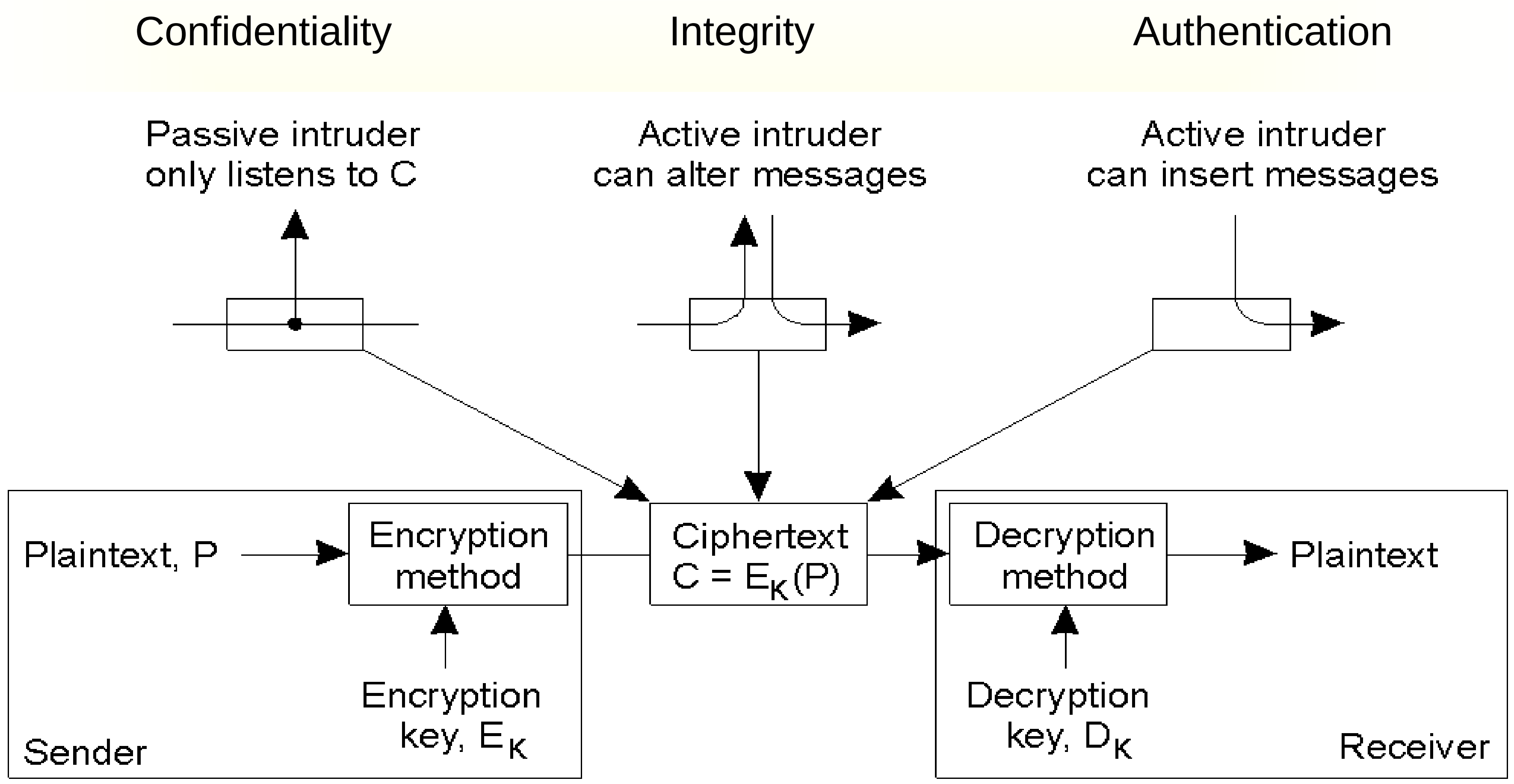
Chapter 3 – Cryptography and Encryption Techniques

3.1 Introduction

- Encryption is required for confidentiality, integrity and authentication (to assure that a message comes from the alleged source); to protect messages from intruders
  - Eavesdropping (listening/spying the message) - Confidentiality
    - An intruder may try to read the message
    - If it is well encrypted, the intruder will not know the content
    - However, just the fact the intruder knows that there is communication may be a threat (Traffic analysis)
  - Modification/Integrity
    - Modifying a plaintext is easy, but modifying encrypted messages is difficult
  - Insertion of Messages/Integrity
    - Inserting new message into a ciphertext is difficult
    - *Milikit sitegn* in some societies of our country

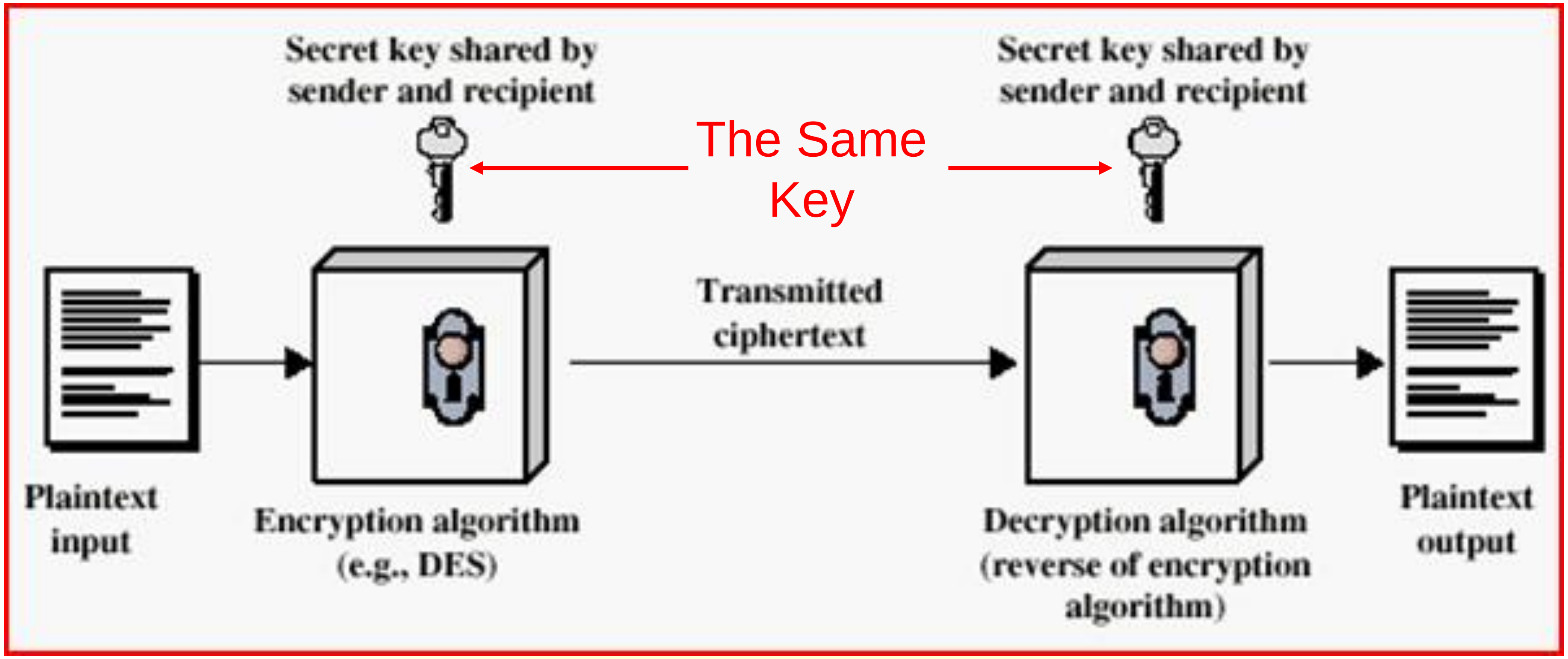
Intruders and eavesdroppers in communication



- Terminology
  - Cryptography: Refers to schemes for encryption and decryption; It comes from the Greek words for secret writing
  - Encryption: The process by which plaintext is converted into ciphertext
  - Decryption: Recovering plaintext from the ciphertext
  - Secret key: Used by the encryption algorithm. In a classical (symmetric key) cryptography, the same secret key is used for encryption and decryption
  - Cryptanalysis: The study of “breaking the code”. Cryptanalysts!
  - Cryptology: Cryptography + cryptanalysis

- Cryptography has five ingredients
  - Plaintext (or Cleartext): the original message that is fed into the algorithm as input
  - Encryption algorithm: performs various substitutions and transformations on the plaintext
  - Secret Key: is also input to the algorithm; the exact substitutions and transformations performed by the algorithm depend on the key; larger key size means greater security but may decrease encryption/decryption speed
  - Ciphertext: the scrambled message produced as output. It depends on the plaintext and the secret key; for a given message, two different keys will produce two different ciphertexts
  - Decryption algorithm: the encryption algorithm run in reverse. It takes the ciphertext and the same secret key (in symmetric key cryptography) and produces the original plaintext

- Simplified Symmetric Encryption Model


Secret key shared by sender and recipient
The Same Key
Secret key shared by sender and recipient
Transmitted ciphertext
Plaintext input
Encryption algorithm (e.g., DES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

- The need for cryptography
  - If you have the best firewall, very tight security policies, hardened operating systems, virus scanners, intrusion-detection/prevention system, antispyware, and every other computer security angle covered but send your data in raw, plain text, then you simply are not secure

- Description
  - A sender S wants to transmit message M to a receiver R
  - To protect the message M, the sender first encrypts it into an unintelligible message M’
  - After receipt of M’, R decrypts the message to obtain M
  - M is called the plaintext: what we want to encrypt
  - M’ is called the ciphertext: the encrypted output

- Note: Steganography is a technique for hiding a secret message within a larger one so that others cannot discern the presence or contents of the hidden message; it is used to claim ownership; it is not encryption

- **Notation**
  - Given
    - P = Plaintext
    - C = Ciphertext
  - $C = E_K(P)$ Encryption
  - $P = D_K(C)$ Decryption

    $\Rightarrow P = D_K(E_K(P))$

    $\Rightarrow C = E_K(D_K(C))$

- The two basic building blocks of all encryption techniques are substitution and transposition
  - Substitution: replace a block of data with another
  - Transposition: rearrange the order in which data is presented
- Caesar Cipher - Early Example of a Substitution Cipher by Julius Caesar
  - Each character of a message is replaced by a character three positions down in the alphabet (the alphabet is wrapped around, so that the letter following Z is A)
    - Plain text: ARE YOU READY
    - Ciphertext: DUH BRX UHDGB
  - If we represent each letter of the alphabet by an integer that corresponds to its position in the alphabet (A=1, B=2, … Z=26), the formula for replacing each character ‘p’ of the plaintext with a character ‘c’ of the ciphertext can be expressed as
    - $c = E_3(p) = (p + 3) \bmod 26$ (If a is an integer and n is a positive integer, we define a mod n to be the remainder when a is divided by n. The integer n is called the modulus)

