half block from 32 to 48 bit, the same number of bit as in compressed key subset, which is important as the next operation is to XOR the two together.
2) It produces a long string of data for the substitution operation that subsequently comprises it.
3) Because in the subsequent substitutions on the first & 4^{th} bits appearing in 2 S-boxes, they affect two substitutions. The effect of this is the dependency of the

output on the input bits is that the dependency of the output on the input bits increases rapidly.

2.11.1.4 XOR:

XOR operation performed with the appropriate subset key for that round & the resulting 48 block.

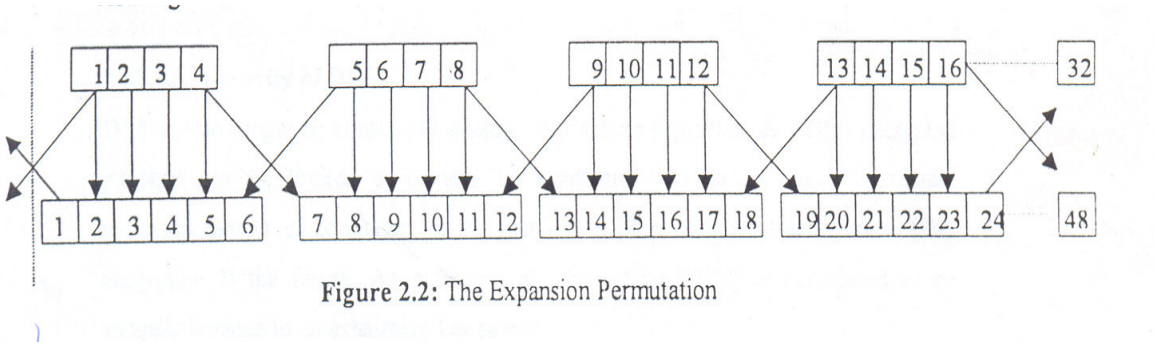


Figure 2.2: The Expansion Permutation

2.11.1.5 Substitution:

After XOR operation the next operation is to perform substitution on the expanded block. There are 8 substitution boxes called S-boxes. The first S-Box operates on the first 6 bits of the 48 bit expanded block, the second S-box on the next 6 & so on. Each S-box operates from a table of 4 rows & 16 columns; each entry on a table is a 4 bit number. The 6 bit number the s-box takes as input is used to look up the appropriate entry in the table in the following way. The first and the 6 bits combine to form a two bit number corresponding to a row number, the second and fifth bit combine to form a 4 bit number corresponding to a particular column. The net result of the substitution phase is 8 4bit blocks that are then combined to form a 32 bit block. It is the non-linear relationship of the S-boxes that really provides DES with its security.

2.11.1.6 Permutation:

The 32 bit output of a substitution phase then undergoes a straight forward transposition using a table called P-Box. After all the round has been completed, the two half blocks of 32 bits are recombined to form a 64 bit output. The final permutation is performed on it, and the resulting 64 bit block is the desired DES encrypted cipher text of the input plain text block.

2.11.1.7 DES Decryption:

If one has the correct key decrypting DES is very easy. The decryption algo is identical to the encryption algo. The only change is to decrypt DES cipher text; the subsets of the keys use in each round are used in reverse, which is the 16^{th} subset is used first.

2.11.1.8 Security of DES:

DES can no longer be considered a sufficiently secured algorithm if the DES secured message can be broken in minutes by a super computer. Then the rapidly increasing power of computer means, it'll be trivial to break DES in future. An extension of DES called DESX is considered virtually immune to key search.

2.12 International Data Encryption Algorithm:

The international data encryption algorithm is a symmetric block cipher developed by Xuejia Lai & James Massey in the Swiss federal institute of Technology in 1990 and was called the proposed encryption standard PES. In 1991, Lai & Massey strengthened the algorithm against differential crypt analysis and called the result improved PES. The IPES name was changed to International Date Encryption Algorithm in 1992.

IDEA is one of a number of conventional encryption algorithms that have been proposed in recent years to replace DES. In terms of adoption, IDEA is one of the most successful of these proposals. IDEA is best known for its use in PGP (pretty good privacy) in network protocol.

2.12.1 The Algorithm

IDEA algorithm with the key length of 128 bits, a block size of the 64 bits and as with DES, the same algorithm provides encryption and decryption. IDEA consists of 8 rounds using 52 sub keys. Each round uses 6 sub keys with the remaining 4 being used for output transformation. The Sub-keys are created as follows:

1) The 128 bit key is divided into 8 16 bit keys to provide the first 8 sub keys.
2) The bits of the original key are then shifted 25 bits to the left, and then it is again split in 8 sub keys.
3) The shifting & splitting is repeated until all 52 sub keys (SK1-SK52) have been created.

The 64 bit plain text is first split into four blocks. A round then consists of the following steps:

Output block-1 (OB1) = b1*sk1 (multiply 1st sub-block with 1st sub key)

(OB2) = b2+sk2 (add 2nd sub-block with 2nd sub key)

(OB3) = b3+sk3 (add 3rd sub-block with 3rd sub key)

(OB4) = b4*sk4 (multiply 4th sub-block with 4th sub key)

(OB5) = OB1 XOR OB3 (XOR results of 1 & 3)

(OB6) = OB2 XOR OB4

(OB7) = OB5 * SK5

(OB8) = OB6+OB7

(OB9) = OB8*SK6

(OB10) = OB7+OB9

(OB11) = OB1 XOR OB9

(OB12) = OB3 XOR OB9

(OB13) = OB2 XOR OB10

(OB14) = OB4 XOR OB10

The input to the next round is the four sub blocks OB11, OB13, OB12, and OB14 in that order.

After the eighth round, the four final output blocks (F1-F4) are used in a final transformation to produce 4 sub blocks of cipher text c1-c4 that are then rejoined to form final 64 bit block of cipher text.

C1=F1*SK49

C2=F2+SK50

C3=F3+SK51

C4=F4+SK52

Cipher Text= C1 C2 C3 C4

2.12.2 Security provided by IDEA:

IDEA is approximately twice as fast as DES and is also considerably more secure. Using a brute force approach there are 2^{128} possible keys. If a billion chips that could each test 1 billion keys a second would try and crack an IDEA encrypted message, it would take them 10^{13} years. Being a fairly new algorithm, it is possible a better attack than brute force will be found, when coupled with more powerful machines in the future may be able to crack a message. However, a long way into future IDEA seems to be a very secure algorithm.

