CHAPTER ONE

INTRODUCTION TO CIPHER SYSTEMS

(1.1) What is cryptography?

Cryptography is the science of using mathematics to encrypt and decrypt data. I.e. cryptography is the study of secret (crypto-) writing (-graphy). Cryptography enables you to store sensitive information or transmit it across insecure channels or networks (like the Internet) so that it cannot be read by anyone except the intended recipient. While cryptography is the science of securing data, cryptanalysis is the science of analyzing and breaking secure communication. Classical cryptanalysis involves an interesting combination of analytical reasoning, application of mathematical tools, pattern finding, patience, determination, and luck. Cryptanalysts are also called attackers.

(1.2) How does cryptography work?

A cryptographic algorithm, or cipher, is a mathematical function used in the encryption and decryption process. A cryptographic algorithm works in combination with a key — a word, number, or phrase to encrypt the plaintext. The same plaintext encrypts to different ciphertext with different keys. The security of encrypted data is entirely dependent on two things: the strength of the cryptographic algorithm and the secrecy of the key. A cryptographic algorithm, plus all possible keys and all the protocols that make it work comprise a cryptosystem.

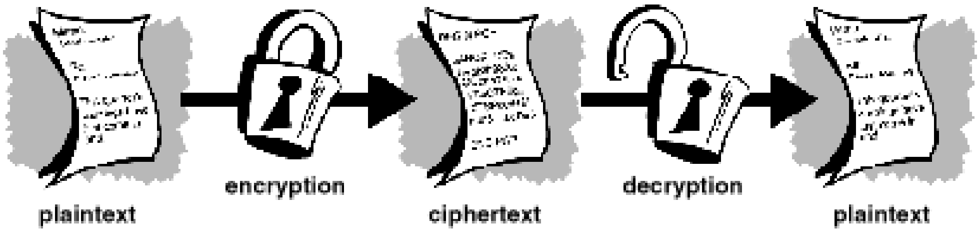


Figure 1: How does cryptography work?

(1.3) Basic Concepts

Encryption domains and codomains

- ***A*** denotes a finite set called the ***alphabet*** *of definition*. For example, A={0;1},the *binary alphabet*, is a frequently used alphabet of definition. Note that any alphabet can be encoded in terms of the binary alphabet. For example, since there are 32 binary strings of length five, each letter of the English alphabet can be assigned a unique binary string of length five.
- **M** denotes a set called the ***message space***. **M** consists of strings of symbols from an alphabet of definition. An element of **M** is called a ***plaintext*** *message* or simply a ***plaintext***. For example, **M** may consist of binary strings, English text, computer code, etc.

- ***C*** denotes a set called the ***ciphertext space***. ***C*** consists of strings of symbols from an alphabet of definition, which may differ from the alphabet of definition for **M**. An element of ***C*** is called a ***ciphertext***.

Encryption and decryption transformations

- **K** denotes a set called the ***key space***. An element of **K** is called a ***key***.
- Each element $e \in$ **K** uniquely determines a bijection from **M** to ***C***, denoted by E_e.

  E_e is called an ***encryption function*** or an ***encryption transformation***. Note that E_e must be a bijection if the process is to be reversed and a unique plaintext message recovered for each distinct ciphertext.

- For each $d \in$ **K**, D_d denotes a bijection from ***C*** to **M** (i.e., D_d: $C \rightarrow M$). D_d is called a ***decryption function*** or ***decryption transformation***.
- The process of applying the transformation E_e to a message $m \in$ ***M*** is usually referred to as ***encrypting*** m or the ***encryption*** of m.
- The process of applying the transformation D_d to a ciphertext c is usually referred to as ***decrypting*** c or the ***decryption*** of c.
- An *encryption scheme* consists of a set $\{E_e: e \in K\}$ of encryption transformations

  and a corresponding set $\{D_d: d \in$ **K**$\}$ of decryption transformations with the property that for each $e \in$ **K** there is a unique key $d \in$ **K** such that $D_d = E_e^{-1}$;that is, $D_d(E_e(m)) = m$ for all $m \in$ **M**. An encryption scheme is sometimes referred to as a ***cipher***.

- The keys e and d in the preceding definition are referred to as a ***key pair*** and some-times denoted by (e; d). Note that e and d could be the same.
- To *construct* an encryption scheme requires one to select a message space **M**, a ciphertext space ***C***, a key space **K**, a set of encryption transformations $\{E_e: e \in$ **K**$\}$, and a corresponding set of decryption transformations $\{D_d: d \in$ **K**$\}$.

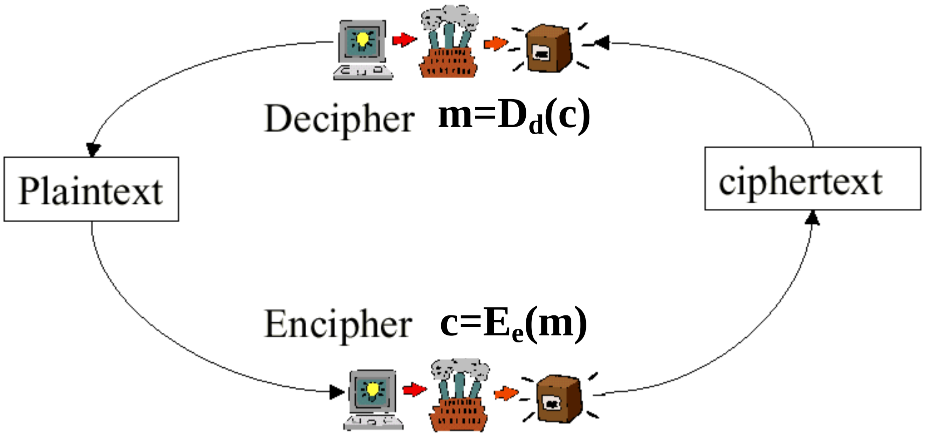


Figure 2: Secret writing

(1.4) Terminology

Cryptography: the art or science encompassing the principles and methods of transforming an intelligible message into one that is unintelligible, and then retransforming that message back to its original form.

Plaintext: the original intelligible message.

Ciphertext: the transformed message, i.e. unintelligible message.

Cipher: an algorithm for transforming an intelligible message into one that is unintelligible by transposition and/or substitution methods.

Key: some critical information used by the cipher, known only to the sender & receiver.

Encipher (Encode): the process of converting plaintext to ciphertext using a cipher and a key.

Decipher (Decode): the process of converting ciphertext back into plaintext using a cipher and a key.

Cryptanalysis: the study of principles and methods of transforming an unintelligible message back into an intelligible message *without* knowledge of the key. Also called codebreaking.

Cryptology: both cryptography and cryptanalysis.

Cryptanalyst: someone who engages in cryptanalysis.

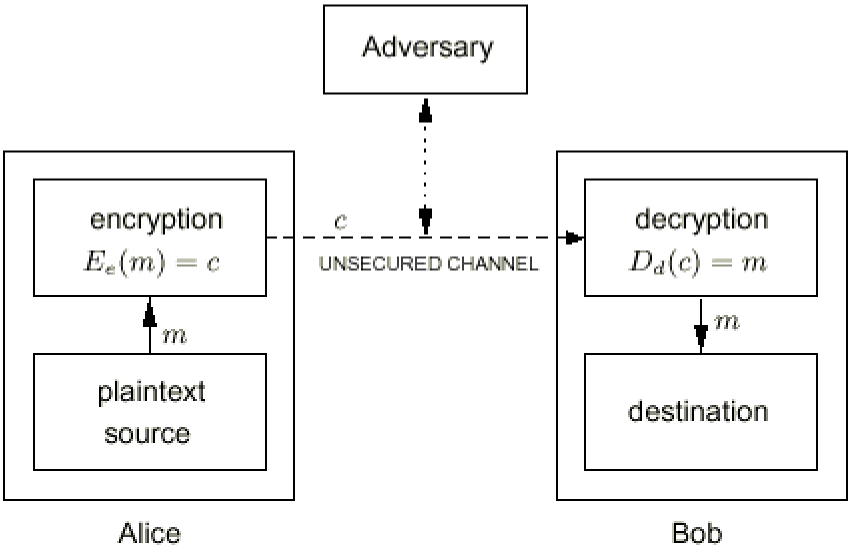


Figure 3: The block diagram to cipher system

(1.5) Classification of cipher systems

In general we can classify the cipher systems as the following:

I. Secret key systems.
   1. Conventional systems (classical).
      a. Transposition cipher.
         i Simple.
           - Message reversal cipher.
           - Columnar transposition.
         ii Double.
      b. Substitution cipher.
         i Monoalphabetic.
           - Simple.
             (a) Direct stander.
             (b) Standard reverse.
             (c) Multiplicative cipher.
             (d) Affine cipher.
             (e) Mixed alphabet.
             (f) Keyword mixed.
             (g) Transposed keyword mixed.
           - Homophonic.
             (a) Beale.
             (b) Higher order.
         ii Polyalphabetic.
           - Vigenere.
           - Beaufort.
         iii Polygraphic.
           - Playfair.
           - Hill cipher.
   2. Modern systems.
      a. Block cipher.
         - DES (Data Encryption Standards).
      b. Stream cipher.
         - LFSR (Linear Feedback Shift Register).

II. Public key systems.
   1. RSA.
   2. Knapsack.

(1.6) Secret key systems

In such type of systems the encipher key and the decipher key must be known **only** by the sender and the receiver, so they must exchange the key over a secure channel.

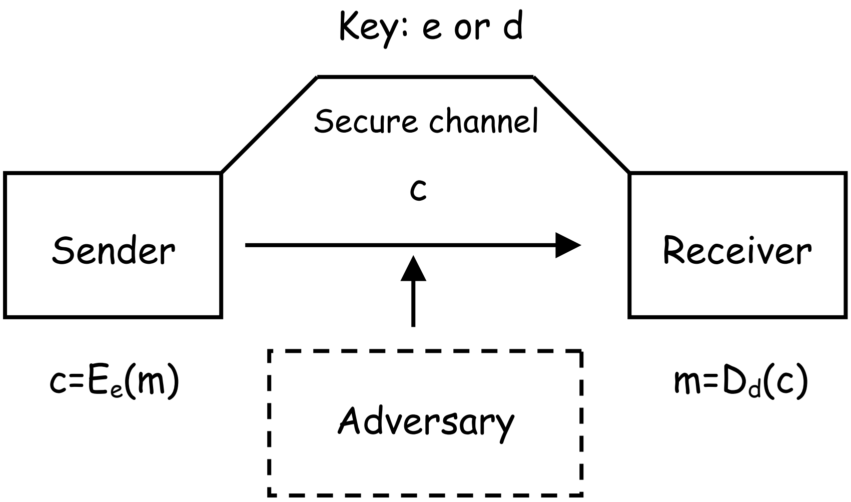


Figure 4: Secret key systems

The problems with secret key cryptography are:

i. Requires establishment of a secure channel for key exchange.
ii. Two parties cannot start communication if they never met.

(1.6.1) Conventional systems (classical)

Before there are computers, cryptography consisted of characters based algorithms. Different cryptography algorithms either substituted characters for another or transposed characters with another. The better algorithm did both.

The primary change is that algorithms work on bits instead of characters; this is actually just a change in the alphabet size from 26 elements (in English) to two elements only. Most good cryptographic algorithms still combine elements of substitution and transposition.

(1.6.1.1) Transposition cipher

In transposition ciphers the letters of the original message (plaintext) are arranged in a different order to get the ciphertext.

Plaintext → Rearrange characters → Ciphertext

i. Simple

➢ Message reversal cipher

In such procedure the plaintext will be written backward to produce the ciphertext. For example if the message is: ALMANSOUR UNIVERSITY COLLEGE, then

Plaintext = ALMANSOUR UNIVERSITY COLLEGE
Ciphertext = EGELLOC YTISREVINU RUOSNAMLA

Mathematically if L is the length of the message then c=E(k)=L+1-k, where k is the position of the letter in the plaintext.

➢ Columnar transposition

We arrange the message as array of 2-dimensition. The number of rows and columns depends on the length of the message, if the length of the message equal to 30 then the probability of the numbers of rows and columns are: 15X2, 2X15, 10X3, 3X10, 5X6, or 6X5. Note that if the length of the message is 29, we must add a dummy letter in the end of the message.

For example if the message is: ALMANSOUR UNIVERSITY COLLEGE, then the length of the message is 26, we will add a dummy letter X to the end of the message and the length will be 27. We can say that 27=9X3 and

1	2	3
A	L	M
A	N	S
O	U	R
U	N	I
V	E	R
S	I	T
Y	C	O
L	L	E
G	E	X

If the key is (2,3,1) then we arrange the columns as the following:

2	3	1
L	M	A
N	S	A
U	R	O
N	I	U
E	R	V
I	T	S
C	O	Y
L	E	L
E	X	G

The ciphertext comes from the reading on the above table by columns

LNUNEICLEMSRIRTOEXAAOUVSYLG

To make the key easy to remember we take a keyword like TWO and rearrange its letters alphabetically

T	W	O
2	3	1

So (2,3,1) is the key that will use to rearrange the array columns.

ii. Double

Double Transposition consists of two applications of columnar transposition to a message. The two applications may use the same key for each of the two steps, or they may use different keys.

Columnar transposition works like this: First pick a keyword, such as DESCRIBE, then write the message under it in rows:

D	E	S	C	R	I	B	E
Y	O	U	R	M	O	T	H
E	R	W	A	S	A	H	A
M	S	T	E	R	A	N	D
Y	O	U	R	F	A	T	H
E	R	S	M	E	L	T	O
F	E	L	D	E	R	B	E
R	R	I	E	S			

Now number the letters in the keyword in alphabetical order.

3	4	8	2	7	6	1	5
D	E	S	C	R	I	B	E
Y	O	U	R	M	O	T	H
E	R	W	A	S	A	H	A
M	S	T	E	R	A	N	D
Y	O	U	R	F	A	T	H
E	R	S	M	E	L	T	O
F	E	L	D	E	R	B	E
R	R	I	E	S			

Then read the cipher off by columns, starting with the lowest-numbered column: Column 1 is THNTTB, followed by RAERMDE YEMYEFR ORSORER HADHOE OAAALR MSRFEES UWTUSLI. This completes the first columnar transposition. Next, select and number a second keyword, and write this intermediate ciphertext under it in rows:

2	7	1	8	9	5	4	6	3
C	O	A	S	T	L	I	N	E
T	H	N	T	T	B	R	A	E
R	M	D	E	Y	E	M	Y	E
F	R	O	R	S	O	R	E	R
H	A	D	H	O	E	O	A	A
A	L	R	M	S	R	F	E	E
S	U	W	T	U	S	L	I	

Finally, take it off by columns again and put it into five-letter groups for transmission.

NDODR WTRFH ASEER AERMR OFLBE OERSA YEAEI HMRAL UTERH MTTYS OSU

To decrypt a double transposition, construct a block with the right number of rows under the keyword, blocking off the short columns. Write the cipher in by columns, and read it out by rows.

(1.6.1.2) Substitution cipher

A system of encryption in which each letter of a message is replaced with another character, but retains its position within the message.

i. Monoalphabetic

A substitution cipher system is the system that use one alphabet throughout encryption.

a. Simple substitution cipher

Simple substitution ciphers replaced each character of plaintext with the corresponding character of the ciphertext; a single one-to-one mapping from plaintext to ciphertext characters is used to encipher an entire message.

➢ Direct standard

The **Caesar cipher** is the one most famous and simplest of all ciphers. It is classified as a substitution cipher because the sender replaces the letters in the actual message with a new set of letters. In the Caesar cipher, each letter is replaced with the third letter following it in the alphabet. The alphabet wraps around, so if the letter in the actual message were X,Y, or Z, it would be replaced with A, B, or C, respectively. The modern English alphabet actually contains several letters not in the Roman alphabet, but we will demonstrate the cipher using the modern English alphabet.

Plaintext alpha.:

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**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**

Ciphertext alpha.:

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
3	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**0**	**1**	**2**

As an example, if the message is: ALMANSOUR UNIVERSITY COLLEGE, then

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

Note that the key to deciphering a message encoded with a Caesar cipher (also called a Caesar shift) is knowing the number of letters by which the alphabet is shifted. As we see, in Caesar cipher the key is k=3, we can choose a different value to the key in the range between 0 and 25.

$$c=E_k(m)=(m+k) \bmod 26$$

For example in the above example $E_3(A) = E_3(0) = (0+3) \bmod 26 = 3$ =D and $E_3(Y) = E_3(24) = (24+3) \bmod 26 = 1$ = B and so on.

If the adversary received the ciphertext and he know that the sender used the shift method, the only thing he need to do, is to try all the possibilities that equal to 25 trials.

➢ Standard reverse

This method is similar to the Direct standard, except that the ciphertext alphabet are written in reversed order from Z to A.

$$c=E_k(m)=(25-m+k) \bmod 26$$

For example if k=0 then,

Plaintext alpha.:

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	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Ciphertext alpha.:

25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

As an example, if the message is: ALMANSOUR UNIVERSITY COLLEGE, then

A L M A N S O U R   U N I V E R S I T Y   C O L L E G E
Z O N Z M H L F I   F M R E V I H R G B   X L O O V T V

➢ Multiplicative cipher

Ciphers based on multiply each character by a key k; that is,

$$E_k(m)=(m*k) \bmod 26$$

Where k and 26 are relatively prime (GCD(k,26)=1), so that the letters of the alphabet produce a complete set of residues, so that in this case the key must be an odd number and not equal to 13. So, if k=9 then,

Plaintext alpha.:

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	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Ciphertext alpha.:

0	9	18	1	10	19	2	11	20	3	12	21	4	13	22	5	14	23	6	15	24	7	16	25	8	17
A	J	S	B	K	T	C	L	U	D	M	V	E	N	W	F	O	X	G	P	Y	H	Q	Z	I	R

As an example, if the message is: ALMANSOUR UNIVERSITY COLLEGE, then

A L M A N S O U R   U N I V E R S I T Y   C O L L E G E
A V E A N G W Y X   Y N U H K X G U P I   S W V V K C K

For example in the above example $E_9(A) = E_9(0) = (0*9) \bmod 26 = 0$ =A and $E_9(Y) = E_9(24) = (24*9) \bmod 26 = 8$ = I and so on.

➢ Affine cipher

Addition (shifting) and multiplication can be combined to give an Affine transformation

$$E_{k1,k2}(m)=(m*k_1+k_2) \bmod 26$$

The conditions on k_1 are the same conditions on the key of the multiplicative cipher, and the conditions on k_2 are the same conditions on the key of the additive cipher.

Now, if k_1=7 and k_2=4 then

Plaintext alpha.:

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	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Ciphertext alpha.:

4	11	18	25	6	13	20	1	8	15	22	3	10	17	24	5	12	19	0	7	14	21	2	9	16	23
E	L	S	Z	G	N	U	B	I	P	W	D	K	R	Y	F	M	T	A	H	O	V	C	J	Q	X

As an example, if the message is: ALMANSOUR UNIVERSITY COLLEGE,

A L M A N S O U R   U N I V E R S I T Y   C O L L E G E
E D K E R A Y O T   O R I V G T A I H Q   S Y D D G U G

For example in the above example $E_{7,4}(A) = E_{7,4}(0) = (0*7+4) \bmod 26 = 4$ =E and $E_{7,4}(Y) = E_{7,4}(24) = (24*7+4) \bmod 26 = 16$ = Q and so on.

➢ Mixed alphabet

If we permit the cipher alphabet to be any rearrangement of the plain alphabet, then we can generate an enormous number of distinct modes of encryption. There are 26! such rearrangements, which is over 400,000,000,000,000,000,000,000,000, which gives rise to an equivalent number of distinct cipher alphabets. Each cipher alphabet is known as a key. If our message is intercepted by the enemy, who correctly assumes that we have used a monoalphabetic substitution cipher, they are still faced with the impossible challenge of checking all possible keys. If an enemy agent could check one of these possible keys every second, it would take roughly one billion times the lifetime of the universe to check all of them and find the correct one.

For example, one of the 26! Is the following

Plaintext alpha.:

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

Ciphertext alpha.:

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

if the message is: ALMANSOUR UNIVERSITY COLLEGE,

A L M A N S O U R U N I V E R S I T Y C O L L E G E

X C Y X Q T P K W K Q B E M W T B F A Z P C C M H M

The disadvantage of this method is that the arrangement is difficult to be remembered.

➢ Keyword mixed

In this method we need a keyword like MATHEMATICS, and a keyletter like S, then:

1st. Remove the repeated letters from the keyword, and you will get MATHEICS.

2nd. Put the first letter of the modified keyword under the keyletter flowed by the remaining letters of the keyword.

3rd. Complete the ciphertext alphabet by the remaining letters without repetitions.

Plaintext alpha.:

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

Ciphertext alpha.:

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

if the message is: ALMANSOUR UNIVERSITY COLLEGE,

A L M A N S O U R U N I V E R S I T Y C O L L E G E

B R U B V M W T Z T V O H J Z M O A C F W R R J L J

➢ Transposed keyword mixed

In this method we need a keyword like MATHEMATICS. After removing the repeated letters, we put it in a matrix with number of columns equal to the number of the letters in the modified keyword

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

Then we take the matrix letters column by column and we will get

Plaintext alpha.:

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

Ciphertext alpha.:

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

if the message is: ALMANSOUR UNIVERSITY COLLEGE,

A L M A N S O U R U N I V E R S I T Y C O L L E G E
M H G M R K E C I C R T L A I K T V N O E H H A P A

b. Homophonic substitution cipher

Homophonic substitution cipher are similar to simple substitution, except the mapping is one-to many, and each plaintext character is enciphered with a variety of ciphertext characters.

The Homophonic Substitution Cipher involves replacing each letter with a variety of substitutes, the number of potential substitutes being proportional to the frequency of the letter. For example, the letter 'a' accounts for roughly 8% of all letters in English, so we assign 8 symbols to represent it. Each time an 'a' appears in the plaintext it is replaced by one of the 8 symbols chosen at random, and so by the end of the encipherment each symbol constitutes roughly 1% of the ciphertext. The letter 'b' accounts for 2% of all letters and so we assign 2 symbols to represent it. Each time 'b' appears in the plaintext either of the two symbols can be chosen, so each symbol will also constitute roughly 1% of the ciphertext. This process continues throughout the alphabet, until we get to 'z', which is so rare that is has only one substitute. In the example below, the substitutes happen to be 2-digit numbers, there are between 1 and 12 substitutes for each letter, depending on the letter's relative abundance.

The point of offering several substitution options for popular letters is to balance out the frequencies of symbols in the ciphertext. Every symbol will constitute roughly 1% of the ciphertext. If none of the symbols appears more frequently than any other, then this cipher would appear to defy any potential attack via straightforward frequency analysis.

A	09	12	33	47	53	67	78	92				
B	48	81										
C	13	41	62									
D	01	03	45	79								
E	14	16	24	44	46	55	57	64	74	82	87	98
F	10	31										
G	06	25										
H	23	39	50	56	65	68						
I	32	70	73	83	88	93						

J	15												
K	04												
L	26	37	51	84									
M	22	27											
N	18	58	59	66	71	91							
O	00	05	07	54	72	90	99						
P	38	95											
Q	94												
R	29	35	40	52	77	80							
S	11	19	36	76	86	96							
T	17	20	30	43	49	69	75	85	97				
U	08	61	63										
V	34												
W	60	89											
X	28												
Y	21	52											
Z	02												

if the message is: ALMANSOUR UNIVERSITY COLLEGE,

A	L	M	A	N	S	O	U	R		U	N	I	V	E	R	S	I	T	Y		C	O	L	L	E	G	E
53	26	27	47	71	76	00	63	29		63	59	32	34	74	35	76	32	75	21		13	00	51	51	74	06	46

➢ Beale cipher

In this method we assign a set of numbers to each letter in the plaintext alphabet by using a specific text, each letter in the plaintext will be replaced by number that represent the location of some word in the text that start with this letter.

For example, if the text is:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
“Christmas, the annual festival of Christ's birth. Christmas Day falls on December 25 and celebrates the
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 3233 34 35
birth of Jesus Christ in Bethlehem as recounted in the Gospels of Matthew and Luke. It is, after Easter,
36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 525354
the most important feast in the Church's year. Since the Gospels make no mention of dates, it is not
55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
certain that Christ was born on this day. In fact, Christmas Day did not officially come into being until
74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91
354 when Pope Gregory proclaimed December 25 as the date of the Nativity. In doing so, he was
92 93 94 95 96 97 98 99
following the early Church's policy of absorbing rather than repressing existing pagan rites which, since

early times, had celebrated the winter solstice and the coming of spring.”

if the message is: ALMANSOUR COLLEGE,

A	L	M	A	N	S	O	U	R	C	O	L	L	E	G	E
03	31	29	14	48	44	05	73	24	15	97	31	31	94	27	35

➢ Higher order homophonic

Recall that, given enough ciphertext, most ciphers are theoretically breakable because there is a single key that deciphers the ciphertext into meaningful plaintext; all other keys produce meaningless sequence of letters.

It is possible to construct higher-order homophonic ciphers where each ciphertext deciphers into more than one meaningful plaintext using different keys. For example, the same ciphertext could decipher into the following 2 different plaintexts using different keys:

THE TREASURE IS BURIED IN GOOSE CREEK
THE BEALE CIPHERS ARE A GIGANTIC HOAX

To construct a second-order homophonic cipher (meaning that for each plaintext there are two possible meaningful plaintexts), arrange the numbers 1 through n^2 into an nXn matrix K whose rows and columns correspond to the characters of the plaintext alphabet. For each plaintext character a, row a of K defines one set of homophones $f_1(a)$, while column a defines another set of homophones $f_2(a)$. A plaintext message $M=m_1\ m_2$ …is enciphered along with a dummy message $X=x_1\ x_2$ … to get ciphertext $C=c_1\ c_2$ …, where $c_i = K(m_i\ ,x_i\ )$, i=1,2,… That is, c_i is in row m_i and column x_i.

For example. Let n=5. The following is 5X5 matrix for the plaintext alphabet {E, I, L, M, S}.

	E	I	L	M	S
E	10	22	18	02	11
I	12	01	25	05	20
L	19	06	23	13	07
M	03	16	08	24	15
S	17	09	21	14	04

And the message that we want to encipher is SMILE which is replace by LIMES, then

M	=	S	M	I	L	E
X	=	L	I	M	E	S
C	=	21	16	05	19	11

ii. Polyalphabetic.

Polyalphabetic substitution cipher is a substitution cipher in which the cipher alphabet changes during the encryption. The change is defined by a key.

➢ Vigenere cipher

The Vigenere Cipher , proposed by Blaise de Vigenere from the court of Henry III of France in the sixteenth century, is a polyalphabetic substitution based on the following tableau:

	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	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
B	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
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
D	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
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
F	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
G	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
H	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
I	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
J	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
K	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
L	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
M	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
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
O	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
P	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
Q	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
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
S	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
T	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
U	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
V	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
W	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
X	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
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
Z	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

Note that each row of the table corresponds to a Caesar Cipher. The first row is a shift of 0; the second is a shift of 1; and the last is a shift of 25. Mathematically, $E_{k_i}(m)=(m+k_i) \bmod 26$.

The Vigenere cipher uses this table together with a keyword to encipher a message. For example, suppose we wish to encipher the plaintext message:

TO BE OR NOT TO BE THAT IS THE QUESTION

using the keyword RELATIONS. We begin by writing the keyword, repeated as many times as necessary, above the plaintext message. To derive the ciphertext using the tableau, for each letter in the plaintext, one finds the intersection of the row given by the corresponding keyword letter and the column given by the plaintext letter itself to pick out the ciphertext letter.

Keyword:	R E L A T I O N S R E L A T I O N S R E L A T I O N S R E L
Plaintext:	T O B E O R N O T T O B E T H A T I S T H E Q U E S T I O N
Ciphertext:	K S M E H Z B B L K S M E M P O G A J X S E J C S F L Z S Y

Decipherment of an encrypted message is equally straightforward. One writes the keyword repeatedly above the message:

Keyword:	R E L A T I O N S R E L A T I O N S R E L A T I O N S R E L
Ciphertext:	K S M E H Z B B L K S M E M P O G A J X S E J C S F L Z S Y
Plaintext:	T O B E O R N O T T O B E T H A T I S T H E Q U E S T I O N

This time one uses the keyword letter to pick a column of the table and then traces down the column to the row containing the ciphertext letter. The index of that row is the plaintext letter.

The strength of the Vigenere cipher against frequency analysis can be seen by examining the above ciphertext. Note that there are 7 'T's in the plaintext message and that they have been encrypted by 'K,' 'L,' 'K,' 'M,' 'G,' 'X,' and 'L' respectively. This successfully masks the frequency characteristics of the English 'T.' One way of looking at this is to notice that each letter of our keyword RELATIONS picks out 1 of the 26 possible substitution alphabets given in the Vigenere tableau. Thus, any message encrypted by a Vigenere cipher is a collection of as many simple substitution ciphers as there are letters in the keyword.

➢ Beaufort cipher

In the Beaufort cipher the table is used in the following way:

- Encryption.
  Locate the plaintext letter in the top row of the table. Search the column immediately under till the keyletter is found. Follow the row of the keyletter to the left. The cryptoletter is found in the leftmost column.

  Mathematically, $E_{k_i}(m)=(k_i-m) \bmod 26$.

- Decryption.
  Locate the cryptoletter in the leftmost column of the table. Search the row to the right till the keyletter is found. Go straight up from the keyletter. The cleartext is found in the top row.

The Beaufort way of using the table is somewhat easier than standard Vigenere, since you only have to follow one route instead of finding an intersection of a row and a column.

Keyword:	R E L A T I O N S R E L A T I O N S R E L A T I O N S R E L
Plaintext:	T O B E O R N O T T O B E T H A T I S T H E Q U E S T I O N
Ciphertext:	Y Q K W F R B Z Z Y Q K W A B O U K Z L E W D O K V Z J Q Y

Note,

Cipher	Enciphering	Deciphering
Vigenere	c = m + k	m = c – k
Beaufort	c = k - m	m = k - c

iii. Polygraphic

Polygram substitution ciphers encipher block of letters at the time, rather than a single letter; this makes cryptanalysis harder, as it destroys the single letter frequency distribution.

➢ Playfair cipher

To encipher a message in Playfair, pick a keyword and write it into a five-by-five square, omitting repeated letters and combining I and J in one cell. In this example, we use the keyword MANCHESTER and write it into the square by rows. It may be written in any other pattern; other popular choices include writing it by columns or writing it in a spiral starting at one corner and ending in the center. Follow the keyword with the rest of the alphabet's letters in alphabetical order.

