Performance Analysis of Hill Cipher and Its Modifications

Gülbahar Akgün

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Applied Mathematics and Computer Science

Eastern Mediterranean University
February 2015
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Serhan Çiftçioğlu
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Prof. Dr. Nazım Mahmudov
Chair, Department of Mathematics

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Prof. Dr. Rza Bashirov
Supervisor

Examining Committee

1. Prof. Dr. Rashad Aliyev ____________________

2. Prof. Dr. Rza Bashirov ____________________

3. Asst. Prof. Dr. Hüseyin Lort ____________________

ABSTRACT

The history of cryptography goes to several thousand years back when ancient Egyptions tried to hide text by using unusual hieroglyphs instead of more ordinary ones here and there on a tablet. Although lot of cryptographic algorithms have been developed and practically used in various areas, the choice of best algorithm is still on focus of researchers. Since encryption/decryption is an expensive operation, the researchers have always tried to compromise between performance measured in terms of time-effectiveness and confidentiality (or secrecy) provided by cryptographic algorithms. The researchers have realized that the best cryptographic algorithm is determined by reasonable trade-off between performance and confidentiality of the cryptosystem.

In this thesis we investigate performance of three cryptographic algorithms, namely Hill cipher, affine Hill cipher and Saeednia’s modification. We perform comparative analysis of aforesaid cryptographic algorithms via measuring run times on different sized problems. Computer experiments are performed in MATLAB, a high-level technical computing language and interactive environment for algorithm development.

Keywords: Cryptography, Hill cipher, affine Hill cipher, Saeednia’s algorithm, linear transformation, permutation matrix

ÖZ

Kriptografinin tarihi birkaç bin yıl önceye antik Mısırlıların alışılmadık hiyeroglifler kullanarak tablet üzerindeki metinleri sakladıkları döneme kadar uzanır. Günümüzde çok sayıda kriptografik algoritma bulunmasına ve bu algoritmaların çeşitli alanlarda kullanılmasına rağmen, en iyi kriptografik algoritma seçimi halen araştırmacıların dikkat ettikleri konulardandır. Şifreleme/şifre çözme pahalı bir işlem olduğundan, en iyi algoritmanın seçimi için şifreleme algoritmalarının performansı ve gizlilik arasındaki bağlantı zemininde seçim yapılır. İyi kriptografik algoritmanın performans ve gizlilik arasında makul seçim yaparak belirlenmesi konusunda araştırmacılar ortak fikir belirlemişlerdir.

Bu tezde Hill şifreleme yöntemi, afin Hill şifreleme yöntemi ve Saeednia yöntemi farklı büyüklükte matrisler kullanarak çalışma sürelerinin ölçülmesi ve kıyaslanması şeklinde karşılaştırmalı olarak irdelenmiştir. Bilgisayar deneyleri için yüksek seviyeli teknik hesaplama dili ve algoritma geliştirme aracı MATLAB kullanılmıştır.

Anahtar Kelimeler: Kriptografi, Hill şifreleme yöntemi, lineer transformasyon, permütasyon matrisi

DEDICATION

Dedicated to the Akgün Family

ACKNOWLEDGMENT

Firstly, I would like to thank Prof. Dr. Rza Bashirov for the help, motivation and patience he has shown throughout my studies. He is a person that I have always taken as a model for his kindness and moral values.

Special thanks to Prof. Dr. Mehmet Akbaş, Prof. Dr. Agamirza Bashirov, Prof. Dr. Rashad Aliyev and Dr. Pembe Sabancıgil who have always helped me with the courses I have taken.

I would like express my gratitude to my father Kemal Akgün, my mother Asiye Akgün, my uncle Temel Akgün, my aunt Samiye Akgün, my brother Fatih Mehmet Akgün and sisters Zeynep and Elanur Akgün for their support throughout my thesis studies. Also special thanks to the whole Akgün family.

Finally I would like to thank all my friends and my moral supporter Fatma Erik for supporting and motivating me during study on the.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

A	set of arcs
x	plaintext
y	ciphertext
H	key matrix
n	number of letters in an alphabet
m	block size; number of letters in polygrafig cipher
KU_A	Alice’s public key
KR_A	Alice’s private key
E_{KU_A}	Encryption with Alice’s public key
D_{KR_A}	Decryption with Alice’s private key
p, q	prime numbers
M	plaintext
C	ciphertext
φ	Euler’s totient function
e	component of a public key
d	components of a private key
V	shift vector
π	random permutation
P_π	permutation matrix induced by π

LIST OF TERMS

Asymmetric key cryptography	a class of algorithms, which require two separate keys, public key and private key that are respectively used for encryption and decryption
Attack	attempt to break the cipher and recover plaintext
Authentication	the process of confirming the identity of a person
Block cipher	processing of the data in a fixed-length groups
Cipher	algorithm that represents information in a scrambled way
Cipherletter	a letter of a ciphertext
Ciphertext	encrypted plaintext
Confidentiality	a set of rules or a promise that limits access or places restrictions on certain types of information
Cryptography	
Decryption	a reverse of encryption i.e., recovering original written document from encrypted one
Decryption algorithms	a method that can be used to decrypt the written document
Encryption	hiding the contents of a written document from those unauthorised persons
Encryption algorithm	a method that can be used to encrypt the written document
Plain letter	
Plaintext	ordinary readable text

Polygraphic cipher	A polygraphic cipher is a cipher where the plaintext is divided into groups of adjacent latters of the same fixed length, and then each such group is transformed into a group of letters of the same size
Secret key	a component of encryption in conventional cryptography
Stream cipher	an algorithm that processes text symbol by symbol
Symmetric key cryptography	same as conventional cryptography
Plaintext letter	readable text to be encrypted
Polygraphic cipher	a conventional cipher that processes text in groups of symbols
Private key	a key that is usually used for decryption
Public key	a key that is usually used for encryption
Unauthorized party	A person who tries to catch and read the messages, but who is not allowed to do so

Chapter 1

INTRODUCTION

The Hill cipher was invented by Leslie Hill in 1929. The algorithm presents an example of a polygraphic substitution cipher. Hill's major contribution was the use of linear algebra in design and analysis of cryptosystems. Although there exist plenty of cryptographic techniques, Hill cipher is the only algorithm that is fully based on linear algebra. The Hill cipher uses matrices and matrix multiplication to represent the plaintext in a scrambled manner.

Given plaintext x, Hill cipher converts x into ciphertext $y = H \cdot x \pmod{n}$ where n is the number of letters in the alphabet. It is well-known that every cryptographic technique is vulnerable to attacks of unauthorized persons. Particularly, Hill cipher can be relatively easily broken [5]. If cryptanalyst knows m pairs (x, y) of successive plaintext and ciphertext, he/she can calculate the key matrix H by computing $H = Y \cdot X^{-1} (\bmod n)$, where X and Y are $m \times m$ matrices of plaintext and ciphertext.

In attempt to correct security flaws detected in Hill cipher several researchers have presented its variants. They have modified Hill cipher to improve its security. The modifications of Hill cipher are based on a variety of techniques. In [7] the authors suggested to use two coprime numbers for securely sharing between the participants. Their algorithm is too time-consuming, requiring a lot of mathematical manipulations. Also the algorithm is not efficient especially when dealing with a bulk data, though the proposed algorithm thwarts the known plaintext attacks. In [6] it is suggested to make Hill cipher more secure using some random permutations of columns and rows of the key matrix. As it was later observed that, this cryptosystem is also vulnerable to known plaintext attacks, the same problem arising in Hill cipher [3]. An algorithm improving the security of Hill cipher introducing several random numbers produced by a hash chain is described in [3]. Another modification of Hill cipher uses an initial vector that multiplies successively by some orders of the key matrix to produce the corresponding key of each block [1]. As it was reported in [8] this algorithm also suffers from same problem, as it has several security problems.

The present thesis is a comprehensive study of Hill cipher and its two modifications, namely affine Hill cipher and Saeednia’s algorithm. Although affine Hill cipher and Saeednia’s modification add more power into Hill cipher by patching the security holes observed in Hill cipher, they also suffer from similar security problem. Performance is another factor that plays important role in selection of cryptographic algorithms. In the current work performance is measured in terms of run time needed to complete task. The best strategy in selection of cryptographic algorithms is perhaps based on trade-off between confidentiality provided by algorithm and cost of the algorithm measured in terms of its time-efficiency. In the present thesis we compare the aforesaid algorithms in terms of time-effectiveness. We measure run time needed to complete encryption and decryption tasks for all three algorithms. It is done for different-sized key matrices. Then we compare the statistical results on run times for all three algorithms and make conclusions regarding their time-effectiveness.

The thesis is organized as follows. In Chapter 2, we provide general information about cryptographic methods: We discuss taxonomy of cryptographic algorithms, succinctly overview classes of cryptographic algorithms based on number of the keys used for encryption and decryption, type of operation used to hide information, the way plaintext/ciphertext is processed, the number of alphabets used to create ciphertext. Then we review Hill cipher, affine Hill cipher and Saeednia’s modification of Hill cipher, and discuss security problems observed for these algorithms. After that we provide results of computer experiments and make conclusions. The list of references is provided at the end of the thesis.