2.13 Blowfish:

Blowfish is a variable-length key block cipher developed by Bruce Schneier. It does not meet all the requirements for a new cryptographic standard discussed above: It is only suitable for applications where the key does not change often, like a communications link or an automatic file encryptor. It is significantly faster than DES when implemented in 32-bit microprocessors with large data caches, such as the Pentium and the Power PC.

2.13.1 Description of the Algorithm:

Blowfish is a variable-length key, 64-bit block cipher. The algorithm consists of two parts; a key-expansion part and a date-encryption part. Key expansion converts a key of at most 448 bits into several sub-key arrays totaling 4168 bytes.

Data encryption occurs via a 16-round Feistel network. Each round consists of a key-dependent permutation, and a key-and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookups per round.

2.13.2 Sub-keys:

Blowfish uses a large number of sub-keys. These keys must be pre-computed before any data encryption or decryption.

1. The P-array consists of 18 32-bit sub-keys:
   P1, P2…P18
2. There are four 32-bit S-boxes with 256 entries each:
   S1, 0, S1, 1 ……., S1, 255;
   S2, 0 S2, 1… S2, 255;

S3, 0, S3, 1… S3, 255;

S4, 0, S4, 1… S4, 255;

The exact method used to calculate these sub-keys will be described later.

2.13.3 Encryption and Decryption:

Blowfish is a Feistel network consisting of 16 rounds. The input is a 64-bit data element, X

Divide X into two 32-bit halves XL, XR

For I = 1 to 16
XL = XL XOR Pi
XR = F (XL) XOR XR
Swap XL and XR
Swap XL and XR (Undo the last swap)
XR = XR XOR P17
XL = XL XOR P18

Recombine XL and XR

Function F:

Divide XL into four eight-bit quarters: a, b, c and d

$F(XL) = ((S_{1,a} + S_{2,b +} \text{ mob } 2^{32}) \text{ XOR } S^{3,C}) + S_{4,d} \text{ MOB } 2^{32}$ (2.25)

Decryption is exactly the same as encryption, except that P1, P2 … P18 are used in the reverse order.

Implementations of Blowfish that require the fastest speeds should unroll the loop and ensure that all sub keys are stored in cache.

Generating the Sub keys:

The sub keys are calculated using the Blowfish algorithm. The exact method is as follows:

1. Initialize first the P-array and then the four S-boxes, in order with fixed string.
   This string consists of the hexadecimal digits of fractional part of Pi
   (Less the initial 3). For example:
   P1 = 243f6a88

P2 = 85a308d3
P3 = 13198a2e
P4 = 03707344

2. XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cycle through the key bits until the entire P-array has been XORed with key bits. (For every short key, there is a least one equivalent longer key' for example, if A is a 64-bit key, then AA, AAA, etc equivalent keys.)

3. Encrypt the all-zero string with the Blowfish algorithm using the subkeys described in steps (1) and (2).

4. Replace P1 and P2 with the output of step (3).

5. Encrypt the output of step (30 using the Blowfish algorithm with the modified sub-keys.

6. Replace P3 and P4 with the output of step (5).

7. Continue the process, replacing all entries of the P-array, and then all four S-boxes in order, with the output of the continuously changing Blowfish algorithm.

In total, 521 iterations are required to generate all required sub-keys. Applications can store the sub-keys rather than derivation process multiple times.

2.14 RC Cipher:

RC stands for *Ron's Code or Rivest Cipher*. These ciphers were designed by Ron Rivest for the RSA Data Security. Different RC Ciphers described briefly below.

2.14.1 RC2:

It was designed as a quick-fix replacement for DES that is more secure. It is a block cipher with a variable key size that has propriety algorithm RC2 is a variable-key-length cipher. However, when using the Microsoft base cryptographic provider, the key-length is hard –coded to 40 bits. When using the Microsoft enhanced

cryptographic provider, the key length is 128 bits by default and can be in the range of 40 to 128 bit in 8-bit increments.

2.14.2 RC4:

It was developed by Ron Rivest in 1987. It is a variable-key-size stream cipher. The details of the algorithm have not been officially published. The algorithm is extremely easy to describe and program just like RC2, 40 bit RC4 is supported by the Microsoft base Cryptography provider, and the enhanced provider allows keys in the range of 40 to 128 bits in 8-bit increments.

2.14.3 RC5:

RC5 is a block designed for speed. It allows a user defined key length, data block size, and number of encryption rounds. In particularly the key size can be as large as 2,048 bits. RSA Data security is working to have RC5 included in numerous internet standards including 1Psec.

2.15 Conclusion:

There are several methods of conventional cryptography, and since it is not possible to present all the methods, very important and popular methods were presented.

It is seen that the modified Hill cipher Encryption and Decryption requires generating random Matrix, which is essentially the power of security. As we know in Hill cipher Decryption requires inverse of the matrix. Hence while decryption one problem arises that is. Inverse of the matrix does not always exist. Then if the matrix is not invertible them encrypted text cannot be decrypted. But this drawback is completely eliminated in modified Hill cipher algorithm.

At the same time, this method requires the cracker to find the inverse of many square matrices which is not computationally easy. So this modified Hill-Cipher method is both easy to implement and difficult to crack.

Chapter 3

PUBLIC-KEY CRYPTOGRAPHY

3.1 Introduction:

Public-Key Algorithms are symmetric, that is to say the key that is used to encrypt the message is different from the key used to decrypt the message. The encryption key, known as the Public key is used to encrypt a message, but the message can only be decoded by the person that has the decryption key, known as the private key.

This type of encryption has a number of advantages over traditional symmetric Ciphers. It means that the recipient can make their public key widely available- anyone wanting to send them a message uses the algorithm and the recipient's public key to do so. An eavesdropper may have both the algorithm and the public key, but will still not be able to decrypt the message. Only the recipient, with the private key can decrypt the message.

A advantage of public-key algorithm is that they are more computationally intensive than symmetric algorithms, and therefore encryption and decryption take longer. This may not be significant for a short text message, but certainly is for bulk data encryption.