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

First we need to prepare the plaintext message for encryption. To encrypt "THIS SECRET MESSAGE IS ENCRYPTED," break it up into two-letter groups. If both letters in a pair are the same, insert an X between them. If there is only one letter in the last group, add an X to it.

TH IS SE CR ET ME SX SA GE IS EN CR YP TE DX

Now we encrypt each two-letter group. Find the T and H in the square and locate the letters at opposite corners of the rectangle they form:

.	.	N	.	H
.	.	T	.	B
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

Replace TH with those letters, starting with the letter on the same row as the first letter of the pair: TH becomes BN. Continue this process with each pair of letters:

TH IS SE CR ET ME SX SA GE IS EN CR YP TE DX
BN FR

Notice that S and E are in the same row. In this case we take the letter immediately to the right of each letter of the pair, so that SE becomes TS.

.	.	.	.	.
E	S	T	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

TH IS SE CR ET ME SX SA GE IS EN CR YP TE DX
BN FR TS

Now we see that C and R are in the same column. Use the letter immediately below each of these letters, so that CR becomes RI. This is the last special case, and the encryption proceeds without further incident.

.	.	.	C	.
.	.	.	R	.
.	.	.	I/J	.
.	.	.	.	.
.	.	.	.	.

TH IS SE CR ET ME SX SA GE IS EN CR YP TE DX
BN FR TS RI SR ED TW FS DT FR TM RI XQ RS GV

To decrypt the message, simply reverse the process: If the two letters are in different rows and columns, take the letters in the opposite corners of their rectangle. If they are in the same row, take the letters to the left. If they are in the same column, take the letters above each of them.

➢ Hill cipher

This method apply a linear transformation on d letters of the plaintext to get d letters of the ciphertext. The message divided onto a number of blocks M, each block contains d letters, then rearrange it in matrix of one column and d rows. And we use a matrix K with the size dXd that contains numbers from the range between 0 and 25, then

$$C=KM \bmod 26$$

For example, if

$$K=\begin{bmatrix}3 & 3\\ 2 & 5\end{bmatrix} \Rightarrow K^{-1}=\begin{bmatrix}15 & 17\\ 20 & 9\end{bmatrix}$$

and we want to encipher the message HELP, then

$$M_1=\begin{bmatrix}H\\ E\end{bmatrix}=\begin{bmatrix}7\\ 4\end{bmatrix}; M_2=\begin{bmatrix}L\\ P\end{bmatrix}=\begin{bmatrix}11\\ 15\end{bmatrix}$$

$$C_1=KM_1=\begin{bmatrix}3 & 3\\ 2 & 5\end{bmatrix}\begin{bmatrix}7\\ 4\end{bmatrix}=\begin{bmatrix}33\\ 34\end{bmatrix}=\begin{bmatrix}7\\ 8\end{bmatrix}=\begin{bmatrix}H\\ I\end{bmatrix}$$

$$C_2=KM_2=\begin{bmatrix}3 & 3\\ 2 & 5\end{bmatrix}\begin{bmatrix}11\\ 15\end{bmatrix}=\begin{bmatrix}78\\ 97\end{bmatrix}=\begin{bmatrix}0\\ 19\end{bmatrix}=\begin{bmatrix}A\\ T\end{bmatrix}$$

so, the ciphertext is HIAT.

Now to decipher the word HIAT

$$C_1 = \begin{bmatrix} H \\ I \end{bmatrix} = \begin{bmatrix} 7 \\ 8 \end{bmatrix};\ C_2 = \begin{bmatrix} A \\ T \end{bmatrix} = \begin{bmatrix} 0 \\ 19 \end{bmatrix}$$

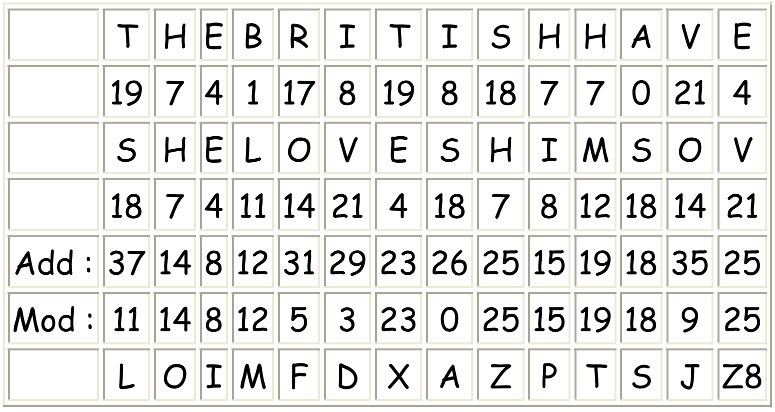
$$M_1 = K^{-1}C_1 = \begin{bmatrix} 15 & 17 \\ 20 & 9 \end{bmatrix}\begin{bmatrix} 7 \\ 8 \end{bmatrix} = \begin{bmatrix} 241 \\ 212 \end{bmatrix} = \begin{bmatrix} 7 \\ 4 \end{bmatrix} = \begin{bmatrix} H \\ E \end{bmatrix}$$

$$M_2 = K^{-1}C_2 = \begin{bmatrix} 15 & 17 \\ 20 & 9 \end{bmatrix}\begin{bmatrix} 0 \\ 19 \end{bmatrix} = \begin{bmatrix} 323 \\ 171 \end{bmatrix} = \begin{bmatrix} 11 \\ 15 \end{bmatrix} = \begin{bmatrix} L \\ P \end{bmatrix}$$

so, the original word is HELP.

(1.7)One-Time Pads

During the war, an AT&T engineer Gilbert Vernam proposed a system called the One-Time Pad that has perfect security. In this system additive ciphers are used to encipher each letter of the plaintext; however, the shift is different for each letter! The shift is determined from a one-time pad, which means some large collection of letters, such as a book. Each day a different page was used for the coded messages. If the plaintext were THE BRITISH HAVE FIFTY TANKS and the relevant part of the one-time pad were SHE LOVES HIM SO VERY MUCH NOW we would use the number of each letter as the shift. For instance since S corresponds to number 18, the cipher for the beginning T would be 18 letters after T, namely, L. The ciphertext is then computed as follows



	T	H	E	B	R	I	T	I	S	H	H	A	V	E
	19	7	4	1	17	8	19	8	18	7	7	0	21	4
	S	H	E	L	O	V	E	S	H	I	M	S	O	V
	18	7	4	11	14	21	4	18	7	8	12	18	14	21
Add :	37	14	8	12	31	29	23	26	25	15	19	18	35	25
Mod :	11	14	8	12	5	3	23	0	25	15	19	18	9	25
	L	O	I	M	F	D	X	A	Z	P	T	S	J	Z8

Different letters of ciphertext could correspond to the same plaintext letter, and vice versa. This cryptosystem is virtually unbreakable. The weakness is the key which must be immense. This must be shared by all communicants. Thus, there is a security problem in transport of the key. However, transport of the keys can usually be carried out at a chosen time and place, while coded messages usually need to be sent in emergency situations. Also, statistical analysis may be possible if the key is a regular text; for this reason some effort is usually made to choose keys which are truly random sequences of characters.

(1.8) Cryptanalysis:

The science of deducing the plaintext from a ciphertext, without knowledge of the key.

(1.8.1) Attacks on encrypted messages

The objective of the following attacks is to systematically recover plaintext from ciphertext, or even more drastically, to deduce the decryption key.

1. **A ciphertext-only attack**: is one where the adversary (or cryptanalyst) tries to deduce the decryption key or plaintext by only observing ciphertext. Any encryption scheme vulnerable to this type of attack is considered to be completely insecure.
2. **A known-plaintext attack**: is one where the adversary has a quantity of plaintext and corresponding ciphertext. This type of attack is typically only marginally more difficult to mount.
3. **A chosen-plaintext attack**: is one where the adversary chooses plaintext and is then given corresponding ciphertext. Subsequently, the adversary uses any information deduced in order to recover plaintext corresponding to previously unseen ciphertext.
4. **An adaptive chosen-plaintext attack**: is a chosen-plaintext attack wherein the choice of plaintext may depend on the ciphertext received from previous requests.
5. **A chosen-ciphertext attack**: is one where the adversary selects the ciphertext and is then given the corresponding plaintext. One way to mount such an attack is for the adversary to gain access to the equipment used for decryption (but not the decryption key, which may be securely embedded in the equipment). The objective is then to be able, without access to such equipment, to deduce the plaintext from (different) ciphertext.
6. **An adaptive chosen-ciphertext attack**: is a chosen-ciphertext attack where the choice of ciphertext may depend on the plaintext received from previous requests.

(1.8.2) Some concepts on cryptanalysis:

- **Frequency**: number of appearance of the letter in the ciphertext, where the frequencies of the ciphertext letters are compared with the frequencies in Table 1 or Figure 5.

- **Repetition**: is the similar parts in the ciphertext that have length not less than three. This helps us to find the length of the key (the number of alphabets that used to enciphering in the polyalphabetic systems).
  Take the Highest Common Factor HCF between the reputations, which represent the length of the key, this method, is called the Kasiski method.

Letter	%
a	8.167
b	1.492
c	2.782
d	4.253
e	12.702
f	2.228
g	2.015
h	6.094
i	6.966
j	0.153
k	0.772
l	4.025
m	2.406
n	6.749
o	7.507
p	1.929
q	0.095
r	5.987
s	6.327
t	9.056
u	2.758
v	0.978
w	2.360
x	0.150
y	1.974
z	0.074

Table 1: English letters frequencies

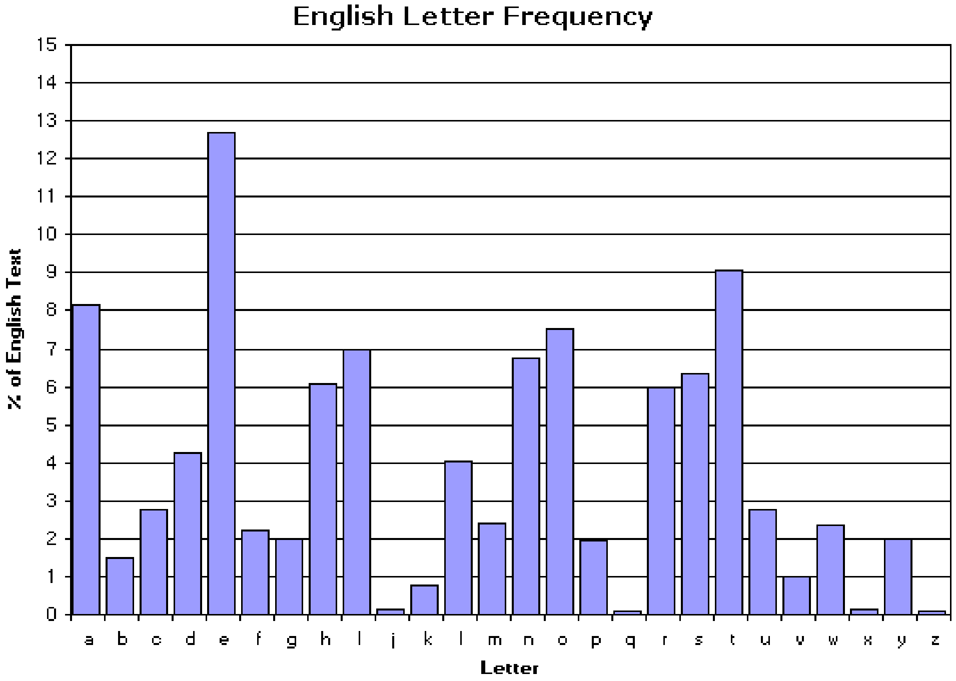


Figure 5: Histogram of English letters frequencies

- **Index of Coincidence (IC)**: is the probability that two letters selected from the text are identical, we can compute the IC from the following equation:

$$IC = \frac{\sum_{A}^{Z} f_{\lambda}(f_{\lambda} - 1)}{n(n-1)},$$

where f_{λ} is the frequency of the letter λ in the ciphertext and n is the length of the letter. The IC value differs from language to another. We can use the IC to discover if the message were enciphered using Monoalphabetic system or polyalphabetic system.

✦ **Coincidence**: is the computing of the coincidence of the ciphertexts, where two messages put one over the other, and the purpose is to discover if the two messages were enciphered using the same key. If there is 7 coincidence letters between 100 letters in the two messages then the two messages were enciphered using the same key, while if there is 4 letters coincidence between every 100 letters then they enciphered with different keys.

(1.9) Cryptanalysis examples:

First of all we must specified the type of the cipher system that was used. If the frequencies of the ciphertext are the same as the frequencies of the language then, a transposition cipher system was used; otherwise a substitution cipher system was used.

(1.9.1) Cryptanalysis of transposition cipher systems:

When we decide that a transposition cipher system were used, we put the cipher text in mXn matrix, m and n depends on the length of the received ciphertext, for example if the length is 500 then one of the possible sizes is 20X25. Then we rearrange the columns to get some known patterns such as (and, the, ion, that,…) in addition to some expected word in the message.

As we know there are two types of transposition cipher system: simple and double transposition, the cryptanalysis of the last one is more complicated because we lose the ability to find the known patterns.

(1.9.2) Cryptanalysis of substitution cipher systems:

If we know that a substitution cipher system was used, the next step is to determine whether a monoalphabetic system or polyalphabetic system was used, by using the IC of the language.

Example: A sample of ordinary English contains the following distribution of letters

Letter	Count	Letter	count
A	141	N	119
B	36	O	132
C	36	P	28
D	103	Q	1
E	188	R	95
F	37	S	64
G	34	T	182
H	102	U	59
I	123	V	13
J	4	W	55
K	18	X	3
L	56	Y	23
M	27	Z	0

What is the probability of selecting an identical pair of letters from this collection? in other word compute the IC.

$$IC = \frac{\sum_{A}^{Z} f_\lambda (f_\lambda - 1)}{n(n-1)}$$

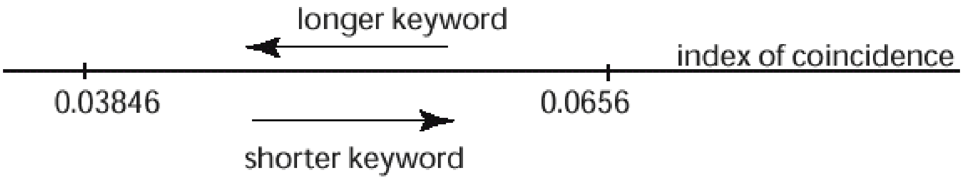
$$IC = \frac{141(141-1) + 36(36-1) + ... + 23(23-1) + 0(0-1)}{1679(1679-1)} = \frac{184838}{2817362} \approx 0.0656 .$$

Example: What is the index of coincidence for a collection of 2600 letters consisting of 100 A 's,100 B 's,100 C 's,...,100 Z 's?

$$IC = \frac{100 \cdot 99 + 100 \cdot 99 + ... + 100 \cdot 99 + 100 \cdot 99}{2600 \cdot 2599} \approx 0.0384615 .$$

As we see from the two examples above the index of coincidence of totally random (uniformly distributed) collection of letters is about 0.0385. Vigenere ciphertexts from longer keywords have a more uniform distribution of letters. For keyword length closer to 1, the index of coincidence will be larger, closer to 0.0656.

Polyalphabeticity Measure for English



If the length of the text is n, we can quantify the connection between index of coincidence and keyword length k, (number of alphabets), where:

$$k \approx \frac{0.0265 \cdot n}{(0.065 - IC) + n(IC - 0.0385)},$$

Example: A polyalphabetic ciphertext has the following letter counts.

Letter	Count	Letter	count
A	60	N	28
B	50	O	83
C	42	P	44
D	64	Q	69
E	51	R	13
F	63	S	29
G	19	T	66
H	48	U	87
I	56	V	63
J	67	W	19
K	23	X	43
L	45	Y	39
M	44	Z	67

Estimate the keyword length.

Solution: There are n=1282 letters.

$$\text{IC}=\frac{60\cdot 59+50\cdot 49+\ldots+67\cdot 66}{1282\cdot 1281}=\frac{35761}{821121}\approx 0.04355.$$

$$\text{K}=\frac{0.0265\cdot 1282}{(0.065-0.04355)+1282(0.04355-0.03846)}=5.1892.$$

Based only on this evidence, a reasonably likely keyword length is 5.

➢ Now, after the above tests if we conclude that a monoalphabetic cipher system was used, then:

❶ If a direct standard or reversed system were used, we compare the frequencies of the ciphertext with the frequencies of the English language, start by putting E against the letter with the higher frequency in the ciphertext, then we put the other letters sequentially.

❷ If a mixed cipher system was used (Random) then we compare the frequencies of the ciphertext with that in Table 1 and Figure 5.

For advanced analysis we can use in addition to Table 1, a table of double letter frequencies TH, HE, IN, ER, RE, ON, AN, EN,…, and triple letter frequencies THE, AND, TIO, ATI, FOR, THA, TER, RES,… and so on.

➢ If a polyalphabetic cipher system was used then we will use the Kasiski method to find the length of the key k (number of alphabets). Then we divide the ciphertext into k parts, each part will analyze as in ❷ above.

The Kasiski method was introduced in 1863 by the Prussian military officer Friedrich W. Kasiski. The method analysis repetitions in the ciphertext to determine the period.

For example, consider the plaintext TO BE OR NOT TO BE enciphered with a Vigenere cipher with key HAM:

K=	H	A	M	H	A	M	H	A	M	H	A	M	H
M=	T	O	B	E	O	R	N	O	T	T	O	B	E
C=	A	O	N	L	O	D	U	O	F	A	O	N	L

The ciphertext contains two occurrences of the sequence AONL 9 characters apart, and the period could be 1,3 or 9 (we know it's 3).

Repetitions in the ciphertext more than two characters long are unlikely to occur by chance. They occur when the plaintext pattern repeats at a distance equal to a multiple of the period.

If there are m ciphertext repetitions that occur at intervals I j ($1\leq j\leq m$) the period is likely to be some number that divides most of the m intervals.

Example: We shall use IC and Kasiski method to analyze the following ciphertext.

ZHYME ZVELK OJUBW CEYIN CUSML RAVSR YARNH CEARI UJPGP VARDU
QZCGR NNCAW JALUH GJPJR YGEGQ FULUS QFFPV EYED**Q** **GOLKA** **LVOSJ**
TFRTR YEJZS RVNCI HYJNM ZDCRO DKHCR MMLNR FFLFN **QGOLK** **ALVOS**
JWMIK QKUBP SAYOJ RRQYI NRNYC YQZSY EDNCA LEILX RCHUG IEBKO
YTHGV VCKHC JE**QGO** **LKALV** **OSJ**ED WEAKS GJHYC LLFTY IGSVT FVPMZ
NRZOL CYUZS FKOQR YRTAR ZFGKI QKRSV IRCEY USKVT MKHCR MYQIL
XRCRL GQARZ OLKHY KSNFN RRNCZ TWUOC JNMKC MDEZP IRJEJ W

When we calculate the frequency distribution, we will find that the IC=0.04343, n=346,

$$k=\frac{0.0265\cdot 346}{(0.065-0.04343)+346(0.04343-0.03846)}=5.2659$$

The IC indicates that this is a polyalphabetic cipher with a period of about 5.
We observe that there are 3 occurrences of the sequence QGOLKALVOSJ, the first two occurrences are separated by 51 and the last two by 72 characters (start to start); the only common divisor of 51 and 72 is 3 - the period is almost certainly 3.

Example: When we calculate the IC of some ciphertext, we find that k=9.34. Also we observe that there is NYX appearance many times in the ciphertext and the distance between them are 30, 50, 90, 110, and 33.
Since these can each be factored as

30=2X3X5
50=2X5X5
90=2X3X3X5
110=2X5X11
33=3X11

there are a number of candidates for key length. 2 and 5 are popular factors among these distance followed by 3 and 11. Note that all but 33 have 2X5=10 as a factor. The cryptanalyst might then disregard 33 as a pure coincidence, and discard that data in favor of conjecture that the key length is a multiple of 2 and/or 5. Combining this with data from the Friedman test that the key approximately 9 letters long, the cryptanalyst guesses that the key is 10 letters long, and not 2 or 5 letters long.

Exercises

Attempt all the following exercises

Q1) In problems 1-5, state whether the following are true or false.

1. 14 = 5 (mod 9)
2. 4 = 16 (mod 12)
3. 7 = 3 (mod 10)
4. -3 = 5 (mod 8)
5. 3 = 9 (mod 12)
6. 90 = 9 (mod 10)

Q2) Try to encipher the following message:

If you have some trouble when you worry you make it double

Using

a) Message reversal.
b) Columnar transposition, key=Software.
c) Double columnar transposition, key$_1$=Microsoft, key$_2$=Samsung.
d) Direct standard, key=7.
e) Multiplicative cipher, key=11.
f) Affine cipher, key=(7,12).
g) Keyword mixed, keyword=Professional, keyletter=S.
h) Transposed keyword mixed, keyword=Marching season.
i) Vigenere cipher, keyword=Yanni.
j) Beaufort cipher, keyword=Yanni.
k) Playfair cipher, keyword=imagination.

Q3) If $K = \begin{bmatrix} 7 & 5 \\ 4 & 9 \end{bmatrix}$, use this matrix to encipher the message:

WINTER LIGHT

then find K^{-1} and decipher the result of the above.

Q4) Encrypt the following message using a direct standard Cipher with key value K=18 and write a modular equation to express this system of encipherment.

MATHEMATICS IS FUN

Q5) Decrypt the following message, which has been encrypted using a direct standard Cipher with key value K=1. Write out a modular equation to express this system of encipherment.

GZUD Z MHBD CZX

Q6) Try to decipher the following ciphertext:

ETNAN XFWN LYK Y RYETNA QF EBWKXF LTX KYQP ETQK YPHQWN QK RXA DXB KXF DXB PXFE LYKT DXBAKNMR LNMM KX DXBA RNNE KCNMM MQUN TNMM QR QF VNP LQET Y ZQAM UNNI DXBA KTXNK XF.

Q7) Consider the ciphertext:

WSPGM HHEHM CMTGP NROVX WISCQ TXHKRVESQT IMMKW BMTKW CSTVL TGOPZ XGTQM CXHCX HSMGX WMNIA XPLVY GROWX LILNF JXTJI RIRVE XRTAX WETUS BITJM CKMCO TWSGR HIRGK PVDNI HWOHL DAIVX JVNUS JX

Calculate the IC, and then estimate the key length.

Q8) How many possible keys does a Playfair cipher have? Express your answer as an approximate power of 2.

Q9) The following message has been encrypted using a direct standard Cipher with an unknown key value. Use the first word of the encrypted message to try all the possible keys. Then decrypt the entire message and determine the correct key value used for encryption.

DZXP XPDDLRPD NLY MP DZWGPO MJ NZXAWPETYR ESP AWLTY NZXAZYPYE

Q10) The following message has been encrypted using a direct standard Cipher with an unknown key value. Use the table of frequency to determine which cipher characters occur frequently and infrequently. Decrypt the entire message and state the key value used to encrypt the message.

BMBLG HMTLX TLRMH WXVKR IMTFX LLTZX PAXGR HNWHG HMDGH PPAXK XMAXP HKWLU XZBGT GWXGW HYMXG FHKXM AHKHN ZATGT ERLBL BLKXJ NBKXW MHWXV KRIML NVAFX LLTZX L

Q11) Decipher:

NSCRG LEXCT OEFNE HNRTL HOAHT OEICY NOIOT TEEGK SGWAO IHIAA NRWEN OTKRT DDPE

if you know that a columnar transposition were used with keyword k=COMPARE.

Q12) Decipher the following cryptogram:

GLZOXA

Knowing that an Affine cipher with k_2=4 was used and that the plaintext is a word of the English language.

Q13) If you know that a Vigenere cipher were used to get the following encrypted message:

TUAEIGTUEISBLNCCUA

And the key was k=RUN. Try to get the plaintext.

Q14) Encipher the message

TO BE OR NOT TO BE

Using the Playfair cipher with the key, k=software engineering.

Q15) Use a second order homophonic cipher to encipher the message **COROLLA** using the dummy message **CAPPRIS**. [Hint: create a table of nXn size, where n is the number of the used letters]

Q16) In a ciphertext we observe that there is a pattern appears several times, and the distance between them are 63, 21, and 56 what are the possibilities of the key length.

Q17) If you know that the keyword mixed cipher was used to encipher a message, and you receive one of the cryptogram. Use the frequencies comparison to find the original message.

YHVEVJLXVSST VI V HZIKSJ DR JCLI HZXZBJ YZNZSDFEZBJ LB JZXCBDSDAT EVBT DR JCZ XLFCZH ITIJZEI JCVJ PZHZ DBXZ XDBLILYZHZY IZXKHZ VHZ BDP WHZVMVWSZ.

Q18) Using the Hill digraph cipher that sends plaintext block PQ to ciphertext CD with

$$C=3P+10Q \pmod{26}$$
$$D=9P+7Q \pmod{16}$$

encipher the message **BEWARE THE MESSENGER**, then compute the inverse transformation and decipher again.

Q19) Decipher the ciphertext message **RD SR QO VU QP CZ AN QW RD DS AK OB** which was enciphered using the Hill digraph cipher which sends plaintext block PQ to ciphertext block CD via

$$C=13P+4Q \pmod{26}$$
$$D=9P+Q \pmod{26}.$$

Q20) A cryptanalyst has determined that the most common digraph appearing in ciphertext enciphered using a Hill digraph cipher is **RH**, followed closely by **NI**. She assumes these correspond to the most common English digraphs, **TH** and **HE**, respectively. If she is correct, given these values, what are the values of a,b,c, and d in the enciphering transformation

$$C = aP + bQ \pmod{26}$$

$$D = cP + dQ \pmod{26}$$

Q21) Explain the difference between a substitution cipher and a transposition cipher.

Q22) A message is enciphered with a transposition cipher. What should we see when we do a frequency analysis of the message?

Chapter Two
Practical Security

(2.1) Introduction:

The discussion of chapter one arise a certain weakness of Monoalphabetic cipher, the encipherment of a letter only involves using a small portion of the letters of key, exactly the one letter which is substituted for it. Then we can break this cipher system by finding small portion of the message and try to decipher them and by using these small portions we can find the way to decipher the overall message.

To make the system more secure, it seems desirable to use a considerable amount of keys to encipher each character of the message. And also it is probably helpful to 'spread' the statistical structure of the ciphertext by enciphering a number of message characters simultaneously.

(2.2) Diffusion and Confusion:

In order to accommodate the points of using a considerable amount of key and spread the statistical structure of ciphertext, and reduce the effectiveness of statistical attacks on ciphertext: Shannon suggests that the cryptographer uses two techniques which he calls **Diffusion** and **Confusion**.

The idea of diffusion is to spread the statistics of the message space into a statistical structure, which involves long combinations of the letter in the ciphertext.

To understand the idea of diffusion assume M=m_1 m_2…, then we pick an integer s and replace m by the sequence y_1 y_2… where

$$y_n = \sum_{i=0}^{s-1} m_{n+i} \mod 26$$

Where n=1, 2, 3,… By doing this we'll get the message space with letter frequencies of the new message space Y will become more equal than in M.

The effect of all this is that the cryptanalyst needs along time so that he can find a certain way to decipher the ciphertext. In practice this means that we are enciphering a number of message characters simultaneously and dependently.

The disadvantage of this type of system is that, at the receiver, each part of the message depends on a number of ciphertext characters. Thus, if one single ciphertext is error transmitted, this may cause many errors in the received message. This diffusing effect of one error in transmission causing many in decipherment is usually called **error propagation**.

The idea of confusion is to hide any relationship between the plaintext, ciphertext and the key. This implies that the message characters will encipher depending on virtually the entire key. This idea will force the cryptanalyst to find the whole key simultaneously and will make him solve considerably more complex equation than when he was able to find the key piece by piece.

The ideas of confusion and diffusion are the principles behind the design of most block ciphers.

The summary of the above discussion is:

- Confusion is produced using substitution; when a long block of plaintext is substituted for a different block of ciphertext, the statistical patterns of plaintext become hard to detect.
- Diffusion dissipates the redundancy of the plaintext by spreading it out over the ciphertext; this can be produced using permutation, i.e. reordering the parts of a plaintext message.

(2.3) Shannon's five criteria:

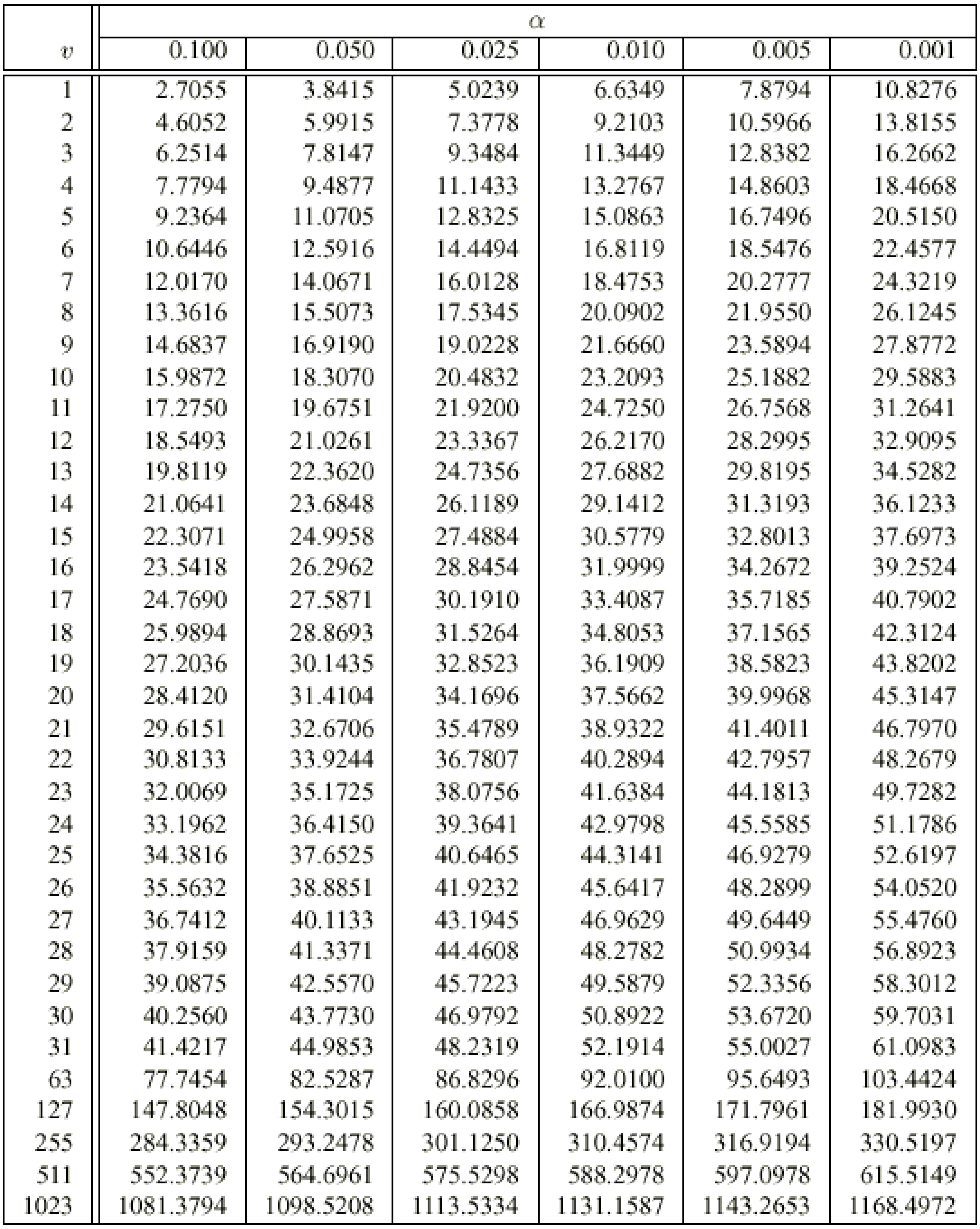
Shannon suggests five important criteria to evaluate the cipher systems, which are:

1. The amount of secrecy offered.
2. The size of the key.
3. The simplicity of the enciphering and deciphering operations.
4. The propagation of errors.
5. Extension of the message.