Chapter 2

CRYPTOGRAPHIC METHODS

2.1 Brief history

Cryptography is a systematic study of hiding techniques that just started around a hundred years ago though it has been used for thousands of years. First known evidence of the use of cryptography was found in Egypt. Ancient Egyptions used some unusual hieroglyphs instead of more ordinary ones here and there on a tablet to hide the text. Over the many years many cryptographic techniques have been proposed and practically used in various areas. Depending on type of transformation, the way the plaintext and ciphertext are processed, and number of the keys used for encryption/decryption the cryptographic techniques can be classified into several important classes. Figure 2.1 demonstrates the taxonomy of cryptographic techniques.

In what follows we succinctly explain each class of cryptographic techniques, and refer readers to [4] for detailed information.

2.2 Symmetric vs Asymmetric Cryptography

In this thesis we repeatedly refer to Alice and Bob, the commonly used placeholder names, which are used as archetypal characters throughout the thesis to explain the major principles of cryptography. For example, "Alice sends an encrypted message to Bob" is understood as "Party A sends an encrypted message to Party B. "The main logic behind of the symmetric cryptography can be explained introducing the following example.

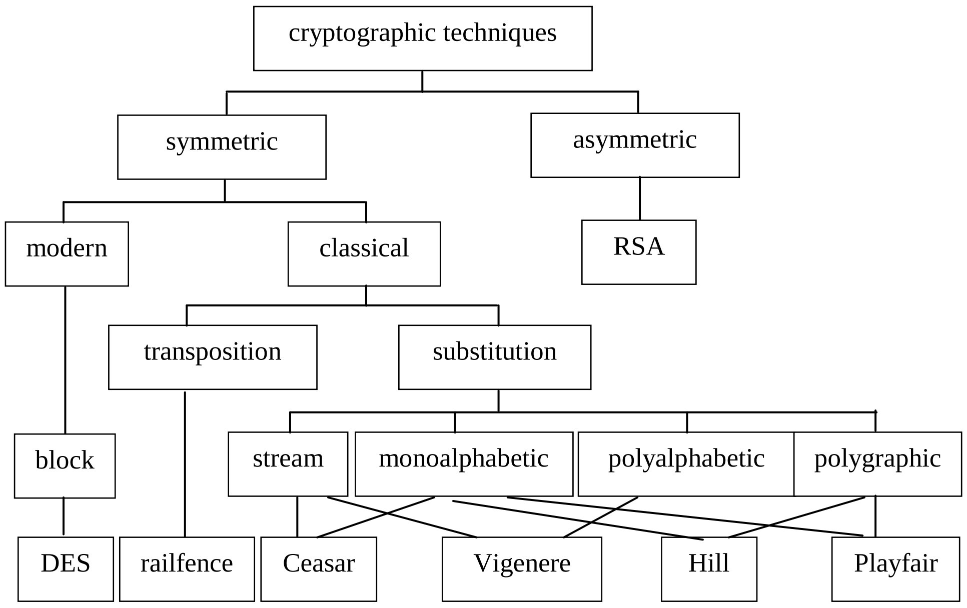

Figure 2.1: Taxonomy of cryptographic techniques.

Let us assume that Alice wants to send secret message to Bob. Alice uses a key to encrypt the message since the message is confidential. Bob is the only one who should gain access to the message. Alice sends ciphertext together with the key to Bob. Bob uses the key and same cipher to decrypt the message. In this example Alice and Bob share the key, that is called symmetric key. Alice and Bob are the only ones who are allowed to know the secret key and read the message. No one else is allowed to know the secret key and read the encrypted message. This is the way how confidentiality is achieved in symmetric cryptography. The way asymmetric-key cryptography works is detailed in Figure 2.2.

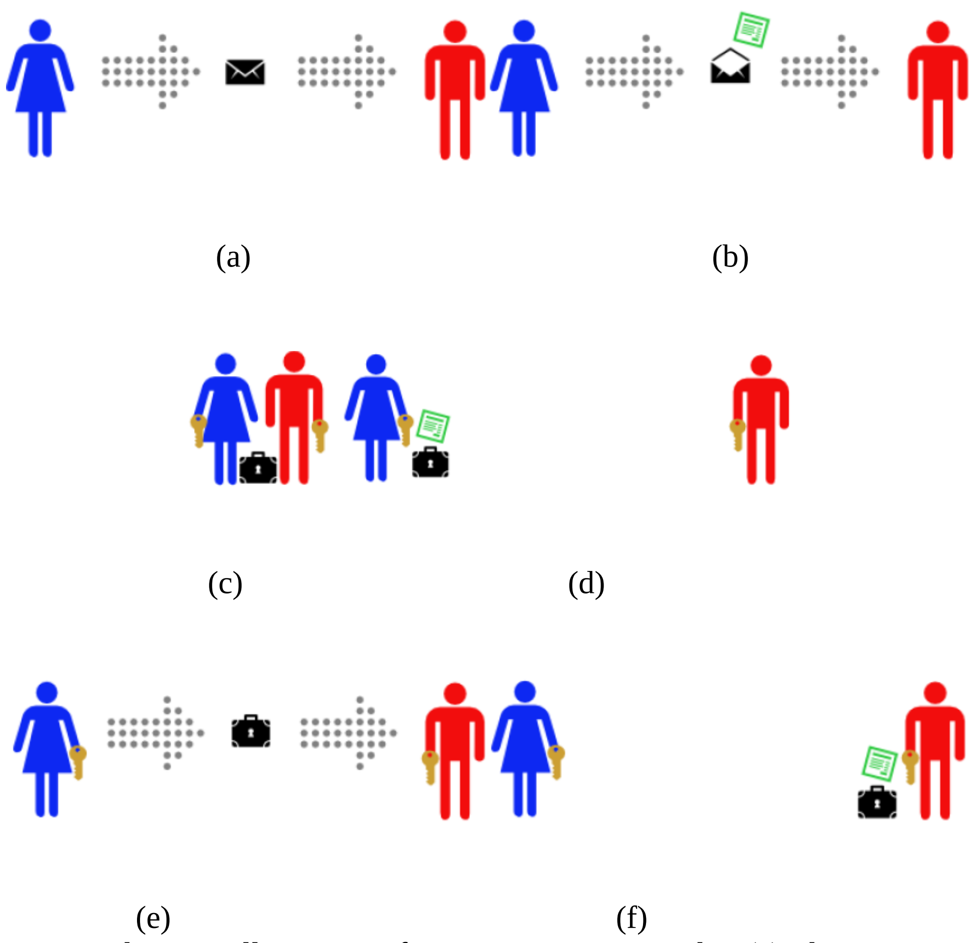
Figure 2.2: Schematic illustration of symmetric cryptography. (a) The easiest way for Alice to send a private message to Bob is via postal mail. (b) Alice knows that the postman is eager to read the mail she will send. Thus, Alice decides to develop a way to send secret messages to Bob so that nobody being able to read them. (c) Alice buys lockbox with two identical keys, puts the message inside a lockbox and sends the lockbox to Bob. She realizes that she needs to send the key to Bob. Since the postman is curious to open the lockbox and take a copy of the key. Alice thereby meets Bob at a nearby bar to give him one of the keys. It is inconvenient, but she only has to do it once. (d) After Alice gets home she uses her key to lock her message into the lockbox. (e) Then she sends the lockbox to Bob. What the mailman could do is to look at the lockbox, hide the lockbox so that Bob doesn't get the message, but what is for sure there is no way for postman to read the message, as he cannot open the lockbox. (f) On the other hand, Bob can simply use the identical key to unlock the lockbox and read the message. It is somewhat like handshaking in sense that Bob can use the same method for securely replying since Alice and Bob have identical keys.