3.2 The Basic Principle:

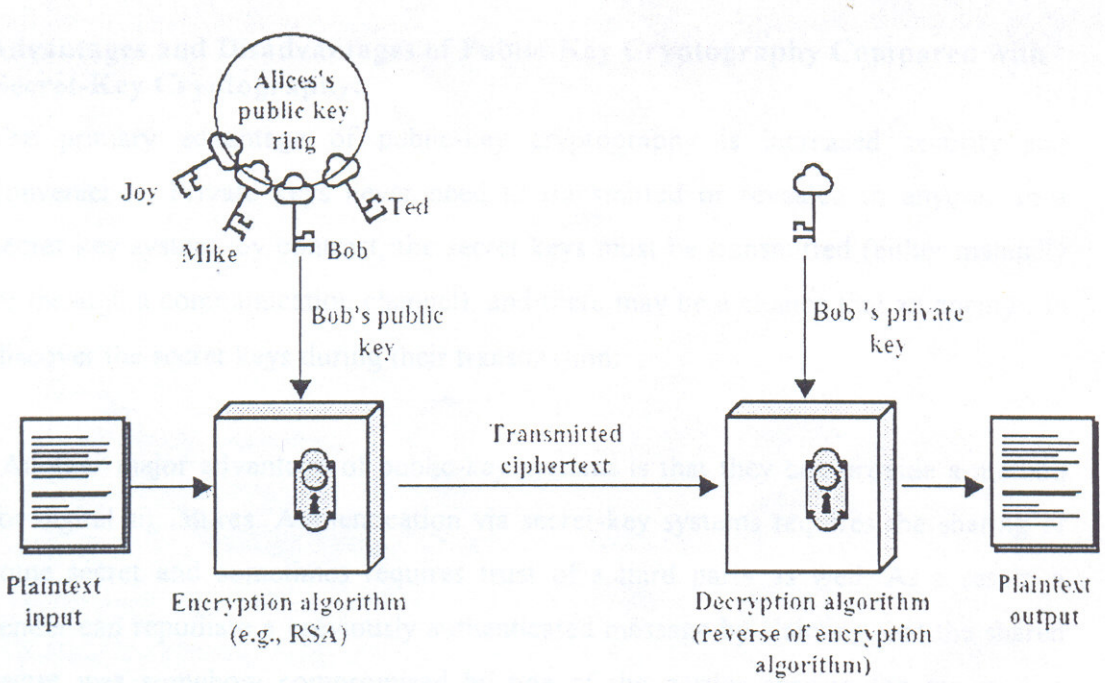
In order to decrypt a message, Bob (the recipient) has to know the key. However, it may be difficult for Alice (the sender) to tell Bob what the key is. If they simply agree on a key bye-mail for example, Eve could be listening in on their e-mail conversation and thus also learn what the key is. Public key cryptography was invented to solve this problem.

When using public-key cryptography, Alice and Bob both have their own key pairs. A key pair consists of a public key and a private-key. If the public-key is used to encrypt

something, then it can be decrypted only using the private-key. And similarly, if the private-key is used to encrypt something, then it can be decrypted only using the

public-key. It is not possible to figure out what the private-key is given only the public-key, or vice versa.

This makes it possible for Alice and Bob to simply send their public keys to one another, even if the channel they are using to do so is insecure. It is no problem that Eve now gets a copy of the public keys. If Alice wants to send a secret message to Bob, she encrypts the message using Bob's public key. Bob then takes his private key to decrypt the message. Since Eve does not have a copy of Bob's private key, she cannot decrypt the message. Of course this means that Bob has to carefully guard his private key. With public key cryptography it is thus possible for two people who have never met to securely exchange messages. Figure 3.1 illustrates the public-key encryption process.



(a) Encryption

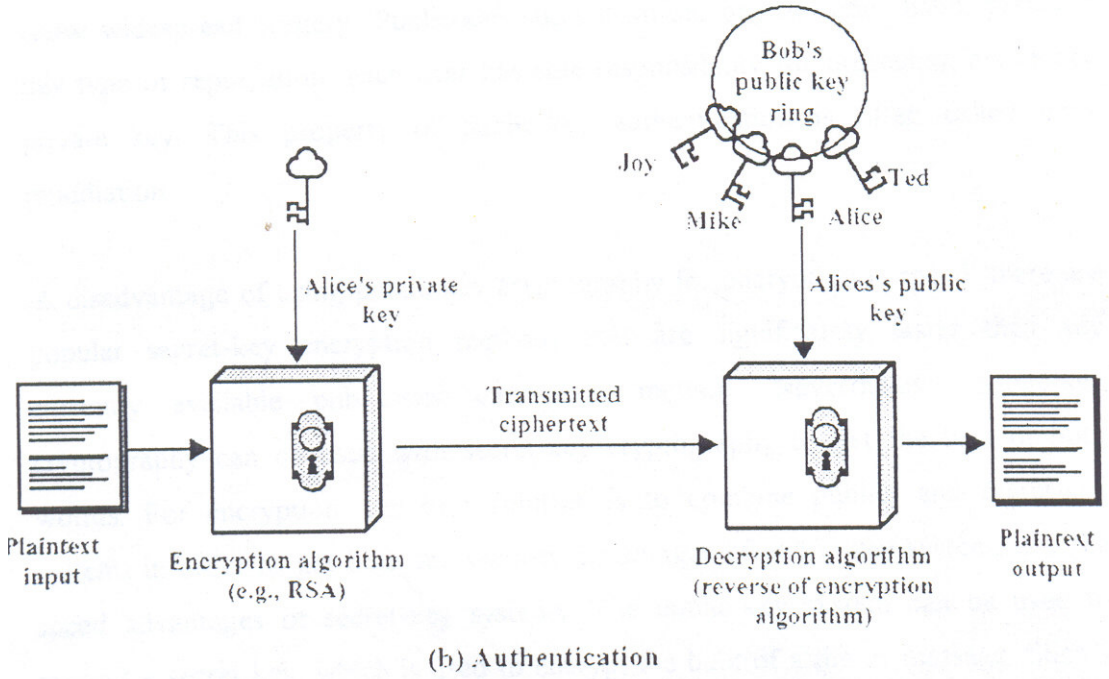


Figure 3.1: Public–Key Encryption

3.3 Advantage and Dis-Advantage of Public-Key Cryptography Compared with Secret-Key Cryptography:

1. The primary advantage of public-key cryptography is increased security and convenience. Private keys never need to transmitted or revealed to anyone. In a secret-key system, by contrast, the secret keys must be transmitted (either manually or through a communication channel), and there may be a chance that an enemy can discover the secret keys during their transmission.
2. Another major advantage of public-key systems is that they can provide a method for digital signatures. Authentication via secret-key systems requires the sharing of some secret and sometimes requires trust of a third party as well. As a result, a sender can repudiate a previously authenticated message by claiming that the shared secret was somehow compromised by one of the parties sharing the secret. For example, the Kerberos secret-key authentication system involves a central database that keeps copies of the secret keys of all users; an attack on the database would allow widespread forgery. Public-key authentication, on the other hand, prevents this type of repudiation; each user has sole responsibility for protecting

his or her private key. This property of public-key authentication is often called non-repudiation.