It is clear that any system has a higher security will be superior than any other system. If the system theoretically can be broken, it might, practically impossible to do so; because there might not be a certain way to analyze the code so that an intruder can't take the original plaintext from the ciphertext.

Some of cipher systems generate a key space i.e. it take all the possibilities of the keys that may solve the problem. A good cipher system has to have a simple encipher and decipher algorithms but the analysis of the key has to be a very complicated one; i.e. the time taken to encipher and decipher the message must be a polynomial time while the time taken by the cryptanalyst to break the message must be an exponential time.

One of the most good things in cipher system is that the key must be simple and can be easily memorized. In many ciphering systems, the error might be propagate and damage or garbled the information, hence we have cut this propagation of errors. Finally the fifth criteria discuss that if the message being a very long, it might be broken, hence the cipher system has to be unbreakable in spite of the massage is long.



v	α					
	0.100	0.050	0.025	0.010	0.005	0.001
1	2.7055	3.8415	5.0239	6.6349	7.8794	10.8276
2	4.6052	5.9915	7.3778	9.2103	10.5966	13.8155
3	6.2514	7.8147	9.3484	11.3449	12.8382	16.2662
4	7.7794	9.4877	11.1433	13.2767	14.8603	18.4668
5	9.2364	11.0705	12.8325	15.0863	16.7496	20.5150
6	10.6446	12.5916	14.4494	16.8119	18.5476	22.4577
7	12.0170	14.0671	16.0128	18.4753	20.2777	24.3219
8	13.3616	15.5073	17.5345	20.0902	21.9550	26.1245
9	14.6837	16.9190	19.0228	21.6660	23.5894	27.8772
10	15.9872	18.3070	20.4832	23.2093	25.1882	29.5883
11	17.2750	19.6751	21.9200	24.7250	26.7568	31.2641
12	18.5493	21.0261	23.3367	26.2170	28.2995	32.9095
13	19.8119	22.3620	24.7356	27.6882	29.8195	34.5282
14	21.0641	23.6848	26.1189	29.1412	31.3193	36.1233
15	22.3071	24.9958	27.4884	30.5779	32.8013	37.6973
16	23.5418	26.2962	28.8454	31.9999	34.2672	39.2524
17	24.7690	27.5871	30.1910	33.4087	35.7185	40.7902
18	25.9894	28.8693	31.5264	34.8053	37.1565	42.3124
19	27.2036	30.1435	32.8523	36.1909	38.5823	43.8202
20	28.4120	31.4104	34.1696	37.5662	39.9968	45.3147
21	29.6151	32.6706	35.4789	38.9322	41.4011	46.7970
22	30.8133	33.9244	36.7807	40.2894	42.7957	48.2679
23	32.0069	35.1725	38.0756	41.6384	44.1813	49.7282
24	33.1962	36.4150	39.3641	42.9798	45.5585	51.1786
25	34.3816	37.6525	40.6465	44.3141	46.9279	52.6197
26	35.5632	38.8851	41.9232	45.6417	48.2899	54.0520
27	36.7412	40.1133	43.1945	46.9629	49.6449	55.4760
28	37.9159	41.3371	44.4608	48.2782	50.9934	56.8923
29	39.0875	42.5570	45.7223	49.5879	52.3356	58.3012
30	40.2560	43.7730	46.9792	50.8922	53.6720	59.7031
31	41.4217	44.9853	48.2319	52.1914	55.0027	61.0983
63	77.7454	82.5287	86.8296	92.0100	95.6493	103.4424
127	147.8048	154.3015	160.0858	166.9874	171.7961	181.9930
255	284.3359	293.2478	301.1250	310.4574	316.9194	330.5197
511	552.3739	564.6961	575.5298	588.2978	597.0978	615.5149
1023	1081.3794	1098.5208	1113.5334	1131.1587	1143.2653	1168.4972

Table 1

Selected percentiles of the χ^2 (chi-square) distribution. A(v,α *)-entry of* x *in the table has the following meaning: if* X *is a random variable having a χ^2 distribution with* v *degrees of freedom, then* P(X>x)=α .

(2.4)Concept of randomness:

Golomb's randomness postulates are presented here for historical reasons they were one of the first attempts to establish some *necessary* conditions for a periodic pseudo random sequence to look random. It is emphasized that these conditions are far from being *sufficient* for such sequences to be considered random. Unless otherwise stated, all sequences are binary sequences.

Definition Let s = s_0, s_1, s_2,… be an infinite sequence. The subsequence consisting of the first n terms of s is denoted by s^n = s_0, s_1, …, s_n.

Definition The sequence s = s_0, s_1, s_2,… is said to be ***N-periodic*** if $s_i = s_{i+N}$ for all $i \geq 0$. The sequence s is ***periodic*** if it is N-periodic for some positive integer N. The ***period*** of a periodic sequence s is the smallest positive integer N for which s is N-periodic. If s is a periodic sequence of period N, then the ***cycle*** of s is the subsequence s^N .

Definition Let s be a sequence. A ***run*** of s is a subsequence of s consisting of consecutive 0's or consecutive 1's which is neither preceded nor succeeded by the same symbol. A run of 0's is called a ***gap***, while a run of 1's is called a ***block***.

Definition Let s = s_0, s_1, s_2,… be a periodic sequence of period N. The ***autocorrelation*** *function* of s is the integer-valued function C(t) defined as

$$C(t) = \frac{1}{N}\sum_{i=0}^{N-1}(2s_i - 1)\cdot(2s_{i+t} - 1), \quad \text{for } 0 \leq t \leq N-1.$$

The autocorrelation function C(t) measures the amount of similarity between the sequence s and a shift of s by t positions. If s is a random periodic sequence of period N, then $|N \cdot C(t)|$ can be expected to be quite small for all values of t, 0 < t <N.

The above equation can be put in another simple form

$$C(t) = \frac{A-D}{N}, \quad \text{for } \; 0 \leq t \leq N-1$$

where A is the number of the similar locations between the original sequence s^N and the shifted one s^{N+t}, while D is the number of the different locations between them.

Example: Consider the following sequence:

0101101011010110101101011

compute the autocorrelation function.

Solution:

As we see N=5, s^N=01011, so

S_0=01011

S_1=10110

S_2=01101

S_3=11010

S_4=10101

1) S = 0 1 0 1 1
   S_0= 0 1 0 1 1

$$A=5,\ D=0,\ \Rightarrow c(0)=\frac{5-0}{5}=1$$

2) S = 0 1 0 1 1
   S_1= 1 0 1 1 0

$$A=1,\ D=4,\ \Rightarrow c(1)=\frac{1-4}{5}=-\frac{3}{5}$$

3) S = 0 1 0 1 1
   S_2= 0 1 1 0 1

$$A=3,\ D=2,\ \Rightarrow c(2)=\frac{3-2}{5}=\frac{1}{5}$$

4) S = 0 1 0 1 1
   S_3= 1 1 0 1 0

$$A=3,\ D=2,\ \Rightarrow c(3)=\frac{3-2}{5}=\frac{1}{5}$$

5) S = 0 1 0 1 1
   S_4= 1 0 1 0 1

$$A=1,\ D=4,\ \Rightarrow c(4)=\frac{1-4}{5}=-\frac{3}{5}.$$

Definition Let s be a periodic sequence of period N. *Golomb's randomness postulates* are the following.

R1: In the cycle s^N of s, the number of 1's differs from the number of 0's by at most 1. In other word if N is an even number then the number of 1's and 0's are equal, while if N is an odd number, then the number of 1's either more by one or less by one than the number of 0's.

R2: In the cycle s^N, at least half the runs have length 1, at least one-fourth have length 2, at least one-eighth have length 3, in general, at least $1/2^i$ have length i. Moreover, for each of these lengths, there are (almost) equally many gaps and blocks.

R3: The autocorrelation function C(t) is two-valued. That is for some integer K,

$$N\cdot C(t)=\sum_{i=0}^{N-1}(2s_i-1)\cdot(2s_{i+t}-1)=\begin{cases}N, & if\ t=0,\\ K, & if\ 1\le t\le N-1.\end{cases}$$

Definition A binary sequence which satisfies Golomb's randomness postulates is called a ***pseudo-noise sequence*** or a ***pn-sequence***.

Example (*pn-sequence*) Consider the periodic sequence s of period N = 15 with cycle

$$s^{15} = 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1.$$

The following shows that the sequence s satisfies Golomb's randomness postulates.

R1: The number of 0's in s^{15} is 7, while the number of 1's is 8.

R2: s^{15} has 8 runs. There are 4 runs of length 1 (2 gaps and 2 blocks), 2 runs of length 2 (1 gap and 1 block), 1 run of length 3 (1 gap), and 1 run of length 4 (1 block).

R3: The autocorrelation function C(t) takes on two values: C(0)=1 and $C(t) = -\frac{1}{15}$ for $1 \le t \le 14$.

Hence, s is a pn-sequence.

(2.5) Statistical tests for randomness:

Let s = $s_0, s_1, s_2, \ldots, s_{n-1}$ be a binary sequence of length n. This subsection presents four statistical tests that are commonly used for determining whether the binary sequence s possesses some specific characteristics that a truly random sequence would be likely to exhibit. It is emphasized again that the outcome of each test is not definite, but rather probabilistic. If a sequence passes all four tests, there is no guarantee that it was indeed produced by a random bit generator.

In the following tests we will take the significant value α to be equal 0.05, so the success degree will be $(100-\alpha)\%=(100-5)\%=95\%$.

(i) Frequency test (monobit test)

The purpose of this test is to determine whether the number of 0's and 1's in s are approximately the same, as would be expected for a random sequence. Let n_0 , n_1 denote the number of 0's and 1's in s, respectively. The statistic used is

$$X_1 = \frac{(n_0 - n_1)^2}{n}$$

which approximately follows a χ^2 distribution with 1 degree of freedom if $n \ge 10$. i.e. when α=0.05, $X_1 \le 3.84$.

(ii) Serial test (two-bit test)

The purpose of this test is to determine whether the number of occurrences of 00, 01, 10, and 11 as subsequences of s are approximately the same, as would be expected for a random sequence. Let n_0, n_1 denote the number of 0's and 1's in s, respectively, and let n_{00}, n_{01}, n_{10}, n_{11} denote the number of occurrences of 00, 01, 10, 11 in s, respectively. Note that $n_{00}+n_{01}+n_{10}+n_{11}=(n-1)$ since the subsequences are allowed to overlap. The statistic used is

$$X_2 = \frac{4}{n-1}\sum_{i=0}^{1}\sum_{j=0}^{1} n_{ij}^2 - \frac{2}{n}\sum_{i=0}^{1} n_i^2 + 1$$

$$= \frac{4}{n-1}\left(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2\right) - \frac{2}{n}\left(n_0^2 + n_1^2\right) + 1$$

which approximately follows a χ^2 distribution with 2 degrees of freedom if $n \ge 21$. i.e. $X_2 \le 5.99$.

(iii) Poker test

Let m be a positive integer such that $\left\lfloor \frac{n}{m} \right\rfloor \geq 5 \cdot 2^m$, and let k = $\left\lfloor \frac{n}{m} \right\rfloor$. Divide the sequence s into k non-overlapping parts each of length m, and let n_i be the number of occurrences of the i^{th} type of sequence of length m, $1 \leq i \leq 2^m$. The poker test determines whether the sequences of length m each appear approximately the same number of times in s, as would be expected for a random sequence. The statistic used is

$$X_3 = \frac{2^m}{k}\left(\sum_{i=1}^{2^m} n_i^2\right) - k$$

which approximately follows a χ^2 distribution with 2^m-1 degrees of freedom. Note that the poker test is a generalization of the frequency test: setting m = 1 in the poker test yields the frequency test.

(iv) Runs test

The purpose of the runs test is to determine whether the number of runs (of either zeros or ones) of various lengths in the sequence s is as expected for a random sequence. The expected number of gaps (or blocks) of length i in a random sequence of length n is $e_i=(n-i+3)/2^{i+2}$ .Let k be equal to the largest integer i for which $e_i \geq 5$. Let B_i, G_i be the number of blocks and gaps, respectively, of length i in s for each i, $1 \leq i \leq k$.

The statistic used is

$$X_4 = \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(G_i - e_i)^2}{e_i}$$

which approximately follows a χ^2 distribution with 2k-2 degrees of freedom.

<u>**Example**</u>: (basic statistical tests) Consider the (non-random) sequence s of length n=160 obtained by replicating the following sequence four times:

11100 01100 01000 10100 11101 11100 10010 01001.

Test the randomness of this sequence.

Solution:

The complete sequence is:

11100 01100 01000 10100 11101 11100 10010 01001
11100 01100 01000 10100 11101 11100 10010 01001
11100 01100 01000 10100 11101 11100 10010 01001
11100 01100 01000 10100 11101 11100 10010 01001

(i) (frequency test) n_0=84, n_1=76, and the value of the statistic X_1 is 0.4.

$$X_1 = \frac{(n_0 - n_1)^2}{n}$$

$$X_1 = \frac{(84-76)^2}{160} = 0.4.$$

(ii) (serial test) n_{00}=44, n_{01}=40, n_{10}=40, n_{11}=35, and the value of the statistic X_2 is 0.6252.

$$\begin{aligned}
X_2 &= \frac{4}{n-1}\sum_{i=0}^{1}\sum_{j=0}^{1} n_{ij}^2 - \frac{2}{n}\sum_{i=0}^{1} n_i^2 + 1 \\
&= \frac{4}{n-1}\left(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2\right) - \frac{2}{n}\left(n_0^2 + n_1^2\right) + 1 \\
&= \frac{4}{159}\left(44^2 + 40^2 + 40^2 + 35^2\right) - \frac{2}{160}\left(84^2 + 76^2\right) + 1 \\
&= \frac{4}{159}(1936 + 1600 + 1600 + 1225) - \frac{2}{160}(7056 + 5776) + 1
\end{aligned}$$

$$\begin{aligned}
&= \frac{4}{159}(6361) - \frac{2}{160}(12832) + 1 \\
&= 160.0252 - 160.4 + 1 = 0.6252.
\end{aligned}$$

(iii) (poker test)

m=1	$\left\lfloor \frac{160}{1} \right\rfloor$=160	>	$5\cdot 2^1$=10
m=2	$\left\lfloor \frac{160}{2} \right\rfloor$=80	>	$5\cdot 2^2$=20
m=3	$\left\lfloor \frac{160}{3} \right\rfloor$=53	>	$5\cdot 2^3$=40
m=4	$\left\lfloor \frac{160}{4} \right\rfloor$=40	<	$5\cdot 2^4$=80

Here m=3 and k=53. The blocks 000, 001, 010, 011, 100, 101, 110, 111 appear 5, 10, 6, 4, 12, 3, 6,and 7 times, respectively, and the value of the statistic X_3 is 9.6415.

$$X_3 = \frac{2^m}{k}\left(\sum_{i=1}^{2^m} n_i^2\right) - k$$

$$\begin{aligned}
X_3 &= \frac{2^3}{53}\left(5^2 + 10^2 + 6^2 + 4^2 + 12^2 + 3^2 + 6^2 + 7^2\right) - 53 \\
&= \frac{8}{53}(25 + 100 + 36 + 16 + 144 + 9 + 36 + 49) - 53 = 9.6415
\end{aligned}$$

(iv) (runs test)

i	$e_i=(n-i+3)/2^{i+2}$		
1	$(160-1+3)/2^3=\frac{162}{8}=20.25$	>	5
2	$(160-2+3)/2^4=\frac{161}{16}=10.0625$	>	5
3	$(160-3+3)/2^5=\frac{160}{32}=5$	=	5
4	$(160-4+3)/2^6=\frac{159}{64}=2.4843$	<	5

Here k=3. There are 25, 4, 5 blocks of lengths 1, 2, 3, respectively, and 8, 20, 12 gaps of lengths 1, 2, 3, respectively. The value of the statistic X_4 is 31.7913.

$$X_4=\sum_{i=1}^{k}\frac{(B_i-e_i)^2}{e_i}+\sum_{i=1}^{k}\frac{(G_i-e_i)^2}{e_i}$$

$$X_4=\frac{(B_1-e_1)^2}{e_1}+\frac{(B_2-e_2)^2}{e_2}+\frac{(B_3-e_3)^2}{e_3}+\frac{(G_1-e_1)^2}{e_1}+\frac{(G_2-e_2)^2}{e_2}+\frac{(G_3-e_3)^2}{e_3}$$

$$X_4=\frac{(25-20.25)^2}{20.25}+\frac{(4-10.0625)^2}{10.0625}+\frac{(5-5)^2}{5}+\frac{(8-20.25)^2}{20.25}+\frac{(20-10.0625)^2}{10.0625}+\frac{(12-5)^2}{5}$$

$$X_4=31.7913.$$

For a significance level of α=0.05, the threshold values for X_1, X_2, X_3, and X_4 are 3.8415 (for one degree of freedom), 5.9915 (for two degree of freedom), 14.0671 (for seven degree of freedom, since $2^m-1=2^3-1=7$), and 9.4877 (for four degree of freedom, since 2k-2=2(3)-2=4), respectively (see Tables 1). Hence, the given sequence s passes the frequency, serial, and poker tests, but fails the runs test.

Example: Consider the following periodic sequence

0101011101100011111001101001000010101110110001111100110100100001010111….

The period of this sequence is 31. Take the four first cycles of this sequence to test if it is a random sequence.

Solution: The first 124 bits of the sequence are

0 1 0 1 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 1 1 0 1 0 0 1 0 0 0
0 1 0 1 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 1 1 0 1 0 0 1 0 0 0
0 1 0 1 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 1 1 0 1 0 0 1 0 0 0
0 1 0 1 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 1 1 0 1 0 0 1 0 0 0

(i) (frequency test) n_0=60, n_1=64, and the value of the statistic X_1 is 0.1290.

$$X_1=\frac{(60-64)^2}{124}=0.1290<3.84,$$

so the sequence pass this test.

(ii) (serial test) n_{00}=27, n_{01}=32, n_{10}=32, n_{11}=32, and the value of the statistic X_2 is

$$X_2 = \frac{4}{123}\left(27^2 + 32^2 + 32^2 + 32^2\right) - \frac{2}{124}\left(60^2 + 64^2\right) + 1$$
$$= \frac{4}{123}(3801) - \frac{2}{124}(7696) + 1$$
$$= 123.6097 - 124.1290 + 1 = 0.4807 < 5.99,$$

so the sequence pass this test also.

(iii) (poker test)

m=1	$\left\lfloor \frac{124}{1} \right\rfloor$=124	>	$5 \cdot 2^1$=10
m=2	$\left\lfloor \frac{124}{2} \right\rfloor$=62	>	$5 \cdot 2^2$=20
m=3	$\left\lfloor \frac{124}{3} \right\rfloor$=41	>	$5 \cdot 2^3$=40
m=4	$\left\lfloor \frac{124}{4} \right\rfloor$=31	<	$5 \cdot 2^4$=80

Here m=3 and k=41.

010	101	110	110	001	111	100	110	100	100
001	010	111	011	000	111	110	011	010	010
000	101	011	101	100	011	111	001	101	001
000	010	101	110	110	001	111	100	110	100
100	0								

The blocks 000, 001, 010, 011, 100, 101, 110, 111 appear 3, 5, 5, 4, 7, 5, 7,and 5 times, respectively, and the value of the statistic X_3 is

$$X_3 = \frac{2^3}{41}\left(3^2 + 5^2 + 5^2 + 4^2 + 7^2 + 5^2 + 7^2 + 5^2\right) - 41$$
$$= \frac{8}{41}(9 + 25 + 25 + 16 + 49 + 25 + 49 + 25) - 41 = 2.5122$$

The degree of freedom here is 2^3-1=7, so χ^2 =14.0671. And since 2.5122<14.0671, so the sequence pass this test also.

(iv) (runs test)

i	$e_i=(n-i+3)/2^{i+2}$		
1	$(124-1+3)/2^3=\frac{126}{8}=15.75$	>	5
2	$(124-2+3)/2^4=\frac{125}{16}=7.8125$	>	5
3	$(124-3+3)/2^5=\frac{124}{32}=3.875$	<	5

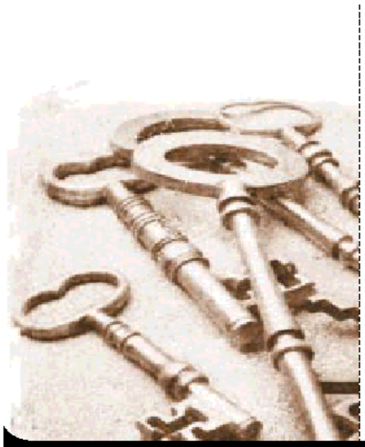
Here k=2.

i	B_i	G_i
1	16	17
2	8	8

There are 16, 8 blocks of lengths 1, 2, respectively, and 17, 8 gaps of lengths 1, 2, respectively. The value of the statistic X_4 is

$$X_4=\frac{(B_1-e_1)^2}{e_1}+\frac{(B_2-e_2)^2}{e_2}+\frac{(G_1-e_1)^2}{e_1}+\frac{(G_2-e_2)^2}{e_2}$$
$$X_4=\frac{(16-15.75)^2}{15.75}+\frac{(8-7.8125)^2}{7.8125}+\frac{(17-15.75)^2}{15.75}+\frac{(8-7.8125)^2}{7.8125}$$
$$X_4=0.11213.$$

The degree of freedom here is 2(2)-2=4, so χ^2=5.99. And since 0.11213<5.99, so the sequence pass this test also.



Exercises

Attempt all the following exercises

Q1) Consider the sequence s of length n=125 obtained by replicating the following sequence five times:

01011 01011 01011 01011 01011

Test the randomness of this sequence.

Q2) Consider the sequence s of length n=144 obtained by replicating the following sequence six times:

110010 111011 010011 011111

Test the randomness of this sequence.

Q3) Consider the following sequence of length n=200:

1010010100 0101100100 1101110010 1001001101 0101001101
0100101100 0101101100 0110101010 1001101010 0100111001
0011010110 1101100001 0011011110 1010010010 1101010100
0100101000 1101101010 0100111001 0000100101 1011010011

Test the randomness of this sequence.

CHAPTER THREE

STREAM CIPHERS

(3.1) Introduction

Stream ciphers are an important class of encryption algorithms. They encrypt individual characters (usually binary digits) of a plaintext message one at a time, using an encryption transformation which varies with time. By contrast, block ciphers tend to simultaneously encrypt groups of characters of a plaintext message using a fixed encryption transformation. Stream ciphers are generally faster than block ciphers in hardware, and have less complex hardware circuitry. They are also more appropriate, and in some cases mandatory (e.g., in some telecommunications applications), when buffering is limited or when characters must be individually processed as they are received. Because they have limited or no error propagation, stream ciphers may also be advantageous in situations where transmission errors are highly probable.

(3.2) One Time Pad

Definition Unconditional Security

A cryptosystem is unconditionally secure if it cannot be broken even with infinite computational resources.

Definition One-time Pad (OTP)

A cryptosystem developed by Mauborgne based on Vernam's stream cipher consisting of:

$|M| = |C| = |K|$, with $m_i; c_i; z_i \in \{0,1\}$.

Encrypt → $e_{ki}(m_i) = m_i \oplus z_i$.

decrypt → $d_{ki}(c_i) = c_i \oplus z_i$.

Remarks:

1. The truth table of the XOR operation $\oplus$ is:

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

2. Encryption and decryption are the same operation (XOR). Why? We show that decryption of ciphertext bit c_i yields the corresponding plaintext bit.

   Decryption: $c_i \oplus z_i = (m_i \oplus z_i) \oplus z_i = m_i \oplus (z_i \oplus z_i) = m_i$.

   Note that $z_i \oplus z_i = 0$ for $z_i = 0$ and for $z_i = 1$.

Example: Encryption of the letter **A** by Alice.

'**A**' is given in ASCII code as $65_{10} = 1000001_2$.

Let's assume that the first key stream bits are → $z_1, \ldots, z_7 = 0101101$

Encryption by Alice: plaintext m_i : 1000001 = '**A**' (ASCII symbol)
key stream z_i: 0101101
ciphertext c_i : 1101100 = 'l' (ASCII symbol)

Decryption by Bob: ciphertext c_i : 1101100 = 'l' (ASCII symbol)
key stream z_i : 0101101
plaintext m_i : 1000001 = '**A**' (ASCII symbol)

Theorem: The OTP is unconditionally secure if keys are only used once.

(3.3) Synchronous stream ciphers

Definition: A ***synchronous*** stream. Cipher is one in which the keystream is generated independently of the plaintext message and of the ciphertext.

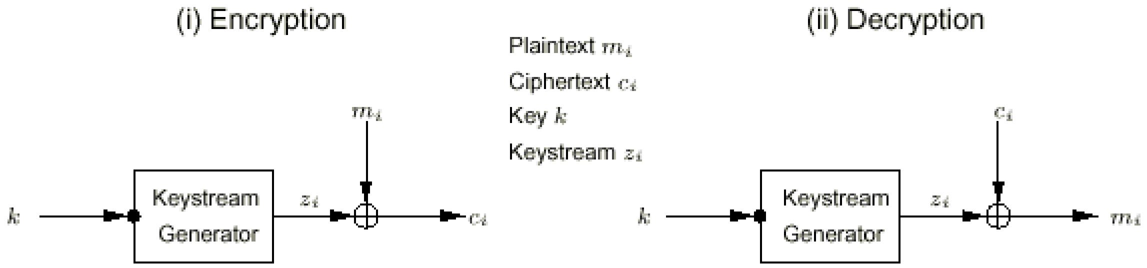


Figure 1: General model of a binary additive synchronous stream cipher.

properties of synchronous stream ciphers:

1. ***synchronization requirements.*** In a synchronous stream cipher, both the sender and receiver must be *synchronized* – using the same key and operating at the same position (state) within that key – to allow for proper decryption. If synchronization is lost due to ciphertext digits being inserted or deleted during transmission, then decryption fails and can only be restored through additional techniques for re-synchronization. Techniques for re-synchronization include re-initialization, placing special markers at regular intervals in the ciphertext, or, if the plaintext contains enough redundancy, trying all possible keystream offsets.
2. ***no error propagation.*** A ciphertext digit that is modified (but not deleted) during transmission does not affect the decryption of other ciphertext digits.
3. ***active attacks.*** As a consequence of property (1), the insertion, deletion, or replay of ciphertext digits by an active adversary causes immediate loss of synchronization, and hence might possibly be detected by the decryptor. As a consequence of property (2), an active adversary might possibly be able to make changes to selected ciphertext digits, and know exactly what affect these changes have on the plaintext.

Most of the stream ciphers that have been proposed to date in the literature are additive stream ciphers, which are defined below.

(3.4) Self-synchronizing stream ciphers

Definition: A ***self-synchronizing*** or ***asynchronous*** stream cipher is one in which the keystream is generated as a function of the key and a fixed number of previous ciphertext digits.

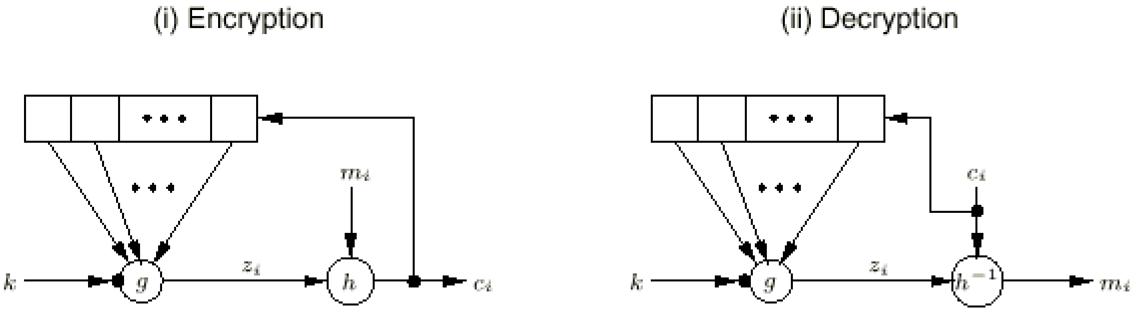


Figure 2: General model of a self-synchronizing stream cipher.

properties of self-synchronizing stream ciphers

1. ***self-synchronization.*** Self-synchronization is possible if ciphertext digits are deleted or inserted, because the decryption mapping depends only on a fixed number of preceding ciphertext characters. Such ciphers are capable of re-establishing proper decryption automatically after loss of synchronization, with only a fixed number of plaintext characters unrecoverable.
2. ***limited error propagation.*** Suppose that the state of a self-synchronization stream cipher depends on t previous ciphertext digits. If a single ciphertext digit is modified (or even deleted or inserted) during transmission, then decryption of up to t subsequent ciphertext digits may be incorrect, after which correct decryption resumes.
3. ***active attacks.*** Property (2) implies that any modification of ciphertext digits by an active adversary causes several other ciphertext digits to be decrypted incorrectly, thereby improving (compared to synchronous stream ciphers) the likelihood of being detected by the decryptor. As a consequence of property (1), it is more difficult (than for synchronous stream ciphers) to detect insertion, deletion, or replay of ciphertext digits by an active adversary.
4. ***diffusion of plaintext statistics.*** Since each plaintext digit influences the entire following ciphertext, the statistical properties of the plaintext are dispersed through the ciphertext. Hence, self-synchronizing stream ciphers may be more resistant than synchronous stream ciphers against attacks based on plaintext redundancy.