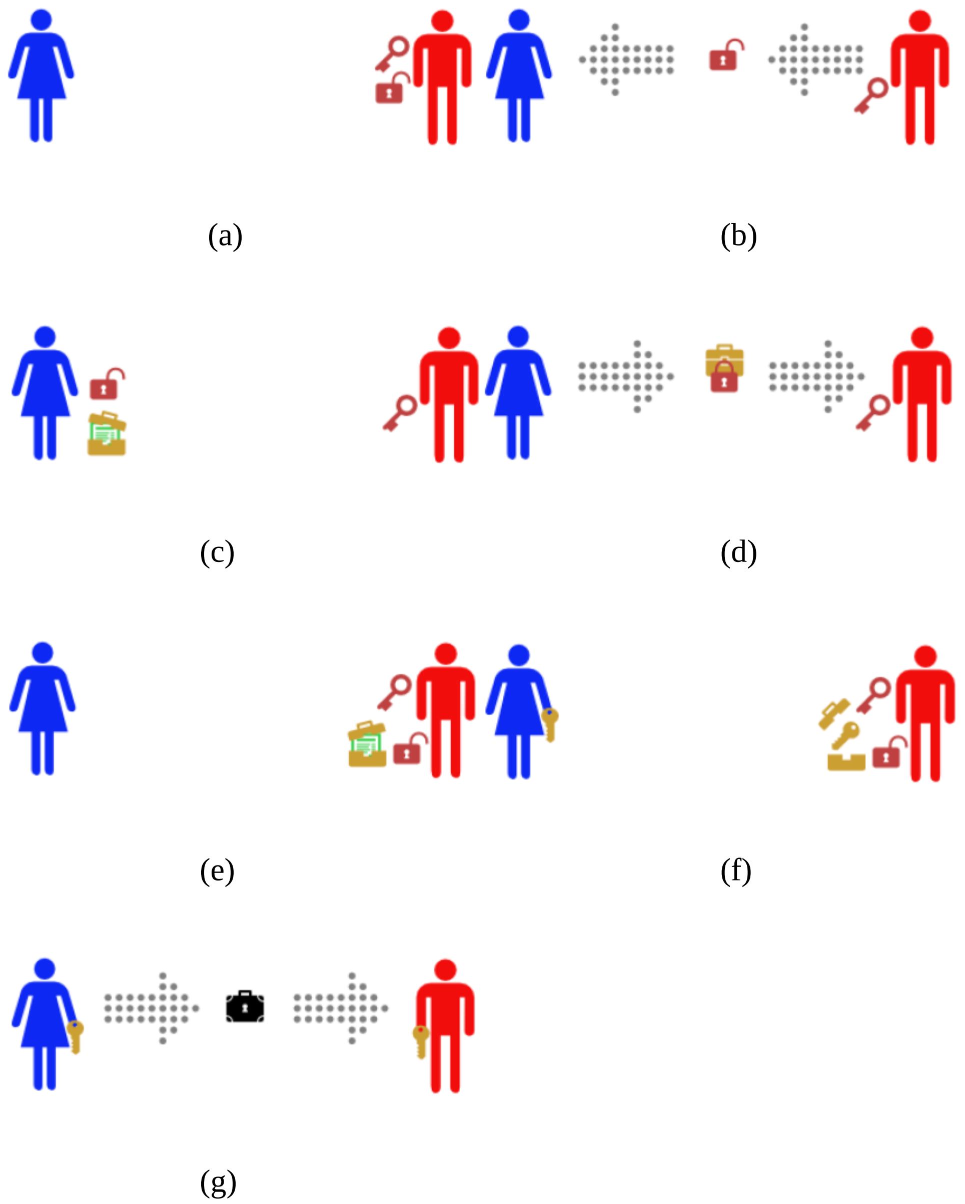

Figure 2.3: Schematic illustration of asymmetric cryptography. (a) This time, Alice and Bob do not meet at all. First Bob gets a lock and a matching key. (b) Then Bob sends the unlocked lock to Alice, keeping the key in secret. (c) Alice gets a lockbox, puts the message in it. (d) After that she locks the lockbox with Bob's lock and mails it to Bob. (e) She is sure that the mailman is not capable anymore to read the message as he has no way to open the lock. Bob receives the lockbox, opens it with the key and reads the message. (f) The above principle works if the messages are sent in one direction. To make the communication link bidirectional Alice firstly buys a blue lock, a key and then mails the lock to Bob so that he can reply. Alice sends Bob one of the keys. (g) Alice and Bob are now able to communicate in terms of symmetric-key lockbox.

In asymmetric cryptography a pair of keys is used to encrypt/decrypt a message. One of the keys called a public-key that is known to everybody. Another key referred as a private-key is known to the recipient of the message. In asymmetric cryptography, anyone can encrypt messages using the public key, but only the holder of the paired private key can decrypt it. Private-key is always kept in secret, while public-key is publicly shared with the others.

Returning to our coined example, Alice and Bob both get a pair of keys $(KU_A,\ KR_A)$ and $(KU_B,\ KR_B)$ one is public key and another is private key. Each of them keeps private key in secret and publishes public key. When Alice wants to send a message X confidentially to Bob, and wants to be sure that only Bob may be able to read it Alice encrypts the message with Bob's public key $Y = E_{KU_B}(X)$ and sends ciphertext Y to Bob. On receipt of the ciphertext Y Bob decrypts it with his own private key, and recovers the original plaintext as $X = D_{KR_B}(Y) = D_{KR_B}(E_{KU_B}(X))$. The logic behind of asymmetric-key cryptography is carefully detailed in Figure 2.3.

Over the many hundred years number theory was one the purest branches of mathematics and it remained so until second half of 20th century when modern cryptography has started developing. Public-key cryptography has turned number theory into area of applied mathematics which particularly played important role in developing RSA algorithm, the only fully described and practically implemented public-key algorithm. RSA, stands for Ron **R**ivest, Adi **S**hamir and Leonard **A**dleman, who were originally developed the algorithm in 1977. Particularly, selection of pair of keys in RSA is done in terms of primality test. Since RSA operates with large primes, fast and trustful primality test algorithms are of ultimate interest in public key cryptography.

If p and q are two primes, block of plaintext in RSA algorithm takes value less than $n = p \cdot q$, meaning that block size is defined by $\log_2 n$. Given plaintext M and ciphertext C, encryption and decryption is carried out as

$$C = M^e (\mathrm{mod}\, n), \tag{1}$$

$$M = C^d (\mathrm{mod}\, n) = (M^e)^d (\mathrm{mod}\, n) = M^{e \cdot d} (\mathrm{mod}\, n). \tag{2}$$

RSA uses corollary to Euler's Theorem: Given two prime numbers, p and two integers, n and m such that $n = pq$ and $0 < m < n$ arbitrary integer k, the following is valid:

$$m^{k\varphi(n)+1} = m^{k(p-1)(q-1)+1} = m(\mathrm{mod}\, n), \tag{3}$$

where $\varphi(n)$ is Euler's totient function defined as the number of integers that are less than or equals to n and that are coprime to n. Thus, e and d can be easily found from (2) and (3) as follows:

$$ed = k\varphi(n) + 1,$$

$$ed \equiv 1(\mathrm{mod}\, \varphi(n)),$$

$$d \equiv e^{-1}(\mathrm{mod}\, \varphi(n)). \tag{4}$$

Thus, e is chosen according to identity $\gcd(\varphi(n), e) = 1$, $1 < e < \varphi(n)$, mean while d is determined from (4). The private and public key are defined as $\{d, n\}$ and $\{e, n\}$ respectively. Once public and private key are determined, encryption and decryption can be performed in accordance with (1) and (2).

2.3 Classical vs Modern Methods

We distinguish between classical and modern symmetric cryptographic ciphers. Classical ciphers as earliest forms of secret writing require a little more than local pen and paper and based on processing of text prepared in natural languages such as English, French, etc. It is well-known that ciphertext produced by a classical cipher always reveals statistical information about the plaintext, which makes it easy for unauthorized party to break the code and recover the plaintext.

As it is believed that frequency analysis method was proposed by Arab mathematician Al-Kindi, who is also known as Alkindus, in the 9th century [2]. The main idea behind of the frequency analysis method is to discover similarities between known relative frequencies of the letters, combinations of two-letters, three-letters, etc. and those in a plaintext, and guess the plain letters by using brute-force attack consequently substituting the cipher letters by candidate plain letters. What is true is that although partially recovered plaintext is not fully readable it may still contain essential information about plaintext. One obvious result of the frequency analysis method is that if a single alphabet is under consideration nearly all ciphers became more or less readily breakable by any informed attacker. This is why ciphers based on use of a single alphabet although still enjoy popularity today, mostly remain as puzzles. Frequency analysis method has remained as the most powerful method until the development of the polyalphabetic cipher. The monoalphabetic and polyalphabetic ciphers are detailed in Subsection 2.3.

Modern cryptographic methods are based on processing of digital information. The development of digital computers and electronics has changed the outlook on cryptography. Transition from natural language to binary or boolean alphabet made it possible to develop much more complex ciphers. Furthermore, computers are capable to encrypt/decrypt multimedia data that is made of text, image and voice information. All these data are representable in binary format. This is essential departure from the classical ciphers which are applicable for documents prepared in natural languages only. Modern ciphers are characterized along several dimensions such as block size, F-function, number of rounds and S-box.

Unlike classical methods, which generally manipulate with characters or small groups of characters, modern methods split text into groups (or blocks) and process group-wise (or block-wise). Nowadays, block size of 32-, 64- or even 128-bit is rather reasonable choice in real life applications. There is direct proportional relationship between block size and message confidentiality. The level of message confidentiality increases with increase of block size. On the other hand, the cost of the algorithm, which is measured in terms of time required to complete the task, also increases with increase of the block size. So, there must be a reasonable compromise between level of confidentiality and price spent to break the cipher. As complex the cipher becomes it is more difficult to break it, hence more time and effort should be spent to break it.

The key size is just another parameter that essentially affects the outcome in modern cryptographic methods. Some researchers claim that in modern techniques the key but not the encryption/decryption algorithm is of primary interest. The key size is in direct proportional relationship with the confidentiality. Increase of the key size increases the level of confidentiality. The key size of 64- and 128-bit is quite reasonable compromise in sense of security and time-effectiveness of encryption/decryption processes.

In modern cryptography, it is quite customary to repeat core of the algorithm called round until reaching desired level of secrecy. It must be noticed all rounds have the same structure. Round is a mixture of Boolean operations and substitutions of the message fragments that integrates diffusion and confusion principles, techniques used for hiding statistical properties of the key and plaintext inside ciphertext. Greater the number of rounds, it is more difficult to perform cryptanalysis. The following criterion remains true in practical applications: the number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple linear cryptanalysis.