3. A disadvantage of using public-key cryptography for encryption is speed; there are popular secret-key encryption methods that are significantly faster than any currently available public-key encryption method. Nevertheless, public-key cryptography can be used with secret-key cryptography to get the best of both worlds. For encryption, the best solution is to combine public- and secret-key systems in order to get both the security advantages of public-key systems and the speed advantages of secret-key systems. The public-key system can be used to encrypt a secret key, which is used to encrypt the bulk of a file or message. Such a protocol is called a digital envelope.

4. Public-key cryptography may be vulnerable to impersonation, however, even if users' private keys are not available. A successful attack on a certification authority will allow an adversary to impersonate whomever the adversary chooses to by using a public-key certificate from the compromised authority to bind a key of the adversary's choice to the name of another user.

5. In some situations, public-key cryptography is not necessary and secret-key cryptography alone is sufficient. This includes environments where secure secret - key agreement can take place, for example by users meeting in private. It also includes environments where a single authority knows and manages all the keys (e.g., a closed banking system) Since the authority knows everyone's keys already, there is not much advantage for some to be "public" and others "private" Also, public-key cryptography is usually not necessary in a single-user environment. For example, if you want to keep your personal files encrypted, you can do so with any secret-key encryption algorithm using, say, your personal password as the secret key. In general, public-key cryptography is best suited for an open multi-user environment.

6. Public-key cryptography is not meant to replace secret-key cryptography, but rather to supplement it, to make it more secure. The first use of public-key techniques was

for secure key exchange in an otherwise secret-key system; this is still one of its primary functions. Secret-key cryptography remains extremely important and is the subject of ongoing study and research. Some secret-key cryptosystems are discussed in the sections on Block Cipher and Stream Cipher.

3.4 The RSA Algorithm:

The RSA cryptosystem, named after its inventors R. Rivest, A. Shamir, and L. Adleman, is the most widely used public key Cryptosystem. It may be used to provide both secrecy and digital signatures and its security is based on the intractability of the integer factorization.

3.4.1 Description of the Algorithm:

Two very large prime numbers, normally of equal length, are randomly chosen then multiplied together.

$$N = A \times B \quad (3.1)$$

$$T = (A-1) \times (B-1) \quad (3.2)$$

A third number is then also chosen randomly as the public key (E) such that it has no common factors (i.e. is relatively prime) with T. The private key (D) is then:

$$D = E^{-1} \bmod T \quad (3.3)$$

To encrypt a block of plaintext (M) into cipher text (C) :

$$C = M^E \bmod N \quad (3.4)$$

To Decrypt:

$$M = C^D \bmod N \quad (3.5)$$

3.4.2 Security of RSA:

The security of RSA algorithm depends on the ability of the hacker to factorize numbers. New, faster and better methods for factoring numbers are constantly being devised. The Trent best for long numbers is the Number Field Sieve. Prime Numbers of a length that was unimaginable a mere decade ago are now factored easily. Obviously the longer a number is, the harder is to factor, and so the better the security of RSA. As theory and computers improve, large and large keys will have to be used. The advantage in using

extremely long keys is the computational overhead involved in encryption / decryption. This will only become a problem if a new factoring technique emerges that requires keys of such lengths to be used that necessary key length increases much faster than the increasing average speed of computers utilizing the RSA algorithm. RSA's future security relies solely on advances in factoring techniques.

3.5 Diffie-Hellman Key Exchange Algorithm:

In 1976, Diffie and Hellman published the first public key based algorithm, which was designed to provide a means to exchange securely a key *K* over a public network. That key *K* can later be used as a session key. Note however that this algorithm applies only to the exchange of keys.

Before the key exchange actually begins, two global public values are generated and made public to everyone: a prime number *q* and a, *a primitive root* of *q,* where a < *q.*

A primitive root a of a prime number p is an integer for which the *successive powers* modulo p:

$$a \bmod p,\ a^2 \bmod p,\ a^3 \bmod p,\ \ldots,\ a^i \bmod p,\ \ldots,\ a(p^{-1}) \bmod p \quad (3.6)$$

generates (p - 1) distinct integers, without repetition. The sequence $\{a' \bmod p\}_{i=0}^{(p^{-1})}$ is thus a permutation of the integers from i = 1 to (p - I).
For any integer $b =$ a' mod p, the exponent i is referred to as the *discrete logarithm* (or index) of *b* in base a modulo p.

Suppose that Alice, A, and Bob, B, want to exchange a key *K.* Alice selects a secret random number *XA* < *q,* compute the public value $YA = a^X A \bmod q$ and makes it public. Bob does the same, that is, he generates his private random number *XB* < *q,* calculate *YB* = $a^X B \bmod q$ and publishes *YB.* Then ~ice computes the key *K* as follows:

$K = (YB)^{XA} \bmod q$ (3.7)
$K = (a^x B \bmod q)^{XA} \bmod q$ (3.8)
$K = (a^x B)^{XA} \bmod q$ (Principle of modular arithmetic) (3.9)
$K = a^x B^x A \bmod q$ (3.10)

Similarly, Bob computes K using the public information and his own private random number:

$K = (YA)^X B \bmod q$ (3.11)

$K = (a^x A \bmod q)^{XB} \bmod q$ (3.12)

$K = (a^X A)^{XB} \bmod q$ (3.13)

$K = a^X A^X B \bmod q = a^X B^X \bmod q$ (same key K) (3.14)

The security of the algorithm is based on the difficulty to compute discrete logarithms While it is easy to compute the modular exponentials Y = ax mod q knowing the secret number X, it is difficult for an attacker to compute X from Y and the global public values a and q:

Y= ax mod q (modular exponentiation: easy) (3.15)

X= logy Y mod q (discrete logarithm: difficult) (3.16)

3.6 EIGamal Public Key System:

The ELGamal cryptographic algorithm is a public key system like the Diffie-Hellman system. It is mainly used to establish common keys and not to encrypt message. The EIGamal cryptographic algorithm is comparable to the Diffie-Hellman system. Although the inventor, Taher Elgamal, did not apply for a patent on his invention, the owners of the Diffie-Hellman patent felt this system was covered by their patent. For no apparent reason everyone calls this the IElGamal" system although Mr. Elgamal's last name does· not have a capital letter 'G'[19]

A disadvantage of the EI Gamal system is that the encrypted message becomes very big, about twice the size of the original message m. For this reason it is only used for small messages such as secret keys.