(3.5) Feedback shift registers

Linear feedback shift registers (LFSRs) are used in many of the keystream generators that have been proposed in the literature. There are several reasons for this:

1. LFSRs are well-suited to hardware implementation.
2. They can produce sequences of large period.

3. They can produce sequences with good statistical properties.
4. Because of their structure, they can be readily analyzed using algebraic techniques.

Definition: A ***linear feedback shift register*** (LFSR)of length L consists of L *stages* (or *delay elements*) numbered 0, 1, …, L-1, each capable of storing one bit and having one input and one output; and a clock which controls the movement of data. During each unit of time the following operations are performed:

(i) The content of stage 0 is output and forms part of the *output sequence*.

(ii) The content of stage i is moved to stage i-1 for each i,$1 \leq i \leq L-1$.

(iii) The new content of stage L-1 is the *feedback bit* s_j which is calculated by adding together modulo 2 the previous contents of a fixed subset of stages 0,1, … ,L-1.

Figure 3 depicts an LFSR. Referring to the figure, each c_i is either 0 or 1; the closed semi-circles are AND gates; and the feedback bit s_j is the XOR of the contents of those stages i, $1 \leq i \leq L-1$, for which c_{L-i} = 1.

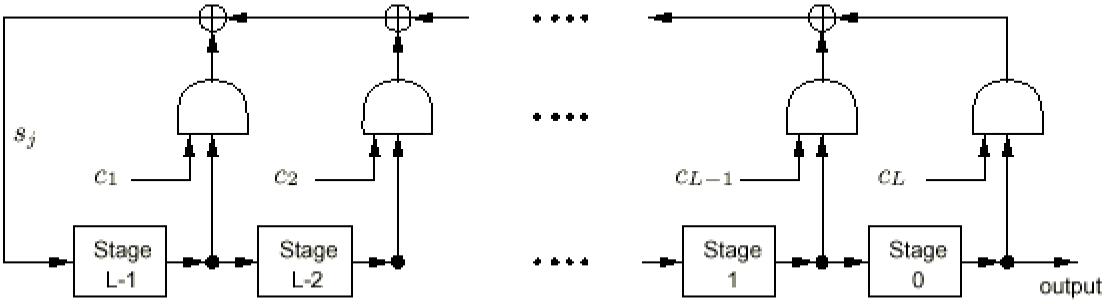


Figure 3: A linear feedback shift register (LFSR) of length L.

Definition: The LFSR of Figure 3 is denoted $\langle L, C(D) \rangle$, where C(D) = $1+c_1D+c_2D^2+\ldots+c_LD^L$ $\in Z_2[D]$ is the ***connection polynomial***. The LFSR is said to be ***non-singular*** if the degree of C(D) is L (that is, c_L=1). If the initial content of stage i is $s_i \in \{0,1\}$ for each i, $1 \leq i \leq L-1$, then [s_{L-1}, … , s_1, s_0] is called the ***initial state*** of the LFSR.

Fact I: The initial state of the LFSR in Figure 3 is [s_{L-1}, … , s_1, s_0], then the output sequence s = s_0, s_1, … is uniquely determined by the following recursion:

$$s_j = (c_1\, s_{j-1} + c_2\, s_{j-2} + \ldots + c_L\, s_{j-L}) \bmod 2 \text{ for } j \geq L.$$

Example: *(output sequence of an LFSR)* Consider the LFSR $\langle 4, 1+D+D^4 \rangle$ depicted in Figure 4. If the initial state of the LFSR is [0, 0, 0, 0], the output sequence is the zero sequence. The following tables show the contents of the stages D_3 , D_2 , D_1 , D_0 at the end of each unit of time t when the initial state is [0, 1, 1, 0].

t	D3	D2	D1	D0
0	0	1	1	0
1	0	0	1	1
2	1	0	0	1
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	1	0	0	0
7	1	1	0	0

t	D3	D2	D1	D0
8	1	1	1	0
9	1	1	1	1
10	0	1	1	1
11	1	0	1	1
12	0	1	0	1
13	1	0	1	0
14	1	1	0	1
15	0	1	1	0

The output sequence is s = 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, … , and is periodic with period 15.

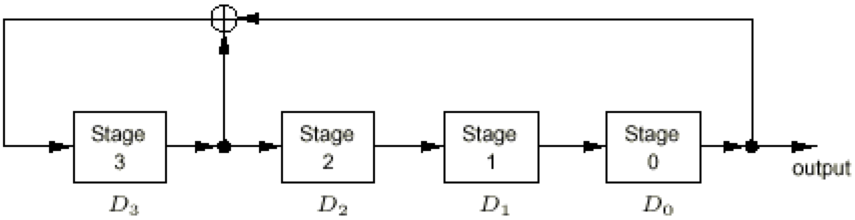


Figure 4: *The LFSR* $\langle 4, 1+D+D^4 \rangle$

Fact I: Every output sequence (i.e., for all possible initial states) of an LFSR $\langle L, C(D) \rangle$ is periodic if and only if the connection polynomial C(D) has degree L.
If an LFSR $\langle L, C(D) \rangle$ is *singular* (i.e., C(D) has degree less than L), then not all output sequences are periodic. However, the output sequences are *ultimately periodic*; that is, the sequences obtained by ignoring a certain finite number of terms at the beginning are periodic. For the remainder of this chapter, it will be assumed that all LFSRs are non-singular. Fact II determines the periods of the output sequences of some special types of non-singular LFSRs.

Fact II: *(periods of LFSR output sequences)* Let C(D) $\in Z_2$[D] be a connection polynomial of degree L.
(i) If C(D) is irreducible over Z_2, then each of the 2^L-1 non-zero initial states of the non-singular LFSR $\langle L, C(D) \rangle$ produces an output sequence with period equal to the least positive integer N such that C(D) divides 1+D^N in Z_2[D]. (Note: it is always the case that this N is a divisor of 2^L-1.)
(ii) If C(D) is a primitive polynomial, then each of the 2^L-1 non-zero initial states of the non-singular LFSR $\langle L, C(D) \rangle$ produces an output sequence with maximum possible period 2^L-1.

Definition: If $C(D) \in Z_2[D]$ is a primitive polynomial of degree L, then $\langle L, C(D) \rangle$ is called a ***maximum-length*** LFSR. The output of a maximum-length LFSR with non-zero initial state is called an ***m-sequence***.

A binary message stream $M=m_1 m_2 \ldots$ is enciphered by computing:

$$c_i=m_i \oplus k_i$$

As the bits of the key stream are generated as shown in the following figure:

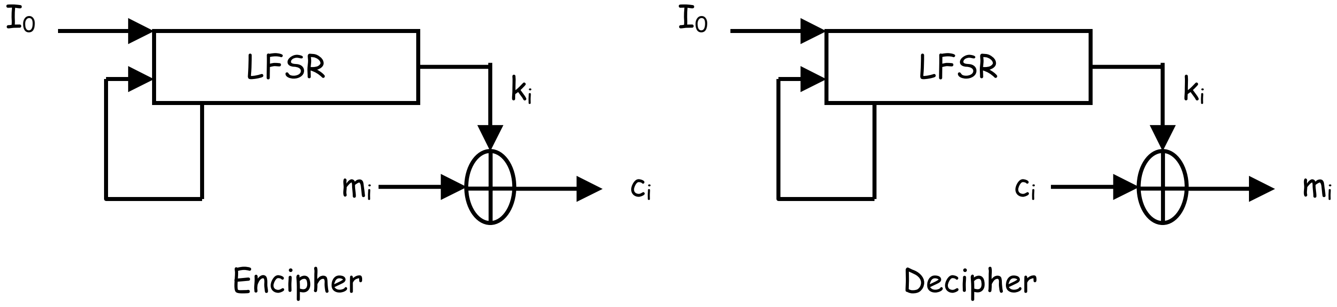


Figure 5: Encryption With LFSR

(3.6) Stream cipher algorithms:

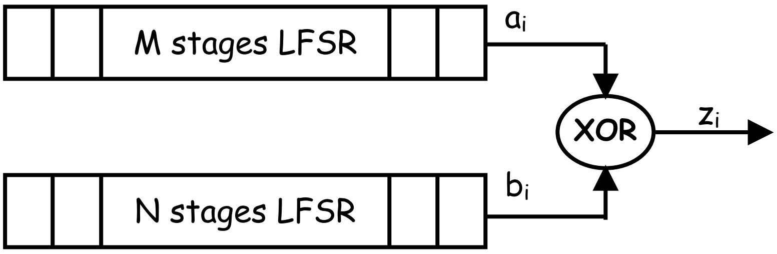
In this part, we'll discuss some of stream cipher algorithms. We'll explain those algorithms in details so that we can recognize their registers and also the type of connections or functions of connections.

(3.6.1) Exclusive-OR algorithm:

This algorithm consists of two linear feedback shift registers; each one has a linear feedback function, which will give the maximum period.

The length of these registers are different but has the property that the greatest common divisor between their length=1, i.e. let M and N equal the length of the shift registers, hence the gcd(M,N)=1.

The following figure will clarify this algorithm:



$$z_i=a_i \oplus b_i$$

Figure 6: XOR system

The output of the algorithm is:

$$Z=A \oplus B=\overline{A}\,B+A\overline{B}\ .$$

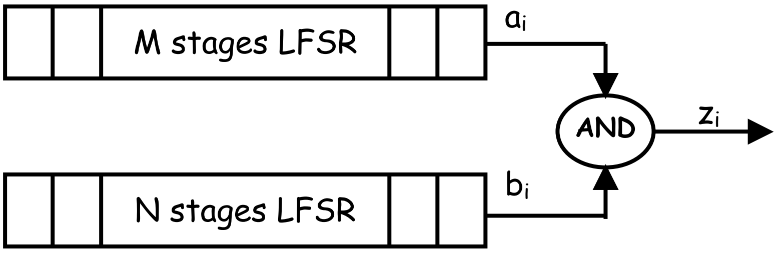
The input parameters for this system are as follows:

1- The no. of shift registers.
2- The length of shift registers.
3- The linear feedback function and the length of the series to be generated.
4- For each shift register:
    a. The linear feedback function applied as usual.
    b. The final result of the series for both shift registers applied to the XOR operation.

The above points repeated many times according to the length of our series to be enciphered.

(3.6.2) Hadamard algorithm:

This algorithm is look like the XOR algorithm but the only difference between them is that the combining function will be changed to AND. Figure 7 will explain the algorithm:



$z_i = a_i\ b_i$

Figure 7: Hadamard system

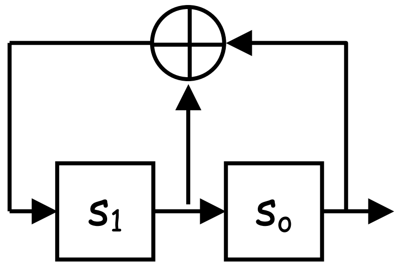
When the gcd(M,N)=1, the period length of the final sequence is $(2^M-1)(2^N-1)$, which is the maximum period.

Note: we can use the OR operation instead of AND, and the equation will be

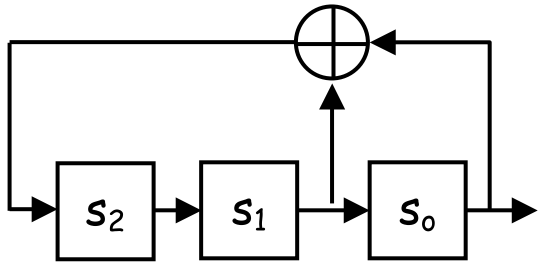
$$z_i = a_i + b_i + a_i\,b_i.$$

Example: We have two linear feedback shift registers with 2 and 3 stages respectively, and the corresponding connection polynomials are $C_1(D)=1+D+D^2$ and $C_2(D)=1+D^2+D^3$, with initial states [1,1] and [1,1,1] respectively. Apply the Hadamard algorithm to find the resulting sequence.

$C_1(D)=1+D+D^2 \Rightarrow s_0+s_1.$



$C_2(D)=1+D^2+D^3 \Rightarrow s_0+s_1$.



	LFSR 1		
T	S_1	S_2	a_i
0	1	1	
1	0	1	1
2	1	0	1
3	1	1	0
4			
5			
6			
7			
Max per.	2^2-1=3		
Output	110110110110…		

LFSR 2			
S_2	S_1	S_0	b_i
1	1	1	
0	1	1	1
0	0	1	1
1	0	0	1
0	1	0	0
1	0	1	0
1	1	0	1
1	1	1	0
2^3-1=7			
11100101110010…			

Since gcd(3,7)=1, hence the period of the resulting sequence =3X7=21.

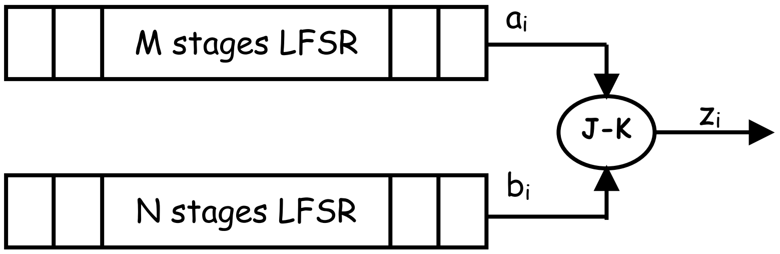
A=	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0
B=	1	1	1	0	0	1	0	1	1	1	0	0	1	0	1	1	1	0	0	1	0
Z=	1	1	0	0	0	0	0	1	0	1	0	0	1	0	0	1	1	0	0	1	0

Note: approximately, three to four from the results are zeros, because of the AND operation. So, the result of the algorithm does not satisfy the first randomness postulate.

(3.6.3) J-K flip-flop algorithm:

In this algorithm, the combining function will be replaced by a J-K flip-flop; hence the final result will be given by the following equation:

$$z_i = (a_i + b_i + 1)\ z_{i-1} + a_i$$



$z_i = (a_i + b_i + 1)\ z_{i-1} + a_i$

Figure 8: J-K flip-flop system

The truth table for J-K flip flop is:

J	K	z_{i+1}
0	0	z_i
0	1	0
1	0	1
1	1	$\overline{z}_i$

(3.6.4) Geffe's algorithm:

The system here consists of three linear feedback shift registers connected as shown in figure 9. The length of the shift registers are M, N, and L, where the gcd(M,N,L)=1. The equation that will use here is:

$$z_i = a_i\, b_i + b_i\, c_i + c_i$$

The keystream generated has period $(2^M-1)(2^N-1)(2^L-1)$.

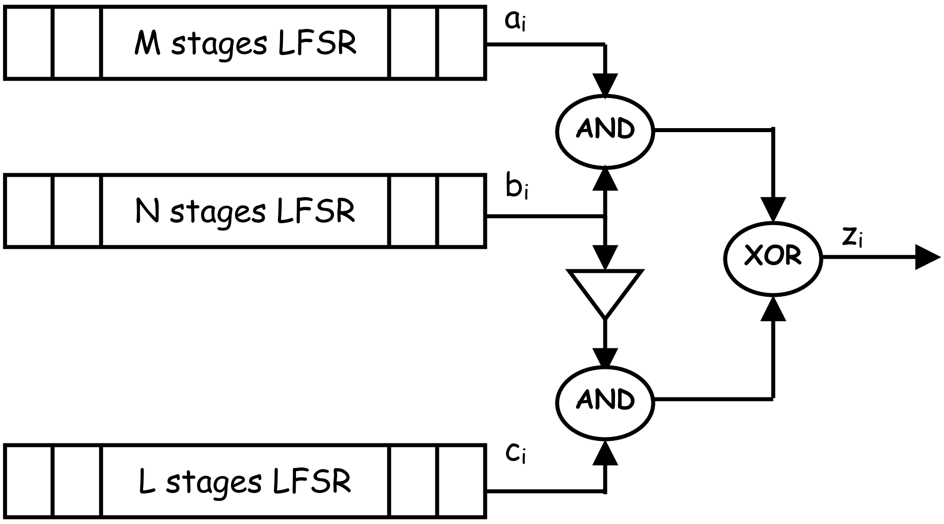


Figure 9: Geffe's system

(3.6.5) Police algorithm:

This system consists of three linear feedback shift registers of lengths M, N, and L, where gcd(M,N,L)=1. The second shift register connected to a police that control the output of the other two shift registers, see figure 10. If b_i=0 then z_i=a_i, and if b_i=1 then z_i=c_i.

The equation that describe this systems is:

$$z_i=a_i\ (b_i+1) + c_i\ b_i$$

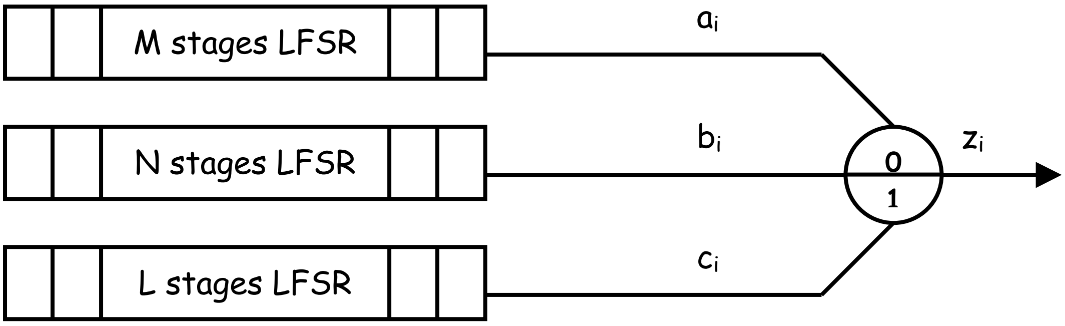
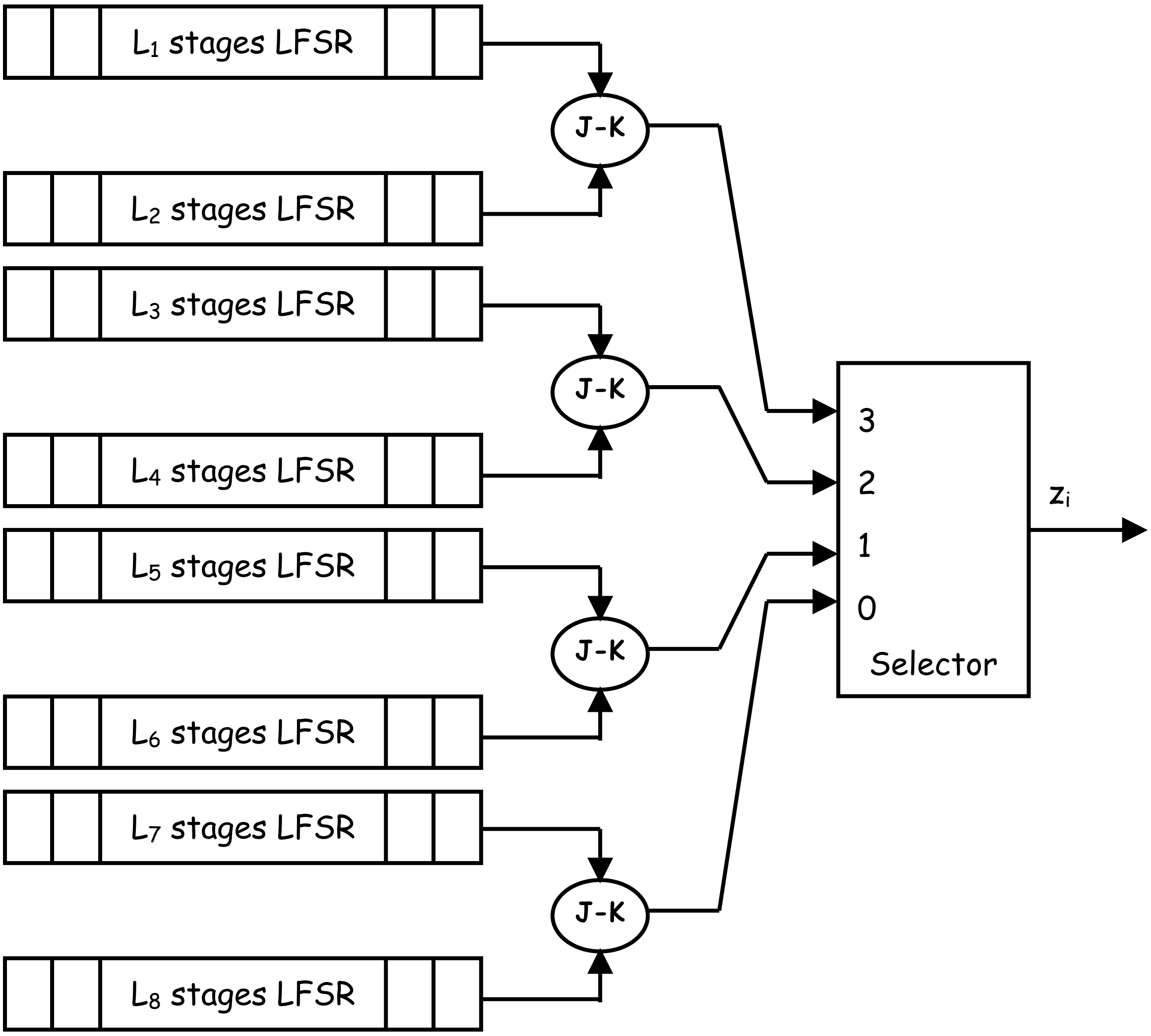


Figure 10: Police system

(3.6.5) Pless's algorithm:

This system consists of eight linear feedback shift registers of deferent lengths. These shift registers are put in four pairs, the gcd of each pair is equal to one. There are four J-K flip-flops and a recycling clock. Each time the recycling clock will choose one bit from the four arrival bits, see figure 11.



47Figure 11: Pless's system

Exercises

Attempt all the following exercises

Q1) You have two linear feedback shift registers with 3 and 5 stages respectively, and the corresponding connection polynomials are $C_1(D)=1+D+D^3$ and $C_2(D)=1+D^3+D^5$, with initial states [1,0,1] and [1,0,0,1] respectively. Apply the J-K flip-flop algorithm to find the resulting sequence.

Q2) You have three linear feedback shift registers with 4, 5 and 3 stages respectively, and the corresponding connection polynomials are $C_1(D)=1+D^2+D^4$, $C_2(D)=1+D+D^5$, and $C_3(D)=1+D^2+D^3$, with initial states [1,1,1,1] , [1,0,1,0,1], and [1,1,0] respectively. Apply the Geffe's algorithm to find the resulting sequence.

Q3) For (Q2), apply the police algorithm to find the resulting sequence.

Q5) Prove that $z_i = (a_i + b_i + 1)\, z_{i-1} + a_i$, represent the result of the J-K flip-flop.

Q6) Prove that $z_i=a_i\,(b_i+1) + c_i\, b_i$, represent the result of the Police algorithm.

Q7) Prove that $z_i = a_i\, b_i + b_i\, c_i + c_i$, represent the result of the Geffe's algorithm.

Q4) In Pless's algorithm, there are 4 shift registers pairs with the following output:

P1 = 0 0 1 0 1 0 1 0 0 1 1 0 0 1 1 0 1 1 0 …

P2 = 0 0 1 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 1 …

P3 = 0 0 1 0 0 1 1 0 1 0 1 1 0 1 1 0 1 0 1 0 1 …

P4 = 1 0 1 0 1 0 1 0 0 0 1 1 1 0 1 1 0 1 1 1 0 …

If the initial of the recycling clock is 2, what are the first 10 bits of the resulting sequence?

CHAPTER FOUR
BLOCK CIPHERS

(4.1) Introduction

A block cipher is a function which maps n-bit plaintext blocks to n-bit ciphertext blocks; n is called the ***block length***. It may be viewed as a simple substitution cipher with large character size. The function is parameterized by a k-bit key , taking values from a subset $\mathcal{K}$ (the key space) of the set of all k-bit vectors V_k. It is generally assumed that the key is chosen at random.

Definition: An n-bit block cipher is a function E: $V_n \times \mathcal{K} \rightarrow V_n$, such that for each key $k \in \mathcal{K}$, E(P;k) is an invertible mapping (the encryption function for k) from V_n to V_n, written $E_k(P)$. The inverse mapping is the decryption function, denoted $D_k(C)$. $C=E_k(P)$ denotes that ciphertext C results from encrypting plaintext P under K.

(4.2) DES

The Data Encryption Standard (DES), known as the Data Encryption Algorithm (DEA) by ANSI (American National Standards Institute) and the DEA-1 by the ISO (International Standards Organization), has been a worldwide standard for over 20 years.

The algorithm was introduces by IBM team cryptographers on 15, March 1973, to meet the following criteria:

- The algorithm must provide a high level of security.
- The algorithm must be completely specified and easy to understand.
- The security of the algorithm must reside completely on the key; the security should not depend on the secrecy of the algorithm,
- The algorithm must be available to all users.
- The algorithm must be economically implementable in electronic devices.

Although it was presented by IBM in 1974 it was not adopted as a federal standard until 23, November 1976. This delay was caused by the time given to the public and the NSA (National security Agency) to check the algorithm and make sure it has no trapdoors or possible weaknesses.

The standard was re-assessed every five years ever since. It was until 1998 that it was doubted- only doubted- that the algorithm was likely to be broken soon. So a need for another, stronger algorithm appeared.

(4.2.1) The algorithm

DES is very much a traditional cipher in the sense that it employs the traditional cryptographic methods of transposition and substitution, but it is designed to interrelate them in such a way so to produce a ciphertext of such complexity that a brute-force key search is, at least in theory, the only- or at least the quickest- way to attack it successfully. A 56-bit key gives a search space of 2^{56}=7.2*10^{16}.

(4.2.1.1) The encryption process

The encryption process consists of two parts, one dealing with the text, and another dealing with the key.

Part One: Processing the key

To begin with, 64 key bits are fed to Permuted-Choice-1 (PC-1) where the eight parity bits (8, 16, 24,…, 64) are discarded to produce an initial 56-bit key K as shown in Figure 1. PC-1 is a single permutation but it is effectively divided into two 28-bit sub-blocks. The first 28 bits of K are usually designated C_0 and the last 28 bits D_0. The two 28-bit sub-blocks are then shifted one place to the left so that e.g. the bit in the first position of C_0 moves to the last position of C_0, the bit in the second position of C_0 moves into the first position of C_0 and so on. The left shifting of C_0 and D_0 produce a re-ordering of each and the results are designated C_1 and D_1 respectively.

C_0

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

D_0

63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Figure 1: Permuted-Choice-1 (PC1).

The 56 bits of C_1 and D_1, taken as a single block, are then sent to second permutation, Permuted-Choice-2 (PC-2). This is shown in Figure 2.

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Figure 2: Permuted-Choice-2 (PC2).

The 48 bits output from PC-2 is the first iteration K_1 of the key. K_1 then enters the main algorithm.

On the second round, C_1 and D_1 are shifted one place to the left to produce C_2 and D_2. The 56-bit block C_2D_2 enters PC-2 to produce the second 48-bit iteration of the key K_2 and so on for each iteration of the key. A block diagram of the key generation process is shown in Figure 3.

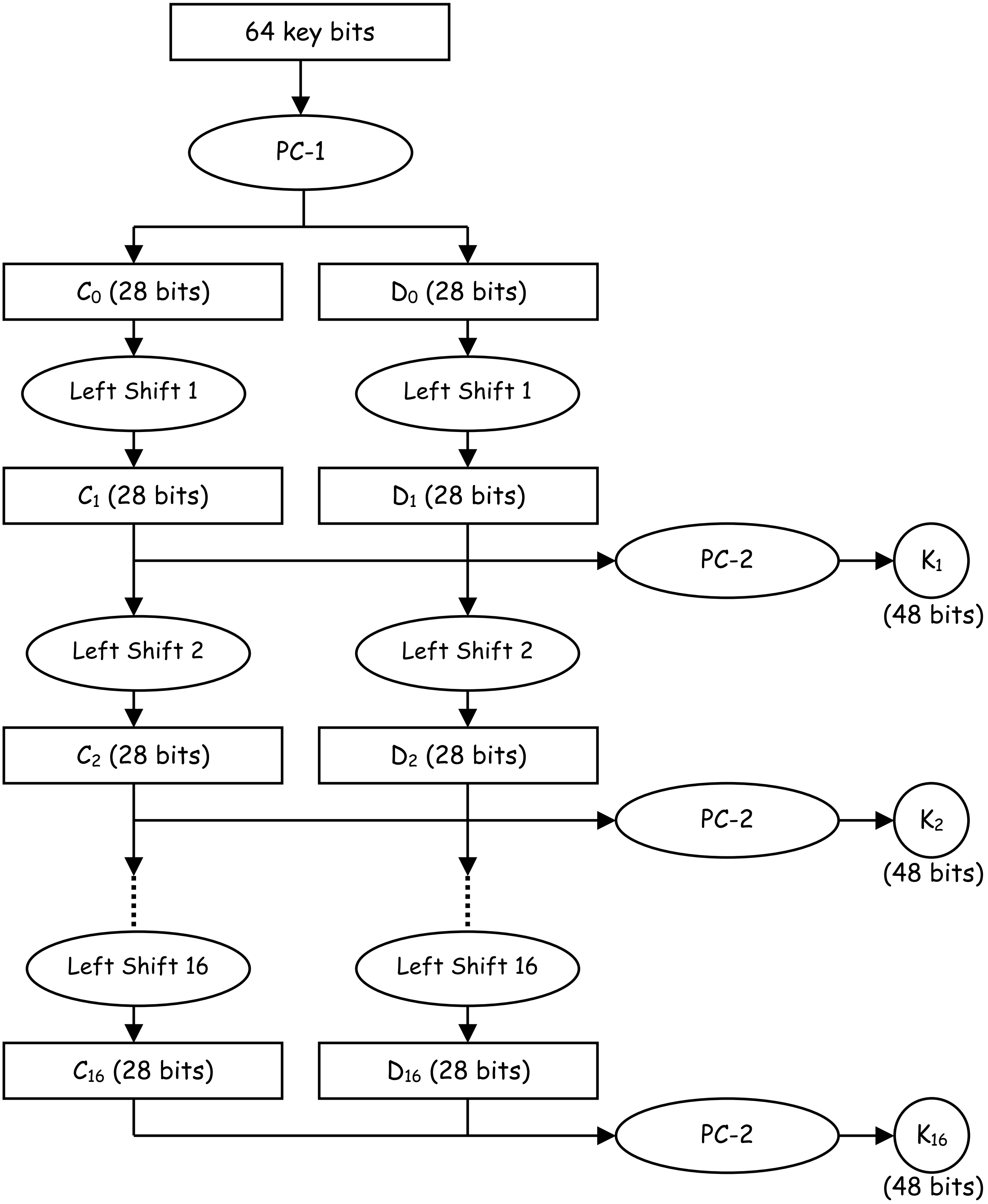


Figure 3: The DES key generation process.