- A more general version of this cipher that allows for any degree of shift
  - $c = E_k(p) = (p + k) \bmod 26$
- The formula for decryption would be
  - $p = D_k(c) = (c - k) \bmod 26$
- In these formulas
  - ‘k’ is the secret key; the symbols ’E’ and ’D’ stand for Encryption and Decryption respectively, and p and c are characters in the plain and ciphertext respectively
- If it is known that a given ciphertext is a Caesar cipher, then a brute-force attack is easily performed: simply try all the 25 possible keys

- Transposition Cipher
  - A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters
  - The simplest of such ciphers is the rail fence technique, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows
  - For example, to encipher the message “MEET ME AFTER THE GOOD PARTY” with a rail fence of depth 2 (number of rows, which is the key), we write the following

```
M   E   M   A   T   R   H   G   O   P   R   Y
  E   T   E   F   E   T   E   O   D   A   T
```

    - The ciphertext is MEMATRHGOPRYETEFETEODAT

- With depth of 3

```
M       M       T       H       O       R
  E   T   E   F   E   T   E   O   D   A   T
    E       A       R       G       P       Y
```

    - The ciphertext is MMTHORETEFETEODATEARGPY

- This would be trivial to cryptanalyze. A more complex scheme is to write the message in a rectangle, row by row, and read the message column by column, but permute the order of the columns. The order of the columns is the key to the algorithm
- Example:
  - Plain text: Attack Postponed Until Two AM
  - Key: 4 3 1 2 5 6 7

A	T	T	A	C	K	P
O	S	T	P	O	N	E
D	U	N	T	I	L	T
W	O	A	M	X	Y	Z

  - To encrypt, start with the column that is labeled 1, in this case column 3. Write down all the letters in that column. Proceed to column 4, which is labeled 2, then column 2, then column 1, then columns 5, 6, and 7
  - Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ
  - There are other more complex methods such as the Rotor Machines which you can read about
  - Exercise: Decrypt the above ciphertext

- There are two forms of encryption systems
  - Symmetric (also called Traditional or Secret-key or Private key or Single key) cryptosystem
  - Asymmetric (also called Public key) cryptosystem

3.2 Symmetric Cryptosystem

- The same key is used to encrypt and decrypt a message
  - $C = E_K(P)$
  - $P = D_K(C) \quad \Rightarrow \quad P = D_K[E_K(P)]$
  - Has been used for centuries in a variety of forms
- The key has to be kept secret
- The key has to be communicated using a secure channel; major problem
- It is still in use in combination with public key cryptosystems due to some of its advantages, mainly efficiency
- Properties of an Encryption Function
  - It is computationally infeasible to find the key K when given the plaintext P and the associated ciphertext C $[E_K(P)]$
  - It should also be computationally infeasible to find another key K' such that $E_K(P) = E_{K'}(P)$; Uniqueness

3.2.1 DES - A Popular Example of Symmetric Cryptosystem

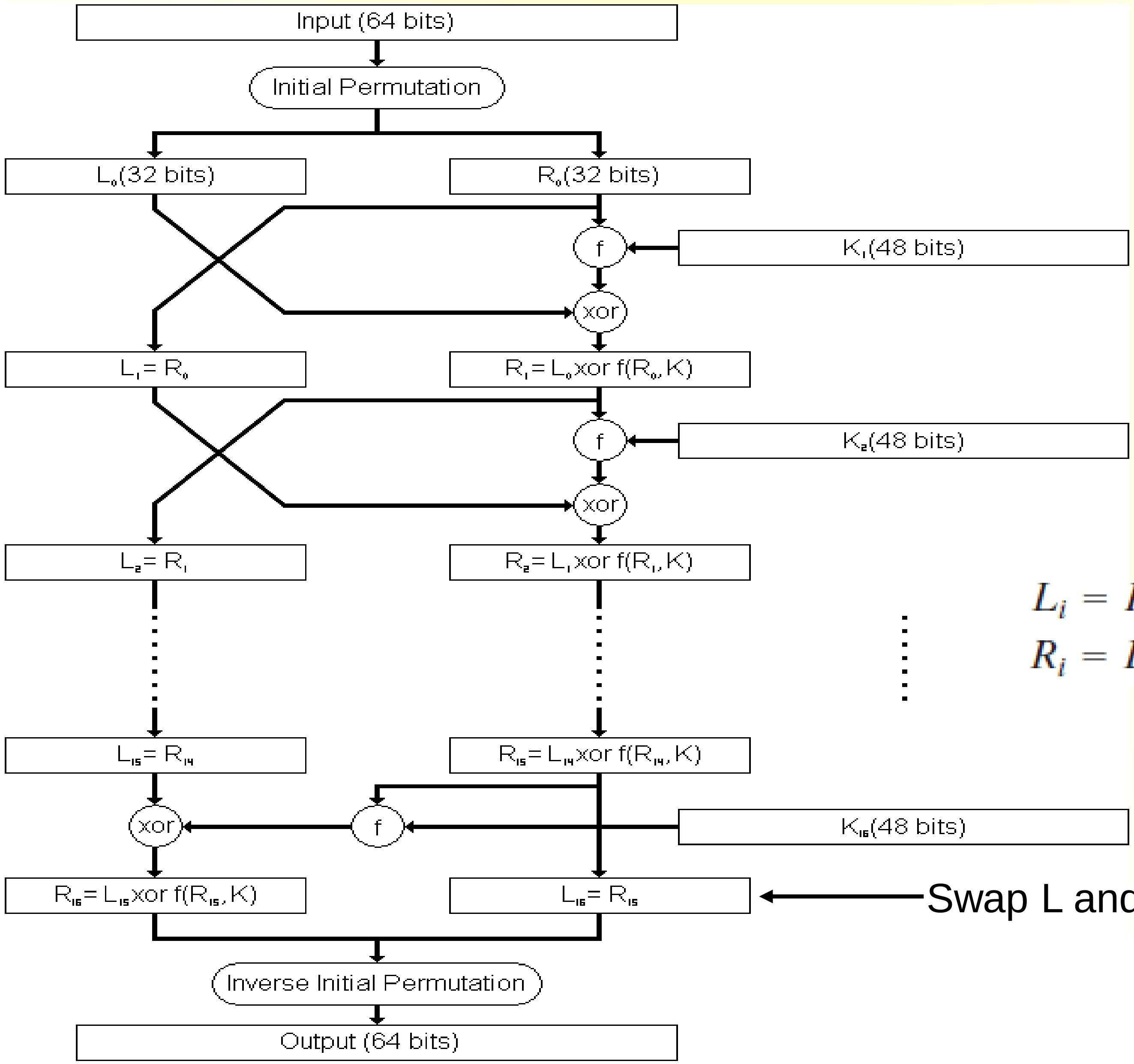
- In 1973, the NBS (National Bureau of Standards, now called NIST - National Institute of Standards and Technology) published a request for an encryption algorithm that would meet the following criteria:
  - have a high security level
  - be easily understood
  - not depend on the algorithm's confidentiality
  - be adaptable and economical
  - be efficient
- In late 1974, IBM proposed "Lucifer", which was then modified by NSA (National Security Agency) in 1976 to become the DES (Data Encryption Standard)
  - DES was then approved by NBS in 1978 and was standardized by ANSI under the name of ANSI X3.92, also known as DEA (Data Encryption Algorithm)

- DES utilizes block cipher, which means that during the encryption process, the plaintext is broken into fixed length blocks of 64 bits
  - A block cipher processes the input one block of elements at a time, producing an output block for each input block; larger block sizes mean greater security but reduced encryption/ decryption speed; a block size of 128 bits is a reasonable tradeoff and is nearly universal among recent block cipher designs
  - A stream cipher processes the input elements continuously, producing output one element at a time, as it goes along
    - e.g., substitution (Caesar Cipher)
- The key in DES is 56 bits; 8-bit out of the total 64-bit block key is used for parity check (for example, if odd parity is used, each byte has an odd number of 1s)
- 56-bit key gives 2^{56} ($\cong 7.2*10^{16}$) possible key variations

- DES algorithm involves carrying out combinations, substitutions and permutations between the text to be encrypted and the key, while making sure the operations can be performed in both directions (for decryption)
- The combination of substitutions and permutations is called a product cipher
- DES was best suited for implementation in hardware, probably to discourage implementations in software, which tend to be slow by comparison during that time
- Modern computers are so fast that satisfactory software implementations for DES are possible
- DES is the most widely used symmetric algorithm despite claims whether 56 bits is long enough to guarantee security

- DES Encryption
  - Data is divided into 64-bit blocks; the key is 56 bits
  - The processing has three phases
  - Phase 1
    - The 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input; no elements are added or deleted or replaced, rather the order in which the elements appear in the sequence is changed
  - Phase 2
    - The 64 bits are then divided into two 32-bit halves called L and R. The encryption then proceeds through 16 rounds of the same function, each using the L and R parts, and a subkey
    - In each round, the new L part is simply a copy of the incoming R part
    - The R and subkeys are processed in the so called f-function, and exclusive-or of the output of the f-function with the existing L part to create the new R part

- Phase 3
  - The L and R parts are swapped
  - The preoutput is passed through a permutation that is the inverse of the initial permutation (IP^{-1}), to produce the 64-bit ciphertext



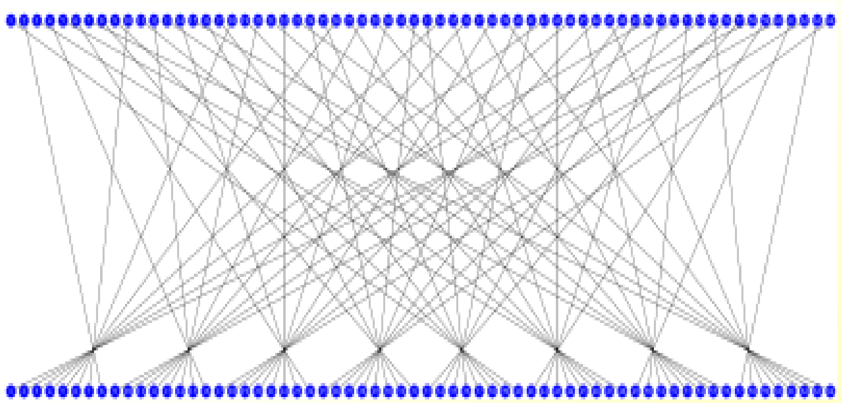
$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

$\longleftarrow$ Swap L and R

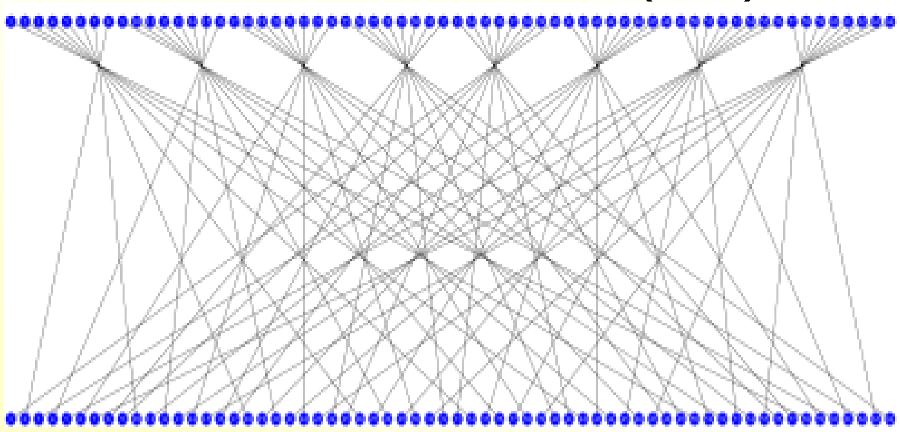
Structure of DES Algorithm

- DES – Permutation: The initial permutation and its inverse are defined by tables; in all tables, the numbers are the bit positions

Initial Permutation (IP)



Inverse Permutation (IP^{-1})



IP (e.g., IP(1) = 58; IP(2) = 50, etc.)