One common misconception concerning public-key encryption is that public-key encryption is more secure compared to conventional encryption. The level of confidentiality depends on such parameters as key length, inner structure of encryption algorithm, etc. In fact, none of two types of cryptography is superior to another from the point of view of resisting cryptanalysis. There is nothing in principle about either conventional or public-key encryption that makes one superior to another from the point of view of resisting cryptanalysis.

Modern cryptographic methods are tightly related with such notions as integrity checking, sender/receiver identity authentication, digital signatures. Let us have a closer look at the way identity authentication is accomplished in public-key cryptography. Let us notice that authentication is the process of confirming the identity of a person. Assume Alice wants to send message X confidentially to Bob, and meantime wants to authenticate the message. Initially Alice and Bob respectively generate a pair of keys, one is public and another is private $\{KU_A, KR_A\}$ and $\{KU_B, KR_B\}$. Each of them publishes own public key and keeps private key in secret. Public-key cryptography allows us to encrypt the message with any of the keys, while only the other key can be used to decrypt it. Since Alice knows Bob's public key KU_B and own public and private keys $\{KU_A, KR_A\}$, she has two choices for encryption: She either encrypts message X with Bob's public key KU_B and sends ciphertext $Y = E_{KU_B}(X)$ to Bob or uses own private key KR_A to authenticate the plaintext X. In the former case the ciphertext can be decrypted with Bob's private key only. Since Bob keeps his private key in secret he is the person who can decrypt ciphertext and read the original message. On receipt of the ciphertext Y Bob decrypts it and recovers the plaintext $X = D_{KR_B}(Y)$. In the latter case authenticated message can be "discovered" by Alice's public key KU_A. Another important observation is that it is hard if not impossible to alter the message without access to Alice's private key, so the message is authenticated both in terms of source and data integrity. Since Bob has access to Alice's public key KU_A, he can easily detect her digital signature on the message by implementing $X = D_{KU_A}(E_{KR_A}(X))$. This is the way how message authentication and its detection are accomplished in public-key cryptography.

Alice however can provide both the authentication and confidentiality by using following schema:

Encryption: $Z = E_{KU_B}(E_{KR_A}(X))$,

Decryption: $X = D_{KU_A}(D_{KR_B}(Z))$.

The above two formulae are read as follows. Alice digitally signs message X with own private key, after this she encrypts the message with Bob's public key. Ciphertext is then sent to Bob. The ciphertext can be decrypted with Bob's private key only and thereby he is the only person who can decrypt the ciphertext. Thus, Bob decrypts the ciphertext and then using Alice's public key finds Alice's digital signature on the message.

2.4 Substitution vs Transposition Ciphers

In academic literature researchers distinguish between two types of classical cryptographic methods. All classical encryption methodsare based on either substitution of the elements in the plaintext by another elements or rearrangements of the elements in accordance to predefined templates. The former methodis called substitution cipher while latter method named transposition method. Some cryptosystems are hybrid involving multiple stages of substitutions and transpositions.

It is believed that one of the earliest examples of substitution cipher is due to Julius Ceasar. When encrypting with Ceasar's cipher each letter of the plaintext is substituted with the letter standing three places further in the alphabet. When decrypting with Ceasar's cipher each cipher letter is replaced according to the same principle but in reverse order. The substitution is performed in wrapped-around manner, with the first letter of the alphabet following the last letter when encrypting and vice versa when decrypting. For instance, Ceasar's cipher replaces a plaintext fragment "meetmeafterthetogaparty" with ciphertext "phhwphdiwhuwkhwrjdsduwb". It must be noticed that number of the shift positions is a key to the cipher.

Ceasar's cipher can be easily extended if we substitute a letter with the letter standing any k places further. So, that the general Caesar algorithm is expressed as $C = E(p) = (p + k)\text{mod } (26)$ where k is in the range 1 to 25. Likewise, decryption is simply expressed as $p = D(C) = (C - k)\text{mod } (26)$. Cryptanalyst needs to try all 25 keys to break the general Ceasar's cipher.

2.5 Monoalphabetic vs Polyalphabetic Ciphers

A monoalphabetic substitution cipher is based on use of a single alphabet. It must be noticed that alphabet is an ordered sequence of letters. Any rearrangement of the letters in known alphabets results in a new alphabet which is different than the original alphabet from point of view considered in cryptography. Consider English alphabet. Each of 26! distinct rearrangements of English letters leads to a new coding scheme. A monoalphabetic cipher uses single replacement scheme for all letters of the plaintext. This is the reason why it is relatively easy to break monoalphabetic cipher. In fact simple linear cryptanalysis is good enough to break monoalphabetic cipher.

Polyalphabetic ciphers are specifically developed to overcome the above mentioned weakness provided by monoalphabetic ciphers. In polyalphabetic cipher multiple alphabets are alternatively used for substituting the plain letters by successive cipher letters. It is quite cumbersome task to collect statistical information on relative frequencies of the cipher letters since they apparently belong to different alphabets and cryptanalyst does not know which letter belongs to which alphabet. It is essential that number of alphabets used to create ciphertext could be easily determined if cryptanalyst knew the length of the keyword. What is interesting the key length but not the keyword itself is of primary importance. Once cryptanalyst knows the key length he/she can easily reduce whole task of breaking the polyalphabetic cipher into n monoalphabetic tasks and reserve then reperately. There exist several tricks for discovering the key length. Let us assume that cryptanalyst somehow deters the length of the keyword. Next he/she collects each *n* th letter, and separates the ciphertext into *n* classes with each class determined by single letter of the keyword.

After application the brute force analysis to groups of letters specified by a keyword letter the cryptanalyst recovers the plaintext.

Vigenere algorithm is a well-known and the simplest among polyalphabetic ciphers. Vigenere algorithm that is based on English alphabet uses 26 rules or distinct alphabets, each obtained from the previous by shifting the letters of the alphabet by one position to the right in a circular manner such that A follows Z. Each alphabet is pointed out by a key letter. A key letter is indeed a cipher letter that substitutes one of the plaintext letters. Given plainletter y and the key letter x, Vigenere algorithm substitutes y by the cipherletter that is at the intersection of the row pointed by x and the column pointed by y.

Here is example illustrating encryption process by Vigenere cipher. Let "encrypt" be the key word and "cryptography" be the plaintext. Related table is shown in Table 2.1. The associated ciphertext created by Vigenere algorithm is "pvagrhvvnryw".

Table 2.1: Vigenere table created by key word "encrypt".

		PLAINTEXT LETTERS																									
		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**
K	**E**	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	**N**	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
Y	**C**	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
W	**R**	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
O	**Y**	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
R	**P**	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
D	**T**	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

2.6 Stream vs Block Ciphers

A stream cipher that symbol-wise processes the text. Block ciphers encrypts fixed-length block of bits using an unvarying transformation. Stream ciphers encrypt streams of bits with varying length and use varying transformation on each bit.

Caesar cipher is perhaps the most famous cipher mentioned in academic literature. It is believed that Julius Ceasar sent scrambled messages to his In a substitution cipher, each character of the plain text (plain text is the message which has to be encrypted) is substituted by another character to form the cipher text (cipher text is the encrypted message). The variant used by Caesar was a shift by 3 cipher. Each character was shifted by 3 places, so the character ‘A’ was replaced by ‘D’, ‘B’ was replaced by ‘E’ and so on. The characters would wrap around at the end, so ‘X’ would be replaced by ‘A’.

Chapter 3

HILL CIPHER AND ITS MODIFICATIONS

Hill cipher was developed by its inventor Lester Hill in 1929 [4]. Hill cipher is known to be the first polygraphic cipher. The method is based on linear matrix transformation of a message space. Given a plaintext message $p = (p_1, p_2, \dots)$ where p_i is a letter in some alphabet and invertible $m \times m$ matrix H, Hill cipher represents p_i by numeric value $x_i \in Z_n (Z_n = \{0,1,\dots,n-1\})$ and encrypts plaintext as $y = H \cdot x \pmod{n}$, where x and y are plaintext and ciphertext column vectors. Similarly, y is decrypted as $x = H^{-1} \cdot y \pmod{n}$, where H^{-1} is the inverse of H. That is, $H \cdot H^{-1} = H^{-1} \cdot H = I$ holds, where I is the identity matrix.

The following exemplifies Hill cipher for $n = 26$, $H = \begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}$ and $H^{-1} = \begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix}$. The plaintext $x = (ACT) = (0\ 2\ 19)$ is encrypted as

$y = H \cdot x \pmod{26} = (15\ 14\ 7) = (POH)$. Likewise, the ciphertext $y = (15\ 14\ 7)$ is decrypted as $x = H^{-1} \cdot y \pmod{26} = (0\ 2\ 19) = (ACT)$.

Hill cipher is vulnerable to cryptanalysis. The cryptanalyst usually sits in tight loop and tries to possess the plaintext of some messages and the corresponding ciphertext of those messages to deduce the key. If cryptanalyst succeeds with gaining access to the flow of messages then he can easily decrypt any new messages encrypted with the same key. The system can be obviously broken, knowing only m distinct plaintext and ciphertext pairs (x, y) and by computing $H = Y \cdot X^{-1} (\operatorname{mod} n)$, where X and Y are the matrices composed of m columns of x and y, respectively. Whenever X is invertible the opponent can obviously compute the unknown key as $H = Y \cdot X^{-1} (\operatorname{mod} n)$ and consequently break the cipher. If the X is not invertible then cryptanalyst keeps on collecting m plaintext and ciphertext pairs until the resulting matrix is invertible. When m is unknown, cryptanalyst might try the procedure for $m = 2,3,4$ until the key is found.