3.6.1 Generating the EI Gamal Public key:

As with Diffie-Hellman, Alice and Bob have a (publicly known) prime number p and a generator g. Alice chooses a random number a and computes A = ga. Bob does the same and computes B = g^b.

Alice's public key is A and her private key is a. Similarly, Bob's public key is B and his private key is b.

3.6.2 Encrypting and Decrypting Messages:

If Bob now wants to send a message m to Alice, he randomly picks a number k, which is smaller than p. He then computes:

$$C_1 = g^k \bmod p \quad (3.17)$$

$$C_2 = A^k * m \bmod p \quad (3.18)$$

And send C_1 and C_2 to Alice. Alice can use this to reconstruct the message m by computing

$$C_1^{-a*} C_2 \bmod p = m \quad (3.19)$$

3.7 Elliptic Curve Cryptography:

Public key cryptography systems are usually based on the assumption that a particular mathematical operation is easy to do, but difficult to undo unless you know some particular secret. This particular secret that serves as the secret key. A recent development in this field is the so-called Elliptic Curve Cryptography.

Elliptic Curve Cryptography works with point on a curve. The security of this type of public key cryptography depends on the elliptic curve discrete logarithm problem. Elliptic curve cryptography was invented by Neil Koblitz in 1987 and by Victor Miller in 1986. The principles of elliptic curve cryptography can be used to adapt many cryptographic algorithms, such as Diffie-Hellman or EIGamal. Although no general patent on elliptic curve cryptography appears to exist, there are several patents that may be relevant depending on the implementation . The main advantage of elliptic curve cryptography is that the keys can be much smaller. Recommended key sizes are in the order of 160 bits rather than 1024 bits for RSA.

3.7.1 Elliptic Curves:

An elliptic curve is a set of points (x, y), for which it is true that

$$y^2 = x^3 + ax + b \quad (3.21)$$

Given certain chosen numbers a and b. Typically the numbers are integers (whole numbers), although in principle the system also works with real (fractional) numbers. Despite what the name suggests, the curves do not have an elliptic shape. For example, -4 and b = 0.67 gives the elliptic curve with equation $y^2 = x^3 - 4x + 0.67$. This curve is illustrated in Figure 3.2.

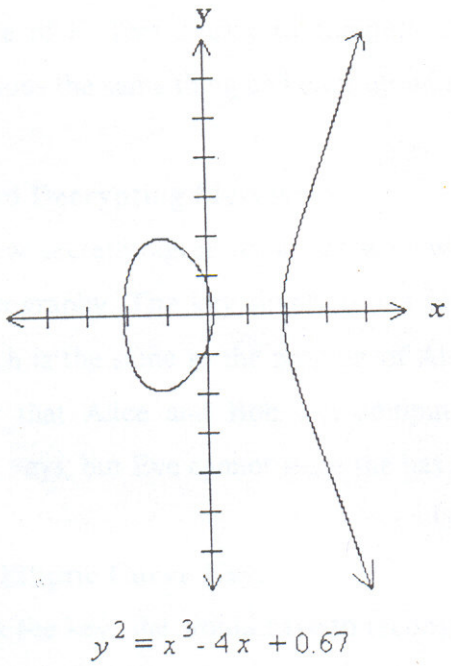


Figure 3.2: Example of Elliptic Curves

If $x^3 + ax + b$ contains no repeated factors, or equivalently if $4a^3 + 27b^2$ is not 0, then the elliptic curve can be used to form a group. A group is simply a set of points on the curve. Because it is a group, it is possible to "add up" points which gives another point on the curve. On the graph, two points are added up by drawing a line through both and taking as the outcome where the line intersects the curve.

For cryptographic purposes, an elliptic curve must have only points with all coordinates whole numbers (integers) in the group.

The trick with elliptic curve cryptography is that if you have a point F on the curve, all multiples of these points are also on the curve [22, 23].

3.7.2 Generating an Elliptic Curve Publickey:

Alice and Bob agree on an elliptic curve and pick a certain point F on this curve. They can do this with Eve listening in. Alice next picks a random number As (which does not have to be a point on the curve) and computes AI' = As*F. The point AI' is on the curve, [because it is a multiple of F. This is easy to compute Alice's public key is AI' and her secret key is As. Bob does the same thing and ends up with Bp and Bs.

3.7.3 Encrypting and Decrypting Messages:

Aice and Bob can now secretly agree on a key with which they can encrypt messages sing secret key cryptography. The key simply is the product of Alice's public key and Bob's secret key (which is the same as the product of Alice's secret key and Bob's public ey). It will be clear that Alice and Bob can compute this product after they have exchanged their public keys, but Eve cannot since she has none of the secret keys.

3.7.4 Cracking the Elliptic Curve Key:

If Eve wanted to crack the key, she would have to reconstruct one of the secret keys This l1eans having to compute As given AI' and F (because AI' = As * F). And that is very difficult.

The number of discrete points on the curve (points with both X and Y coordinates being integers) is called the order of the curve. If the order of the point F is a prime number of n bits, then computing As from As*F and F takes roughly 2"'2 operations. If F is, say, 160 bits long, then Eve needs about 2^{80} operations. If she can do a billion operations per second, this takes about 38 million years. This problem is commonly referred to as the elliptic curve discrete logarithm problem.

3.8 Conclusion:

Brief review of public-key cryptography and implemented RSA algorithm are presented in this chapter. Also some popular public-key cryptography methods presented briefly. Public- key algorithms are based on mathematical functions rather than on substitution and permutation. Hence the advantage of public key algorithm is that they are more computationally intensive than symmetric algorithms. More important, public-key cryptography is asymmetric involving the use of two separate keys, in contrast to symmetric conventional encryption, which uses only one key. The use of two keys has profound consequences in the areas of confidentiality, key distribution and authentication.