58	50	42	34	26	18	10	2	L
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	R
59	51	43	35	27	19	11	3	
61	53	45	37	29	21	13	5	
63	55	47	39	31	23	15	7	

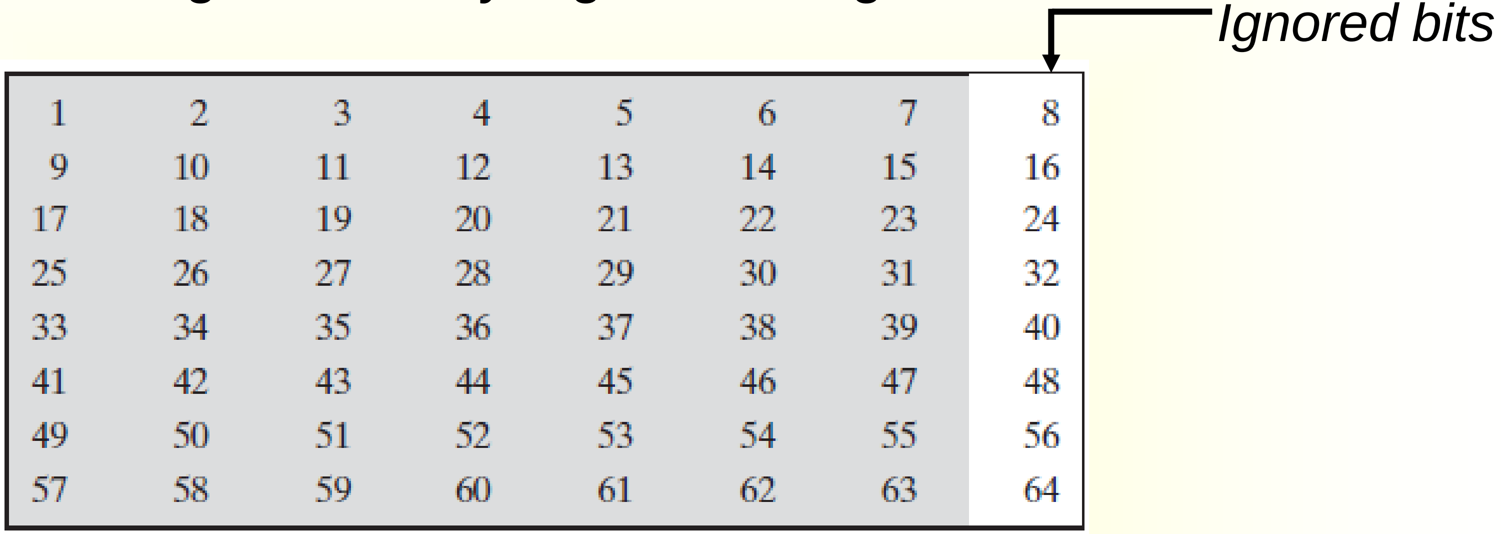
IP^{-1} (e.g., $IP^{-1}(1) = 40$; $IP^{-1}(2) = 8$, etc.)

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

"First Bit of the output is taken from the 58th bit of the input, etc...

- Generating Subkeys
  - Initially, the key is passed through a permutation function
  - Then, for each of the sixteen rounds, a subkey (K_i) is produced by the combination of a left circular shift and a permutation
  - The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits

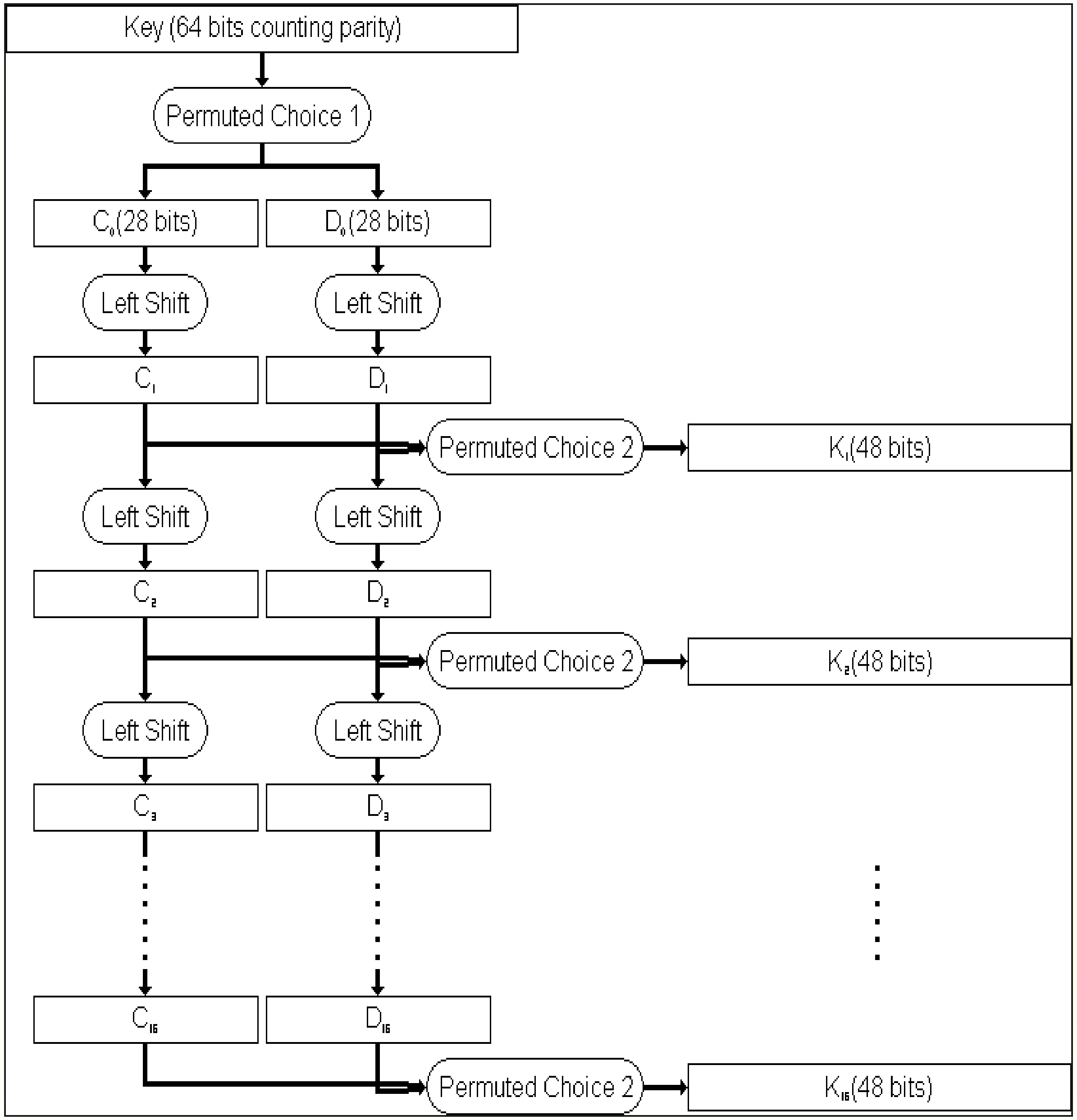
- To generate the subkeys, we start with the 56-bit key (64 bits - with parity bits); the bits of the key are numbered from 1 through 64; every eighth bit is ignored


Ignored bits

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	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Input Key (the numbers are the bit positions)

- These are permuted and divided into two halves called C and D (Permutation Choice 1)
- For each round, C and D are each shifted left circularly one or two bits (the number of bits depending on the round)
- The 48-bit subkey is then selected from the current C and D bits using Permutation Choice 2



DES- Algorithm - Key Schedule and Subkey Generation

PC-1: Permutation Choice 1

- Extracts and permutes only 56-bit of the original 64-bit key (excluding parity bits 8,16, 24, 32, 40, 48, 56, 64)

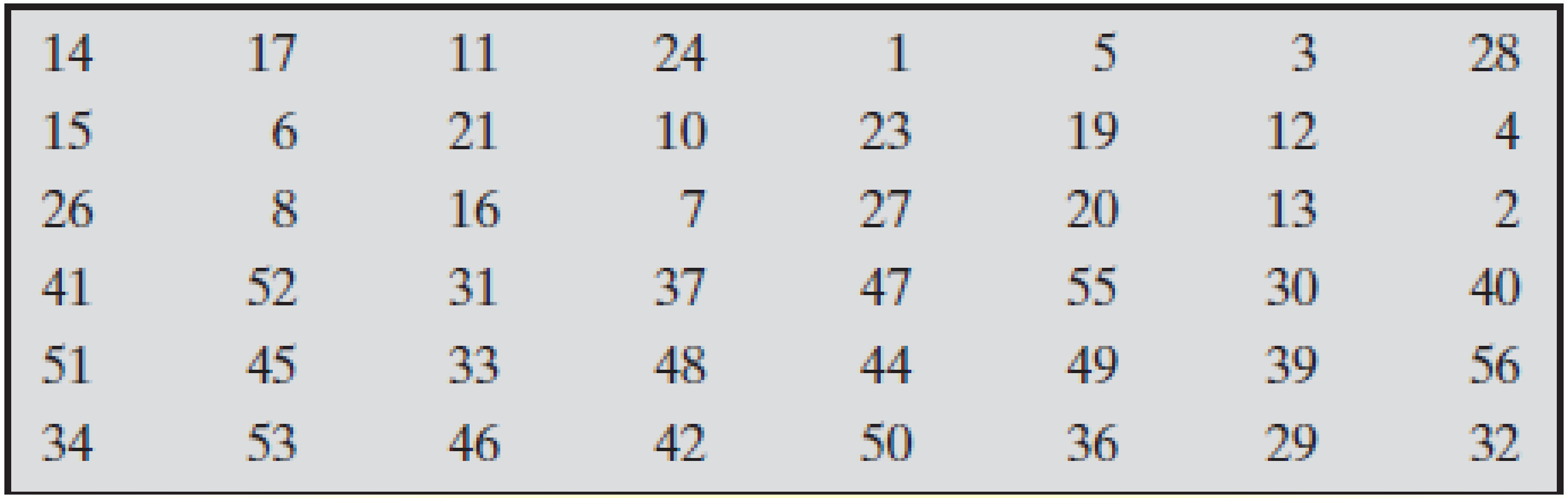
57	49	41	33	25	17	9	
1	58	50	42	34	26	18	C
10	2	59	51	43	35	27	
19	11	3	60	52	44	36	
63	55	47	39	31	23	15	
7	62	54	46	38	30	22	D
14	6	61	53	45	37	29	
21	13	5	28	20	12	4	