The affine Hill cipher was proposed to overcome this drawback [5]. The affine Hill cipher is a secure variant of Hill cipher in which the concept of Hill cipher is extended by mixing it up with an affine transformation. Similar to the Hill cipher the affine Hill cipher is polygraphic cipher, encrypting/decrypting m letters at a time. Given key matrix H and vector V, in affine Hill cipher the encryption expression is represented by $y = H \cdot x + V\ (\operatorname{mod} n)$. Similarly, the decryption is performed by $x = H^{-1} \cdot (y - V) (\operatorname{mod} n)$. The following example illustrates the way encryption and decryption is performed in affine Hill cipher. The following example exemplifies affine Hill cipher. Let $n = 26$, $H = \begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}$, $H^{-1} = \begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix}$ and $V = \begin{pmatrix} 5 \\ 0 \\ 7 \end{pmatrix}$. The encryption $x = (ACT) = (0\ 2\ 19)$ is possessed by

$$y = H \cdot x + V\ (\operatorname{mod} 26) = \begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix} + \begin{pmatrix} 5 \\ 0 \\ 7 \end{pmatrix} (\operatorname{mod} 26) = \begin{pmatrix} 20 \\ 14 \\ 14 \end{pmatrix}.$$

Likewise, the decryption is performed as follows:

$$x = H^{-1} \cdot (x - V) \pmod{26} = \begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix} \cdot \begin{pmatrix} 15 \\ 14 \\ 7 \end{pmatrix} \pmod{26} = \begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix}.$$

Suppose Alice chooses affine Hill cipher to send confidential message to Bob. Firstly, she selects a pair (H, V) to encrypt the plaintext. Then she sends ciphertext as well as (H, V) to Bob. When Bob receives ciphertext and a pair (H, V) he creates H^{-1} and then decrypts the ciphertext with (H^{-1}, V).

In 2000, Saeednia proposed an interesting modification of Hill cipher [6]. The main idea behind of his algorithm is to modify the key matrix each time Hill cipher is implemented. Encrypting a message by a one-time used matrix would make the algorithm more secure compared to the original Hill cipher and affine Hill cipher.

Assume Alice decides to send confidential message of size $m \times s$ to Bob, and she chooses Saeednia's algorithm to encrypt the message. Then she firstly selects random permutation π of size m and creates $m \times m$ permutation matrix P_π by permuting the rows of identity matrix of the same size. Such a matrix is always row equivalent to an identity matrix. Then she creates its inverse P_π^{-1} by permuting the columns of the identity matrix. Likewise, P_π^{-1} is column equivalent to the row-permuted matrix. After that, she creates one-time used matrix H_π from the key matrix H as $H_\pi = P_\pi^{-1} \cdot H \cdot P_\pi$. She further encrypts x as $y = H_\pi \cdot x \pmod{n}$ and sends a pair $(y, \pi^{'})$ to Bob where $\pi^{'} = H \cdot \pi \pmod{n}$. Upon receipt of the message, Bob computes permutation π from $\pi^{'}$ and H^{-1} as follows $\pi = H^{-1} \cdot \pi^{'} \pmod{n}$. Bob next calculates $(H^{-1})_\pi = P_\pi^{-1} \cdot H^{-1} \cdot P_\pi$ and decrypts the ciphertext as $x = (H_\pi)^{-1} \cdot y \pmod{n}$ keeping in mind that $(H_\pi)^{-1} = (H^{-1})_\pi$.

It should be noticed that the permutation of any pair of rows (or columns) of matrix H yields a matrix whose inverse is obtained by the permutation of the same columns (or rows) of H^{-1}. This is the reason why Bob does not need to use transposition algorithm to find $(H_\pi)^{-1}$; it can be easily obtained from equality $(H_\pi)^{-1} = (H^{-1})_\pi$. This observation essentially decreases computational cost of Saeednia's algorithm.

Below we exemplify encryption and decryption with Saeednia's algorithm for $\pi = \begin{pmatrix}123\\213\end{pmatrix}$, $H = \begin{pmatrix}6 & 24 & 1\\13 & 16 & 10\\20 & 17 & 15\end{pmatrix}$ and $x = \begin{pmatrix}0\\2\\19\end{pmatrix}$. By using permutation matrix $P_\pi = \begin{pmatrix}0 & 1 & 0\\1 & 0 & 0\\0 & 0 & 1\end{pmatrix}$ and its inverse $P_\pi^{-1} = \begin{pmatrix}0 & 1 & 0\\1 & 0 & 0\\0 & 0 & 1\end{pmatrix}$ we obtain

$$H_\pi = P_\pi^{-1} \cdot H \cdot P_\pi = \begin{pmatrix}0 & 1 & 0\\1 & 0 & 0\\0 & 0 & 1\end{pmatrix} \cdot \begin{pmatrix}6 & 24 & 1\\13 & 16 & 10\\20 & 17 & 15\end{pmatrix} \cdot \begin{pmatrix}0 & 1 & 0\\1 & 0 & 0\\0 & 0 & 1\end{pmatrix} =$$

$$\begin{pmatrix}16 & 13 & 10\\24 & 6 & 1\\17 & 20 & 15\end{pmatrix}.$$

After that we encrypt the plaintext as follows

$$y = H_\pi \cdot x \pmod{n} = \begin{pmatrix}16 & 13 & 10\\24 & 6 & 1\\17 & 20 & 15\end{pmatrix} \cdot \begin{pmatrix}0\\2\\19\end{pmatrix} \pmod{26} = \begin{pmatrix}8\\5\\13\end{pmatrix}.$$

Decryption is carried out as follows

$$(H_\pi)^{-1} = (H^{-1})_\pi = P_\pi^{-1} \cdot H^{-1} \cdot P_\pi = \begin{pmatrix}0 & 1 & 0\\1 & 0 & 0\\0 & 0 & 1\end{pmatrix} \cdot \begin{pmatrix}8 & 5 & 10\\21 & 8 & 21\\21 & 12 & 8\end{pmatrix} \cdot \begin{pmatrix}0 & 1 & 0\\1 & 0 & 0\\0 & 0 & 1\end{pmatrix}.$$

It was reported in [3] that Saeednia's algorithm has the same problem as the original Hill cipher. By collecting m pairs of (π, π') and simultaneously solving m equations

$\pi = H \cdot \pi'$, a cryptanalyst can obtain the key matrix H. Further, he can use P_π and P_π^{-1} to calculate H_π. In the same paper it was noticed that Saeednia's algorithm is time consuming since it is based on frequent use of matrix operations. Matrix operations require a lot of time to compute when the matrix size is large enough.

Chapter 4

COMPUTER EXPERIMENTS

To analyze the time-effectiveness of Hill algorithm and its modifications we performed a series of computer experiments on PC/Windows 7 platform using an interactive environment and high-level computing tool MATLAB. The relative time-effectiveness of above three methods is assessed in terms of run times spent for encryption and decryption. The steps carried out in computer experiments are outlined below.

The main steps of Hill cipher are indicated below:

Encryption

1. Select invertible matrix $\boldsymbol{H}$.
2. Calculate $\boldsymbol{y} = \boldsymbol{H} \cdot \boldsymbol{x}$ (mod $\boldsymbol{n}$).

   Alice sends $(\boldsymbol{y}, \boldsymbol{H})$ to Bob.

Decryption

1. Calculate $\boldsymbol{H}^{-\mathbf{1}}$.
2. Calculate $\boldsymbol{x} = \boldsymbol{H}^{-\mathbf{1}} \cdot \boldsymbol{y}$ (mod $\boldsymbol{n}$).

The affine Hill cipher is represented by the following steps:

Encryption

1. Select invertible matrix $\boldsymbol{H}$.

2. Select vector $\boldsymbol{V}$.
3. Calculate $\boldsymbol{y} = \boldsymbol{H} \cdot \boldsymbol{x} + \boldsymbol{V} \pmod{\boldsymbol{n}}$.

   Alice sends $\boldsymbol{H}, \boldsymbol{V}, \boldsymbol{y}$ to Bob.

<u>Decryption</u>

1. Calculate $\boldsymbol{H}^{-1}$.
2. Calculate $\boldsymbol{x} = \boldsymbol{H}^{-1}(\boldsymbol{y} - \boldsymbol{V})$.