For the full 16 rounds of DES, the schedule of left shifts is:

Key iteration	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Left Shift	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Algorithm 4.1: DES key schedule

INPUT: 64-bit key $K = k_1 \ldots k_{64}$ (including 8 odd-parity bits).

OUTPUT: sixteen 48-bit keys K_i, $1 \le i \le 16$.

1. Define v_i, $1 \le i \le 16$ as follows: $v_i = 1$ for $i \in \{1, 2, 9, 16\}$; $v_i = 2$ otherwise. (These are left-shift values for 28-bit circular rotations below).
2. $T \leftarrow PC(K)$; represent T as 28-bit halves (C_0, D_0). (Use PC1 Table to select bits from K: $C_0 = k_{57}\, k_{49} \ldots k_{36}$, $D_0 = k_{63}\, k_{55} \ldots k_4$).
3. For i from 1 to 16, compute K_i as follows: $C_i \leftarrow (C_{i-1} ↵ v_i)$, $D_i \leftarrow (D_{i-1} ↵ v_i)$, $K_i \leftarrow PC2(C_i, D_i)$. (Use PC2 Table to select 48 bits from the concatenation $b_1\, b_2 \ldots b_{56}$ of C_i and D_i: $K_i = b_{14}\, b_{17} \ldots b_{32}$. '↵' denotes left circular shift.)

Part Two: Processing the text

DES takes a 64-bit block of permuted ASCII plaintext, breaks it into two parts and processes the two 32-bit halves for 16 iterations (rounds) of a complex product algorithm to produce 64 bits of ciphertext. Full encryption thus lies in repeatedly enciphering each 64-bit ciphertext block until the entire message has been converted into ciphertext.

To begin with, the 64 bits of plaintext are subjected to the Initial Permutation (IP) and the output enters the main algorithm. The IP is implemented as shown in Figure 4.

IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

IP^{-1}							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Figure 4: DES initial permutation and inverse (IP and IP^{-1}).

The permutation tables are read from left to right in descending rows. The permuted text is called the initial text or T_0 and is split into two equal parts to form a 32-bit left-hand half (L_0) and a 32-bit right-hand half (R_0). In the IP table shown above, the plaintext bytes are sorted into columns so that the odd numbered bits are in the lower half, which is R_0, and the even numbered bits are in the upper half, which is L_0.

The 32 bits of R_0 now enter a four-stage function. The first step of the function is the E-bit selection table shown in Figure 5.

E					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Figure 5: DES per-round functions: expansion E and permutation P.

At this step, R_0 is expanded from 32 bits to 48 bits of R_0. Note the duplication of 16 bit positions in the E-table. It is this, which gives the expansion of the 32-bit R_0 into a new 48-bit block. The 48 bits of R0 are now XORed with the 48 bits of the first iteration of the key k_1. This XORing happens in a bit-by-bit form. The resulting block of 48 bits is now broken up into eight, six-bit sub blocks. Each sub-block moves into its assigned selection function S_1, S_2, …, S_8 respectively. The output is eight, four-bit sub blocks in the range from 0000 to 1111 which then combine to yield a 32-bit block (see Figure 6).

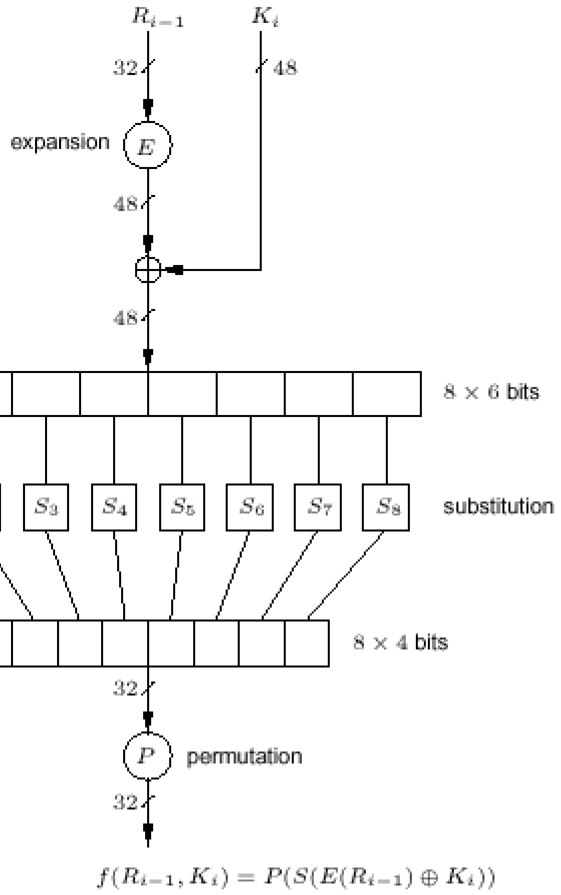
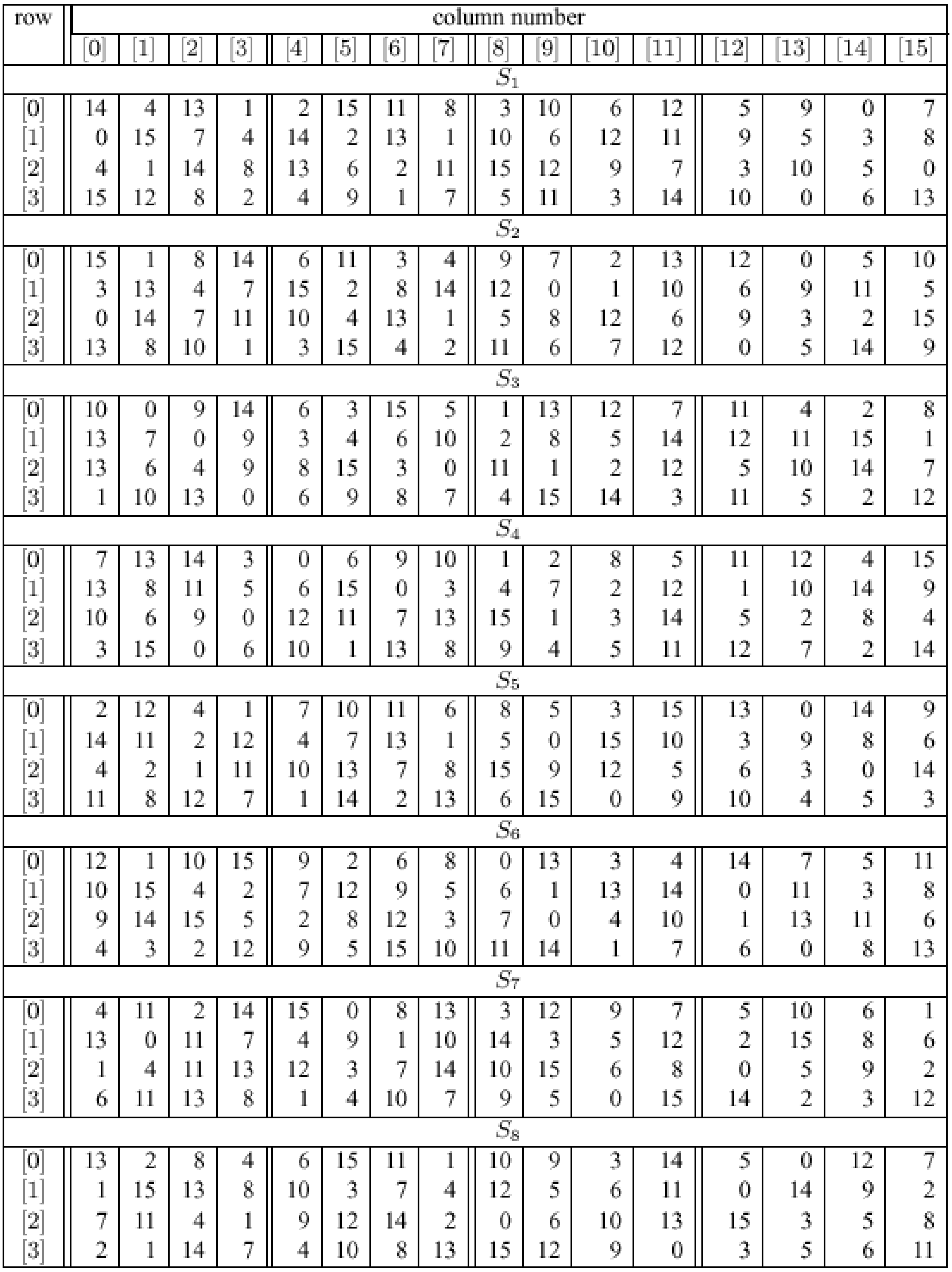


Figure 6: DES inner function f.

As they are non-linear, the selection functions (sometimes called the S-Boxes) lie at the core of DES security and they are set out in Figure 7.



row	column number															
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]
S_1																
[0]	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
[1]	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
[2]	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
[3]	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2																
[0]	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
[1]	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
[2]	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
[3]	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3																
[0]	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
[1]	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
[2]	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
[3]	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4																
[0]	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
[1]	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
[2]	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
[3]	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5																
[0]	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
[1]	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
[2]	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
[3]	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6																
[0]	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
[1]	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
[2]	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
[3]	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7																
[0]	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
[1]	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
[2]	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
[3]	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8																
[0]	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
[1]	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
[2]	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
[3]	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Figure 7: The S-Boxes of DES.

If for example the first six-bit sub-block is 001011, then the first and the sixth bits determine the row in S_1 while the second to the fifth bits determine the column in S_1.

The number (in binary) at the intersection of the designated row and column is the output. In our example, the first and sixth bits are 0 and 1, which is decimal 1, while the second to fifth bits are 0101 equal 5. Thus, the substitution value is found in the intersection of row 1 and column 5 of S_1 and 0010 (2) is the assigned replacement for 001011. Were 001011 input to S_5, however, the output would be 0111 (7).

Having left the selection boxes, the new 32-bit block of R_0 is now transposed by the permutation P (Figure 5), to give the new 32-bit block of R_0. This is the last stage of the four-step function on R_0. This four-step function is thus a highly complex product sub-algorithm. To sum up; the 32-bit R_0 is expanded by the E-table to 48 bits and the result is XORed with a 48-bit iteration of the 56-bit key. The 48-bit outcome is broken up into eight sub-blocks and each is substituted in selection boxes. The resulting eight, four-bit, sub-blocks are recombined to give a 32-bit block that is fed into the permutation P.

Up to now, the 32-bit left-side block L_0 has not played any part in the encryption procedure. This is remedied by moving L_0 over to the right where it is XORed with the 32-bit result of permutation P. The outcome is the new 32-bit right-side block.
The original R_0 moves leftwards to become L_1. This completes the first DES iteration with a 64-bit text T_1 consisting of a 32-bit left-hand block L_1 and a 32-bit right-hand block R_1 such that $T_1=L_1R_1$.

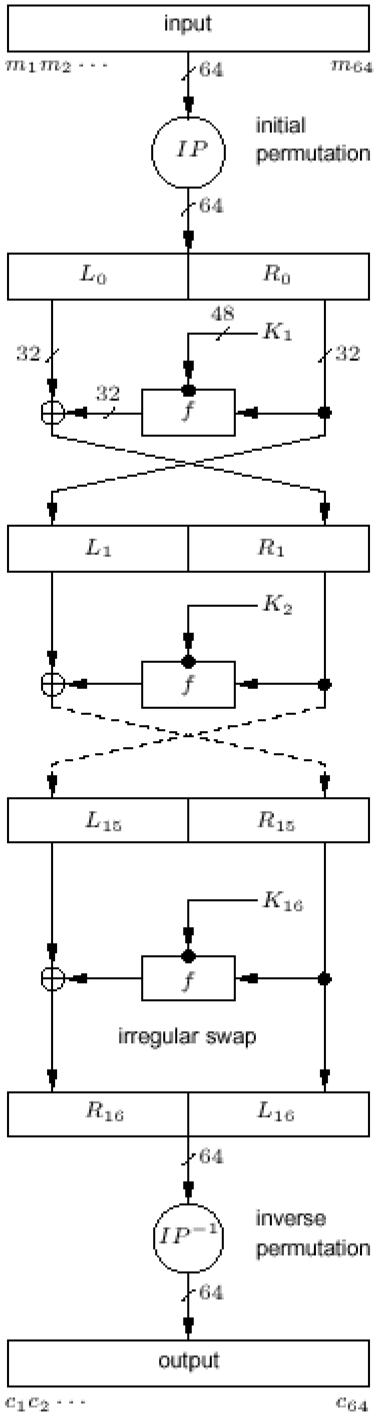
The 32-bit block R_1 now enters the four-step function for the second round of encryption.
Each round is functionally equivalent; taking 32-bit input L_{i-1} and R_{i-1} from the previous round and producing 32-bit outputs L_i and R_i for $1\leq i \leq 16$, as follows:

$$L_i = R_{i-1}; \tag{4.1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i), \text{ where } f(R_{i-1}, K_i)= P(S(E(R_{i-1})\oplus K_i)) \tag{4.2}$$

In the last iteration, however, the left- and right-hand halves do not exchange places. They are combined and sent into the inverse IP^{-1} of the initial permutation IP.

The output is the final ciphertext and the process encrypts eight bytes of the message.
The next eight bytes of plaintext then enter into the algorithm for their 16-round encryption and the process continues until the entire message is has been enciphered. Figure 8 shows the main step of the DES algorithm.



$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

Figure 8: DES computation path.

Algorithm 4.2: Data Encryption Standard (DES)

INPUT: plaintext $m_1 \ldots m_{64}$; 64-bit key $K = k_1 \ldots k_{64}$ (includes 8 parity bits).

OUTPUT: 64-bit ciphertext block $C = c_1 \ldots c_{64}$.

1. (key schedule) Compute sixteen 48-bit round keys K_i from K using Algorithm 4.1.
2. $(L_0, R_0) \leftarrow IP(m_1\ m_2 \ldots m_{64})$. (Use IP Table in Figure 4 to permute bits; split the result into left and right 32-bit halves $L_0 = m_{58}\ m_{50} \ldots m_8$, $R_0 = m_{57}\ m_{49} \ldots m_7$).
3. (16 rounds) for i from 1 to 16, compute L_i and R_i using Equations (4.1) and (4.2) above, computing $f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$ as follows:
    a. Expand $R_{i-1} = r_1\ r_2 \ldots r_{32}$ from 32 to 48 bits using E per Table in Figure 5: $T \leftarrow E(R_{i-1})$. (Thus $T = r_{32}\ r_1\ r_2 \ldots r_{32}\ r_1$).
    b. $T^{'} \leftarrow T \oplus K_i$. Represent $T^{'}$ as eight 6-bit character strings: $(B_1 , \ldots , B_8)=T^{'}$.
    c. $T^{''} \leftarrow (S_1(B_1), S_2(B_2), \ldots , S_8(B_8))$. (Here $S_i(B_i)$ maps $B_i = b_1\ b_2 \ldots b_6$ to the 4-bit entry in row r and column c of S_i in Figure 7 , where $r = 2 \bullet b_1 + b_6$, and $b_2 b_3 b_4 b_5$ is the radix-2 representation of $0 \leq c \leq 15$. Thus $S_1(011011)$ yields r = 1, c = 13, and output 5, i.e., binary 0101).
    d. $T^{'''} \leftarrow P(T^{''})$. (Use P per Table in Figure 5 to permute the 32 bits of $T^{''} = t_1\ t_2 \ldots t_{32}$, yielding $t_{16}\ t_7 \ldots t_{25}$).
4. $b_1\ b_2 \ldots b_{64}\ (R_{16},\ L_{16})$. (Exchange final blocks L_{16}, R_{16}).
5. $C \leftarrow IP^{-1}\ (b_1\ b_2 \ldots b_{64})$. (Transpose using IP^{-1} from Table in Figure 4; $C = b_{40}\ b_8 \ldots b_{25}$).

(4.2.1.2) The decryption process

The substitution and transposition of the Des seem like numbers chosen at random; there is no apparent pattern to the table describing the various changes. However, the change, which were chosen with extreme care, produce a surprising but intended result. The same DES algorithm is used both for encryption and decryption.

This result is true because cycle (i) drives from cycle (i-1) in the following manner:

$$L_i = R_{i-1},$$

$$R_i = L_{i-1} \oplus f(R_{i-1},K_i)$$

Where $\oplus$ is the exclusive-or operation, and f is the function computed in an expand shift substitute permute cycle. These two equations show that the result of each cycle depends only on the previous cycle.

By rewriting these equations in terms of R_{i-1} and L_{i-1}, we get

$$R_{i-1} = L_i,$$

and

$$L_{i-1} = R_i \oplus f(R_{i-1},K_i) = R_i \oplus f(L_i,K_i).$$

The last two equations show that these same values could be obtained from the results of later cycles. It is this property that makes the DES a reversible procedure; we can encrypt a string and also decrypt the result to derive the plaintext again.
With the DES it is possible to go forward and encrypt or to go backwards and decrypt using the same function f. the only change is that the keys must be taken in reverse order (K_{16}, K_{15},... ,K_1) for decryption. That one algorithm can be used either to encrypt or decrypt is very convenient for a hardware or software implementation of the DES.

(4.3) Practical example

To get a slightly better idea of what goes on, it is easy to take 64 bits of plaintext through 'one round' of a DES.

Let the plaintext be (**Caligula**) and the key (**Claudius**), each of them made up of eight letters and will thereby encrypt in a single block.

The ASCII character of (**Claudius**), may be used to generate the one-round, 47-bit key K_1. Thus, with a 'parity' bit inserted into the rightmost position, we have:

C= 67 =10000110 l=108=11011001 a=97 =11000010 u=117=11101010
d=100 =11001000 i=105=11010011 u=117=11101010 s=115= 11100110

The 64 bits are fed into the 56-element permutation PC-1,i.e.

Permuted-choice PC-1

1	1	1	1	1	1	1	1	1	1	0	1	1	0
1	1	1	1	1	1	1	1	1	0	0	0	0	0
1	0	1	1	0	0	1	0	1	0	1	0	1	1
0	0	0	0	0	1	0	0	1	0	0	0	1	0
			C_0							D_0			

The parity bits are 'lost' leaving two 28-bit subsets C_0 and D_0 which are then shifted one place to the left. The output is:

C_1 = 1111111111111101100100000101
D_1 = 1101101100000010101101000101

The 56-bit string C_1D_1, it is now sent to the permutation PC-2.This gives

Permuted-choice PC-2	1	1	1	0	1	1	
	1	1	0	1	0	1	
	0	0	1	1	1	1	
	1	1	0	1	1	1	K_1
	0	0	0	0	1	0	
	1	0	0	1	1	1	
	0	0	0	1	0	0	
	0	0	1	1	1	1	

and, reading off the rows in descending order, the output is the 48-bit key

K_1=111011110101001111110111000010100111000100001111

which now moves into the main algorithm.

Now we begin with the plaintext (**Caligula**), the ASCII codes

C=67 =01000011	A=97 =01100001	l=108=01101100	i=105=01101001
g=103=01100111	U=117=01110101	l=108=01101100	a=97 =01100001

for the eight characters of (**Caligula**) are (with zero in the leftmost position and taken as a bit string) sent to the initial permutation IP.

Initial Permutation IP	1	1	1	1	1	1	1	1	
	0	0	1	0	0	0	0	0	
	0	1	1	1	0	1	0	0	L_0
	1	0	1	1	1	0	1	1	
	0	0	0	0	0	0	0	0	
	1	1	1	1	1	1	1	0	
	0	1	0	0	1	1	0	0	R_0
	0	0	0	1	0	0	0	1	

Reading off the rows from top to bottom gives the 64-bit initial text T_0 which, in turn, is broken up into the two 32-bit blocks:

L_0=11111111001000000111010010111011

R_0=00000000111111100100110000010001

R_0 now proceeds into the 4-stage function f where it is first read into the E-table and expanded to the 48-bit block ${}^1R_0$.

$$\text{E-table expansion}\quad\begin{matrix}1&0&0&0&0&0\\0&0&0&0&0&1\\0&1&1&1&1&1\\1&1&1&1&0&0\\0&0&1&0&0&1\\0&1&1&0&0&0\\0&0&0&0&1&0\\1&0&0&0&1&0\end{matrix}\quad {}^1R_0$$

and ${}^1R_0$=100000000001011111111100001001011000000010100010.
Having left its E-table expansion, the 48-bit block ${}^1R_0$ meets and is XORed with the 48-bit key K_1 to yield ${}^2R_0={}^1R_0\oplus K_1$, i.e.

${}^1R_0$= 100000000001011111111100001001011000000010100010
K_1 = 111011110101001111110111000010100111000100001111
${}^2R_0$= 011011110100010000001011001011111111000110101101

The 48-bit block ${}^2R_0$ is now broken up into the eight, 6-bit sub-blocks r_1 to r_8. Each sub-block enters its appropriate substitution box S_1 to S_8 for the non-linear replacement:

Selection box

	1	2	3	4	5	6	7	8
Input	011011	110100	010000	001011	001011	111111	000110	101101
Output	0101	1100	0001	1111	0111	1101	1110	1000

In S_1, the first and sixth bits of r_1 (0 and 1) determine the row while the second, third, fourth and fifth bits determine the column. As binary 01 is decimal 1 and binary 1101 is decimal 13, the substitution value for r_1 is found in the intersection of the [1] row and the [13] column in S_1. The designated number is decimal 5 or binary 101 and, with a 0 added in the leftmost position, the 4-bit output block is ρ_1=0010. When the substitution are done for the seven remaining sub-blocks, the outputs is the eight 4-bit blocks ρ_1, ρ_2,... ,ρ_8 which combine to give:

${}^3R_0$= 01011100000111110111110 1 11101000

${}^3R_0$ now enters into the permutation P to give

$$\text{Permutation P}\quad\begin{matrix}1&0&1&1\\1&1&0&0\\0&1&0&1\\1&1&0&0\\1&0&1&1\\0&1&0&0\\1&1&0&1\\1&0&1&1\end{matrix}\quad {}^4R_0$$

and the output:

$$^4R_0 = f(R_0,K_1) = 10111100010111001011010011011011.$$

This completes the 4-stage function on R_0.

$^4R_0$ is now XORed with the as yet untouched left-half L_0 to obtain the right-hand block $R_1=L_0\oplus{}^4R_0$, i.e.

$$\begin{aligned} ^4R_0 &= 10111100010111001011010011011011 \\ L_0 &= 11111111001000000111010010111011 \\ R_1 &= 01000011011111001100000001100000 \end{aligned}$$

In a sense, R_0 has been converted into the complex pseudorandom bit string $^4R_0$ by means of the function f and the key K_1. This 'secondary key' then 'encrypts' L_0 as R_1 while the original R_0 becomes L_1.

Since this is a variant 'one-round' DES, the two 32-bit blocks R1 and L1 do not change position but form the 64-bit pre-output block T_1 which is:

T_1	
$R_1=L_0\oplus{}^4R_0$	$L_1=R_0$
01000011011111001100000001100000	00000000111111100100110000010001

Sending T_1 to the inverse permutation IP^{-1} gives:

									Bit position		ASCII ciphertext
Inverse initial Permutation	0	1	0	0	0	0	1	0	1-8	Output	B
	0	1	1	0	0	0	0	0	9-16		'
	0	0	1	1	1	0	0	0	17-24		8
	0	0	1	1	1	0	0	0	25-32		8
	0	0	1	1	0	0	1	0	33-40		2
	0	0	1	1	0	0	0	1	41-48		1
	0	1	1	1	1	1	0	1	49-56		}
	0	0	1	0	0	1	0	0	57-64		$

Reading off the rows yields (US keyboard) the ciphertext (B'8821}$).

(4.4) Questions about the security of DES:

(4.4.1) Key Length

The length of the key is the most serious objection raised. The key in the original IBM implementation was 128 bits, while the DES key is effectively only 56 bits long. The argument for a longer key centers around the feasibility of an exhaustive search for a key.

Given a piece of plaintext known to be enciphered as a particular piece of ciphertext the goal for the interceptor is to find the key under which the decipherment was done. This attack assumes that the same key will be used to encipher other (unknown) plaintext. Knowing the key will allow the interceptor to decipher intercepted ciphertext easily.

The attack strategy is the "brute force" attack Encipher the known plaintext with an orderly series of keys, repeating with a new key until the enciphered plaintext matches the known ciphertext. There are 2^{56} 56-bit keys. If it were possible 10 test one every 100 ms, the time to test all keys would be about 7.2 x 10^{15} sec., or about 228 million years. If the test took only 1 μs, then the total time for the search is (only!) about 2,280 years.

Even supposing the-test-time to be 1 ns, infeasible on current technology machines, the search time is still in excess of two years, working full time with no hardware or software failures!

Diffie and Hellman suggest a parallel attack. With a parallel design, multiple processors can be assigned the same problem simultaneously. If one chip, working at a rate of one key per microsecond, can check about 8.6x 10^{10} keys in one day, it would take 10^6 days to try all $2^{56} \approx 7 \times 10^{16}$ keys. However, 10^6 chips working in parallel at that rate could check all keys in one day.

One estimate of the cost of such a machine is $50 million. Assuming a “key shop” existed where people would bring their plaintext/ciphertext pairs to obtain keys, and assuming that there was enough business to keep this machine busy 24 hours a day for five years, the proportionate cost would be only about $20,000 per solution. As hardware costs continue to fall, the cost of such a machine becomes lower.

(4.4.2) Two keys give the effect of a 112-bit key

Although the key length is believed by most analysts to be long enough, some people are still unsure about its 56-bit length. There is a method for increasing the effective length of the key. The method requires no change to the algorithm itself, which is convenient in case the algorithm is to be implemented by a hardware device or in an unmodifiable piece of software.

Because there is considerable concern for the security available with only one 56-bit key, a reasonable approach may involve using two keys. If somehow an exhaustive search defeats one key, the second lock should double the time required to break in (or so the analogies to the physical world would imply.) Unfortunately, this is not quite so, Merkie argues that two 56-bit keys in series can be broken with a chosen plaintext attack in 2^{57}

tries, instead of the 2^{112} that would be expected. Therefore, the second encryption adds almost no security.

Tuchman counters that two keys used in a special way enhances security, Tuchmnn uses a technique invented by Matyas and Meyer for use-by IBM in encrypting master keys in some of their encryption systems. With two keys, K_1 and K_2, the sender encrypts with K_1, decrypts with K_2, and encrypts with K_1 again. The receiver decrypts with K_1, encrypts with K_2, and decrypts with K_1 again.

This approach is desirable for use with an automatic encrypting device (which might be either hardware or software). If the device expects two keys and the user wants to use only one, the user supplies K_1 twice. The device encrypts with K_1, decrypts with K_1 (which returns the original plaintext), and finally encrypts with K_1. In that way one device can produce both single and double encryptions.

(4.4.3) Weaknesses of the DES

There are known weaknesses of the DES, but these weaknesses are not believed to limit the effectiveness of the algorithm seriously.

(4.4.3.1) Complements

The first known weakness concern complements. (Throughout this discussion, "compliment" means "ones complement", the result obtained by replacing all 1s by 0s and 0s by 1s.) If the message is encrypted with a particular key, the complement of the encryption will be the encryption of the complement message under the complement key. Stated formally, let p represent a plaintext message and k a key, and let the symbol $\bar{x}$ mean the compliment of the binary string x. If c =DES(p, k) (meaning c is the DES encryption of p using key k), then $\bar{c} = DES(\bar{p}, \bar{k})$. Since most applications of encryption do not deal with complement messages, and since users can be warned not to use complement keys, this is not a serious problem.

(4.4.3.2) Weak keys

A second known weakness concerns choice of keys. Because the initial key is split into two halves, and the two haves arc independently shifted circularly, if the value being shifted is all 0s or all 1s, the key used for encryption in each cycle is the same as for all other cycles. Remember that the difference between encryption and decryption is that the key shifts are applied in reverse. Key shifts are right shifts and the number of positions shifted is taken from the bottom of the table up, instead of top down. But if the keys are al1 0s or all 1s anyway, right or left shifts by 0, 1, or 2 portions are all the same. For these keys, encryption is the same as decryption: c =DES(p,k), and p=DES(c,k). These keys are called "weak keys." The same thing happen, if one half of the key is all 0s and the other half is all 1s (see figure 9). Since these keys are known, they can simply be avoided, so this is not a serious problem

C_0	D_0
$\{0\}^{28}$	$\{0\}^{28}$
$\{1\}^{28}$	$\{1\}^{28}$
$\{0\}^{28}$	$\{1\}^{28}$
$\{1\}^{28}$	$\{0\}^{28}$

Figure 9: Four DES weak keys.

(4.4.3.3) Semi-weak keys

A third difficulty is similar: There arc identifiable pairs of keys that have more that one identical decryption. That is, there are two different keys k_1 and k_2 for which c=DES(p,k_1) and c =DES(p,k_2). This implies that k_1 can decrypt a message encrypted under k_2. These so-called "Semi-weak" keys are shown in Figure 10.

C0	D0	C0	D0
$\{01\}^{14}$	$\{01\}^{14}$	$\{10\}^{14}$	$\{10\}^{14}$
$\{01\}^{14}$	$\{10\}^{14}$	$\{10\}^{14}$	$\{01\}^{14}$
$\{01\}^{14}$	$\{0\}^{28}$	$\{10\}^{14}$	$\{0\}^{28}$
$\{01\}^{14}$	$\{1\}^{28}$	$\{10\}^{14}$	$\{1\}^{28}$
$\{0\}^{28}$	$\{01\}^{14}$	$\{0\}^{28}$	$\{10\}^{14}$
$\{1\}^{28}$	$\{01\}^{14}$	$\{1\}^{28}$	$\{10\}^{14}$

Figure 10: Six pairs of DES semi-weak keys (one pair per line).