Schedule of Left Shifts

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

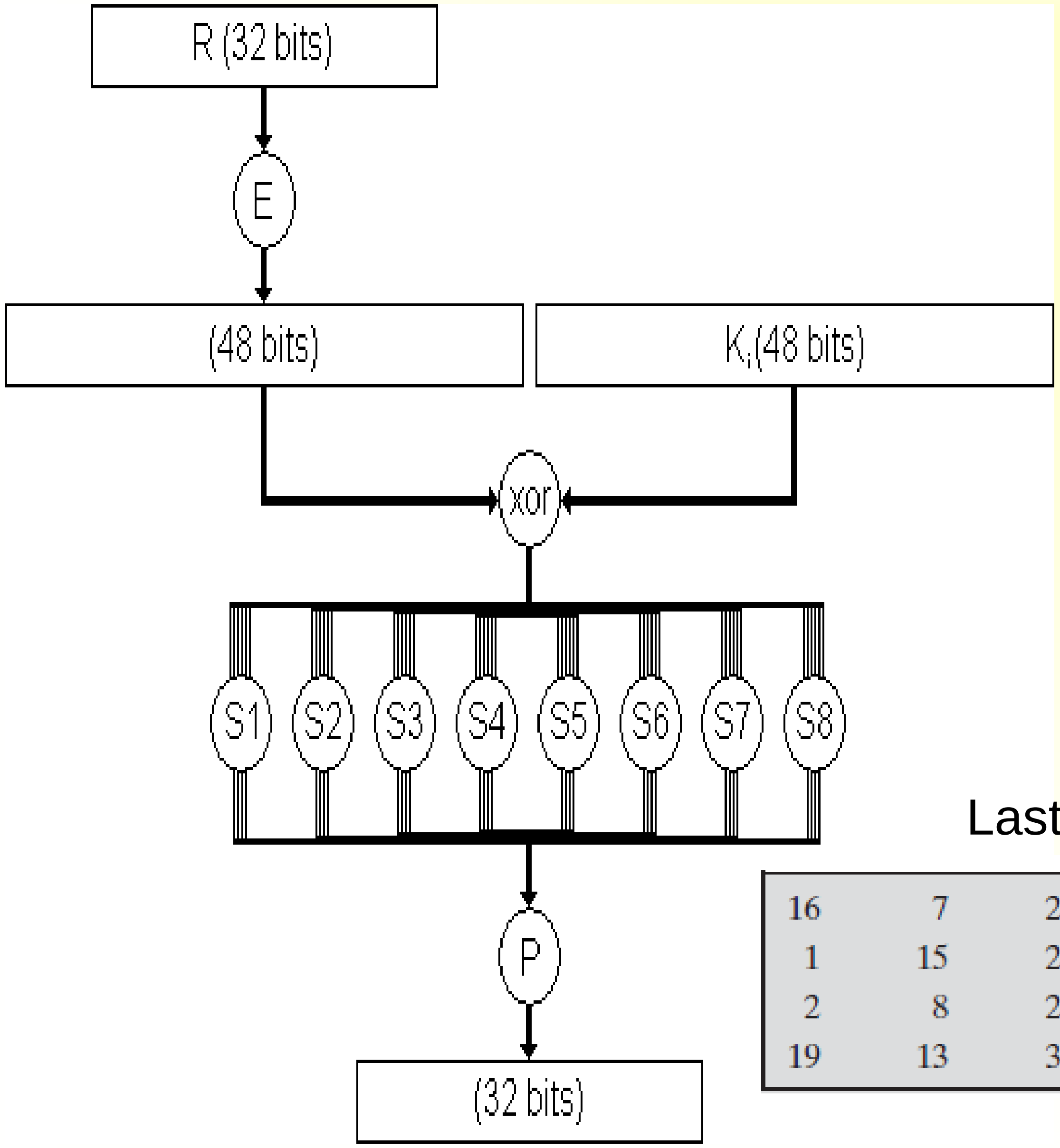
PC-2: Permutation Choice 2

- Selects or extracts the 48-bit subkey for each round from the 56-bit key-schedule



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

- The f-function (Some call it the mangler function)
  - The f-function mixes the bits of the R portion using the Subkey for the current round. First the 32-bit R value is expanded to 48-bits using a permutation E. That value is then exclusive-or'ed with the subkey
  - The 48-bits are then divided into eight 6-bit chunks, each of which is fed into an S-Box (Substitution-Box or Substitution Table) that mixes the bits and produces a 4-bit output. A little bit funny operation here!!
  - Those eight 4-bit outputs are combined into a 32-bit value (8*4 = 32), and permuted once again to give the output of the f-function

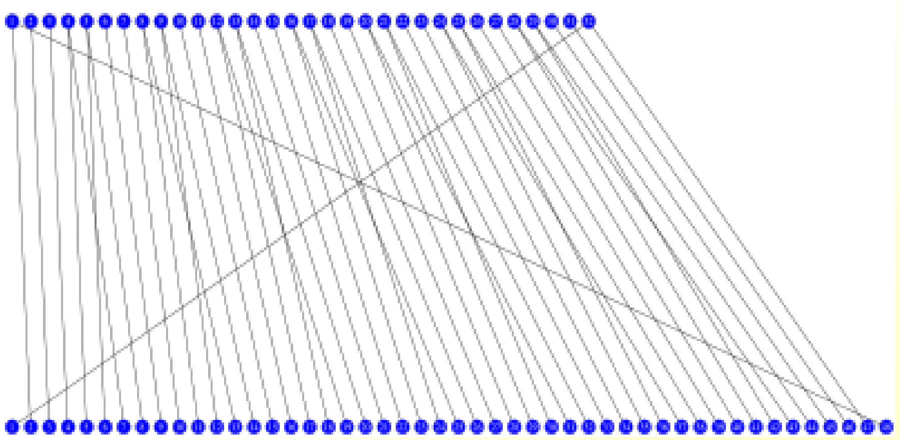


Last Permutation

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

DES- Algorithm, the f-function

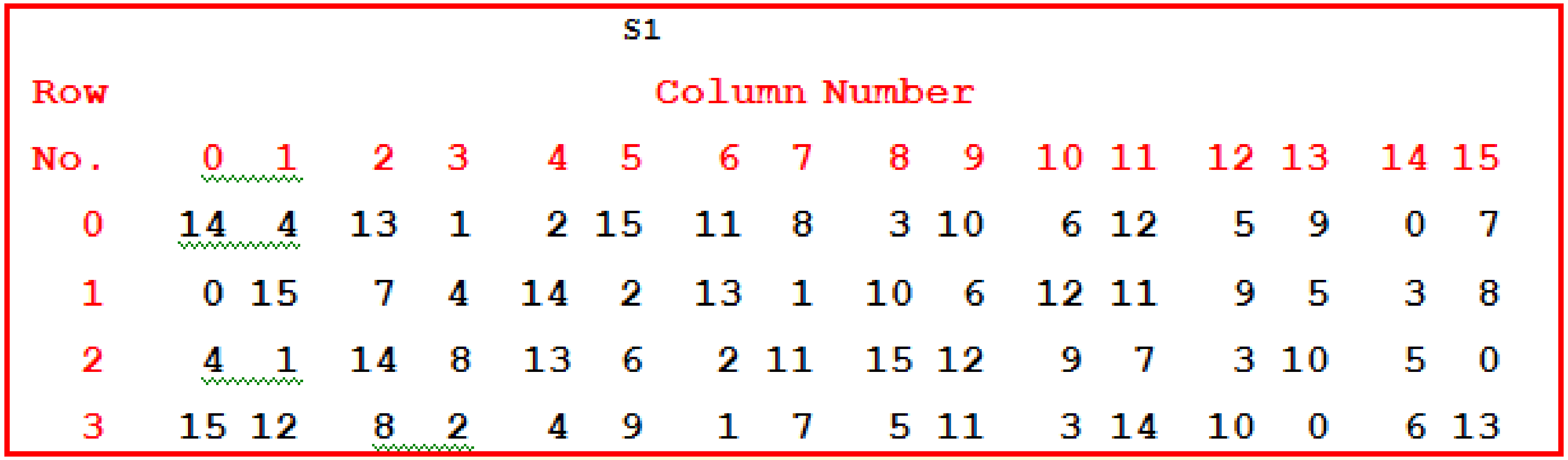
f-function: Expansion/Permutation (E)



The 32-bit half-block of data is expanded to 48 bits; those in red are all repeated at different positions

E table

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


S1

Row No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

DES S-Box for S1- There is one box for each of the 8 S-Boxes

- If ***S_1*** is the function defined in this table and ***B*** is a block of **6 bits**, then ***S_1(B)*** is determined as follows: The first and last bits of ***B*** represent in base 2 a number in the decimal range 0 to 3 (or binary 00 to 11). Let that number be ***i***. The middle 4 bits of ***B*** represent in base 2 a number in the decimal range 0 to 15 (binary 0000 to 1111). Let that number be ***j***. Look up in the table the number in the ***i***-th row and ***j***-th column. It is a number in the range 0 to 15 and is uniquely represented by a 4 bit block. That block is the output ***S_1(B)*** of ***S_1*** for the input ***B***. For example, for input block ***B*** = 011011 the first bit is "0" and the last bit "1" giving 01 as the row. This is row 1. The middle four bits are "1101". This is the binary equivalent of decimal 13, so the column is column number 13. In row 1, column 13 appears 5. This determines the output; 5 is binary 0101, so that the output is 0101. Hence ***S1***(011011) = 0101. **Find S1(101011)?**
- **The design criteria for the S-boxes and for the entire algorithm were not made public**

Note on XOR

- It outputs true only when both inputs differ

  1 1 0 1

  $\underline{1\ 0\ 0\ 1}$

  0 1 0 0

- XORing has a very interesting property in that it is reversible; XORing the result with the second number gives back the first number; XORing the result with the first number gives the second number

0 1 0 0	result
$\underline{1\ 0\ 0\ 1}$	second number
1 1 0 1	first number

- The XOR ($\oplus$) operation opens the door for a simple encryption
  - Encryption: (Plain text in Binary) $\oplus$ Key = Ciphertext
  - Decryption: (Ciphertext) $\oplus$ Key = Plain text in Binary

- Decryption is identical to encryption, except that the subkeys are used in the opposite order. That is, subkey 16 is used in round 1, subkey 15 is used in round 2, etc., ending with subkey 1 being used in round 16
- The Avalanche Effect
  - A desirable property of any encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in the ciphertext
  - In particular, a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext. This is referred to as the avalanche effect
  - If the change were small, this might provide a way to reduce the size of the plaintext or key space to be searched leading to brute-force attack