The main steps of Saeednia's method are as follows:

<u>Encryption</u>

1. Select random permutation $\boldsymbol{\pi}$.
2. Select matrix $\boldsymbol{H}$.
3. Calculate permutation matrix $\boldsymbol{P}_{\boldsymbol{\pi}}$ by permuting the rows of identity matrix $\boldsymbol{I}$.
4. Calculate permutation matrix $\boldsymbol{P}_{\boldsymbol{\pi}}^{-1}$ by permuting the columns of identity matrix $\boldsymbol{I}$.
5. Calculate $\boldsymbol{H}_{\boldsymbol{\pi}} = \boldsymbol{P}_{\boldsymbol{\pi}}^{-1} \cdot \boldsymbol{H} \cdot \boldsymbol{P}_{\boldsymbol{\pi}}$.
6. Calculate $\boldsymbol{y} = \boldsymbol{H}_{\boldsymbol{\pi}} \cdot \boldsymbol{x} \pmod{\boldsymbol{n}}$
7. Calculate $\boldsymbol{\pi}' = \boldsymbol{H} \cdot \boldsymbol{\pi} \pmod{\boldsymbol{n}}$

Alice sends $\boldsymbol{\pi}', \boldsymbol{H}, \boldsymbol{y}$ to Bob.

<u>Decryption</u>

1. Calculate $\boldsymbol{H}^{-1}$.
2. Calculate $\boldsymbol{\pi} = \boldsymbol{H}^{-1} \cdot \boldsymbol{\pi}' \pmod{\boldsymbol{n}}$.
3. Calculate permutation matrix $\boldsymbol{P}_{\boldsymbol{\pi}}$ by permuting the rows of identity matrix $\boldsymbol{I}$.
4. Calculate permutation matrix $\boldsymbol{P}_{\boldsymbol{\pi}}^{-1}$ by permuting the columns of identity matrix $\boldsymbol{I}$.
5. Calculate $(\boldsymbol{H}^{-1})_{\boldsymbol{\pi}} = \boldsymbol{P}_{\boldsymbol{\pi}}^{-1} \cdot \boldsymbol{H} \cdot \boldsymbol{P}_{\boldsymbol{\pi}}$.

6. Calculate $\boldsymbol{x} = (\boldsymbol{H_\pi})^{-\mathbf{1}} \cdot \boldsymbol{y} \pmod{\boldsymbol{n}}$.

We measured run times separately for encryption and decryption procedures. For each matrix size we performed a series of computer experiments. We collected the results of computer experiments and calculated average run time for each matrix size. The results of computer experiments for encryption and decryption are illustrated in Table 4.1 and Table 4.2 respectively. The encryption and decryption run times for Hill cipher, affine Hill cipher and Saeednia's modification are respectively represented in Figures 4.1 - 4.4 using blue, brown and green shapes. In these figures Series 1, Series 2 and Series 3 respectively stand for the results for Hill cipher, affine Hill cipher and Saeednia's algorithm.

Pairwise comparison of time-effectiveness of the three algorithms are shown for encryption in Table 4.3 and for decryption in Table 4.4. Simulation results in Tables 4.1 - 4.4 show that encryption and decryption with Saeednia's algorithm requires essentially more time compared to affine Hill cipher and Hill cipher. This is expected result because Saeednia's algorithm involves a lot of matrix operations which increases its run time. Bar charts and line charts drawn for simulation results clearly demonstrate increasing tendency for all three algorithms: Rate of increase in Saeednia's modification (green shapes) is much more than that in former two algorithms (blue and brown shapes).

On the other hand, Tables 4.1 – 4.4 reveal that the difference between Hill cipher (blue shapes) and affine Hill cipher (brown shapes) is hard to see by the naked eye- the maximum and average difference between run times of affine Hill cipher and Hill cipher is 0.0629 and 0.0362 sec for encryption and 0.1058 and 0.0591 for decryption, which is negligibly small.

Although Saeednia's modification is strongest in sense of confidentiality among three algorithms considered in the present thesis, it is the worst in terms of time-efficiency. On the other hand, Hill cipher and affine Hill cipher take more or less same time for encryption and decryption. However, affine Hill cipher is a bit more secure than Hill cipher.

Table 4.1: Results of computer experiments: Run times for encryption by Hill cipher, affine Hill cipher and Saeednia's algorithms.

Algorithm	Matrix size, *m*									
	10	20	30	40	50	60	70	80	90	100
Hill cipher	0.1248	0.2652	0.2841	0.3193	0.3276	0.3337	0.3349	0.3451	0.3512	0.3588
Affine Hill cipher	0.1327	0.2741	0.3039	0.3544	0.3756	0.3756	0.3948	0.4047	0.4123	0.4217
Saeednia's modification	0.1560	0.2761	0.3423	0.3947	0.4507	0.5311	0.6122	0.7013	0.7707	0.8576

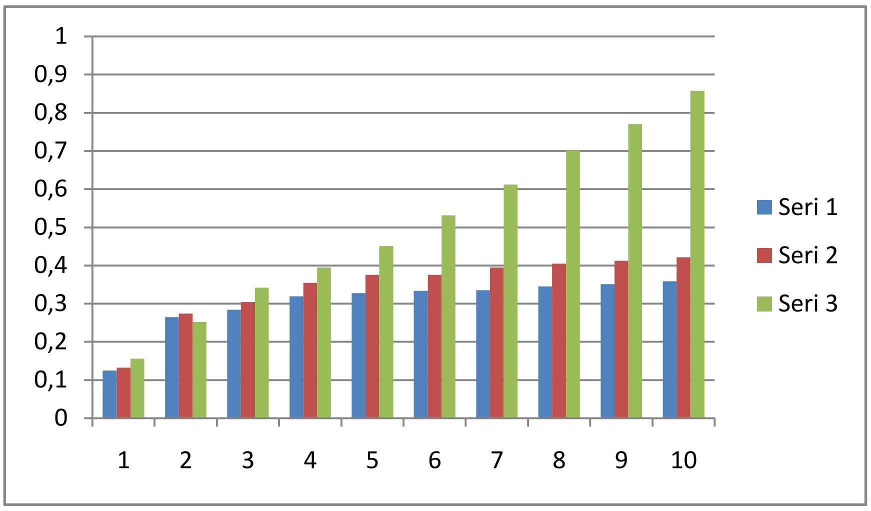

Figure 4.1: Bar chart representation of run times for encryption by Hill cipher, affine Hill cipher and Saeednia's algorithms.

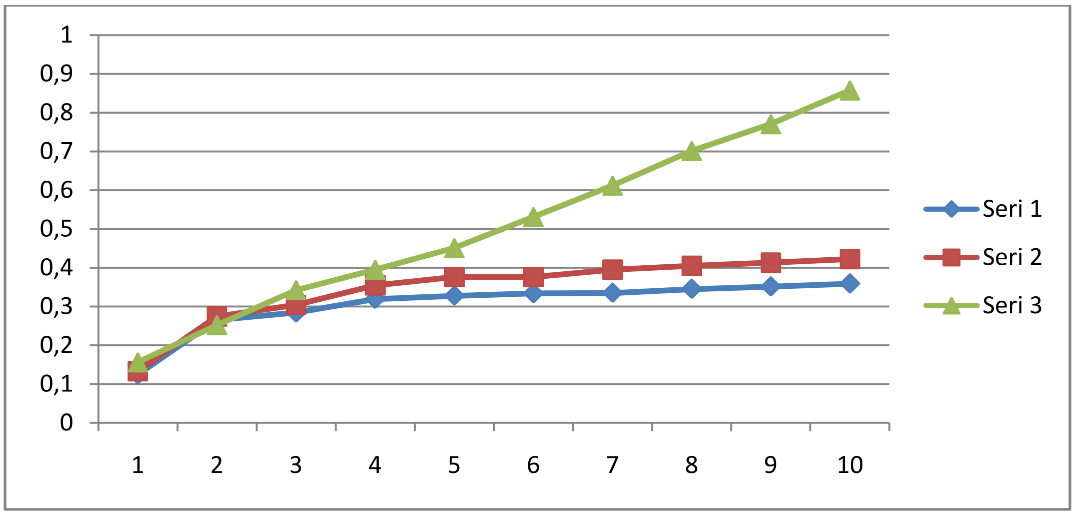

Figure 4.2: Line chart representation of run times for encryption by Hill cipher, affine Hill cipher and Saeednia's algorithms.

Table 4.2: Results of computer experiments: Run times for decryption by Hill cipher, affine Hill cipher and Saeednia's algorithms.