Exercises

Attempt all the following exercises

Q1) In a DES algorithm you have:

C_1=1100101010111100011010101111
D_1=1100000011111100010100001111

Find k_3.

Q2) In a DES algorithm you have the initial key=Baghdadi, find the key of the first round k_1.

Q3) In a DES algorithm you have the initial key=Shankoti, find the key of the first round k_1.

Q4) In a DES algorithm you have:

C_8=1100101010111100011010101111
D_8=1100000011111100010100001111

Find k_{12}.

Q5) If the input to the S-Box is

110101010101010011001110100101001000110011110011

what is the output.

Q6) Use the one-round DES algorithm to encrypt (Get busy) using the key (Sean Paul).

Q7) In the 16th round of the DES algorithm you get:

R_{15}=10110111100000011010011110100011
L_{15}=01111100011100011010101100010101

and the result of the S-Box is

${}^3R_{15}$=00110011010100111100101101101011

what is the ciphertext.

Q8) The input to the inner function of the DES algorithm is:

R_7= 10111100101010001110100001001011

and

K_8= 0111110110111110110111100000001111101110010101010

what is the output [$f(R_7,k_8)$].

CHAPTER FIVE

PUBLIC KEY CRYPTOGRAPHY

(5.1) Modular arithmetic – cryptographer's mathematics

Introduction

Mod-arithmetic is the central mathematical concept in cryptography. Almost any cipher from the Caesar Cipher to the RSA Cipher use it. Thus, I will show you here how to perform Mod addition, Mod subtraction, Mod multiplication, Mod Division and Mod Exponentiation. It is a very easy concept to understand as you will see. Use this page as a reference page and open it whenever you encounter any mod-calculations or mod-terminology that leave questions behind.

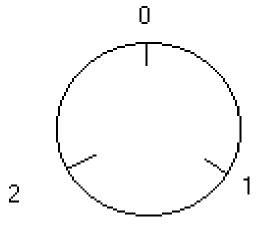
What is meant by Mod, Modulus and Modular Arithmetic?

"Modulus" (abbreviated as "mod") is the Latin word for "remainder, residue" or more in "what is left after parts of the whole are taken". Thus, "modular" or "mod arithmetic" is really "remainder arithmetic". More precise: We are looking for the integer that occurs as a remainder (or the "left-over") when one integers is divided by another integer. Let's do three examples:

<u>**Example:**</u> When 7 is divided by 3 it leaves a remainder of 1. Think of $1 as a left over after $7 are equally split among 3 people. Surely, there is a mathematical notation for mod arithmetic: Instead of writing 7 = 3*2 + 1 where 1 is the integer remainder we will write: 7 mod 3 = 1, which reads as: "7 modulo 3 is 1" and 3 is called the "modulus". Sure, this notation does not reveal the $2 that every person gets as his share. True, however, we are solely interested in the left over part, the remainder of $1 in our example.

<u>**Example:**</u> When 8 is divided by 3 it leaves a remainder of 2. Thus, we write: 8 mod 3 = 2.

<u>**Example:**</u> When 9 is divided by 3 it leaves no remainder. Thus, we write: 9 mod 3 = 0.



Arithmetic MOD 3
yields the answers
0, 1 or 2. I.e.:
1+2 = 0 mod 3
2+2 = 1 mod 3
10+2 = 0 mod 3
10+4 = 2 mod 3

Figure 1: Arithmetic MOD 3 can be performed on a clock with 3 different times: 0, 1 and 2.

Computations involving the modulus to determine remainders are called "Modular Arithmetic". It was first studied by the German Mathematician Karl Friedrich Gauss (1777-1855) in 1801. You may have heard this anecdote about Gauss when he went to school: His Mathematics teacher tried to keep the bored genius busy, so he asked him to add up the first 100 integers hoping that he would keep him quiet for a little while.

However, young Karl promptly responded "5050 and the formula for the sum of the first n integers is n*(n+1) / 2". Do you know why?

Great, we have the principle of Mod Arithmetic straight: To find the remainder simply divide the larger integer by the smaller integer. This surely works for large numbers as well: I.e. 365 MOD 7 = 1 (since 365 = 52*7 +1) .

What is the usage of Mod arithmetic?

365 MOD 7 = 1 tells us that if Christmas will fall on Thursday and we don't have a leap year it will fall on a Friday next year. The same for your birthday and any other day as well: every week day will fall on the following weekday the next year. Notice again that we only care about the remainder 1 and not the completed 52 weeks in a year. In fact if a year would consist of only 358 or 351 or 15 or 8 days, we would still have the same "shift by 1" effect. Apparently, solely the length of each week (called the modulus) determines the "shift by 1". What does 366 MOD 7 = 2 explain for leap years? (answer: Shift by 2 days).

Congruent numbers

Integers that leave the same remainder when divided by the modulus m are somehow similar, however, not identical. Such numbers are called "congruent" . For instance, 1 and 13 and 25 and 37 are congruent mod 12 since they all leave the same remainder when divided by 12. We write this as 1 = 13 = 25 = 37 mod 12. However, they are not congruent mod 13. Why not? Because it yield a different remainder when divided by 13.

Modular Arithmetic is also called Clock Arithmetic

The classical example for mod arithmetic is clock arithmetic: Look at the 12-hour clock in your room. You see 12 numbers on the clock. Here, the modulus is 12 with the twelve remainders 0,1,2,..11. So, when you give the time you actually give a remainder between 0 and 11. Again, the modulus m=12 is in charge of these reminders. What time is it 50 hours after midnight? It is 2 (a.m.) since 50 hours equal 2 full days and 2 hours.

In Modular Arithmetic, we add, subtract, multiply, divide and exponentiate as follows:

A) Mod Addition

Let's start simple: What time is it 10 hours after 11:00? It is 11+10 = 21 o'clock, and 21 minus the modulus 12 leaves a remainder of 9, thus 9 o'clock. What time is it 22 hours after 11:00? It is 11+22 = 33 and subtracting the modulus 12 repeatedly (which is also called "dividing") yields again 9. Ignoring a.m. and p.m., we are performing mod arithmetic on the clock. Let's write the two examples in mod notation: 11+10 = 21 mod 12 = 9 and 11 + 22 = 33 mod 12 = 9.

How to perform Mod Addition: First add the two numbers, secondly, divide the sum by the modulus to compute the remainder.

B) Mod Subtraction

Subtraction is performed in a similar fashion: First subtract, secondly compute the remainder.

Example: 25 - 8 = 17 MOD 12 = 5
Example: 50 - 11 = 39 MOD 12 = 3
What if we obtain a negative answer? Say it is 2 o'clock in New York, what time is it in L.A.? Turning the hand on a clock 3 hours backwards shows that it is 11 o'clock: 2 - 3 = -1 MOD 12 = 11
Thus, if the answer is negative, add the modulus you get a positive number. That number must be between 0 and the modulus.
Example: 3 - 50 = -47 MOD 12 = 1 since - 1 + 12 =11.
Example: 14 - 77 = -63 MOD 12 = 9 since -63 + 12 + 12 + 12 + 12 + 12 + 12 = 9.
Example: 11 - 50 = -39 MOD 15 = 6 since -39 + 15 + 15 + 15 = 6

C) Mod Multiplication

Since multiplication of positive numbers is repeated addition it can be reduced to the above mod addition.
How do we compute 5 * 8 MOD 12? First we multiply: 5 * 8 = 40, secondly we find the remainder: 40 MOD 12 = 4.

A useful shortcut:
*A mod expert would find the answer to 123 * 62 mod 12 immediately. It is 6. How does she know? Without being a Gauss genius, she computes 123 mod 12 = 3 and 62 mod 12 = 2 and multiplies those two answers.* To verify this: 123*62 mod 12 = 7626 mod 12 = 6. This computation aid is true for addition and subtraction as well. Therefore, we may write them as:

Computation Rules for Mod Arithmetic

1) a + b mod m = (a mod m) + (b mod m)
2) a - b mod m = (a mod m) - (b mod m)
3) a * b mod m = (a mod m) * (b mod m)

D) Mod Division

Division is the inverse operation of multiplication. This means that every division question can be answered by answering a "find the missing number" multiplication question. I.e. Since 5*8 = 4 MOD 12 dividing by 5 yields

8 = 4/5 MOD 12.

Thus, if I had asked you: Compute 4/5 MOD 12, the answer is apparently 8.

Example:

To compute 5 / 7 mod 12, we introduce an x

x = 5 / 7 mod 12 to multiply both sides by 7.

7x = 5 mod 12.

We find x by testing the 12 different remainders 0, 1, ...11.

Trial and error yields x=11 since
7 * 11 mod 12 = 77 mod 12 = 5.

Some Mod Divisions have no Solution

Unlike the division of real numbers, mod division does not always yield an answer. The reason for that is that the mod-multiplication does not always yield all possible remainders less than the modulus. Let's investigate this fact. For example:

Using a modulus of m=6, we set up a multiplication table that displays the multiplications of the remainders 0,1,2,3,4 and 5.

* mod 6	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

Two Observations:

1) Some divisions have no answer

The rows created by the remainders 0,2,3,4 do not contain all six remainders. Consequently, some divisions have no answer. I.e. consider division by 2:

4 / 2 = 2 mod 6 since 2 * 2 = 4 mod 6.

2 / 2 = 1 mod 6 since 1 * 2 = 2 mod 6.

However, 3 / 2 = x mod 6 has no answer x since there exists no remainder x such that 2 * x yields 3 mod 6.

Also, 5 / 2 mod 6 has no answer. Why not? In fact no odd integer could possibly be divided by 2 mod. If, however, we use a modulus of 7 any odd (and any even) integer less than 7 can be divided by 2. Explain why by using the multiplication for mod 7 below.

2) Some divisions have many answers

I.e. 4 / 2 = 2 mod 6 , since 2 * 2 = 4 mod 6,

also 4 / 2 = 5 mod 6, since 2 * 5 = 4 mod 6.

Even seemingly odd divisions like 0/3 or even worse 0/0 are legal mod 6. Both have the answer 2. Explain this.

If we don't limit us to the six remainders as answers, we actually find an infinite number of answers. Notice that also 2*8, 2*11, 2*14, 2*17, ... yield 4 mod 6. Consequently, when dividing 4 by 2 mod 6 8,11,14,17,...are correct answers as well. Thus, don't be surprised if your partner finds a different answer than you. It might be just as correct as yours.

Example: If the modulus is m=7, the divisions yield unique solutions.

* mod 7	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Perform the following divisions. Which one has multiple answers between 0 and the modulus? Find them if they exist. Which division has a unique answer? Which division has no answer?

1) 7 / 4 = x mod 12 (or 7 = 4*x mod 12) No answer.
2) 6 / 9 = x mod 12 (or 6 = 9*x mod 12) x=2.
3) 7 / 5 = x mod 13 (or 7 = 5*x mod 13) x=4.
4) 3 / 13 = x mod 26 (or 3 = 13*x mod 26) No answer.
5) 4 / 10 = x mod 26 (or 4 = 10*x mod 26) x=3.
6) 12 / 10 = x mod 29 (or 12 = 10*x mod 29) x=7.

E) Mod Exponentiation

In the encryption process of the RSA Cipher the plain message is raised to the power of e mod m, where e and m (commonly a 200 digit number) make up the public key. To account for huge numbers, one of our goals in this section is to learn some shortcuts when performing mod exponentiation.

Since mod exponentiation is repeated multiplication, it can be reduced to the above mod multiplication.

How do we compute 34 MOD 12? First we multiply: 3 * 3 * 3 * 3 = 81, secondly we find the remainder: 81 mod 12 = 9.

Shortcut 1: Instead of first computing the (large) power and secondly finding the remainder, it is easier to find the remainders of smaller powers and mod multiply them to get the final answer.

There is a fast way to compute 23^{77} mod 24 . Since 23 = -1 mod 24, we may write $(-1)^{77}$ mod 24 which simplifies to -1 mod 24. To get a positive answer, we add 24 to get 23 as the final answer.

Shortcut 2: If the base is a little less than the modulus, then rewrite the base as a small negative number which when exponentiated yields a smaller answer then the original power.

Powers such as 12345676 would yield an overflow on your calculator. However, performing modular arithmetic using the modulus m=1234569 we are able to compute the answer 64. Why?

Answer: 1234567 = -2 mod 1234569. Thus, $(-2)^6$ = 64 MOD 1234569

We conclude the Mod Exponentiation with one last shortcut. There is a fast way to compute 2^{11} mod 15. Since $2^{11} = ((2^2)^2)^2 * 2^3$, we compute 2^{11} mod 15 as $(2^4)^2 * 2^3$ mod 15 $=((2^4)^2$ mod 15) * (2^3 mod 15) = 1 * 8 mod 15 = 8. Check: Dividing 2048 by 15 leaves a remainder of 8. Correct.

Shortcut 3: (Repeated Squaring) To compute an, divide the exponent n by the greatest power of 2 that is less than n. This yields an exponent e and a remainder r. Finally, compute $a^e * a^r$. To compute a^e, use shortcuts if necessary. a^r is a fairly small number.

Example:

1) 3^{11} mod 12=3. Since $3^2 = 9$, $3^4=(9)^2 = 81 = 9$ mod 12. Also: $3^8=(3^4)^2=(9)^2 = 81 = 9$ mod 12. Since, $3^3 = 27 = 3$ mod 12, $3^{11} = 3^8 * 3^3 = 9 * 3 = 27 = 3$ mod 12.

2) 4^{11} mod 12=4. Since $4^2 =4$, $4^4=4^8=4$. $4^8 * 4^3 = 4$.

3) 5^{13} mod 17=3. Since $5^2=8$, $5^4 = 13 = -4$, $5^8 = 16 =-1$, $5^{13} = -1 * -4 * 5 =20$.

4) 3^{33} mod 17=3. Since $3^2 = 9$, $3^4 = -4$, $3^8 = -1$, $3^{16} =1$, $3^{32} = 1$

5) 16^5 mod 19=4. Since 16 = -3, $16^2 = 9$, $16^4 = 81 = 5$.

(5.2) Euler's φ-function

Definition: Euler's φ-function yields the number of integers less than a given integer n that have no common factor with n. I.e. the gcd equal to 1.

Example: Say n=4, then 1 and 3 - a total of 2 integers - are relative prime to 4 and less than 4. Thus, Euler's φ-function yields 2 which is commonly written as φ(4)=2.

Example: Say n=5, then 1, 2, 3 and 4 - a total of 4 integers - are relative prime to 5 and less than 5. Thus, φ(5)=4. In fact, whenever n is a prime number, none of the n-1 numbers less than n can have a common factor with n. Thus, we may write: φ(p)=p-1 when p is a prime number.

Example: Say n=7, then 1, 2, 3, 4, 5, 6 - a total of 6 integers - are relative prime to 7 and less than 7. Thus, φ(7)=6.

Example: Say n=12, then 1, 5, 7, 11 - a total of 4 integers - are relative prime to 12 and less than 12. Thus, $\varphi(12)=4$.
Example: Say n=15, then 1,2,4,7,8,11,13,14 - a total of 4 integers - are relative prime to 15 and less than 15. Thus, $\varphi(15)=8$.

Observe a summary of the φ–values for the first twenty integers:

M	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
φ(M)	1	2	2	4	2	6	4	8	4	10	4	12	6	8	8	16	6	18	8

Four Computation Rules for Euler's φ-Function

1) $\varphi(p) = p-1$	for a prime p.
2) $\varphi(p^k) = p^k - p^{k-1}$	for a prime power p^k .
3) $\varphi(p*q) = (p-1)*(q-1)$	for two distinct primes p and q. This rule is used for the RSA cipher.
4) $\varphi(k*m) = \varphi(k)*\varphi(m)$	when k and m are relative prime.

Example: $\varphi(3)=3-1=2$ as 1 and 2 are relative prime to 3.
Example: $\varphi(8)= \varphi(2^3)=2^3 -2^2 =4$ as 1,3,5,7 are relative prime to 8.
Example: $\varphi(15)=\varphi(3*5)=(3-1)*(5-1)=2*4=8$ as 1,2,4,7,8,11,13,14 are relative prime to 15.
Example: $\varphi(24) = \varphi(2^3*3) = \varphi(2^3)*\varphi(3) = (2^3-2^2)*(3-1) = 4*2 = 8$ as 1,5,7,11,13,17,19,23 are relative prime to 24.

Computation of φ(n) in a single formula
The 4 computation rules can be combined into one formula that yields φ(n) for any given n. Here it is:

$$\varphi(n) = n * (1- 1/p_1) * (1- 1/p_2) *\ldots* (1-1/p_k),$$
where p_1 , p_2, ..., p_k are the prime divisors of n.

Let me show you two examples of this formula:
Example:
Say n=180, then a prime check program yields the prime factors 2,3 and 5, so that
$\varphi(180) = 180 * (1-1/2) * (1-1/3) * (1-1/5)$
$= 180 * (1/2) * (2/3) * (4/5)$
$= 90 * (2/3) * (4/5) = 60 * (4/5) = 48.$
Example:
Say n=360, since 360=2*180 the prime factors are again 2,3 and 5, so that
$\varphi(360) = 360 * (1-1/2) * (1-1/3) * (1-1/5) = 360 * (1/2) * (2/3) * (4/5)$
$= 180 * (2/3) * (4/5) = 120 * (4/5)= 96$

(5.3) Euler's theorem:

For a given integer n, each integer a that is relative prime to n yields 1 MOD n when raised to the power of φ(n):

$$a^{\varphi(n)} = 1 \text{ MOD } n.$$

If n is a prime, let's call it p, we know that φ(p) = p-1 (rule 1 of phi-function) which simplifies Euler's Theorem to:

$$a^{p-1} = 1 \text{ MOD } p.$$

This special case of Euler's Theorem is also known as Fermat's Little Theorem.

Example: Say M=5, then 1,2,3,4 are relative prime so that φ(5)=4. When raising 1,2,3 and 4 to the power of 4 we compute mod 5:

$1^4 = 1$ mod 5, $2^4 = 16 = 1$ mod 5, $3^4 = 81 = 1$ mod 5 and $4^4 = 256 = 1$ mod 5. This is just what Euler teaches us.

Example: Say M=8, then 1,3,5 and 7 are relative prime φ(8)=4. When raising 1,3,5 and 7 to the power of 4 we compute mod 8:

$1^4 = 1$ mod 8, $3^4 = 81 = 1$ mod 8, $5^4 = 625 = 1$ mod 8 and $7^4 = (7^2)^2 = (49)^2 = (-1)^2 = 1$ mod 5.

(5.4) Euclidean algorithm:

We now present an algorithm, called the Euclidean algorithm, for finding gcd(a,b). Suppose that a>b>0 (otherwise, interchange a and b), and write

$$\begin{aligned}
a &= k_1 b + r_1 & 0 &\le r_1 < b\\
b &= k_2 r_1 + r_2 & 0 &\le r_2 < r_1\\
r_1 &= k_3 r_2 + r_3 & 0 &\le r_3 < r_2\\
r_2 &= k_4 r_3 + r_4 & 0 &\le r_4 < r_3\\
&\vdots & &\vdots\\
r_{n-2} &= k_n r_{n-1} + r_n & 0 &\le r_n < r_{n-1}\\
r_{n-1} &= k_{n+1} r_n + r_{n+1} & 0 &\le r_{n+1} < r_n
\end{aligned}$$

Since a>b>r1>r2>r3>r4>..., the remainders will eventually become zero, so at some point we obtain r_{n+1}=0. r_n=gcd(a,b).

Example:

a) Let a=190 and b=34. Then

$$\begin{aligned}
190 &= 5 \cdot 34 + 20\\
34 &= 1 \cdot 20 + 14\\
20 &= 1 \cdot 14 + 6\\
14 &= 2 \cdot 6 + 2\\
6 &= 3 \cdot 2 + 0
\end{aligned}$$

so gcd(190,34)=2.

b) Let a=108 and b=60. Then

$$\begin{aligned}
108 &= 1 \cdot 60 + 48\\
60 &= 1 \cdot 48 + 12\\
48 &= 4 \cdot 12 + 0
\end{aligned}$$

so gcd(108,60)=12.

(5.5) Extended Euclid algorithm:

Given a and b, the extended Euclid algorithm computes g, u, and v such that

$$g = \gcd(a, b) = u . a + v . b$$

This algorithm is used to compute the modular inverse. If g = 1, then

$$1 = u . a +v . b$$

implies that

$$1 = u . a \pmod b$$
$$1 = v . b \pmod a$$

and therefore

$$u = a^{-1} \pmod b$$
$$v = b^{-1} \pmod a$$

Algorithm 5.1: Extended Euclid Algorithm (EEA)

```
EEA(a, b, g, u, v)
begin
(g0 , g1) = (a, b)
(u0 , u1) = (1, 0)
(v0 , v1) = (0, 1)
while g1 ≠ 0 do
        begin
        q = g0 div g1
        (g0 , g1) = (g1 , g0-g1 . q)
        (u0 , u1) = (u1 , u0-u1 . q)
        (v0 , v1) = (v1 , v0-v1 . q)
        end
g = g0 ; u = u0 ; v = v0
end
```

Example: a = 21 and b = 16 apply the extended Euclid algorithm

Iteration	q	g_0	g_1	u_0	u_1	v_0	v_1
0	-	21	16	1	0	0	1
1	1	16	5	0	1	1	-1
2	3	5	1	1	-3	-1	4
3	5	1	0	-3	16	4	-21
		•		•		•	

EEA returns g = 1, u = -3 and v = 4

This implies

$$1 = -3 . 21 + 4 . 16$$

Therefore

$$21^{-1} = -3 \pmod{16}$$
$$16^{-1} = 4 \pmod{21}$$

(5.6) RSA cipher

In this and the following sections I am going to show you how highly secure communication is realized. We are leaving behind the breakable ciphers that we studied in the previous chapters. Rather, we are going to use learned concepts such as MOD arithmetic, Euler's φ-function, Euler's Theorem and the Euclidean Algorithm in order to create an unbreakable cipher: The RSA Cipher. It is the most popular cryptosystem among today's secure ciphers. It leads us to **Two-Key Cryptography** – commonly called **Public-Key Cryptography**.

The US-Cryptographer Bruce Schneier distinguishes roughly between One-Key and Two-Key Cryptography as follows:

> **"There are two kinds of cryptography in this world: cryptography that will stop your kid sister from reading your files, and cryptography that will stop major governments from reading your files."**

Public Key Cryptography started in 1976 when 31-year-old computer wizard named Whitfield Diffie came up with a new system that surprised the world of ciphers. As a child, Diffie devoured all the books he could find on the subject of cryptography. Certainly there is something about codes -- secret rings, intrigue -- that appeals to youngsters. Diffie, son of an historian, took them very seriously.

Diffie solved the major problem of existing cryptography by making the key transfer unnecessary

The problem with the existing system of cryptography was the key transfer. In order to communicate secretly the sender has to deliver the secret key to the recipient. In that way the recipient can decrypt the encrypted message. The dilemma: In order to transfer a secret a different secret (the key) had to be delivered before. How can the key be sent from one party to another? If you sent it over an insecure channel (i.e. telephone, internet, etc.) what's to stop someone from intercepting it and using it to decode all subsequent messages? Diffie showed in a clever manner how to overcome the key transfer dilemma.

(5.6.1) The Diffie-Hellman key exchange

In 1976, the two Mathematicians Whitfield Diffie and Martin Hellman devised the notion of Public-Key-Cryptography in "New Directions in Cryptography". They showed how two communication partners that are at different locations can publicly create a common key without fearing that a third person who observes the key generation could crack it. The public creation of a common secret key was the first major step in Public Key Cryptography. It enables two parties to establish secure communication *without* having to deliver the secret key at some earlier point in time.

Since you know how to perform MOD arithmetic you will understand the Diffie-Hellman Key Exchange between Alice and Bob easily. Here it is in 4 steps. The two left columns show the general key exchange protocol. The two right columns display an example for the key exchange protocol:

1) Alice and Bob publicly pick 2 integers: a) a prime number **p** b) and an integer **s** between **1** and **p**.		1) Alice and Bob publicly pick a) p = 11 and b) s = 3.	
2) Alice picks a random number **a** that is less than **p**.	2) Bob picks a random number **b** that is less than **p**.	2) Alice picks a=2.	2) Bob picks b=4.
3) Alice computes **A** = $\mathbf{s}^{a}$ MOD **p** and sends **A** to Bob.	3) Bob computes **B** = $\mathbf{s}^{b}$ MOD **p** and sends **B** to Alice.	3) Alice computes A = 3^2 MOD 11 = 9	3) Bob computes B = 3^4 = 81 = 4 MOD 11
4) Alice computes the key **K** = $\mathbf{B}^{a}$ MOD **p**.	4) Bob computes the key **K** = $\mathbf{A}^{b}$ MOD **p**.	4) Alice computes K = 4^2 MOD 11 = **5**	4) Bob computes K = 9^4 = 6561 MOD 11 = **5**.

We have to answer the following two questions:

1) Why do Alice and Bob always end up with the same key K ?
2) Why can't an eavesdropper compute K ?

Answer to questions 1):

Alice and Bob compute the key K in the final step as follows:

Alice: K = $B^a = s^{ba}$ MOD p.

Bob: K = $A^b = s^{ab}$ MOD p.

Since $s^{ba} = s^{ab}$ MOD p both Alice and Bob compute the same key K.

Answer to questions 2):

Say, an eavesdropper did a great job by intercepting the values **A** and **B, p** and **s**. In order to uncover the key **K** he has to compute either **a** or **b**. Mathematics teaches that this is impossible if **a, b, p, s, A, B** were chosen as of 100-digits numbers or larger.

The reason: although it is quite simple to compute i.e. **A** = $\mathbf{s}^{a}$ MOD **p**, however, solving this equation for **a** is impossible. More formally stated: Although the "discrete exponential function" can be executed, its inverse, the "discrete logarithm function" can not. Hence, the "discrete exponential function" is an example of a "One-Way function". We will study such functions later on this chapter.

(5.6.2) Diffie's search for a public key cryptosystem

While Hellman was working on the public key exchange algorithm, Diffie continued his research. His public key exchange idea is feasible, however, it has one major practical limitation: Both parties have to first create their secret key in a mutual process. As a consequence, the sender has to wait until the common key is created to send the encrypted message. Diffie tried to overcome this deficit as follows: Each party shall possess a **key pair**, a **public** and a **private key**. I.e. Alice's public key is used by Bob to encrypt the message for Alice whereas her private key is used to decrypt Bob's encrypted message.

Although it was not Diffie who actually designed an algorithm that realizes his provoking idea of a public and private key, he deserves credit for making the inconceivable conceivable. He initiated mathematical research groups worldwide to find an appropriate mathematical setting that realizes his vision. The research question was:

> **Which mathematical function allows anybody to encrypt a secret message for Alice using her publicly known encoding key *e* and prevents everybody but Alice to decrypt such cipher message?**

It requires an experienced mathematician to find such **one-way** functions since most functions are two-way and can thus be reversed. I.e. A Caesar right shift can be reversed by a corresponding left shift. Mathematically, since addition can be reversed by subtraction it is (the simplest) two-way function.

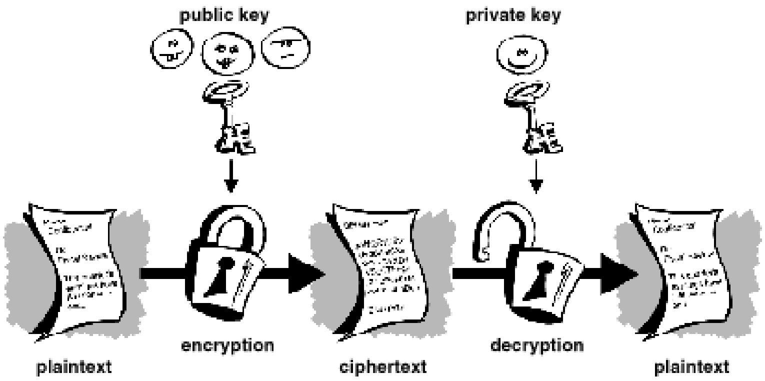


Figure 2: This diagram shows that Public–Key Cryptography uses two different keys for en- and decryption.

The winners of the worldwide search for appropriate one-way functions are the 3 Israelis Ronald Rivest, Adi Shamir and Leonard Adleman. Let's talk for a moment about how the three MIT researchers Rivest, Shamir and Adleman came up with their idea for the RSA Ciphers in 1977.

The irony of the process to create the RSA Cipher is that Rivest, Shamir and Adleman originally tried to prove that Diffie's idea of a public and a private key could NOT be realized by showing that there are no appropriate one-way functions. Thus, while being unsuccessful the three Cryptographers became really successful: While Rivest devised encryption ideas, Adleman tried to attack them and Shamir contributed to both. The final RSA idea was ground-breaking as it simultaneously answers the contemporary quests for a public key cipher as well as a scheme to perform digital signatures.

(5.6.3) The RSA cipher is a public-key-cryptosystem

Given sufficiently large key lengths, RSA realizes the

Public-Key Property:

> The knowledge of the encoding key does not reveal the knowledge of the decoding key. Even the usage of the most powerful computers combined will not suffice to crack the secret decoding key based on the knowledge of the publicly known encoding key.

Although the encoding keys are publicly known, nobody in the world can make use of that knowledge in order to crack the secret decoding key. WHY is that? How can the knowledge of the encoding key NOT reveal the decoding key? What do we have to consider when choosing the encoding key so that its decoding key remains secret? The next section will tell you.

As a consequence:

> Only one key pair per person is needed. Therefore, a total of only n key pairs are needed for n communicating people.

In particular, 100 people communicating just need 100 key pairs. The 100 encoding keys are publicly known. Think of them being listed in a directory. These days the internet provides the means to locate the public encoding key of any correspondent. Currently, the public keys of the most popular encryption software, "Pretty Good Privacy" or simply "PGP", can be looked up at http://pgp.mit.edu . The corresponding 100 private decoding keys must be kept absolutely secure (i.e. in a vault) by its respective owner. They should not be saved on the hard-drive or other accessible devices.

(5.6.4) An Example for RSA Encryption

The RSA Cipher is actually quite easy to understand. Don't be mislead: The fact that a cipher can be executed easily does not make it insecure. On the other hand, ciphers that are hard to execute are not necessarily secure. The 4 steps involved in the RSA Cipher are displayed in the middle column. In the right column I demonstrate how the word "SAFE" is en- and decrypted using the RSA Cipher. We use the same letter values as in the previous chapters to encrypt: S=18, A=0, F=5, E=4. Again, the choice of the letter values is arbitrary. It just matters that sender and recipient use the same letters values. Let's go ahead and perform the RSA encryption.