- DES - Attacks
  - Types of attacks (cracking) in all types of encryption
    - The attacker has only the ciphertext and his/her goal is to find the corresponding plaintext
    - The attacker has the ciphertext and the corresponding plaintext and his/her goal is to find the key
    - In both cases the attacker may or may not know the algorithm
  - A good cryptosystem protects against all types of attacks

- The security of encryption depends on the secrecy of the key, not the secrecy of the algorithm (Security through obscurity is not a good strategy)
  - Keeping the algorithm secret means to invent, test, and install a new one when the old is discovered which is very difficult
  - Keep only the key secret; so that it can be changed as often as needed
- The two types of attacks on an encryption algorithm are
  - Cryptanalysis: based on properties of the encryption algorithm
  - Brute-force: also called exhaustive key search, involves trying all possible keys; This is the most basic method of attack for any cipher

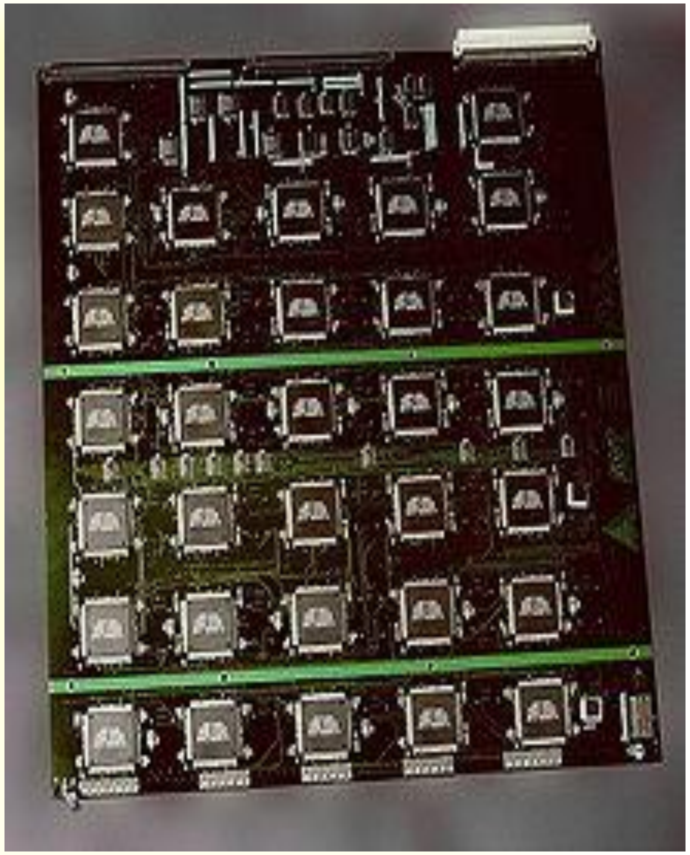
- An encryption scheme is said to be computationally secure if either of the following two criteria are met
  - The cost of breaking the cipher exceeds the value of the encrypted information
  - The time required to break the cipher exceeds the useful lifetime of the information
- Unfortunately, it is very difficult to estimate the amount of effort required to cryptanalyze ciphertext successfully
- The following is the average time required for exhaustive key search (brute-force attack) for various key sizes

Key Size (bits)	**Number of Alternative Keys**	**Time required at 1 Decryption/µs**	**Time required at 10^6 Decryption/µs**
32	$2^{32} = 4.3 \times 10^9$	2^{31}µs = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	2^{55}µs = 1142 years	10 hours
128	$2^{128} = 3.4 \times 10^{38}$	2^{127}µs = 5.4×10^{24} years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	2^{167}µs = 5.9×10^{36} years	5.9×10^{30} years

- The length of the key determines the number of possible keys, and hence the feasibility of the approach; with a key length of 56 bits, there are 2^{56} possible keys, which is approximately 7.2×10^{16} keys. Thus, a brute-force attack appears impractical
- Assuming that, on average, half the key space has to be searched, a single machine performing one DES encryption per microsecond would take more than a thousand years
- However, the assumption of one encryption per microsecond is overly conservative. As far back as 1977, Diffie and Hellman postulated that the technology existed to build a parallel machine with 1 million encryption devices, each of which could perform one encryption per microsecond. This would bring the average search time down to about 10 hours. They estimated that the cost of such a machine would be about $20 million in 1977 dollars

- By 1993, Wiener had proposed a key-search machine costing US$1 million which would find a key within 7 hours
- However, none of these early proposals were ever implemented
- The vulnerability of DES was practically demonstrated in 1997, where RSA Security sponsored a series of contests, offering a $10,000 prize to the first team that broke a message encrypted with DES for the contest. That contest was won by the DESCHALL Project, led by Rocke Verser, Matt Curtin, and Justin Dolske, using idle cycles of thousands of computers across the Internet
- The feasibility of cracking DES quickly was demonstrated in 1998 when a custom DES-cracker was built by the Electronic Frontier Foundation (EFF), a cyberspace civil rights group, at the cost of approximately US$250,000. Their motivation was to show that DES was breakable in practice as well as in theory

- The EFF's DES cracking machine contained 1,856 custom chips and could brute force a DES key in a matter of days
- Note: to supplement the brute-force approach, some degree of knowledge about the expected plaintext is needed. If the message is just plain text in English, then the result pops out easily, although the task of recognizing English would have to be automated. If the text message has been compressed before encryption, then recognition is more difficult. If the message is some more general type of data, such as a numerical file, and this has been compressed, the problem becomes even more difficult to automate



a DES Cracker circuit board fitted with several Deep Crack chips

- DES – Variants
  - Given the potential vulnerability of DES to a brute-force attack, there has been considerable interest in finding an alternative
  - One approach is to design a completely new algorithm, of which AES is a prime example
  - AES (Advanced Encryption Standard)
    - Is a block cipher that works on 128 bit blocks
    - It can have one of three key sizes of 128, 192, or 256 bits
    - NIST estimates that a machine that could crack 56-bit DES in one second (that is, try all 2^{56} keys in one second) would take approximately 149 trillion years to crack a 128-bit AES key
    - AES was selected by the US Government to be the replacement for DES and is now the most widely used symmetric key algorithm

- Triple DES (3DES)
  - Another alternative, which would preserve the existing investment in software and equipment, is to use multiple encryption with DES and multiple keys – Triple DES
  - Provides enhanced security by executing the core algorithm three times and the key length becomes 56*3 = 168-bits
  - With triple length key of three 56-bit keys K_1, K_2 & K_3, encryption follows an encrypt-decrypt-encrypt (EDE) sequence
    - Encrypt with K_1 Decrypt with K_2 Encrypt with K_3

    $C = E(K_3, D(K_2, E(K_1, P)))$
  - Decryption requires that the keys be applied in reverse order
    - Decrypt with K_3 Encrypt with K_2 Decrypt with K_1

    $P = D(K_1, E(K_2, D(K_3, C)))$

- As an alternative, we can use only two keys, i.e., setting K_3 equal to K_1 gives us a double length key K_1, K_2

  $C = E(K_1, D(K_2, E(K_1, P)))$

  $P = D(K_1, E(K_2, D(K_1, C)))$

- There is no cryptographic significance to the use of decryption for the second stage; its only advantage is that it allows users of 3DES to decrypt data encrypted by users of the older single DES by setting $K_2 = K_1$

  $C = E(K_1, D(K_1, E(K_1, P))) = E(K_1,P)$

  $P = D(K_1, E(K_1, D(K_1, C))) = D(K_1, C)$

- Advantages and disadvantages of DES
  - One advantage that DES offers is efficiency
  - The problem with DES is the same problem that all symmetric key algorithms have: How do you transmit the key without risking it becoming compromised? This issue led to the development of public key encryption
  - Another problem is that over time it has become no longer viable for modern encryption. The 56-bit key is simply not long enough

- Read about Cipher-Block Chaining

3.2.2 Other Symmetric Encryption Methods

- Blowfish
  - Is a symmetric block cipher
  - It uses a variable-length key ranging from 32 to 448 bits
  - This flexibility in key size allows to use it in various situations
  - It was designed in 1993 by Bruce Schneier
  - It has been analyzed extensively by the cryptography community and has gained wide acceptance
  - It is also a non-commercial (i.e., free of charge) product, thus making it attractive to budget-conscious organizations
- RC4
  - Is a symmetric stream cipher developed by Ron Rivest
  - The RC is an acronym for Ron’s Cipher
  - There are other RC versions such as RC5 and RC6

3.3 Asymmetric (Public-key) Cryptosystem

- It is a form of cryptosystem in which encryption and decryption are performed using different keys - one public key (KE) and one private key (KD) - that form a unique pair
  - $C = E_{KE}(P)$
  - $P = D_{KD}(C) \quad \Rightarrow \quad P = D_{KD}[E_{KE}(P)]$
  - The two keys have the property that deriving the private key from the public key is computationally infeasible
- Proposed by Diffie and Hellman in 1976
- Uses Mathematical functions whose inverse is not known by Mathematicians of the day
- It is a revolutionary concept since it avoids the need of using a secure channel to communicate the key
- It has made cryptography available for the general public and made many of today’s online applications feasible

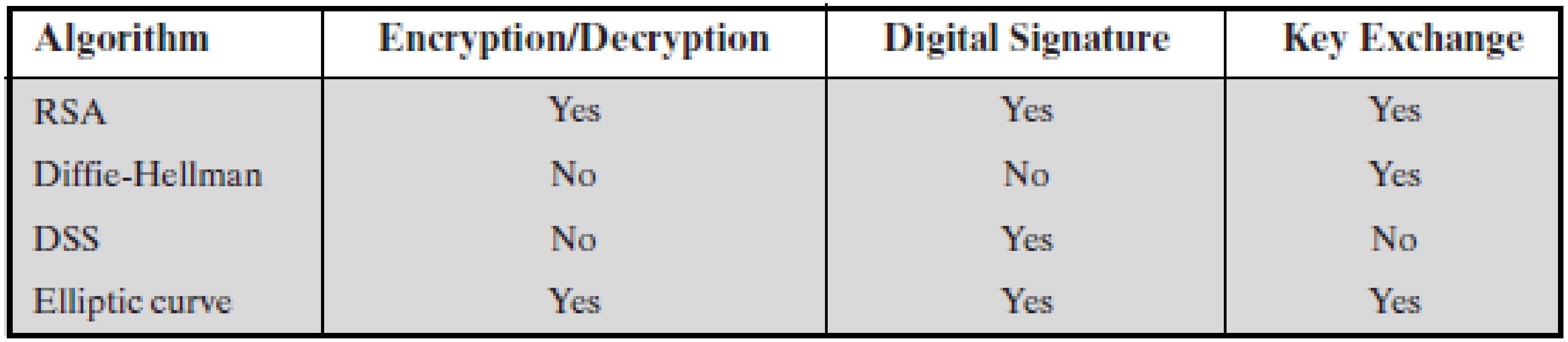
- It provides a radical departure from the past
  - Public-key algorithms are based on mathematical functions rather than on substitution and permutation
  - Public-key cryptography is asymmetric, involving the use of two separate keys, in contrast to symmetric encryption, which uses only one key. The use of two keys has profound consequences in the areas of confidentiality, key distribution, and authentication
- But the authenticity of a secret message is not guaranteed since anyone can send secret messages to the owner of a private key because the corresponding public key is known
- Properties of Public Key Cryptosystem
  - If you have the private key, you can easily decrypt what is encrypted by the public key
  - Otherwise, it is computationally infeasible to decrypt what has been encrypted by the public key