Algorithm	Matrix size, m									
	10	20	30	40	50	60	70	80	90	100
Hill cipher	0.1872	0.3744	0.3803	0.3847	0.3888	0.3998	0.4021	0.4031	0.4042	0.4056
Affine Hill cipher	0.2184	0.4202	0.4391	0.4502	0.4603	0.4693	0.4724	0.4853	0.5001	0.5114
Saeednia's modification	0.2571	0.5001	0.5501	0.6002	0.7096	0.8075	0.9292	1.0196	1.1596	1.3572

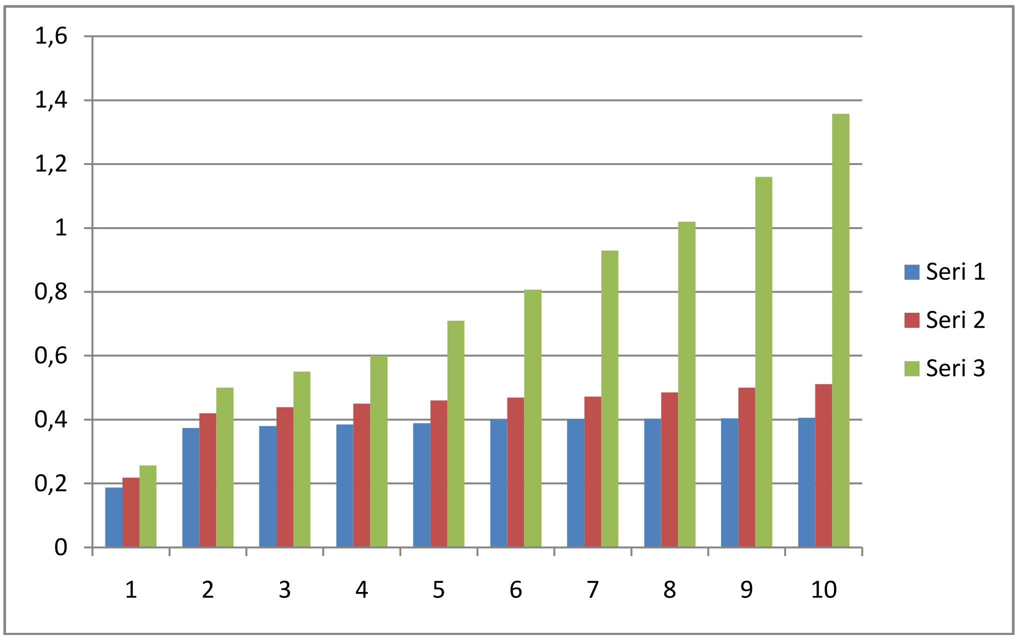

Figure 4.3: Bar chart representation of run times for decryption by Hill cipher, affine Hill cipher and Saeednia's algorithms.

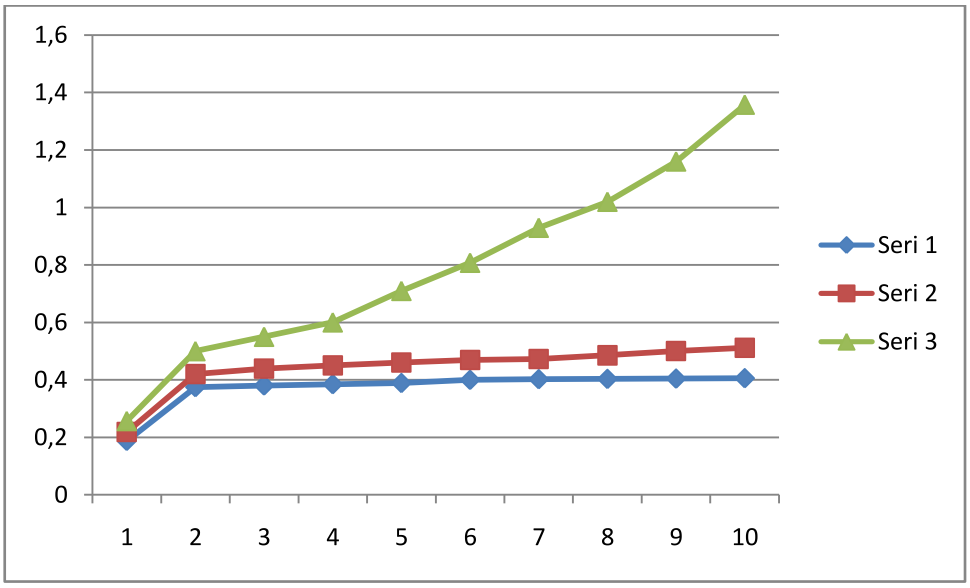

Figure 4.4: Line chart representation of run times for decryption.

Table 4.3: Pairwise comparison of algorithms regarding encryption run times.

Pairwise comparison of algorithms	Difference between run times	
	Maximum	Average
Saeednia's vs Hill	0.4988	0.2048
Saeednia's vs affine Hill	0.4359	0.1619
Affine Hill vs Hill	0.0629	0.0362

Table 4.4: Pairwise comparison of algorithms regarding decryption run times.

Pairwise comparison of algorithms	Difference between run times	
	Maximum	Average
Saeednia’s vs Hill	0.9516	0.3208
Saeednia’s vs affine Hill	0.8458	0.2619
Affine Hill vs Hill	0.1058	0.0591

Chapter 5

CONCLUSION

It is the aim of the present thesis to provide a comparative analysis of cryptographic algorithms in terms of time-effectiveness. The analysis is done for Hill cipher, affine Hill cipher and Saeednia's algorithm.

The main outcomes of this thesis are summarized as follows:

1. Encryption and decryption with Saeednia's algorithm requires essentially more time compared to affine Hill cipher and Hill cipher.
2. Encryption and decryption run times for affine Hill cipher and Hill cipher are negligibly small, not showing essential difference.
3. Since Saeednia's modification provides more confidentiality compared to remaining two algorithms but takes more encryption/decryption time, and that affine Hill cipher a bit more secure than Hill cipher, there should be a reasonable trade-off in selection of cryptographic algorithm. The best choice depends on level of confidentiality of a particular application and traffic between sender and receiver.

REFERENCES

[1] I.A. Ismail, M. Amin and H. Diab, How to repair the Hill cipher, Journal of Zhejiang University-Science A 7(12) (2006) 2022-2030.

[2] A.I. Al-Kadi, The origins of cryptology: The Arab contributions, Cryptologia 16(2), 1992, 97–126.

[3] C-H. Lin, C-Y Lee, C-Y. Lee, Comments on Saeednia’s improved scheme for the Hill cipher, Journal of the Chinese Institute of Engineers 27(5) (2004) 743-746.

[4] W. Stallings, Cryptography and Network Security: Principles and Practice, 6th edition, Upper Saddle River, NJ: Prentice, 2013.

[5] D.R. Stinson, Cryptography Theory and Practice, 3rd edition, Chapman & Hall/CRC, 2006.

[6] S. Saeednia, How to Make the Hill Cipher Secure, Cryptologia Journal 24(4) (2000) 353-360.

[7] Y.S. Yeh, T.C. Wu, C.C. Chang, and W.C. Yang, A New Cryptosystem Using Matrix Transformation, Proceedings of the 25th IEEE International Carnahan Conference on Security Technology, pp.131-138, Oct. 1991.

[8] D. Zhang, and G. Chen, Cryptanalysis of an image encryption scheme based on the Hill cipher, Journal of Zhejiang University - Science A 9(8) (2008) 1118-1123.

APPENDICES

Appendix A: Hill Cipher Encryption

HILL CIPHER ENCRYPTION (10x10)

```
>> t=cputime
a=0+100*rand(10,10)
b=round(a)
x=0+25*rand(10,1)
k=round(x)
g=b*k
l=mod(g,26)
zaman=cputime-t
t  = 0.1248
```

****

HILL CIPHER ENCRYPTION (20x20)

```
>> t=cputime
a=0+100*rand(20,20)
b=round(a)
x=0+25*rand(20,1)
k=round(x)
g=b*k
l=mod(g,26)
zaman=cputime-t
t  = 0.2652
```

****

HILL CIPHER ENCRYPTION (50x50)

```
>> t=cputime
a=0+100*rand(50,50)
b=round(a)
x=0+25*rand(50,1)
k=round(x)
g=b*k
c=inv(b)
y=0+25*rand(50,1)
j=c*y
l=round(j)
p=mod(l,26)
zaman=cputime-t
t = 0.3276
```

****

HILL CIPHER ENCRYPTION (100x100)

```
>> t=cputime
a=0+100*rand(100,100)
```

```
b=round(a)
x=0+25*rand(100,1)
k=round(x)
g=b*k
l=mod(g,26)
zaman=cputime-t
t = 0.3588
```

Appendix B: Hill Cipher Decryption

HILL CIPHER DECRYPTION (10x10)

```
>> t=cputime
a=0+100*rand(10,10)
b=round(a)
x=0+25*rand(10,1)
k=round(x)
g=b*k
c=inv(b)
y=0+25*rand(10,1)
j=c*y
l=round(j)
p=mod(l,26)
zaman=cputime-t
t = 0.1872
```

****

HILL CIPHER DECRYPTION (20x20)

```
>> t=cputime
a=0+100*rand(20,20)
b=round(a)
x=0+25*rand(20,1)
k=round(x)
g=b*k
c=inv(b)
y=0+25*rand(20,1)
j=c*y
l=round(j)
p=mod(l,26)
zaman=cputime-t
t = 0.3744
```

****

HILL CIPHER DECRYPTION (50x50)

```
>> t=cputime
a=0+100*rand(50,50)
b=round(a)
x=0+25*rand(50,1)
k=round(x)
g=b*k
c=inv(b)
y=0+25*rand(50,1)
j=c*y
l=round(j)
p=mod(l,26)
```

```
zaman=cputime-t
t = 0.3888
```

****

HILL CIPHER DECRYPTION (100x100)

```
>> t=cputime
a=0+100*rand(100,100)
b=round(a)
x=0+25*rand(100,1)
k=round(x)
g=b*k
c=inv(b)
y=0+25*rand(100,1)
j=c*y
l=round(j)
p=mod(l,26)
zaman=cputime-t
t = 0.4056
```