Example for RSA Encryption and Decryption:

Step: Preparation	a) Choose two primes p and q so that their product n=p*q is greater than the used alphabet length M (i.e. here M=26). b) Compute $\varphi(n)$.	a) Say p=3 and q=11, then n=33 b) $\varphi(33) = (3-1)*(11-1) = 20$

2. Step:
Encryption
uses the public key (n,e)
	a) Choose a public encoding key e that has to be relative prime to $\varphi(n)$.
	b) We now encrypt each plain letter P by computing
	$C=P^e$ MOD n.
		a) Here, possible values for e are 3, 7, 9, 11, 13, 17, 19. Let's pick e=3.
		b) We encrypt as follows:
		S =18: 18^3 = 24 MOD 33
		A = 0: 0^3 = 0 MOD 33
		F = 5: 5^3 = 26 MOD 33
		E = 4: 4^3 = 31 MOD 33

3. Step: Decryption uses the private key (d,n)	a) The private decoding key d is chosen as the inverse of e MOD $\varphi(n)$: e * d = 1 MOD $\varphi(n)$ Mathematically, find integers d and k that fulfill: e * d = 1 + k * $\varphi(n)$ via the Extended Euclidean Algorithm. b) We decrypt by computing $P=C^d$ MOD n	a) d=7 since 3*7 = 1 MOD 20. b) 24^7 = 18 MOD 33, 18=S. 0^7 = 0 MOD 33, 0=A. 26^7 = 5 MOD 33, 5=F. 31^7 = 4 MOD 33, 4=E.

Remark: I computed the above powers MOD 33 using the Windows XP calculator. Let's verify by hand that i.e. the computation of 31^7 actually yields 4 MOD 33:

Since 31= -2 MOD 33 I can just multiply –2 by itself 7 times to compute $(-2)^7$ and obtain –128. Now, –128 = -95 = -62 = -29 = 4 MOD 33 which produces the calculator answer.

(5.6.5) Why does RSA work? – A Proof

Why does the RSA Cipher work? With other words: for what miraculous reason does the final exponentiation C^d MOD n in step 3 yield the correct plain letter P ? Nobody would use the RSA encryption to encrypt highly sensitive data if RSA's encryption scheme is not guaranteed to yield the original plain text. Therefore, it is important to mathematically prove its correctness. The proof demonstrates how Rivest, Shamir and Adleman took advantage of Euler's Theorem.

Proof of Correctness of the RSA Cipher

We saw in the above example that the decryption yields the proper plain text SAFE. Will it always – for any given plain text – work ? To be sure, we have to prove that the final exponentiation in the decryption process, C^d, is guaranteed to yield the proper plain letter P. In order to do so we have to distinguish two cases:

I) The vast majority of plain letters values P are relative prime to the modulus n. For this case we take advantage of Euler's Theorem to prove the correct functioning of RSA.

II) We also have to prove the correctness of RSA for the rare instances when P is not relative prime to n. i.e. P and n possess a common divisor that is greater than 1.

I) RSA-proof when P and n are relative prime

We establish that C^d actually yields P by using a chain of identities that involves simple exponent rules and Euler's Theorem. Verify each step in the proof.

C^d	The cipher letter C was encrypted by raising the original plain letter P to the power of e: $C=P^e$. We substitute:
$= (P^e)^d$ MOD n	When raising a power to a power, we multiply the exponents:
$= P^{d*e}$ MOD n	The decoding key d was chosen to be relative prime to the encoding key e which can be stated as: $d*e = 1 + k * \varphi(n)$. Therefore,
$= P^{1 + k * \varphi(n)}$ MOD n	Adding exponents allows to multiply powers:
$= P^1 * P^{k * \varphi(n)}$ MOD n	Multiplying exponents allows raising a power to a power:
$= P * (P^{\varphi(n)})^k$ MOD n	What is the purpose of setting up $(P^{\varphi(n)})$? Answer: To make use of Euler's Theorem: $P^{\varphi(n)} = 1$ MOD n when P and n are relative prime. If P and n don't happen to be relative prime RSA still works properly. I will prove this case in part II) of the proof.
$= P * (1)^k$ MOD n	Raising 1 to any power yields 1. Guaranteed.
$= P$ MOD n	The proof is complete. **We verified that C^d yields P in case P and n are relative prime.**

II) RSA-proof when P and n are not relative prime

In that case P and n have a common divisor. Consequently, either p or q must be a divisor of P. This results from the choice of n as the product of the primes p and q. In other words, p and q are the only divisors of n and any divisor that n has in common with P must be either p or q or a multiple of p or q. Without limiting the generality of our proof, we assume that p is a divisor of P. That means that there exists an integer x such that P=x*p.

In the first proof, we made use of Euler's Theorem. We now make use of its simplified version, the so-called "Fermat's Little Theorem". If the modulus is a prime number p, the exponent simplifies to φ(p) = p-1. Why is that? Recall that Euler's φ-function φ(n) gives the number of integers less than n that are relative prime to n.

Fermat's Little Theorem

Given a prime p, then

$$a^{p-1} = 1 \text{ MOD } p$$

holds true for any integer a.

We start with Fermat's Little Theorem to prove that P^{de} = P MOD n. I will simultaneously show you an example in the 3rd column.

Fermat's Little Theorem with the prime q holds true for any integer **P**. We manipulate the exponent by raising both sides to the power of (p-1)*k for some integer k.	$\mathbf{P}^{q-1} = 1$ MOD q	If p=5 and q=7 then n=p*q=35. Let's encrypt the message **P**=10. Then $10^{7-1} = 10^6 = 1$ MOD 7 is guaranteed true because of Fermat's Little Theorem.
Raising powers to powers allows multiplying exponents.	$(\mathbf{P}^{q-1})^{(p-1)*k} = 1^{(p-1)*k}$ MOD q	$(10^6)^{4*k} = 1^{4*k} = 1$ MOD 7
Why did we do so? Because (p-1)*(q-1)*k = φ(p*q) * k = φ(n) * k = de –1 when choosing d and e to be inverse mod φ(n) . We may therefore write	$\mathbf{P}^{(q-1)*(p-1)*k} = 1$ MOD q	$10^{6*4*k} = 1$ MOD 7 We choose d=5 and e=5 so that they are inverse MOD 24 where 24 = φ(35) = (7-1)*(5-1) since 5 * 5 = 25 = 1 MOD 24.

Subtracting 1 on both sides yields	$\mathbf{P}^{de-1} = 1$ MOD q	With k=1: 10^{6*4*k} = 10^{d*e-1} = 10^{5*5-1} = 1 MOD 7
(Adjusting the modulus) If $\mathbf{P}^{de-1}$ - 1 is a multiple of q then **p*** ($\mathbf{P}^{de-1}$-1) must be a multiple of p*q. Consider a simple example with p=2: If 15 is a multiple of 5 then 30 is a multiple of 10.	$P^{de-1} - 1 = 0$ MOD q	10^{5*5-1} –1 = 0 MOD 7
Can you finish the proof without looking at the remaining proof? Try it, it is not difficult.	p*($\mathbf{P}^{de-1}$ - 1) = 0 MOD p*q	5* (10^{5*5-1}–1) =0 MOD 7* 5
(Obtaining $\mathbf{P}^{de}$ =**P**) Since **P** is a multiple of p **P***($\mathbf{P}^{de-1}$ - 1) must also be a multiple of p*q. Continuing our example P=7*p: If 30 is a multiple of 10 then 7*30 = 210 is also a multiple of 10.	p*($\mathbf{P}^{de-1}$ - 1) = 0 MOD p*q	5* (10^{5*5-1} –1) = 0 MOD 35
Distributing yields	**P***($\mathbf{P}^{de-1}$ - 1) = 0 MOD p*q	10* (10^{5*5-1}–1) = 0 MOD 35
Adding **P** on both sides and substituting n for p*q concludes our proof.	$\mathbf{P}^{de}$ - **P** = 0 MOD p*q	10^{5*5} – 10 = 0 MOD 35
We verified that **P'** = **P** when **P** and n are not relative prime. Combining the proofs I) and II), we understand why the RSA encryption works properly in any case.	$\mathbf{P}^{de}$ = **P** MOD n	10^{5*5} = 10 MOD 35

We considered already an example when P and n are relative prime. Let's now consider two simple examples for the case that P is a multiple of p.

<u>Example:</u> Say we choose p=3 and q =5 as two small distinct primes. Then: n = p*q = 15 (allowing us to only encrypt 15 letters) and φ(15) = 2*4 = 8. We may therefore choose the keys to be e = 3 and d = 3 since their product yields 9 and 9 = 1 MOD 8 where 8 = φ(15). It is coincidence that the encryption key equals the decryption key. Instead, we could have also chosen e = 3 and d = 11 (since 33 = 1 MOD 8) or e = 3 and d = 19 (since 57=1 MOD 8). Or e = 5 and d = __ ? Do not miss to fill in the answer. Then continue!

Let's now select a plain letter P such that it has a common divisor with n = 15, say we have to encrypt letter P = 6. We have to verify that P^{de} = P MOD n (here 6^9 = 6 MOD 15).

Since 6^2 = 36 = 6 MOD 15, we deduce that $6^8 = 6^2 * 6^2 * 6^2 * 6^2 = 6 * 6 * 6 * 6 = 6^2 * 6^2 = 6 * 6 = 6^2$ = 6 MOD 15.

Thus, $6^9 = 6^8 * 6 = 6 * 6 = 6$ which shows that the RSA scheme decrypts the correct plain letter P = 6.

<u>Exercise:</u> (Test your understanding) Why would it be incorrect to argue as follows: Euler's Theorem yields $6^{\varphi(15)} = 6^8$ = 1 MOD 15 and therefore $6^9 = 6^8 * 6 = 1 * 6 = 6$ MOD 15. Even though the final answer is correct, where is a mistake in my argument? We computed correctly in example 1 that 6^8 = 6 MOD 15.

<u>Example:</u> Let's now verify that RSA decrypts the plain letter P = 12 correctly. For that, we have to verify that 12^9 = 12 MOD 15. Since 12^2 = 144 = 9 MOD 15 and $12^4 = (12^2)^2 = 9^2$ = 81 = 6 MOD 15, we know that $12^8 = (12^4)^2 = 6^2$ = 36 = 6 MOD 15. This leads us to the final result: $12^9 = 12^8 *12 = 6 *12 = 72 = 12$. Correct.

(5.6.6) Why is RSA secure?

You may say now: RSA does not fulfill the above mentioned Public-Key-Property since we could derive the decoding key d=7 from the encoding key e=3 and φ(33)=20. I admit, you are right. However, the crux of the RSA cryptosystem is that it can be upgraded so that even though everybody knows the modulus n, not even the best eavesdropper is able to compute φ(n).

How can the RSA Cipher be upgraded to a secure cipher?
Answer: Choose the two primes p and q to be at least 100-digit numbers.

Why does that make RSA secure? Because no eavesdropper (not even the NSA or the FBI) is able to compute φ(n) from the publicly known modulus n since its factors p and q – required to compute φ(n) as φ(p*q) = (p-1) * (q-1) - can not be derived from n. As experienced computer experts, Rivest, Shamir and Adleman knew that the multiplication of two large numbers is not difficult, however, finding the factors of a given large integer is a difficult computer problem.
I.e. we easily find that

35 = 7 * 5	or
70 = 7 * 5 * 2	or
69 = 3 * 23	or
221 = 13 * 17	and using some trial and error even
11413 = 113 * 101	

A factoring program quickly finds the factors of 20-digits numbers such as

107262914177797115873 = 1223233789 * 8768799157

However, even today's best factoring programs can not find the factors of 196-digit numbers such as
1072629139784206448651876669948737985105534479415491719546499789238601
95309909991719546489160623556900202902061728413351851858054814816308432
0989148925926075485185203509876558141111112930000042837

Exercise: Realize that the multiplication of large numbers can easily be performed by a computer. Verify that the product of the following two 98-digit numbers
122323378977777777788888888893333333334444444444555555555566666666667
77777777888888888900000000327
and
876879913511111111112222222223333333334444444444555555555566666666677
77777777888888888900000000131
yields the above 196-digit number by using the java applet prime.htm .

Reflection on the Security of the RSA Cipher:

Among the many possible ways of attacking the RSA Cipher, factoring is the most promising one. Fact is that sufficiently large integers can not be factored, even when using the best factoring algorithms on the fastest computers in the world. Consequently, if n is chosen sufficiently large nobody is able to find the factors p and q of n and thus nobody can compute φ(n) in order to then identify the decoding key d. I will show you mathematically that the ability to find the factors of n is equivalent to the ability to compute φ(n). The equivalence is based on the following two facts:

I) Knowing the factors p and q is sufficient to compute φ(n) since φ(n) = φ(p*q) = (p-1) * (q-1). I showed you already that if p=3 and q=5 then φ(15) = φ(5)* φ(3) = 4 * 2 = 8.

II) "Knowing φ(n) and the publicly known value n is sufficient to find the factors p and q" can be seen as follows:

We know that n = p*q and that φ(n) = (p-1)*(q-1). How could Mr. X use the two identities to compute p and q? We know how to solve two equations with two variables if only we can set up two equations that allow us to eliminate either p or q. Here is how:

$$
\begin{aligned}
\varphi(n) &= (p-1)*(q-1) \\
&= (p*q - p - q + 1) \\
&= (p*q - [p + q] + 1) \\
&= (n - [p + q] + 1)
\end{aligned}
$$

Solving for p+q yields:

$$p+q = n - \varphi(n) + 1 \qquad (1)$$

We managed to express p + q in terms of n and φ(n). If additionally we manage to express the difference of the two primes, p – q, in terms of n and φ(n) we can use those two equations to compute p and q. So, let's express p – q in terms of n and φ(n).

$(p-q)^2 = (p-q) * (p-q) = p^2 - 2*p*q + q^2$.	
$(p+q)^2 = (p+q) * (p+q) = p^2 + 2*p*q + q^2$.	
$(p-q)^2 - (p+q)^2 = - 4*p*q$.	This equation stems from subtracting the two previous equations. We add $(p+q)^2$ on both sides.
$(p-q)^2 = (p+q)^2 - 4*p*q$	Taking the square root yields.
$p-q = [(p+q)^2 - 4*p*q]^{1/2}$	We are done since we can express (p+q) as (n - φ(n) +1) replace and p*q as n
$p-q = [(n - \varphi(n) +1)^2 - 4*n]^{1/2}$ (2)	Combining the equations (1) and (2) yields
$p+q = n - \varphi(n) +1$ $p -q = [(n - \varphi(n) +1)^2 - 4*n]^{1/2}$	p and q can be derived by solving a 2 by 2 system.

Example: Say an eavesdropper knows that φ(n) = 30600 in addition to the publicly known n = 31003. To find p and q, he uses the identities (1) and (2) to set up the 2 by 2 system

$$p + q = n - \varphi(n) +1 = 31003 - 30600 + 1 = 404 \qquad (1)$$

$$p - q = [(n - \varphi(n) +1)^2 - 4*n]^{1/2} = (404)^2 - 4*31003 = 198 \qquad (2)$$

He just has to solve the 2 by 2 system for p and q.

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} 404 \\ 198 \end{bmatrix}$$

and the answer should be p=301 and q=103.

This section can be summarized as follows:

🔒 SECURITY OF THE RSA-CIPHER:

Since finding the factors of a huge integer n [without knowing φ(n)] is difficult, the RSA encryption is secure.

The inability to crack RSA is based mathematically on the inability to factor the product n into the two huge primes p and q. However, the reverse process, computing the product given the two primes was easy as I showed you above. A function with such property is called a one-way function.

Definition of a one-way function:

A one-way function is a function that computes the result (i.e. the product of two primes) easily. However, given the result it is practically impossible to compute the original values. Mathematically stated: Given x, f(x) can be computed easily. However, given f(x), computing x is practically impossible.

Example: RSA's one-way function is $f(p,q) = p*q = n$ since n does not yield p and q if chosen sufficiently large.

Among other conceivable one-way functions, Rivest, Shamir and Adleman's choice turned out to be a good one since no Mathematician has been able to design a fast factoring algorithm to make the one-way function "Multiplication of large numbers" a two way function. Nevertheless, if somebody devises an algorithm to factor huge integers quickly the RSA encryption is not secure anymore. Believe me: Designing fast factoring algorithms is a hot topic in current mathematics research, particularly to assess the security of RSA. Although there has not been a major breakthrough, the combination of refining the existing factoring algorithms (such as the number field sieve to name the currently best factoring algorithm for numbers consisting of 110 digits and more) with the ever-increasing power of computers yields new factoring records.

(5.7) Digital Signatures and RSA

How can a sender prove his authenticity? Besides eliminating the key exchange, proving that authenticity was a major quest that RSA realized in the late 70's. Understanding how the RSA encryption scheme works and why it properly decrypts cipher messages will help you understand how to digitally sign documents. In fact, you may be able to discover it for yourself since RSA's digital signature uses both the public and private keys that are used for the RSA encryption.

Hint: The RSA encryption requires you to use the publicly known encryption key (e_R,n_R) of the recipient, however, the RSA signature requires the sender to use his private decryption key (d_S,n_S).

An additional reason for the popularity of the RSA cryptosystem is the following: Not only can we encrypt or digitally sign a document; we can even do both for the same document! We first encrypt using the recipient's encryption key (e_R,n_R) to afterwards sign the encrypted message using our private decryption key (d_S,n_S). In this section I will demonstrate to you how this can be achieved.

(5.7.1) How to Sign a Digital Document using RSA

Recall one of the quests of cryptography: How can a sender prove his authenticity?

A hand-written text is signed by pen to document the authenticity of the sender. Observe that the sender uses unique personal information - a secret that only he possesses to prove his authenticity. Similarly, in order to digitally sign a document the sender has to possess a unique information that nobody else is able to figure out or copy. Now it is your turn: How can RSA be used to digitally sign a document? What does an RSA user possess that nobody else knows about? Of course, it is his private decryption key (d_S,n_S). Again, if only the modulus n is chosen sufficiently large nobody and no computer in the world is capable to compute the private key (d_S,n_S) based on the publicly known encryption key (e_S,n_S). In fact, since private keys can not be forged whereas pen-signatures can, digital signatures are even more trustworthy than traditional signatures.

It is now time to sign a document. Say Alice wants to send a signed letter to Bob.

Example of the RSA Digital Signature scheme:

	The RSA Digital Signature	Example:
1. Step: Preparation:	Alice has to generate a key pair (as explained previously). As before, he has to keep the private key (d,n) secret and makes the public key (e,n) publicly known.	Using the key pair in the introductory example we have (d,n) = (7,33) (e,n) = (3,33)

2. Step: Signing the document P uses the private key (d,n)	The sender applies his private key to the document P in S = P^d Mod n to obtain the signature S which is sent to the recipient. Additionally, the original message P is sent to the recipient as well.	SAFE is signed with d=7:
		S =18: 18^7 = 6 MOD 33
		A = 0: 0^7 = 0 MOD 33
		F = 5: 5^7 = 14 MOD 33
		E = 4: 4^7 = 16 MOD 33

3. Step: Verifying the authenticity of the sender by applying the public key (e,n)	Bob verifies the authenticity of the sender by computing S^e MOD n. He then matches S^e with P. If they are not equal, the document is not authentic. Attention: fraud or a possible computation error has occurred!!	The authenticity of the message is verified by the recipient using the senders public encoding key e=3 6^3 = 18 MOD 33, 18=S. 0^3 = 0 MOD 33, 0=A. 14^3 = 5 MOD 33, 5=F. 16^3 = 4 MOD 33, 4=E.

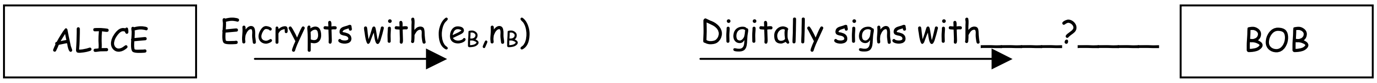
__Exercise:__ Say you are Alice. You want to digitally sign the document "FRAUD" using n=5*7 = 35. First you will have to generate a key pair, secondly you sign FRAUD and send "FRAUD" together with the signed version of "FRAUD". Now, pretend you are the recipient Bob. Firstly, you un-sign Alice's signed message. Secondly, you check the authenticity of Alice's message simply by matching the un-signed message with the original message "FRAUD". For the necessary MOD calculations make use of the windows calculator.

(5.7.2) How to Encrypt AND Digitally Sign with RSA

In the previous section, we learned how Alice may sign a document that can be sent i.e. as an email to the recipient. Since she did encrypt "FRAUD", it was like sending a signed postcard to authenticate the sender. What if she wants to add security and decides to sign and encrypt her message? Sending an encrypted and signed letter electronically corresponds to signed postcard that is mailed in a sealed envelope. How can Alice accomplish this electronically?

__Example:__

1) Devise a scheme with which a sender can encrypt and sign a message before sending it?
2) How does the recipient of the message decrypt and verify the authentication? Before you start complete the following diagram.



Bob verifies authenticity with (e_A,n_A)

and __?____ with (d_B,n_B).

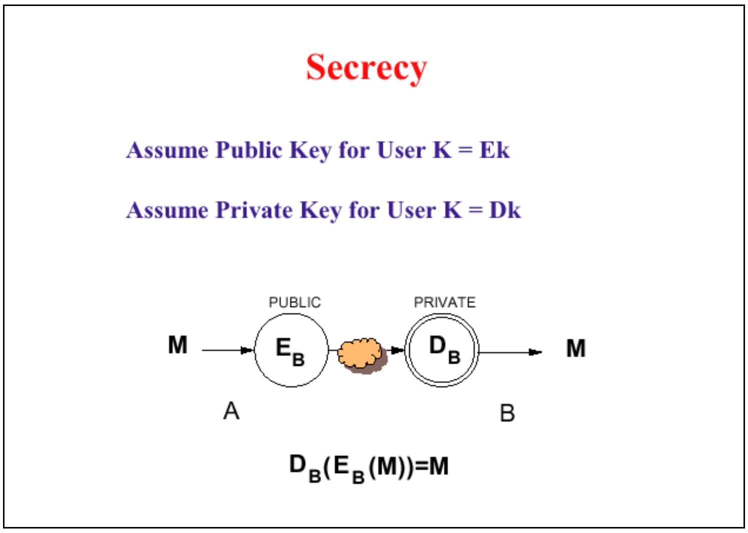
The combined RSA scheme: Encryption and Digital Signature

	The combined RSA scheme	Example:
1. Step: Preparation:	a) Alice has to generate a key pair (as explained previously). She has to keep her private key (d_A,n_A) secret and makes her public key (e_A,n_A) publicly known. b) She has to know Bob's public encoding key (e_B,n_B) .	Alice wants to encrypt and sign "FRAUD". Alice uses the key pair $(d_A,n_A) = (5,35)$ and $(e_A,n_A) = (5,35)$ Since n = 35 = 5 * 7 we obtain φ(35)= (5-1)*(7-1) = 24 and choose e=5 (which is relative prime to 24) and d=5. Bob's key pair is $(d_B,n_B)=(7,33)$ and $(e_B,n_B) = (3,33)$

2. Step: Encrypting the document P uses the public encoding key (e_R,n_R) of the recipient.	Alice uses Bob's public encoding key (e_B,n_B) to encrypt the document P: $C = P^e$ MOD n_B	She encrypts FRAUD using (e_B,n_B) = (3,33): $C = P^e$ MOD n_B
		F = 5: 5^3 = 26 MOD 33
		R =17: 17^3 = 29 MOD 33
		A = 0: 0^3 = 0 MOD 33
		U =19: 19^3 = 28 MOD 33
		D = 3: 3^3 = 27 MOD 33

3. Step: Signing the document C Uses the private key (d_S,n_S).	Alice applies her private key (d_A,n_A) to the document C: $S = C^d$ MOD n_A and sends the signature S to the recipient Bob. Additionally, the encrypted message C is sent to Bob. (Why not P ?)	To sign the document C she uses her private key (d_A,n_A) = (5,35): C^d = S MOD n_A 26^5 = 31 MOD 35 29^5 = 29 MOD 35 0^5 = 0 MOD 35 28^5 = 28 MOD 35 27^5 = 27 MOD 35
4. Step: Verifying the authenticity of the sender by applying the public key (e_B,n_B) of the recipient.	Bob may verify the authenticity of the sender by computing S^e MOD n_A and checking if $S^e = C$. If they don't match, the document is not authentic. Attention: fraud or a possible computation error has occurred!!	Bob verifies the authenticity of the document with (e_A,n_A) = (5,35). S^e = C MOD n_A 31^5 = 26 MOD 35 29^5 = 29 MOD 35 0^5 = 0 MOD 35 28^5 = 28 MOD 35 27^5 = 27 MOD 35 We observe that $C'=C$ which authenticates the sender.
5. Step: Decryption The recipient uses his private key (d_B,n_B) .	The recipient finally decrypts the cipher C by computing $P' = C^d$ MOD n_B. We proved already that $P' = P$ for all P.	Bob decrypts using (d_B,n_B) = (7,33). C^d = P' MOD n_B 26^7 = 5 MOD 33 5 = F 29^7 = 17 MOD 33 17 = R 0^7 = 0 MOD 33 0 = A 28^7 = 19 MOD 33 19 = U 27^7 = 3 MOD 33 3 = D

The encryption and authentication process is straightforward and does not seem to cause any trouble - even if the recipient's modulus n_R turns out to be larger than the modulus of the sender n_A. I.e. imagine we switch the above moduli to n_B = 35 and n_A= 33. When reaching the signing stage (step 3) we handle the 33 integers 0-32 and after the encrypting stage (step 2) we may obtain the 35 integers between 0-34. Here, the integers 0 and 33, 1 and 34 as well as 2 and 35 are signed in the same manner so that $C' = C$ allowing a proper digital signature.



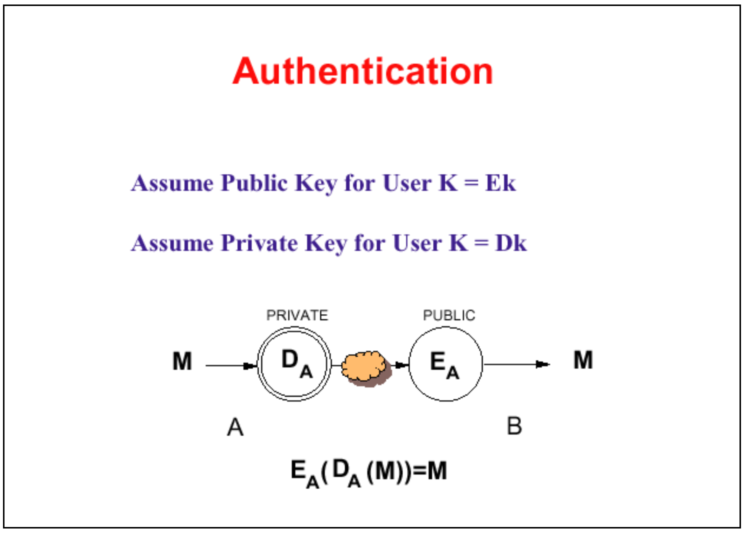
Secrecy

Assume Public Key for User K = Ek

Assume Private Key for User K = Dk

$D_B(E_B(M))=M$

Figure 3: Ciphering using RSA.



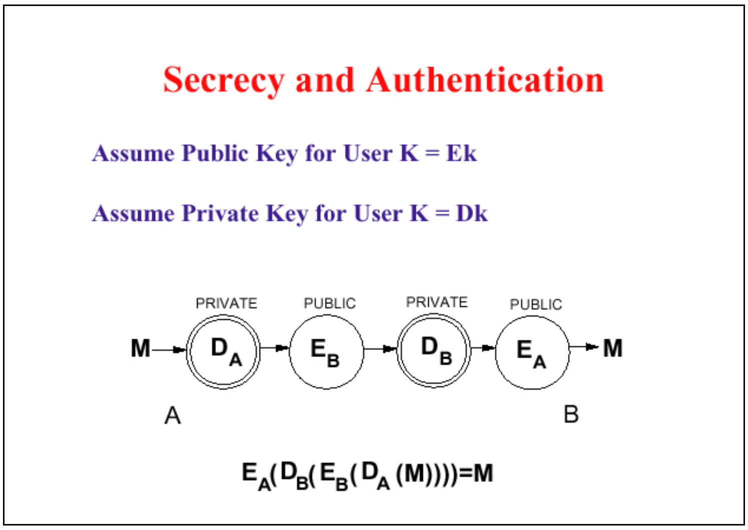
Authentication

Assume Public Key for User K = Ek

Assume Private Key for User K = Dk

$E_A(D_A(M))=M$

Figure 4: Digital signature using RSA.



Secrecy and Authentication

Assume Public Key for User K = Ek

Assume Private Key for User K = Dk

$E_A(D_B(E_B(D_A(M))))=M$

Figure 5: Ciphering and Digital signature using RSA.

(5.8) Knapsack algorithm

The knapsack problem is a simple one. Given a pile of items, each with different weights, is it possible to put some of those items into a knapsack so that the knapsack weighs a given amount? More formally: Given a set of values M_1, M_2,..., M_n, and a sum S. Compute the values of bi such that

$$S=b_1M_1+b_2M_2+...+b_nM_n$$

The values of b_i can be either zero or one. A One indicates that the item is in the Knapsack; a zero indicates that it isn't.

For example, the items might have weights of 1, 5, 6, 11. 14. And 20. You could Pack a knapsack that weighs 22, use weights 5, 6. And 11. You could not pack a knapsack that weighs 24. In general, the time required to solve this problem seems to grow exponentially with the number of items in the pile.

The idea behind the Merkle-Hellman knapsack algorithm is to encode a message as a solution to a series of knapsack problems. A block of plaintext equal in length to the number of items in the pile would select the items in the knapsack (plaintext bits corresponding to the b values), and the ciphertext would be the resulting sum. Figure 6 shows a plaintext encrypted with a sample knapsack problem.

The trick is that there are actually two different knapsack problems, one solvable in linear time and the other believed not to be. The easy knapsack can be modified to crate the hard knap sack. The public key is the hard knapsack, which can easily be used to encrypt but cannot be used to decrypt messages, the private key is the easy knapsack, which gives an easy way to decrypt messages. People who don't know the private key are forced to try to solve the hard Knapsack problem.