- Steps in Asymmetric Cryptosystems
  1. Each user generates a pair of keys to be used for the encryption and decryption of messages
  2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private
  3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice’s public key
  4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice’s private key
- At any time, a user can change its private key and publish the companion public key to replace its old public key

- Common misconceptions concerning public-key encryption
  1. Public-key encryption is not more secure from cryptanalysis than is symmetric encryption. The security of any encryption scheme depends on the length of the key and the computational work involved in breaking a cipher. There is nothing in principle about either symmetric or public-key encryption that makes one superior to another from the point of view of resisting cryptanalysis
  2. Public-key encryption has not made symmetric encryption obsolete. On the contrary, because of the computational overhead of current public-key encryption schemes, there seems no foreseeable likelihood that symmetric encryption will be abandoned
  3. Finally, there is a feeling that key distribution is trivial when using public-key encryption, compared to the rather cumbersome handshaking involved with key distribution centers for symmetric encryption. In fact, some form of protocol is needed, generally involving a central agent, and the procedures involved are not simpler nor any more efficient than those required for symmetric encryption

- Why public-key cryptography?
  - In an attempt to attack two of the most difficult problems associated with symmetric encryption
  - The first problem is that of key distribution since key distribution under symmetric encryption requires either (1) that two communicants already share a key, which somehow has been distributed to them; or (2) the use of a Key Distribution Center (KDC)
    - The second requirement negates the very essence of cryptography: the ability to maintain total secrecy over your own communication; “what good would it do after all to develop impenetrable cryptosystems, if their users were forced to share their keys with a KDC that could be compromised by either burglary or other means?”
  - The second problem is that of digital signatures. If the use of cryptography was to become widespread, then electronic messages and documents would need the equivalent of signatures used in paper documents

- Use of Public-key Cryptosystems
  - It can be used for confidentiality, authentication, or both
  - Depending on the application, the sender uses either the sender's private key, the receiver's public key, or both to perform some type of cryptographic function
  - The use of public-key cryptosystems can be classified into three:
    - Encryption/decryption: For confidentiality, the sender encrypts a message with the recipient's public key
    - Digital signature: For authentication, the sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message
    - Key exchange: Two sides cooperate to exchange a session key, (i.e., a user wishes to set up a connection with another user and uses a secret key to encrypt messages on that connection); Different approaches are possible, involving the private key(s) of one or both parties
  - Some algorithms are suitable for all three applications, whereas others can be used only for one or two of these applications



Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No
Elliptic curve	Yes	Yes	Yes

Applications of Public-Key Cryptosystems

- RSA - Rivest-Shamir-Adleman
- DSS - Digital Signature Standard

- You can read about Diffie-Hellman, DSS, and Elliptic Curve Cryptography (ECC) algorithms

3.3.1 RSA - Asymmetric Cryptosystem Example

- The most widely used public-key cryptosystem is RSA
- RSA is from Ron Rivesh, Adi Shamir and Leonard Adleman (in 1977)
- It is a block cipher in which the plaintext and ciphertext are integers between 0 and m-1 for some m
- The private and public keys are constructed from very large prime numbers (consisting of hundreds of decimal digits)
- Principle: No mathematical method is yet known to efficiently find the prime factors of large numbers
- Breaking RSA is equivalent to finding the prime factors: this is known to be computationally infeasible, i.e., security is based on the difficulty of factoring large integers
- It is only the person who has produced the keys from the prime numbers who can decrypt messages

- RSA - Key Generating Algorithm
  1. Choose two large prime numbers, p and q
  2. Compute n = pq and (phi) φ = (p-1)(q-1)
  3. Choose an integer *e*, 1 < e < φ, such that GCD(e, φ) = 1 (Note: The Greatest Common Divisor of two integers is the largest positive integer that exactly divides both integers) or e and φ are relatively prime (two integers are relatively prime if their only common positive integer factor is 1)
  4. Determine the secret exponent *d*, 1 < d < φ, such that φ divides (ed-1); i.e., the remainder of (ed-1)/φ is zero
  5. The public key is the pair of integers (e, n) and the private key is (d, n), i.e., both sender and receiver must know the value of n. The sender knows the value of e, and only the receiver knows the value of d
     - Keep all the values d, p, q and φ secret
     - n is known as the modulus
     - e is known as the *public exponent* or encryption exponent
     - d is known as the *secret exponent* or decryption exponent

- RSA- Encryption
  - Sender A does the following
    - Obtains the recipient B's public key (e, n)
    - Represents the plaintext message as a positive integer M
    - Computes the ciphertext $C = M^e \bmod n$
    - Sends the ciphertext C to B
- RSA- Decryption
  - Recipient B does the following
    - Uses his/her private key (d, n) to compute $M = C^d \bmod n$
    - Extracts the plaintext from the message representative M
- Compared to DES, RSA is computationally more complex; encryption is 100-1000 times slower than DES
- Hence encryption systems use RSA to exchange only shared keys (for symmetric cryptosystems) in a secure way

- RSA Simple Example - Key Generation

1. Choose two prime numbers: p=11, q=3
2. n = pq = 11*3 = 33
   φ = (p-1)(q-1) = 10*2 = 20
3. Choose e, 1 < e < φ; we choose e=3
   Check GCD(e, φ) = GCD(3, 20) = 1 (i.e., 3 and 20 are relatively prime)
4. Determine d, 1<d<φ, such that φ divides ed-1 (or 20 divides 3d-1)
   Simple testing (d = 2, 3, ...) gives d = 7
   Check: ed-1 = 3*7 - 1 = 20, which is divisible by φ (20)
5. Public key = (e, n) = (3, 33)
   Private key = (d, n) = (7, 33)

- RSA- Encryption Example
  - Now say we want to encrypt the message M = 7
    - $C = M^e \bmod n = 7^3 \bmod 33 = 343 \bmod 33 = 13$
    - Hence the ciphertext C = 13
- RSA- Decryption Example
  - For decryption, we compute
    - $M = C^d \bmod n = 13^7 \bmod 33 = 7$

RSA - More Meaningful Example

- Message: ATTACKxATxSEVEN
- Group the characters into blocks of three and compute a message representative integer for each block
  - <u>ATT</u> <u>ACK</u> <u>XAT</u> <u>XSE</u> <u>VEN</u>
  - In the same way that a decimal number can be represented as the sum of powers of ten, (e.g., $135 = 1 \times 10^2 + 3 \times 10^1 + 5 \times 10^0$), we could represent our blocks of three characters in base 26 using A=0, B=1, C=2, ..., Z=25
- $ATT = 0 \times 26^2 + 19 \times 26^1 + 19 \times 26^0 = 513$
  $ACK = 0 \times 26^2 + 2 \times 26^1 + 10 \times 26^0 = 62$
  $XAT = 23 \times 26^2 + 0 \times 26^1 + 19 \times 26^0 = 15567$
  $XSE = 23 \times 26^2 + 18 \times 26^1 + 4 \times 26^0 = 16020$
  $VEN = 21 \times 26^2 + 4 \times 26^1 + 13 \times 26^0 = 14313$

1. Generate two prime numbers: p=137 and q=131
2. n = pq = 137*131 = 17,947
   φ = (p-1)(q-1) = 136*130 = 17680
3. Choose e = 3
   Check GCD(3,17680)=1 (i.e., e and φ are relatively prime)
4. Determine d, 1<d<φ, such that φ divides ed-1 (or 17680 divides 3d-1); d = 11787; (11787*3-1)/17680 = 2
5. Hence
   - Public key, (e, n) = (3, 17947) and
   - Private key (d, n) = (11787, 17947)

- To encrypt the first integer that represents "ATT“ (513), we have
  - $C = M^e \bmod n = 513^3 \bmod 17947 = 8363$
- We can verify that our private key is valid by decrypting
  - $M = C^d \bmod n = 8363^{11787} \bmod 17947 = 513$
- Overall, our plaintext is represented by the set of integers m
  - (513, 62, 15567, 16020, 14313)
  - After decryption, these numbers are converted to their textual equivalents by successively dividing by 26 and taking the remainders
- We compute the corresponding ciphertext integers
  $C = M^e \bmod n$
  - (8363, 5017, 11884, 9546, 13366)

Do public and private keys form a unique pair?

Iteration on i using d=(φ*i+1)/e, i.e., φ divides ed-1

		i	d
n	33	1	7.00000
φ	20	2	13.66667
e	3	3	20.33333
d	7	4	27.00000
φ divides ed-1		5	33.66667
20 divides 3*d-1		6	40.33333
		7	47.00000
		8	53.66667
		PubK	(3, 33)
		PrvK	(7, 33)

		i	d
n	17947	1	5,893.66667
e	3	2	11,787.00000
φ	17680	3	17,680.33333
d	11787	4	23,573.66667
φ divides ed-1		5	29,467.00000
17680 divides 3*d-1		6	35,360.33333
		7	41,253.66667
		8	47,147.00000
		PubK	(3, 17947)
		PrvK	(11787, 17947)

- e and d form no unique pair!, i.e., with RSA, we can make several private keys for single public key and vice versa
- We have to avoid to make more than one public key for a single private key because combining two such keys helps revealing the private key

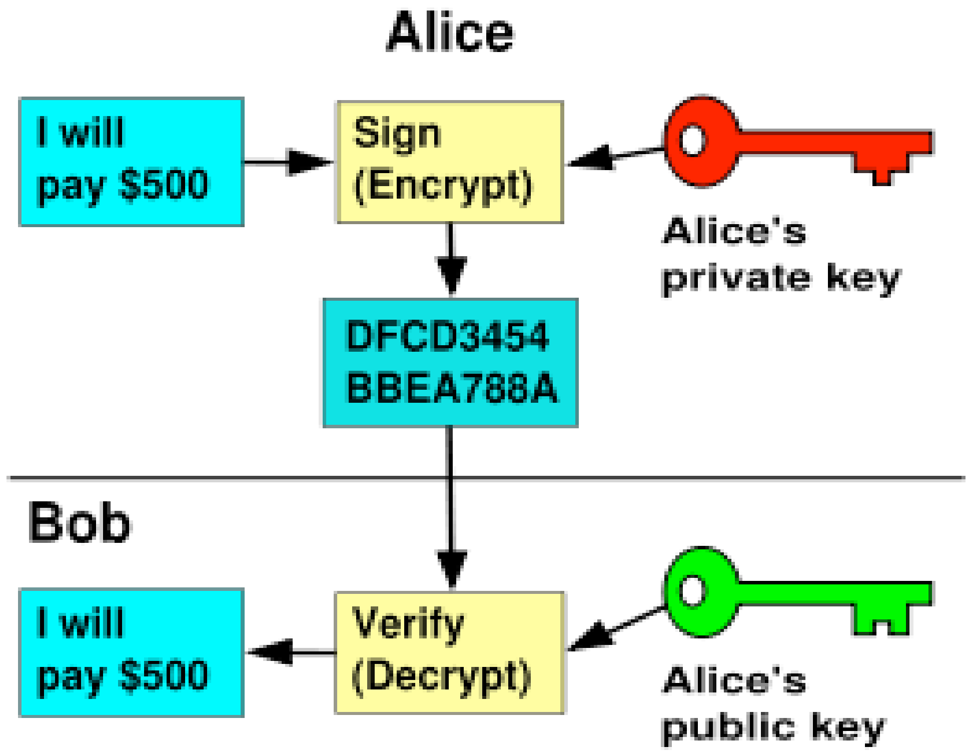
- How large should p and q be?
  - The larger the values, the more difficult it is to break RSA, but the longer it takes to perform the encoding and decoding
  - RSA Laboratories recommends that the product of p and q be on the order of 1,024 bits for corporate use and 768 bits for use with “less valuable information”
- Other issues
  - How do we choose large prime numbers?
  - How do we then choose e and d?
  - How do we perform exponentiation with large prime numbers?
  - You can read about the methods used for the above issues