Appendix C: Affine Hill Cipher Encryption

AFFINE HILL CIPHER ENCRYPTION (10x10)

```
t=cputime
a=0+100*rand(10,10)
b=round(a)
x=0+25*rand(10,1)
k=round(x)
l=b*k
m=0+25*rand(10,1)
n=round(m)
p=l+n
r=mod(p,26)
zaman=cputime-t
t = 0.1327
```

****

AFFINE HILL CIPHER ENCRYPTION (20x20)

```
t=cputime
a=0+100*rand(20,20)
b=round(a)
x=0+25*rand(20,1)
k=round(x)
l=b*k
m=0+25*rand(20,1)
n=round(m)
p=l+n
r=mod(p,26)
zaman=cputime-t
t = 0.2741
```

****

AFFINE HILL CIPHER ENCRYPTION (50x50)

```
>> t=cputime
a=0+100*rand(50,50)
b=round(a)
x=0+25*rand(50,1)
k=round(x)
l=b*k
m=0+25*rand(50,1)
n=round(m)
p=l+n
r=mod(p,26)
zaman=cputime-t
t = 0.3756
```

****

AFFINE HILL CIPHER ENCRYPTION (100x100)

```
>> t=cputime
a=0+100*rand(100,100)
b=round(a)
x=0+25*rand(100,1)
k=round(x)
l=b*k
m=0+25*rand(100,1)
n=round(m)
p=l+n
r=mod(p,26)
zaman=cputime-t
t = 0.4217
```

Appendix D: Affine Hill Cipher Decryption

```
AFFINE HILL CIPHER DECRYPTION (10x10)
t=cputime
y=0+25*rand(10,1)
k=round(y)
m=0+25*rand(10,1)
n=round(m)
p=k-n
a=0+100*rand(10,10)
b=round(a)
c=inv(b)
l=c*p
r=mod(l,26)
zaman=cputime-t
t = 0.2184
```

****

```
AFFINE HILL CIPHER DECRYPTION (20x20)
t=cputime
y=0+25*rand(20,1)
k=round(y)
m=0+25*rand(20,1)
n=round(m)
p=k-n
a=0+100*rand(20,20)
b=round(a)
c=inv(b)
l=c*p
r=mod(l,26)
zaman=cputime-t
t = 0.4202
```

****

```
AFFINE HILL CIPHER DECRYPTION (50x50)
t=cputime
y=0+25*rand(50,1)
k=round(y)
m=0+25*rand(50,1)
n=round(m)
p=k-n
a=0+100*rand(50,50)
b=round(a)
c=inv(b)
l=c*p
r=mod(l,26)
```

```
zaman=cputime-t
t = 0.4603
```

****

AFFINE HILL CIPHER DECRYPTION (100x100)

```
t=cputime
y=0+25*rand(100,1)
k=round(y)
m=0+25*rand(100,1)
n=round(m)
p=k-n
a=0+100*rand(100,100)
b=round(a)
c=inv(b)
l=c*p
r=mod(l,26)
zaman=cputime-t
t = 0.5114
```

Appendix E: S. Saeednia's Algorithm Encryption

S. SAEEDNIA'S ALGORITHM ENCRYPTION (10x10)

```
t=cputime
a=eye(10)
a(10,10)=0
a(1,1)=0
a(1,10)=1
a(10,1)=1
a(8,8)=0
a(2,2)=0
a(8,2)=1
a(2,8)=1
a(6,6)=0
a(3,3)=0
a(6,3)=1
a(3,6)=1
a(5,5)=0
a(4,4)=0
a(5,4)=1
a(4,5)=1
b=inv(a)
x=0+25*rand(10,10)
k=round(x)
l=a*k*b
p=0+25*rand(10,1)
r=round(p)
v=l*r
y=mod(v,26)
zaman=cputime-t
t = 0.1560
```

****

S. SAEEDNIA'S ALGORITHM ENCRYPTION (20x20)

```
>> t=cputime
a=eye(20)
a(20,20)=0
a(1,1)=0
a(1,20)=1
a(20,1)=1
a(12,12)=0
a(4,4)=0
a(4,12)=1
a(12,4)=1
a(6,6)=0
a(3,3)=0
a(6,3)=1
```

```
a(3,6)=1
a(9,9)=0
a(15,15)=0
a(15,9)=1
a(9,15)=1
b=inv(a)
x=0+25*rand(20,20)
k=round(x)
l=a*k*b
p=0+25*rand(20,1)
r=round(p)
v=l*r
y=mod(v,26)
zaman=cputime-t
t = 0.2521
```

****

S. SAEEDNIA’S ALGORITHM ENCRYPTION (50x50)

```
>> t=cputime
a=eye(50)
a(50,50)=0
a(1,1)=0
a(1,50)=1
a(50,1)=1
a(12,12)=0
a(28,28)=0
a(4,28)=1
a(28,4)=1
a(36,36)=0
a(30,30)=0
a(36,30)=1
a(30,36)=1
a(9,9)=0
a(45,45)=0
a(45,9)=1
a(9,45)=1
b=inv(a)
x=0+25*rand(50,50)
k=round(x)
l=a*k*b
p=0+25*rand(50,1)
r=round(p)
v=l*r;
y=mod(v,26)
zaman=cputime-t
t = 0.4507
```

****

S. SAEEDNIA’S ALGORITHM ENCRYPTION (100x100)

```
>> t=cputime
a=eye(100)
a(100,100)=0
a(1,1)=0
a(1,100)=1
a(100,1)=1
a(12,12)=0
a(28,28)=0
a(4,28)=1
a(28,4)=1
a(95,95)=0
a(30,30)=0
a(95,30)=1
a(30,95)=1
a(27,27)=0
a(45,45)=0
a(45,27)=1
a(27,45)=1
a(61,61)=0
a(63,63)=0
a(61,63)=1
a(63,61)=1
b=inv(a)
x=0+25*rand(100,100)
k=round(x)
l=a*k*b
p=0+25*rand(100,1)
r=round(p)
v=l*r;
y=mod(v,26)
zaman=cputime-t
t = 0.8576
```

Appendix F: S. Saeednia's Algorithm Decryption

S. SAEEDNIA'S ALGORITHM DECRYPTION (10x10)

```
>>  t=cputime
a=eye(10)
a(10,10)=0
a(1,1)=0
a(1,10)=1
a(10,1)=1
a(8,8)=0
a(2,2)=0
a(8,2)=1
a(2,8)=1
a(6,6)=0
a(3,3)=0
a(6,3)=1
a(3,6)=1
a(5,5)=0
a(4,4)=0
a(5,4)=1
a(4,5)=1
b=inv(a)
x=0+25*rand(10,10)
k=inv(x)
m=a*k*b
q=0+25*rand(10,1)
v=m*q
s=round(v)
y=mod(s,26)
zaman=cputime-t
t = 0.2571
```

****

S. SAEEDNIA'S ALGORITHM DECRYPTION (20x20)

```
>> t=cputime
a=eye(20)
a(18,18)=0
a(4,4)=0
a(4,18)=1
a(18,4)=1
a(8,8)=0
a(11,11)=0
a(8,11)=1
a(11,8)=1
a(6,6)=0
a(13,13)=0
a(6,13)=1
```

```
a(13,6)=1
a(5,5)=0
a(14,14)=0
a(5,14)=1
a(14,5)=1
b=inv(a)
x=0+25*rand(20,20)
k=inv(x)
m=a*k*b
q=0+25*rand(20,1)
v=m*q
s=round(v)
y=mod(s,26)
zaman=cputime-t
t = 0.5001
```

****

S. SAEEDNIA’S ALGORITHM DECRYPTION (50x50)

```
>> t=cputime
a=eye(50)
a(48,48)=0
a(12,12)=0
a(12,48)=1
a(48,12)=1
a(38,38)=0
a(17,17)=0
a(38,17)=1
a(17,38)=1
a(25,25)=0
a(37,37)=0
a(25,37)=1
a(37,25)=1
a(15,15)=0
a(24,24)=0
a(15,24)=1
a(24,15)=1
a(21,21)=0
a(16,16)=0
a(21,16)=1
a(16,21)=1
a(11,11)=0
a(23,23)=0
a(11,23)=1
a(23,11)=1
b=inv(a)
x=0+25*rand(50,50)
k=inv(x)
m=a*k*b
q=0+25*rand(50,1)
```

```
v=m*q
s=round(v)
y=mod(s,26)
zaman=cputime-t
t = 0.7096
```

****

S. SAEEDNIA’S ALGORITHM DECRYPTION (100x100)

```
>>  t=cputime
a=eye(100)
a(90,90)=0
a(2,2)=0
a(2,90)=1
a(90,2)=1
a(8,8)=0
a(17,17)=0
a(8,17)=1
a(17,8)=1
a(65,65)=0
a(37,37)=0
a(65,37)=1
a(37,65)=1
a(45,45)=0
a(24,24)=0
a(45,24)=1
a(24,45)=1
a(61,61)=0
a(16,16)=0
a(61,16)=1
a(16,61)=1
a(21,21)=0
a(63,63)=0
a(21,63)=1
a(63,21)=1
b=inv(a)
x=0+25*rand(100,100)
k=inv(x)
m=a*k*b
q=0+25*rand(100,1)
v=m*q
s=round(v)
y=mod(s,26)
zaman=cputime-t
t = 1.3572
```