(5.8.1) Superincreasing Knapsacks

What is the easy knapsack problem? If the list of weights is a superincreasing sequence, then the resulting knapsack problem is easy to solve. A superincreasing sequence is a sequence in which every term is greater than the sum of all the previous terms. For example, {1,3,6,13,27,52} is a superincreasing sequence, but {1,3,4,9,15,25} is not.

The solution 10 a superincreasing knapsack is easy to find. Take the local weight and compare it with the largest number in the sequence. If the total weight is less than the number, then it is not in the knapsack. If the total weight is greater than or equal to the number, then it is in the knapsack. Reduce the weight of the knapsack by the value and move to the next largest number in the sequence. Repeat until finished. If the total weight has been brought to zero, then there is a solution. If the total weigh has not, there isn't.

For example, consider a total knapsack weight of 70 and a sequence of weights or {2,3,6,13,27,52). The largest weight, 52, is less than 70, so 52 is in the Knapsack. Subtracting 52 from 70 leaves 18, the next weight, 27, is greater than 18 so 27 is not in the knapsack. The next weight, 13, is less than 18, so 13 is in the knapsack. Subtracting 13 from 18 leaves 5. The next weight, 6, is greater than 5, so 6 is not in the Knapsack. Continuing this process will show, that both 2 and 5 are in the knapsack and the total weight is brought to 0. Which indicates that a solution has been found. Were this a Merkle-Hellman knapsack encryption block, the plaintext that resulted from a ciphertext value of 70 would be 110101.

Non-superincreasing, or normal, knapsacks are hard problems; they have no known quick algorithm. The only known way to determine which items are in the knapsack is to methodically test possible solution until you stumble on the correct one.

Plaintext	1	1	1	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0
Knapsack	1	5	6	11	14	20	1	5	6	11	14	20	1	5	6	11	14	20
Ciphertext	1+5+6+20= 32						5+11+14= 30						0= 0					

Figure 6: Encryption with knapsacks

In knapsack algorithm the private key is a sequence of weights for a superincreasing knapsack problem. The public key is a sequence of weights for a normal knapsack problem with the same solution. Merkle and Hellman developed a technique for converting a superincreasing knapsack problem into a normal knapsack problem. They did this using modular arithmetic.

(5.8.2) Creating the public key from the private key

To get a normal knapsack sequence, take a superincreasing knapsack sequence, for example {2,3,6,13,27,52}, and multiply all of the values by a number n, mod m. The modulus should be a number greater than the sum of all the numbers in the sequence: for example, 105. The multiplier should have no factors in common with the modulus: for example, 31. The normal knapsack sequence would then be

2*31 mod 105=62
3*31 mod 105 =93
6*31 mod 105=81
13*31 mod 105 =38
27*31 mod 105 =102
52*31 mod 105=37

The knapsack would then be {62,93,81,88,102,37}.

The superincreasing knapsack sequence is the private key. The normal knapsack sequence is the public key.

(5.8.3) Encryption

To encrypt a binary message, first break it up into blocks equal to the number of items in the knapsack sequence. Then, allowing a one to indicate die item is present and a zero to indicate that the item is absent, compute the total weights of the knapsacks—one for every message block.

For example, if the message were 011000110101101110 in binary, encryption using the previous knapsack would proceed like this:

Message =011000 110101 101110

011000 corresponds to 93 + 81 = 174

110101 corresponds to 62 + 93 +-38 + 37 = 280

101110 corresponds to 62 + 81 + 88 + 102 = 333

The ciphertext would be

174,280,333

(5.8.4) Decryption

A legitimate recipient of this message knows the private key: the original superincreasing knapsack, as well as the values of n and m used to transform it into a normal knapsack. To decrypt the message, the recipient must first determine n^{-1} such that $n(n^{-1}) \equiv 1(\text{mod m})$. Multiply each of the ciphertext values by n^{-1} mod m, and then partition with the private knapsack to get the plaintext values.

In our example, the super-increasing knapsack is {2,3,6,13,27,52}, m is equal to 105, and n is equal to 31. The ciphertext message is 174,280,333. In this case n^{-1} is equal to 61, so the ciphertext values must be multiplied by 61 mod 105.

174*61 mod 105 = 9 = 3+6, which corresponds to 011000

280*61mod 105 = 70 = 2+3+13+52, which corresponds to 110101

333*61mod 105 = 48 = 2+6+13+27, which corresponds to 101110

The recovered plaintext is 011000 110101 101110.

(5.8.5) Additional Knapsack example

User R: Publishes B = {17, 34, 31, 25, 13}

Keeps A = {1, 2, 4, 8, 16}, n = 17, n^{-1} = 24, and m = 37 secret

User S: wants to send the message M = 12 to User R

User S: Takes M = 12 = $(01100)_2$

Computes

C := 0 * 17 +1 * 34 +1 * 31 +0 * 25 +0 * 13

which gives C = 65

Send C = 65 to User R

User R: Receives C = 65

Computes C' = $65n^{-1}$ = 65 * 24 = 6 mod 37

Solves the easy knapsack problem:

6 = $\underline{0}$ * 1 +$\underline{1}$ * 2 +$\underline{1}$ * 4 +$\underline{0}$ * 8 +$\underline{0}$ * 16

This gives the message as $(01100)_2$ = 12

Exercises

Attempt all the following exercises

Q1) Find 5 numbers that are congruent to

1) 7 mod 5
2) 7 mod 25
3) 17 mod 25

Q2) Reduce the following:

1) 40 mod 12
2) 50 mod 12
3) 50 mod 24
4) 40 mod 24
5) 100 mod 33
6) 1000 mod 33
7) 113 mod 12
8) 154 mod 17
9) 5416 mod 55
10) 827 mod 84

Q3) Compute the following:

1) 73 + 58 = x mod 12
2) 1411 - 285 = x mod 141
3) 74 * 93 = x mod 13
4) 33 * 266 = x mod 26
5) 2590 * 5253= x mod 26
6) 133 * 5202 = x mod 26

Q4) Do these problems:

1) 7 / 5 = x mod 12

2) 7 / 11 = x mod 12
3) 29 / 7 = x mod 12
4) x * 7 = 8 mod 12
5) 7 * x = 9 mod 12
6) x * 7 = 5 mod 12

Q5) Compute $\varphi(21)$, $\varphi(22)$, $\varphi(23)$, $\varphi(25)$, and $\varphi(28)$

Q6) Complete

Say M=90, since 90=____ the prime factors are _______, so that
$\varphi(90)$ = 90 * (1-1/__) * (1-1/__) * (1-1/__)
= 90 * ____________________
= _______________
= _______________
= ___ 24

Q7) In RSA, prove that the plain letter P = 10 will be decrypted correctly by showing that $10^9 = 10$ MOD 15.

Q8) In RSA, prove that the plain letter P = 8 will be decrypted correctly by showing that $8^9 = 8$ MOD 15.

Q10) Use the RSA algorithm to encrypt, sign, authenticate and decrypt the document "CODE" using n_B=35 and n_A=39.

Q11) You have a superincreasing sequence A={2,4,7,15,29,57}, use it to create a Knapsack system. Then use this system to encrypt the plaintext (URUK).

Q12) Decrypt the result of Q11.

CHAPTER SIX
STEGANOGRAPHY AND CASE STUDIES

(6.1) Introduction

Information hiding represents a class of process used to embed data into various forms of media such as image, audio, or text. There are two types of information hiding the first one is, steganography and the second is digital watermarking.

Steganography serves to hide secret messages in other messages, such that the secret 's very existence is concealed. Generally the sender writes an innocuous message and then conceals a secret message on the same piece of paper.

Steganography has its place in security. It is not intended to replace cryptography but to supplement it. Hiding message with steganography reduces the chance of a message being detected.

Steganography and encryption are both used to ensure data confidentiality. However the main difference between them is that with encryption anybody can see that both parties are communicating in secret. Steganography hides the existence of a secret message and in the best case nobody can see that both parties are communicating in secret.

Adding encrypted copyright information to a file could be easy to remove but embedding it within the contents of the file itself can prevent it being easily identified and removed.

Digital watermarking, some of the objectives of using this techniques are confirmation of property, follow up of unauthorized copies, validation of identification and verification of integrity, labeling, usage control and protection of contents.

In other word, we can classify information hiding into two types: steganography and digital watermarking.

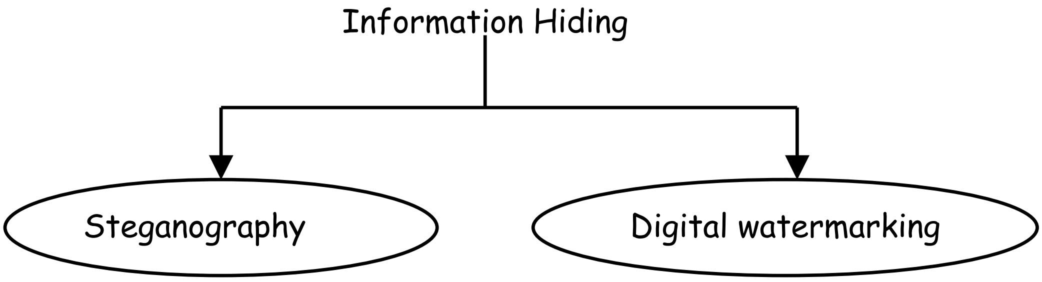


Figure 1: Information hiding classification

(6.2) History

One of the earliest uses of steganography was documented in Histories. Herodotus tells how around 440 B.C. Histiaeus shaved the head of his most trusted slave and tattooed it with a message which disappeared after the hair had regrown. The purpose of this message was to instigate a revolt against the Persians. Another slave could be used to send a reply.

During the American Revolution, invisible ink which would glow over a flame was used by both the British and Americans to communicate secretly.

Steganography was also used in both World Wars. German spies hid text by using invisible ink to print small dots above or below letters and by changing the heights of letter-strokes in cover texts.

During World War II, the Germans would hide data as microdots. This involved photographing the message to be hidden and reducing the size so that that it could be used as a period within another document.

A message sent by a German spy during World War II read:

"Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects for pretext embargo on by-products, ejecting suets and vegetable oils."

By taking the second letter of every word the hidden message "Pershing sails for NY June 1" can be retrieved.

(6.3) Terminology

In this chapter the important terminology used are:-

1. Embedded <data-type>: something to be hidden in something else.
2. Stego-<data-type>: the output of the hiding process; something that has the embedded.
3. Cover <data-type>: An input with "original'' form of the stego-message. In some applications such a cover message is given from the outside, in others, it can be chosen during the hiding process. The letter is represented by the dashed extension to the inner hiding process.
4. Stego-object: The output from stego-system, sometimes hidden in it.

(6.4) Steganography

Steganography literally means "covered writing", it is the art of hiding the very existence of a message.

Though steganography is an ancient craft, the onset of computer technology has given it new life. Computer-based steganographic techniques introduce changes to digital covers to embed information foreign to the native covers.

Most applications of steganography follow one general principle, as illustrated in Figure 2.

Sender, who wants to share a secret message m with recipient, randomly chooses (using the private random source r) a harmless message C called cover object, which can be transmitted to Recipient without raising suspicion, and embeds the secret message into C, probably by using key K, called stego-key. Sender therefore changes the cover C to a stego-object S. This must be done in a very careful way, so that third party, knowing only the apparently harmless message S, can not detect the existence of the secret.

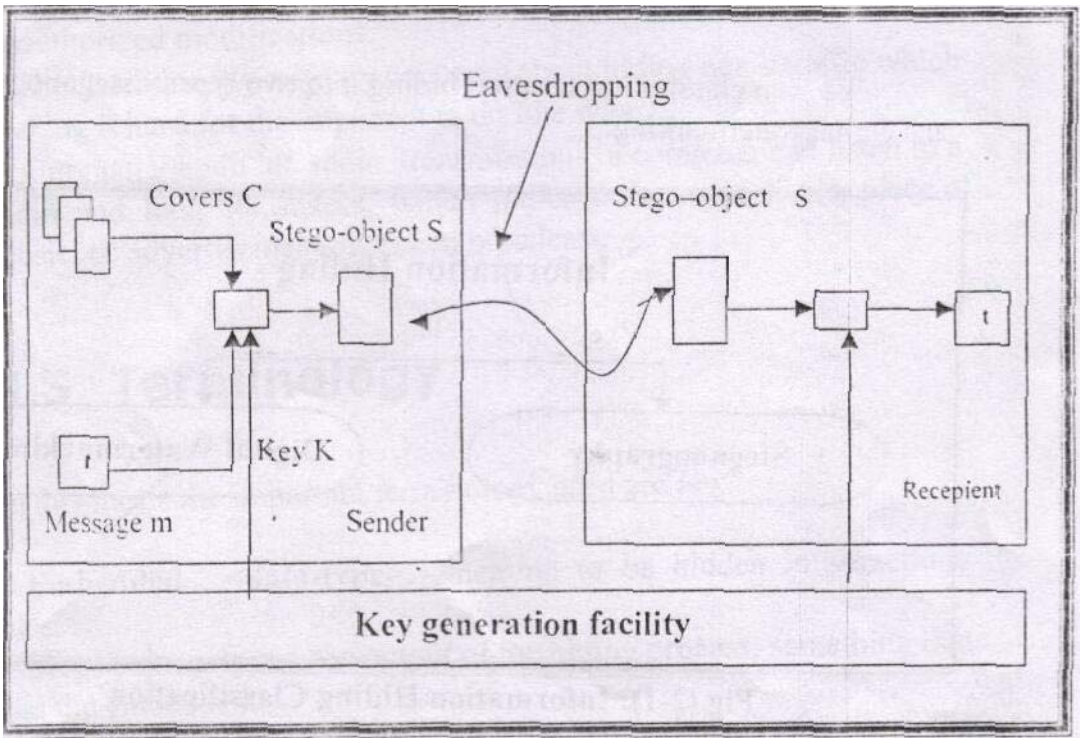


Figure 2: General principle of steganography

Cover could be any computer readable data such as image files, digital sound or written text.
Only Recipient should be able to extract the message in the correct way, of course this is possible only if there is a shared secret between the sender and the recipinent.

(6.4.1) Steganography Types

There are three main types of Steganography as illustrated in Figure 3.

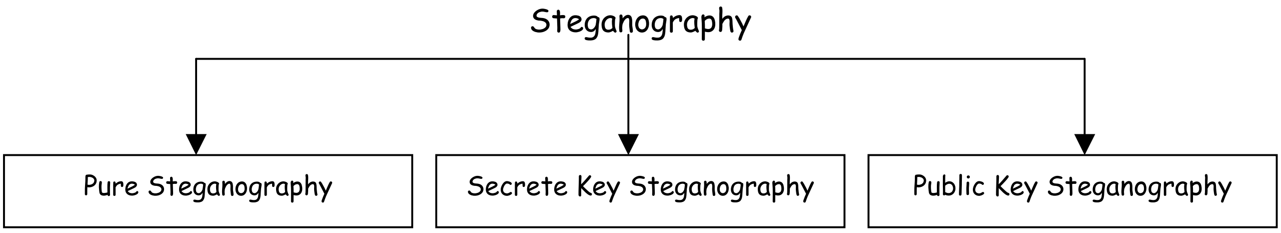


Figure 3: Steganography Types

(a) Pure Steganography

We call a Steganography system which does not require the prior exchange of some secret information (like a stego-key) pure steganography. Formally, the embedding process can be described as a mapping E: C X M → C, where C is the set of possible covers and M the set of possible messages. The extraction process consists of a mapping D: C → M , extracting the secret message out of a cover.

(b) Secret Key Steganography

A secret key Steganography system is similar to a symmetric cipher: the sender chooses cover C and embeds the secret message into C using a secret Key K. If the key used in the embedding process is known to the receiver, he can reverse the process and extract the secret message.

Anyone who does not known the secret key should not be able to obtain evidence of the encoded information. Again, the cover C and the stego-object can be perceptually similar. The mapping of process, if K. the set of secret keys,

$E_k : C \times M \times K \rightarrow C$ and $D_k : C \times K \rightarrow M$ with the property that $D_k(E_K(c, m, k), k) = m$ for all $m \in M$, $c \in C$ and $k \in K$, is called a secret key steganography system.

(C) Public key Steganography

Public Key steganography system require the use of two keys, one is private and the other one is public key; the public key is stored in a public database whereas the public key is used in the embedding process, the secret key is used to reconstruct the secret message. In public -key cryptography, it is not necessary for two people to share a secret key to establish a secure channel. One only needs to know the other's public key. This suggests a possible approach to steganography in which a secret key does not have to be agreed upon by sender and recipient prior to imprisonment.

(6.4.2) Steganography in text

- **Line sfiift coiling**

  In this method, text lines are vertically shifted to encode the document uniquely. Encoding and decoding can generally be applied either to the format file of document, or the bitmap of page image.

  By moving every second line of document either 1/300 of an inch up or down. The line-shift coding worked particularly well, and documents could still be completely decoded even after the tenth photocopy.

- **Word-Shift Coding**

  In word shift coding, codewords are coded into a document by shifting the horizontal locations of words. Within text lines, while maintaining a natural spacing appearance. This encoding can also be applied to either the formal file or the page image bitmap. The method, of course, is only applicable to documents with variable spacing between adjacent words, such as in documents that have been text-justified, as result of this variable spacing, it is necessary to have the original image, or to at least know the spacing between words in the uncoded document.

(6.4.3) Case study: Text in image

Image files

Digital technology offers new ways to apply steganography techniques, including the ability to hide information inside digital images. A digital image is "an array of numbers that represent light intensities at various points". Combined, these light intensities or pixels form the image's raster data. Images with 640 x 480 pixels and 256 colors can contain up to 300 kilo-bits of data. But it is more typical to see digital images in sizes of eight-bit or 24-bit files. This provides an excellent opportunity for hiding information, especially in image sizes of 24-bits.

Each pixel on a computer monitor selects from three primary color variations: red, blue, and green. Each color is represented by a single storage byte. With 24-bit images, three bytes are allocated for each primary color (hence eight bits per byte multiplied by three bytes).

Represented in binary values, for instance, a white background is 11111111, 11111111, 11111111. Pixel representation makes up a file's size. Thus a 24-bit image displayed in high resolution (1,024 x 768) has more than 2 million pixels, producing a file over 2MB in size. The larger the file, the greater opportunity there is to apply steganography techniques. The downside to this of course is that large file sizes might induce unwanted suspicions.

To deal with this, file compression is used. There are two kinds available today: lossy and lossless. Both methods compress files to save storage space, but do so differently. This is important because certain compression applications can interfere with hidden messages. Lossy compression is the most efficient space saver, but does not retain the original image's exactness. JPEG (Joint Photographic Experts Group) is an example of such compression. A lossless approach, in contrast, retains the integrity of the original image. Images saved as GIF (Graphic Interchange Format) or BMP (bitmap file) apply lossless compression.

Inserting Hidden Data

Two files are required for steganography to work. The first file is an innocuous cover image that will host the second file containing hidden information. The hidden message can be anything that is embeddable into a bit stream such as plain text or cipher text. There are several methods to hide information in digital images, from taking advantage of noisy areas that draw less attention in an image, to scattering the message randomly throughout the image. A brief discussion on each of these approaches is in order before continuing.

Least Significant Bit

A color byte contains eight bits, each of which varies in terms of impact on the resulting color. The least significant bit (or the final bit in a stream) affects the smallest change of the eight bits. On the other hand, the first bit of the stream has the largest influence on color selection. For instance, as Denning illustrates, the least and most significant bit are similar to the hour and second hand on a clock. While a change in the second hand alters time very slightly, a change in the hour hand is extreme.

If the least significant bit of every bit stream were to be allocated for a hidden message, the resulting image file would appear unaltered. Moreover, the larger the bit size, the more subtle the change. For example, using the least significant bit in a 24-bit image size, the original raster data for three pixels is:

00101100	10101100	10101000
00101011	01101000	00101011
00101010	00111001	00101010

Inserting the binary value for A (10000011) would result in:

0010110**1**	1010110**0**	1010100**0**
0010101**0**	0110100**0**	0010101**0**
0010101**0**	0011100**1**	0010101**1**

The least significant bit approach is simple to understand and use, but it is largely vulnerable to changes in data due to file compression. Since JPEG compression is so efficient in reducing file sizes, most image traffic over the Internet utilizes it. However, as stated previously, the lossy compression technique JPEG employs may alter the least significant bit.

(6.5) Watermarking

Through the use of advanced computer software, authors of images, music and software can place, a hidden "trademark" in their product, allowing them to keep a check on piracy. This is commonly known as watermarking.

Watermarking is typically short in length or bits, once added, a watermarking must be robust to removal, and reliably detected even after typical image transformation such as rotation, translation, cropping, and quantization.

The watermarking can be of any nature such as a number, text, or an image.

(6.5.1) Watermarking types

(a) Visible Watermark

The Visible watermarking purpose is to discourage unauthorized distribution of digital images, By seeing the watermark, a person is less likely, to copy or distribute the image without permission.

(b) Invisible Watermark

The invisible watermark not revealed until some action is taken upon the source to uncover it.

(6.5.2) Watermarking technique

- **Simple Watermarking**

A very simple yet widely used technique for watermarking images is to add a pattern on top of an existing image. Usually this pattern is an image itself - a logo or something similar, which distorts the underlying image.

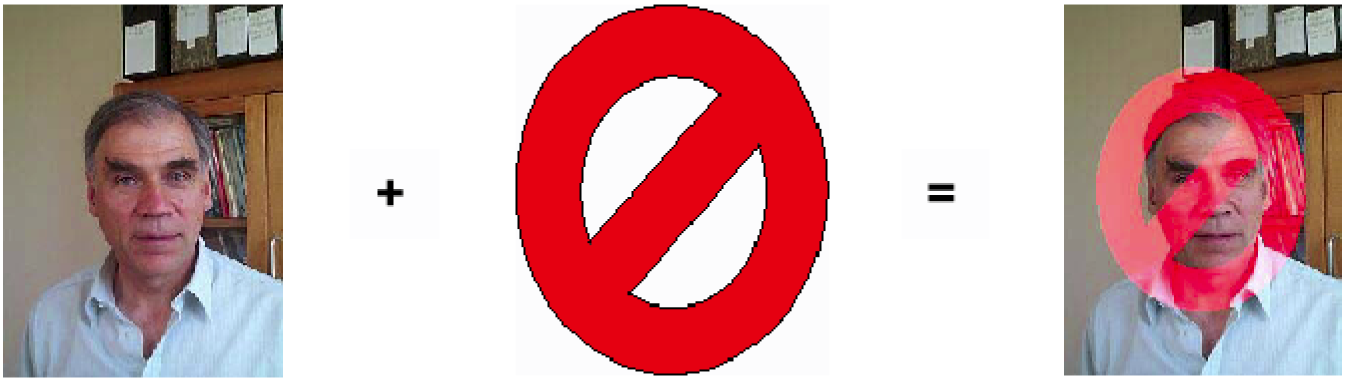


Figure 4. Visible watermarking.

In the example above, the pattern is the middle image while the portrait picture of Mr.X is the image being watermarked. In a standard image editor it is possible to merge both images and get a watermarked image. As long as you know the watermark, it is possible to reverse any adverse effects so that the original doesn't need to be kept. This method is only really applicable to watermarking, as the pattern is visible and even without the original watermark, it is possible to remove the pattern from the watermarked image with some effort and skill.
The equations that can used her is

Additive watermarking:

$C=A+\mu B$ where $\mu=0.001$ for example. μ i known as scaling factor.

Or Multiplicative watermarking:

$$C=A+\mu*A*B$$

- **LSB – Least Significant Bit Hiding (Image Hiding)**

This method is probably the easiest way of hiding information in an image and yet it is surprisingly effective. It works by using the least significant bits of each pixel in one image to hide the most significant bits of another. So in a JPEG image for example, the following steps would need to be taken

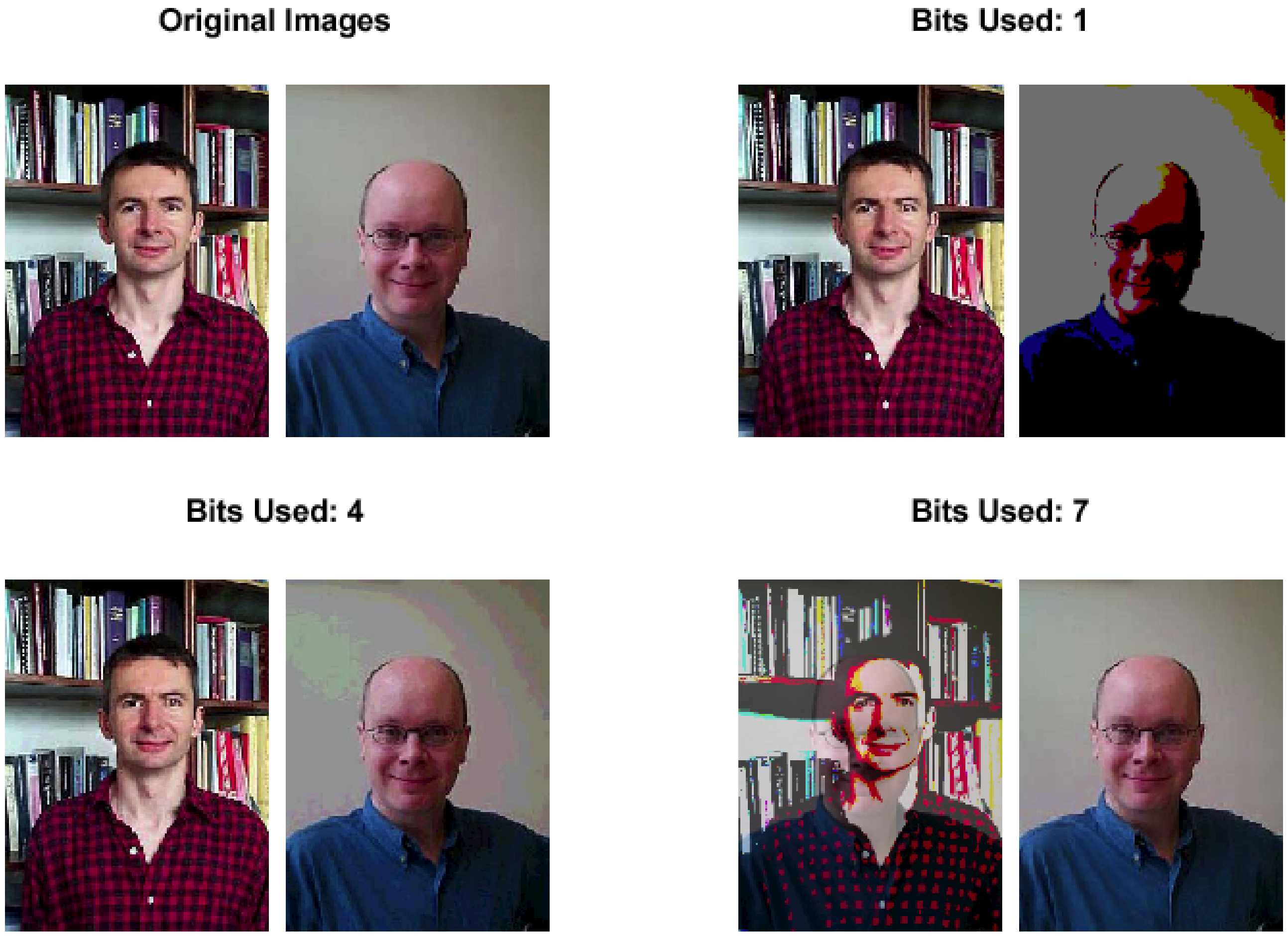
1. First load up both the host image and the image you need to hide.
2. Next chose the number of bits you wish to hide the secret image in. The more bits used in the host image, the more it deteriorates. Increasing the number of bits used though obviously has a beneficial reaction on the secret image increasing its clarity.

3. Now you have to create a new image by combining the pixels from both images. If you decide for example, to use 4 bits to hide the secret image, there will be four bits left for the host image. (PGM - one byte per pixel, JPEG - one byte each for red, green, blue and one byte for alpha channel in some image types)

   Host Pixel: 10110001
   Secret Pixel: 00111111
   New Image Pixel: 10110011

4. To get the original image back you just need to know how many bits were used to store the secret image. You then scan through the host image, pick out the least significant bits according the number used and then use them to create a new image with one change - the bits extracted now become the most significant bits.

   Host Pixel: 10110011
   Bits used: 4
   New Image: 00110000



Original Images

Bits Used: 1

Bits Used: 4

Bits Used: 7

Figure 5: Least significant bit hiding

To show how this technique affects images, Figure 6 shows examples using different bit values. Mr.X image on the left is the host image while Mr.Y on the right is the secret one we wish to hide.

This method works well when both the host and secret images are given equal priority. When one has significantly more room than another, quality is sacrificed. Also while in this example an image has been hidden, the least significant bits could be used to store text or even a small amount of sound. All you need to do is change how the least significant bits are filled in the host image. However this technique makes it very easy to find and remove the hidden data.

(6.6) Difference between Steganography and Digital watermarking

There are many differences between steganography and digital walermarkuig and the major differences are:

	Steganography	**Digital watermarking**
1	It is always invisible.	Sometimes visible and sometimes i invisible.
2	The information hiding by steganography just hides any information.	The information hiding by watermarking is always associated to be digital object you be protected or to its owner.
3	The robustness of steganography is mainly concerned with detection of the hidden message.	The robustness of digital watermarking concerns potential removal by a pirate.
4	Steganography communication are usually point-to-point.	Watermarking techniques are usually one to many.

(6.7) Requirements Of Hiding Information Digitally

There are many different protocols and embedding techniques that enable us to hide data in a given object. However, all of the protocols and techniques must satisfy a number of requirements so that steganography can be applied correctly. The following is a list of main requirements that steganography techniques must satisfy:

- The integrity of the hidden information after it has been embedded inside the stego object must be correct. The secret message must not change in any way, such as additional information being added, loss of information or changes to the secret information after it has been hidden. If secret information is changed during steganography, it would defeat the whole point of the process.

- The stego object must remain unchanged or almost unchanged to the naked eye. If the stego object changes significantly and can be noticed, a third party may see that information is being hidden and therefore could attempt to extract or to destroy it.
- In watermarking, changes in the stego object must have no effect on the watermark. Imagine if you had an illegal copy of an image that you would like to manipulate in various ways. These manipulations can be simple processes such as resizing, trimming or rotating the image. The watermark inside the image must survive these manipulations, otherwise the attackers can very easily remove the watermark and the point of steganography will be broken.
- Finally, we always assume that the attacker knows that there is hidden information inside the stego object.