3.3.2 Digital Signature

- Confidentiality ensures that messages cannot be intercepted and read by eavesdroppers, i.e., encryption protects against passive attack
- A different requirement is to protect against active attack (falsification of data and transactions). Protection against such attacks is known as message authentication
- A message, file, document, or other collection of data is said to be authentic when it is genuine (not altered) and comes from its alleged source
- A digital signature is not used to ensure the confidentiality of a message, but rather to guarantee who sent the message, i.e., authentication (nonrepudiation); it proves who the sender is
- Nonrepudiation can be source repudiation (denial of transmission of message by source) or destination repudiation (denial of receipt of message by destination)
- Just as with handwritten signatures, digital signing should be done in a way that is verifiable and nonforgeable

- Digital signature is also used for Message Integrity; it ensures that messages are protected against modification
- Note: authentication may mean both nonrepudiation and data integrity and sometimes only data integrity
- Digital Signature for Assurance
  - Consider the situation where Bob has just sold Alice something for 500 Birr through a deal that is made by e-mail
  - Alice sends an e-mail accepting to pay 500 Birr
  - Two issues need to be taken care of in addition to authentication
    - Alice needs to be assured that Bob will not modify the amount and show that Alice promised to pay more than 500 Birr
    - Bob needs to be assured that Alice will not deny that she sends the message, i.e., source repudiation

- If Alice signs the message digitally, the two issues will be solved so that her signature is uniquely tied to its content ➲ Bob's change will be noticed and Alice also cannot deny
- There are several ways to place digital signatures; One popular way is to use public-key cryptosystem such as RSA, i.e., message encryption by itself can provide measure of authentication
- Digital signature reverses the asymmetric encryption process



Notation: K_X- : Private key of X

K_X+ : Public key of X

- Alice encrypts the message using her private key

  $C = E(K_A-, M)$ – this is Alice's signature
- Sends the encrypted message to Bob
- Bob then decrypts the signature using Alice's public key

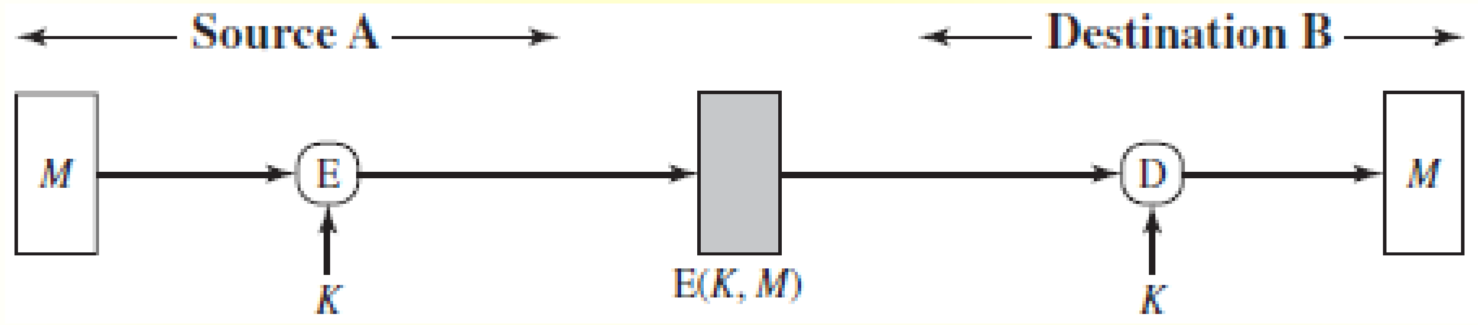
  $M = D(K_A+, C)$
- If Bob can decrypt it with Alice's public key, the message must have been encrypted by Alice; No one else has Alice's private key, and therefore no one else could have created a ciphertext that could be decrypted with Alice's public key – nonforgeable and verifiable
- Therefore, the encrypted message serves as a digital signature
- In addition, it is impossible to alter the message without access to Alice's private key, so the message is authenticated both in terms of source and in terms of data integrity

- But anyone can decrypt the message using Alice's public key if it is not important that the message be kept secret
- To combine both confidentiality and authentication
  - Alice has to first encrypt the message using her private key
  - Then encrypt the message with Bob's public key

    $C = E(K_B+, E(K_A-, M))$
  - Sends the encrypted message to Bob
  - Bob decrypts the message using his private key
  - Bob then decrypts the message using Alice's public key

    $M = D(K_A+, D(K_B-, C))$
- Disadvantage: The public-key algorithm must be applied four times rather than two which has an impact on efficiency

- Symmetric encryption can also be used for authentication
    - A message transmitted from source A to destination B is encrypted using a secret key shared by only A and B. If no other party knows the key, then confidentiality is provided: No other party can recover the plaintext of the message; B is also assured that the message was generated by A (authentication)
    - But, Alice can deny that she has sent the message; Bob can also modify the amount



Symmetric Encryption: Authentication and Confidentiality

- Digital Signature Using Message Digest
  - Problems in Digital Signature
    - Alice may claim that her private key has been stolen before the message was sent
    - Alice may change her private key; a solution could be to have a central authority that keeps track of changes in keys and that signed messages be timestamped
    - Alice's entire message is encrypted which may be expensive in terms of processing requirements
    - It also requires a great deal of storage. Each document must be kept in plaintext to be used for practical purposes. A copy also must be stored in ciphertext so that the origin and contents can be verified in case of a dispute
    - A better and cheaper method is to use a message digest

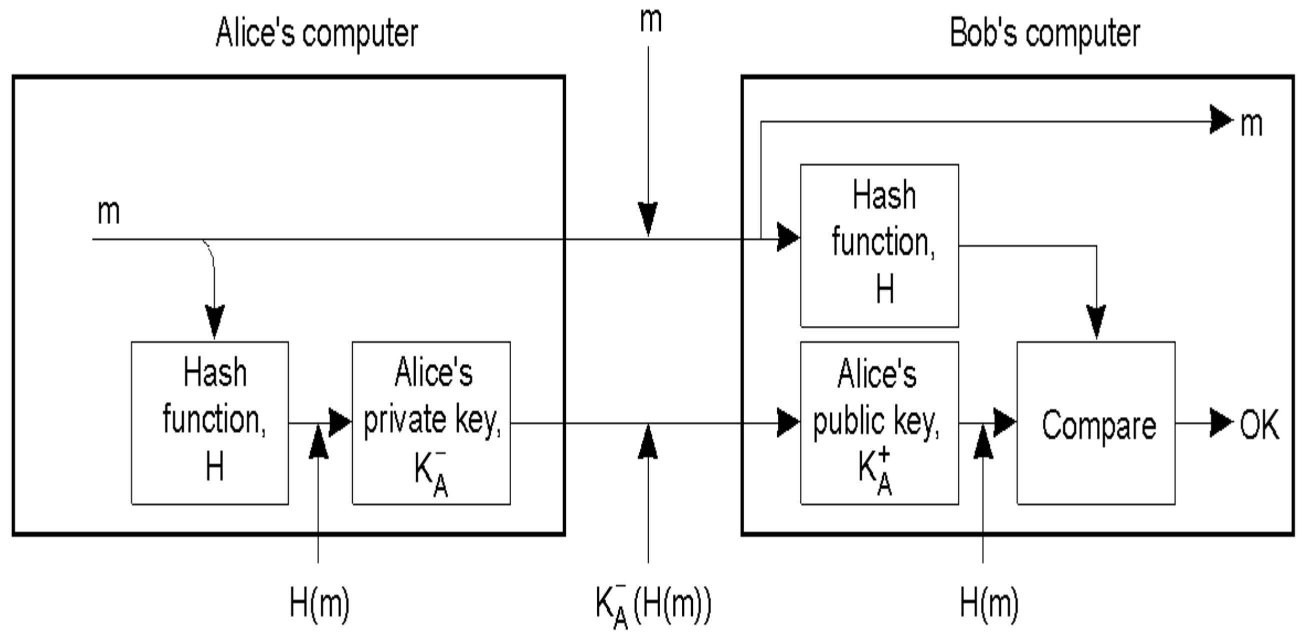
- Hash Functions
  - A hash function H takes a message m of arbitrary length and produces a fixed size bit string h, h = H(m)
  - When the hash value h is sent with the message m (not encrypted), it enables to determine whether m has been modified or not; the principal objective of a hash function is data integrity
  - When a hash function is used to provide message integrity, the hash function value h is often referred to as a message digest
  - The two most common hashing algorithms are MD5 (Message Digest version 5) which produces a 128-bit hash and Secure Hash Algorithm or SHA (SHA-1 and later versions like SHA-256) by NIST which produces a 160-bit message digest

- Example
  - Assume we want to send the number 12345 and use hashing to make sure there were no changes to this transmission
  - The chosen algorithm (highly simplified) is
    - Multiply the data by 56,789
    - Invert the result
    - Chop off all but the first four characters

Multiply:	12345 x 56789 = 701060205
Invert:	502060107
Truncate:	5020

  - Hence 5020 is the hash value that is sent along with 12345
  - The receiver follows the same steps to hash the message; if the results match then there was no modification
- A typical hash combines encryption and truncation or padding to get to a fixed-size authentication value