Chapter 4

IMPLEMENTATION

Software implementation of the Data encryption and decryption technique.

NOTE: The Mathematical background behind the steps taken below related to the hill cipher technique have been discussed earlier.

4.1 GENERAL STEPS:

1. SELECT A PRIME NUMBER WHOSE MODULAR ARITHMETIC WILL DETERMINE THE EVENTUAL COURSE OF DATA TRANSFER.

2. LIKE IN THIS CASE THE WHOLE ASCII CHARACTER SET IS TO TAKEN AS A RESULT 97 HAS BEEN CHOSEN AND ALL THE ASCII CHARACTERS CORRESPOND TO A PARTICULAR RESIDUE. E.G. 0=A, 1=B…….

3. SELECT A VALUE OF N YOU LIKE TO WORK ON…..LIKE HERE IT HAS BEEN CHOOSEN AS 96

4. DECIDE THE BLOCK SIZE I.E. NO OF CHARACTERS TO BE TRANSFERRED AT THE SAME TIME.GREATER THE BLOCK SIZE GREATER WILL BE DIMENSIONS OF THE MATRIX…..HARDER IT WILL BE TO FIND ITS INVERSE AND HARDER IT WILL BE TO CRACK THE CODE BY BRUTE FORCE. HERE THE BLOCK SIZE HAS BEEN CHOOSEN AS 5.

5. GENERATE A RANDOM DIAGONAL MATRIX 'A' WHOSE DIAGONAL ELEMENTS GIVE A N VALUE LCM OF 96.(AS PREVIOUSLY CHOSEN)

6. NOW SELECT A SQUARE INVERTIBLE MATRIX B AND GENERATE $C=B^{-1}AB$

7. NOW C IS THE DESIRED HILL CIPHER MATRIX.

4.2 STEPS AT THE TRANSMITTER SIDE:

1. TAKE THE DATA IN BLOCKS OF 5 AS A COLUMN MATRIX DENOTED BY E

2. MULTIPLY E THE MATRIX GENERATED EARLIER WITH $C^{(N-M)}$ TO GENERATE THE ENCRYPTION CODE.(C WAS GENERATED EARLIER)

3. M IS SOME RANDOM NUMBER SELECTED AND KNOWN TO BOTH THE ENDS. M<N

4. NOW CONVERT THIS CODE INTO MACHINE CODE TO GIVE BETTER COMPRESSION AND BIT SAVING.

NOTE:THE ALGORITHM FOR CONVERSION INTO MACHINE CODE HAS BEEN DISCUSSED LATER.

5.TRANSMIT (USING MANCHESTER CODING OR ELSE)

4.3 STEPS AT THE RECEIVER SIDE:

1.DECOMPRESS THE HEX CODE INTO MOD-97 CODE

2. THEN MULTIPLY THE CODE COLUMN MATRIX WITH C^{M}.

3. THE CORRESPONDING SUSTITUTIONS ARE MADE AND TEXT RECOVERED..

NOTE: MATHEMATICAL BACKGROUND FOR THE ABOVE TECHNIQUE HAS BEEN GIVEN IN THE SECTION FOR HILL-CIPHER PREVIOUSLY

4.4 METHOD FOR COMPRESSION INTO HEX CODE:

1. TAKE THE CODE IN THE PLACE VALUE OF 97 AND GENERATE A POLYNOMIAL.

2. NOW DIVIDE THE POLYNOMIAL BY 16 TO GENERATE A REMAINDER.

3. THE QUOTIENT GENERATED FORMS THE NEXT DIVIDEND POLYNOMIAL AND DIVISON IS CARRIED OUT ONCE MORE AND REMAINDER COLLECTED.

4. THIS PROCESS IS CARRIED ON UNTILL DIVISOR IS LARGER THAN THE DIVIDEND.

5. ALL THE REMAINDERS ARE COLLECTED AND THE ARRAY IS INVERTED. THIS IS THE COMPRESSED CODE.

4.5 IMPLEMENTATION PROGRAM IN MATLAB

Block Transfer Mode:

TRANSMITTER

```
% transmission in blocks of 5
close all;
clc;
clear all;
mod_val=97;
%Look_up_table=char['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','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','0','
1','2','3','4','5','6','7','8','9','!','@','#','$','%','^','&','*','(',')','-
','_','=','+','[',']','{','}','\','|',';',':','"',',','.','/','<','>','?','`','~'];
% Lookup table forms the basis of assignment of characters into mod 97
```

```
% codes

% Taking the input

Block=input('Enter the block of string','s');
A=Block;
for i=1:5
   if isletter(A(i))
      ascii=char(A(i));   % 65 to 90 capital alphabets
      if(ascii < 91)      % 97 to 122 small alphabets
        B(i)=ascii-65;
      else
        B(i)=ascii-71;
      end
   else
      switch A(i)
         case '0'
            B(i)=52;
            break;
         case '1'
            B(i)=53;
            break;
         case '2'
            B(i)=54;
         case '3'
            B(i)=55;
         case '4'
            B(i)=56;
         case '5'
            B(i)=57;
         case '6'
            B(i)=58;
```

```
case '7'
   B(i)=59;
case '8'
   B(i)=60;
case '9'
   B(i)=61;
case '!'
   B(i)=62;
case '@'
   B(i)=63;
case '#'
   B(i)=64;
case '$'
   B(i)=65;
case '%'
   B(i)=66;
case '^'
   B(i)=67;
case '&'
   B(i)=68;
case '*'
   B(i)=69;
case '('
   B(i)=70;
case ')'
   B(i)=71;
case '-'
   B(i)=72;
case '_'
   B(i)=73;
case '='
   B(i)=74;
```

```
case '+'
   B(i)=75;
case '['
   B(i)=76;
case ']'
   B(i)=77;
case '{'
   B(i)=78;
case '}'
   B(i)=79;
case '\'
   B(i)=80;
case '|'
   B(i)=81;
case ';'
   B(i)=82;
case ':'
   B(i)=83;
case '"'
   B(i)=84;
case ','
   B(i)=85;
case '.'
   B(i)=86;
case '/'
   B(i)=87;
case '<'
   B(i)=88;
case '>'
   B(i)=89;
case '?'
   B(i)=90;
```

```
            case '`'
               B(i)=91;
            case '~'
               B(i)=92;
            case ' '
               B(i)=93;
            end
         end
      end
       B
      % done by using B(inv)AB method in Ngenerator.m
       %P=[02 00 00 00 00;00 03 00 00 00;00 00 04 00 00;00 00 00 05 00;00 00 00 00 06];
    % Stands for the public key

       % Encryption of code

       P=[81 33 72 01 14;01 56 91 20 33;20 72 18 39 52;39 91 33 64 71;57 11 49 74 92];
        Secret_key=40;
       % computing A to the power m
       E=[01 00 00 00 00;00 01 00 00 00;00 00 01 00 00;00 00 00 01 00;00 00 00 00 01];
       for i=1:40
          E=E*P;
          E=mod(E,mod_val);
       end
       E
       % Encryption code

       Code=E*(B'); % Encrypted data
       Code=mod(Code,mod_val);
       Code

    %COMPRESSION INTO HEX CODE
```

```
B1=[0 0 0 0 0 0 0 0 0 0 0 0 0];  % mods i.e. hex data
n=4;
k=1;
j=1;
while(n~=-1)
   quotient=floor((Code(k))/16);
   %quotient
   rem=mod(Code(k),16);
   %rem
   Code(k)=quotient;
   %k
   for i=(k+1):5
      quotient=floor(((rem*97)+Code(i))/16);
      %quotient
      rem=mod(((rem*97)+Code(i)),16);
      Code(i)=quotient;
   end
   %j
   B1(j)=rem;
   %rem
   j=j+1;
   %A
   if(Code(k)==0)
      n=n-1;
      k=k+1;
   end
end
B1
%l=length(B1);
fid=fopen('Results2.txt','w');
fprintf(fid,'%d\n',B1);
fclose(fid);
```

RECEPTION BLOCK WISE

```
% Receiver's software implementation
%Conversion from hex code to mod(97) code
%Receiver.m
close all;
clc;
clear all;
mod_val=97;
%A=input('Enter the values received');
%A=[1 4 13 0 0 13 12 4 6];      % hex data

%Take input from file itself
fid=fopen('Results2.txt','r');
A1=fscanf(fid,'%d');
fclose(fid);
for i=1:9
   A(i)=A1(i);
end
A=fliplr(A);
A

% uncompress the code
B=[0 0 0 0 0];  % mod 97 data
n=8;
k=1;
j=1;
while(n~=-1)
   quotient=floor((A(k))/97);
   %quotient
   rem=mod(A(k),97);
   %rem
```

```
    A(k)=quotient;
    %k
    for i=(k+1):9
        quotient=floor(((rem*16)+A(i))/97);
     %   quotient
        rem=mod(((rem*16)+A(i)),97);
        A(i)=quotient;
    end
    %j
    B(j)=rem;
    %rem
    j=j+1;
    %A
    if(A(k)==0)
        n=n-1;
        k=k+1;
    end
end
%B
%length(B)
for i=1:5
    B1(i)=B(6-i);
end
B1
%length(B1)
% done by using B(inv)AB method in Ngenerator.m
    %P=[02 00 00 00 00;00 03 00 00 00;00 00 04 00 00;00 00 00 05 00;00 00 00 00 06];
% Stands for the public key

    P=[81 33 72 01 14;01 56 91 20 33;20 72 18 39 52;39 91 33 64 71;57 11 49 74 92];

    % Decryption of code
```

```
Secret_key=40;
% computing A to the power m
n=96;

E=[01 00 00 00 00;00 01 00 00 00;00 00 01 00 00;00 00 00 01 00;00 00 00 00 01];
for i=1:56
   E=E*P;
   E=mod(E,mod_val);
end
E
% Encryption code

Code=E*((B1)'); % Encrypted data
Code=mod(Code,mod_val);
Code
Look_up_table=['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','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','0','1','2','
3','4','5','6','7','8','9','!','@','#','$','%','^','&','*','(',')','-
','_','=','+','[',']','{','}','\','|',';',':','"',',','.','/','<','>','?','`','~',' '];
for i=1:5
   m=Code(i);
   Ans(i)=Look_up_table(m+1);
end
   Ans
```

TRANSMISSION WITH FILE:

```
close all;
clc;
clear all;
mod_val=97;
```

```
%Look_up_table=char['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','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','0','
1','2','3','4','5','6','7','8','9','!','@','#','$','%','^','&','*','(',')','-
','_','=','+','[',']','{','}','\','|',';',':','"',',','.','/','<','>','?','`','~'];
% Lookup table forms the basis of assignment of characters into mod 97
% codes
fid=fopen('amogh.txt','r');
   Data=fscanf(fid,'%c');
fclose(fid);
Data
fid=fopen('Results1.txt','w');
%Block=input('Enter the block of string','s');
%A=Block;
% Entering the Data into the amount of encryption required
for x=1:20
   for y=1:5
     A(y)=Data((x-1)*5+y);
   end
%
for i=1:5
   if isletter(A(i))
      ascii=char(A(i));   % 65 to 90 capital alphabets
      if(ascii < 91)      % 97 to 122 small alphabets
        B(i)=ascii-65;
      else
        B(i)=ascii-71;
      end
   else
      switch A(i)
         case '0'
            B(i)=52;
            break;
```

```
case '1'
   B(i)=53;
   break;
case '2'
   B(i)=54;
case '3'
   B(i)=55;
case '4'
   B(i)=56;
case '5'
   B(i)=57;
case '6'
   B(i)=58;
case '7'
   B(i)=59;
case '8'
   B(i)=60;
case '9'
   B(i)=61;
case '!'
   B(i)=62;
case '@'
   B(i)=63;
case '#'
   B(i)=64;
case '$'
   B(i)=65;
case '%'
   B(i)=66;
case '^'
   B(i)=67;
case '&'
```

```
    B(i)=68;
case '*'
    B(i)=69;
case '('
    B(i)=70;
case ')'
    B(i)=71;
case '-'
    B(i)=72;
case '_'
    B(i)=73;
case '='
    B(i)=74;
case '+'
    B(i)=75;
case '['
    B(i)=76;
case ']'
    B(i)=77;
case '{'
    B(i)=78;
case '}'
    B(i)=79;
case '\'
    B(i)=80;
case '|'
    B(i)=81;
case ';'
    B(i)=82;
case ':'
    B(i)=83;
case '"'
```

```
            B(i)=84;
        case ','
            B(i)=85;
        case '.'
            B(i)=86;
        case '/'
            B(i)=87;
        case '<'
            B(i)=88;
        case '>'
            B(i)=89;
        case '?'
            B(i)=90;
        case '`'
            B(i)=91;
        case '~'
            B(i)=92;
        case ' '
            B(i)=93;
        end
    end
  end
   B

   % done by using B(inv)AB method in Ngenerator.m
   %P=[02 00 00 00 00;00 03 00 00 00;00 00 04 00 00;00 00 00 05 00;00 00 00 00 06];
srands 4 the diagonal matrix
   P=[81 33 72 01 14;01 56 91 20 33;20 72 18 39 52;39 91 33 64 71;57 11 49 74 92];
   % Encryption of code
   Secret_key=40;
   % computing A to the power m
   E=[01 00 00 00 00;00 01 00 00 00;00 00 01 00 00;00 00 00 01 00;00 00 00 00 01];
```

```
    for i=1:40
        E=E*P;
        E=mod(E,mod_val);
    end
    E
    % Encryption code

    Code=E*(B'); % Encrypted data
    Code=mod(Code,mod_val);
    Code

B1=[0 0 0 0 0 0 0 0 0 0 0 0 0];  % mods i.e. hex data
n=4;
k=1;
j=1;
while(n~=-1)
    quotient=floor((Code(k))/16);
    %quotient
    rem=mod(Code(k),16);
    %rem
    Code(k)=quotient;
    %k
    for i=(k+1):5
        quotient=floor(((rem*97)+Code(i))/16);
        %quotient
        rem=mod(((rem*97)+Code(i)),16);
        Code(i)=quotient;
    end
    %j
    B1(j)=rem;
    %rem
    j=j+1;
```

```
   %A
   if(Code(k)==0)
      n=n-1;
      k=k+1;
   end
end
B1
%fid=fopen('Results1.txt','w');
fprintf(fid,'%d\n',B1);
end
fclose(fid);
```

RECEPTION WITH FILE:

```
% Receiver's software implementation
%Conversion from hex code to mod(97) code
%Receiver.m
close all;
clc;
clear all;
mod_val=97;
%A=input('Enter the values received');
%A=[1 4 13 0 0 13 12 4 6];      % hex data
o=1;
fid=fopen('Results1.txt','r');
D1=fscanf(fid,'%d');
fclose(fid);
%****D1

for x=1:20
   for y=1:13
   A1(y)=D1(((x-1)*13)+y);
   end
```

```
   for i=1:9
   A(i)=A1(i);
   end
A=fliplr(A);
%***A

% uptil  now input taken

%now uncompression done

B=[0 0 0 0 0];  % mod 97 data
n=8;
k=1;
j=1;
while(n~=-1)
   quotient=floor((A(k))/97);
   %quotient
   rem=mod(A(k),97);
   %rem
   A(k)=quotient;
   %k
   for i=(k+1):9
      quotient=floor(((rem*16)+A(i))/97);
    %   quotient
      rem=mod(((rem*16)+A(i)),97);
      A(i)=quotient;
   end
   %j
   B(j)=rem;
   %rem
   j=j+1;
   %A
```

```
   if(A(k)==0)
      n=n-1;
      k=k+1;
   end
end
%B
%length(B)
for i=1:5
   B1(i)=B(6-i);
end

%DECRYPTION
%***B1
%length(B1)
% done by using B(inv)AB method in Ngenerator.m
   %P=[02 00 00 00 00;00 03 00 00 00;00 00 04 00 00;00 00 00 05 00;00 00 00 00 06];
% Stands for the public key
   P=[81 33 72 01 14;01 56 91 20 33;20 72 18 39 52;39 91 33 64 71;57 11 49 74 92];
   % Decryption of code
   Secret_key=40;
   % computing A to the power m
   n=96;

   E=[01 00 00 00 00;00 01 00 00 00;00 00 01 00 00;00 00 00 01 00;00 00 00 00 01];
   for i=1:56
      E=E*P;
      E=mod(E,mod_val);
   end
   %****E
   % Encryption code

Code=E*((B1)'); % Encrypted data
```

```
Code=mod(Code,mod_val);
% ****Code
Look_up_table=['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','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','0','1','2','
3','4','5','6','7','8','9','!','@','#','$','%','^','&','*','(',')','-
','_','=','+','[',']','{','}','\','|',';',':','"',',','.','/','<','>','?','`','~',' '];
%length(Look_up_table)

for p=1:5
   m=Code(p);
   Ans(((o-1)*5)+p)=Look_up_table(m+1);
end
   o=o+1;
%***Ans
end
Ans
```

CONCLUSION:

The HILL cipher technique using a novel method of self repetitive matrix was successfully implemented. A transmitter-receiver pair was successfully modeled which used proper decompression techniques for effective communication. The numerical method suggested to find N value of a matrix was successfully tested and used in the implementation. Another line of investigation also opened some correlation between Eigen values and N value of a matrix.

REFERENCES:

[1] W.Stallings; "Cryptography and Network Security" 2nd Edition, Prentice Hall,1999

[2] Bruce Schneir: Applied Cryptography, 2nd edition, John Wiley & Sons, 1996

[3] Piper,F "Encryption". Security and Detection, Ecos 97. European Conference;

[4] Abrams, M., and Podell, H. "Cryptography" Potentials, IEEE Page No 36-38. Issue:1,Volume: 20, Feb-Mar,2001

[5] Eskiciogiu,A. Litwin,L " Cryptography and Network Security" LOS Alamitos,CA: IEEE computer society press,1987

[6] Garfinkel, S.L; "Public Key Cryptography" , Computer, IEEE, Volume: 29, Issue:6, June 1996.

[7] W.Diffie; M.E.Hell man, " New Directions in Cryptography" IEEE Transactions Information Theory, Nov, pp 644-654

[8] E.Biham and A.Shmir; "Differential C for Cryptanalysis of the Encryption Standard"; Springer- Verilag,1993

[9] Bidjos,J; "Threats to Private and Public Key Protection" , Compcon Spring '91. Digest of Papers, 25 Feb-1 March 1991

[10] V. Miller ; "Uses of Ellptic Curves in Cryptography. In advances in Crptography, Springer Verlag Crypto 95.