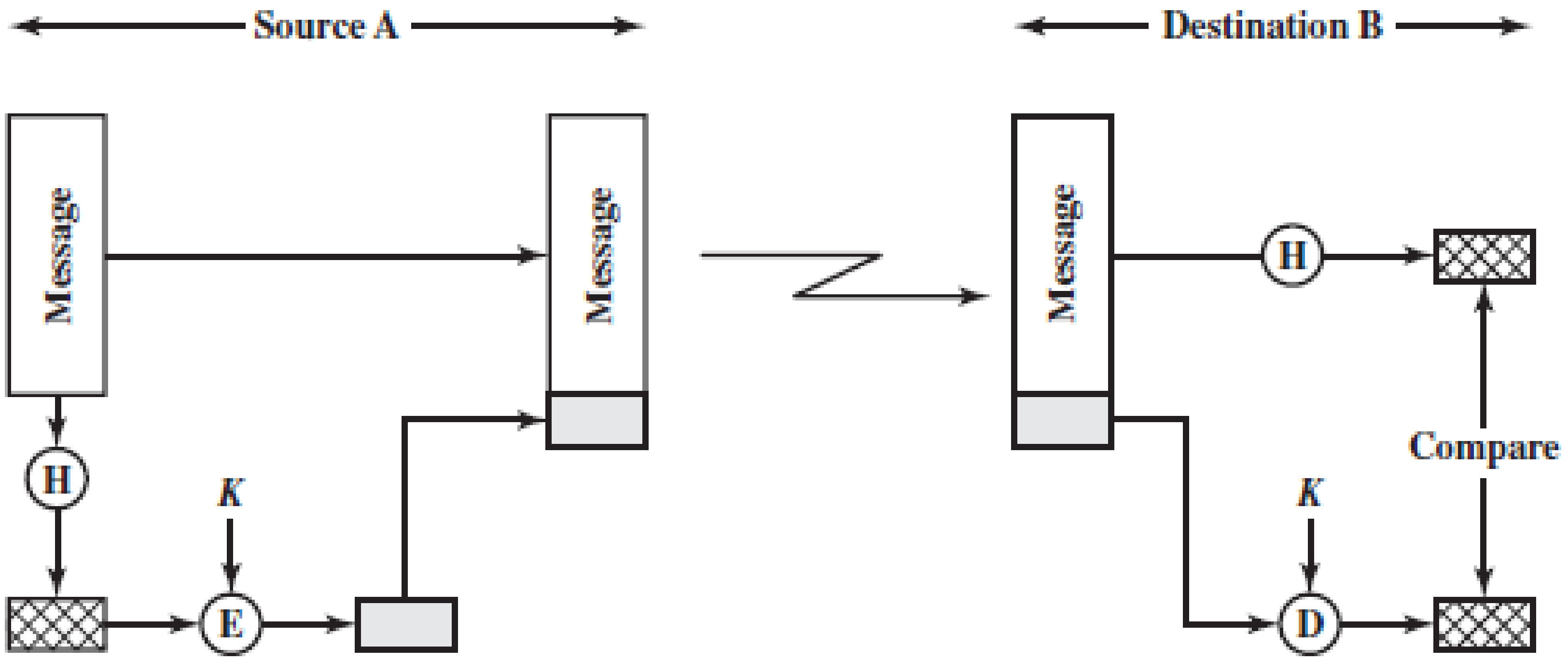
- If m is changed to m', its hash h' = H(m') will be different from h = H(m) and can be easily detected
- Alice first computes a message digest and encrypts it with her private key
- E(K_A-, H(m)) is sent with m so that Bob knows that it comes from Alice by decrypting it with her public key
- Bob decrypts the digest and calculates the message digest; if they match he knows the message has not been altered



Digitally signing a message using a message digest

- m can be sent as plaintext or if confidentiality is required can be encrypted using Bob's public key, but with an implication on performance
- $E(K_A-, H(m))$ and $E(K_B+, m)$ are sent so that Bob knows that it comes from Alice by decrypting the message digest with her public key and m is also protected from eavesdropping

- The message digest can be encrypted using symmetric encryption if it is assumed that only the sender and receiver share the encryption key



Using Symmetric Encryption

- The public-key approach has two advantages: (1) It provides a digital signature as well as message integrity. (2) It does not require the distribution of keys to communicating parties

- Message Authentication Code (MAC)
  - Some hash functions require a key; others do not
  - When encryption is used with hashing, it is extremely expensive
  - Without encryption, Trudy can claim to be Alice and send a bogus message m’ and H(M’) to Bob - Masquerading
  - To perform message integrity, in addition to a hash function, Alice and Bob will need a shared secret s, which is just a string of bits called the authentication key, but distributing s has the same problem as a symmetric key – see later for the solution
  - Steps
    - Alice concatenates s with m to create m+s and calculates the hash h=H(M+s); h is called a Message Authentication Code (MAC)
    - Alice appends the MAC to m and sends the extended message (m, h) to Bob – no encryption
    - Bob calculates H(m+s) and compares it with h since he knows s

- Properties of Hash Functions
  - One-way function: It is computationally infeasible to find m that corresponds to a known output of h
    - Or given a hash value h it should be difficult to find the message m such that h = H(m)
    - That means you cannot “unhash” something
  - Collision resistance
    - Weak-collision resistance: It is computationally infeasible, given m and H, to find m’ ≠ m such that H(m) = H(m’)
    - Strong-collision resistance: Given H, it is computationally infeasible to find any two different input values m and m’, such that H(m) = H(m’)
  - The output is of fixed-length no matter what input is given. This is exactly how Windows stores passwords. For example, if the password is password, then Windows will first hash it producing something like:

    0BD181063899C9239016320B50D3E896693A96DF

- It then stores it in the SAM (Security Accounts Manager) file in the Windows System directory. When you log on, Windows cannot unhash your password (remember it is one-way). So, what Windows does is take whatever password you type in, hash it, and then compare the result with what is in the SAM file. If they match (exactly), then you can log in
  - Caution: password remains invisible while being entered, but
    - it is transferred in clear from keyboard to memory
    - it is present in clear in memory for a short time
  - Hence, beware of eavesdropping and password sniffers!

- Hashing also has other applications
  - For example, it can be used for intrusion detection and virus detection. Store H(F) for each file on a system and secure the hash values (e.g., on a CD-R that is kept secure). One can later determine if a file has been modified by recomputing H(F). An intruder would need to change F without changing H(F)

3.3.3 Symmetric Key Distribution

  - For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others
  - Frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key

- Symmetric Key Distribution Using Symmetric Encryption
  - Key distribution can be achieved in a number of ways. For two parties A and B, the following can be used
    1. A key could be selected by A and physically delivered to B
    2. A third party could select the key and physically deliver it to A and B
        - The above two are manual delivery of a key and difficult in a distributed system where any given host or terminal may need to engage in exchanges with many other hosts and terminals over time and each device needs a number of keys supplied dynamically

3. If A and B have previously and recently used a key, one party could transmit the new key to the other, using the old key to encrypt the new key
   - The problem with this option is if an attacker ever succeeds in gaining access to one of the keys
4. If A and B each have an encrypted connection to a third party C, C could deliver a key on the encrypted links to A and B
   - This is preferable and two kinds of keys are used
     - Permanent key: used between entities for the purpose of distributing session keys
     - Session key: when two end systems (hosts, terminals, etc.) wish to communicate, they establish a logical connection (e.g., virtual circuit). For the duration of that logical connection, called a session, all user data are encrypted with a one-time session key. At the conclusion of the session, the session key is destroyed

- Option 4 requires a Key Distribution Center (KDC) that determines which systems are allowed to communicate with each other
- The operation of a KDC is as follows
  1. When host A wishes to set up a connection to host B, it transmits a connection request packet to the KDC. The communication between A and the KDC is encrypted using a master key (or permanent key) shared only by A and the KDC
  2. If the KDC approves the connection request, it generates a unique one-time session key. It encrypts the session key using the permanent key it shares with A and delivers the encrypted session key to A. Similarly, it encrypts the session key using the permanent key it shares with B and delivers the encrypted session key to B
  3. A and B can now set up a logical connection and exchange messages and data, all encrypted using the temporary session key

- The automated key distribution approach provides the flexibility and dynamic characteristics needed to allow a number of users to access a number of servers and for the servers to exchange data with each other. The most widely used application that implements this approach is Kerberos (details later in Chapter 5)
- Benefits of Session Keys
  - The session key is safely discarded when the channel is no longer used
  - When a key is used very often it becomes vulnerable. Thus by using the permanent key less often, we make them less vulnerable
  - Replay attacks can be avoided (i.e., using the key later after the session ends to pretend as one of the communicating parties)
  - Such a combination of long-lasting and cheaper (more temporary) session keys is a good choice

Symmetric Key Distribution Using Asymmetric Encryption

- Because of the inefficiency of public key cryptosystems, they are almost never used for the direct encryption of sizable block of data, but are limited to relatively small blocks
- One of the most important uses of a public-key cryptosystem is to encrypt secret keys for distribution
- Assume that A and B have exchanged public keys
  1. A uses B's public key to encrypt a message (m1) to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely

     $m1 = E(K_B^+, ID_A+N_1)$
  2. B sends a message (m2) to A encrypted with A's public key and containing A's nonce N_1 as well as a new nonce N_2 generated by B. Because only B could have decrypted message m1, the presence of N_1 in message m2 assures A that the correspondent is B

     $m2 = E(K_A^+, N_1+N_2)$

3. A returns N_2, encrypted using B's public key, to assure B that its correspondent is A

   $m = E(K_B^+, N_2)$

4. A selects a secret key K_s and sends $M = E(K_B^+, E(K_A^-, K_s))$ to B. Encryption of this message with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it

5. B computes to recover the secret key

- This scheme ensures both confidentiality and authentication (steps 1 and 2) in the exchange of a secret key

3.3.4 Public Key Distribution

- Public Announcement of Public Keys
  - Send a public key to any other participant or broadcast the key to the community
  - But anyone can forge such a public announcement, i.e., some user could pretend to be a legitimate user and send a public key to another participant or broadcast it; or Trudy can send Alice a public key pretending to be Bob
- Public-key Infrastructure
  - We need a body that certifies the public key is that of the party (a person, a router, etc.) we wish to communicate with, i.e., Certification/Certificate Authority (CA) that signs (certifies) the public key; an example is VeriSign
  - Public-Key Infrastructure (PKI) is the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography

- Users publish certificates with the X.509 standard (for formatting certificates)
- A certificate is a public key and some naming “stuff”, digitally signed by someone you trust (third party), i.e., the CA
- The resulting certificate will contain information like user’s name/ID, user’s public key, name of CA, start date of certificate, and length of time it is valid
- When Bob sends a message (encrypted with his private key) and his CA-signed certificate, Alice uses the CA’s public key to check the validity of Bob’s certificate and extract Bob’s public key
- The Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) working group has been the driving force for deploying a certificate-based architecture on the Internet
- Read more about the Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX)

3.4 Concluding Remarks about Encryption

- Symmetric Cryptography
  - Advantage: It is efficient
  - Disadvantage: It is impractical for exchanging messages with a large group of previously unknown correspondents over a public network, e.g., in e-commerce, for a merchant to conduct transactions securely with millions of customers, each customer would need a distinct key assigned by that merchant and transmitted over a separate secure channel
- Asymmetric Cryptography
  - Advantage: It allows for secrecy between two parties who have not arranged in advance to have a shared key (or trusted some third party to give it to them)
  - Disadvantage: inefficient
- Therefore, in practice, hybrid systems use public-key to establish session key for symmetric encryption

- Legitimate Versus Fraudulent Encryption Methods
  - Dozens of encryption methods are released to the public for free or are patented and sold for profit every year. However, it is important to realize that this particular area of the computer industry is full of fraud
  - Search (Google) for encryption to find many advertisements for the latest and greatest "unbreakable" encryption
  - How do you separate legitimate encryption methods from frauds?
  - Here are some warning signs
    - Unbreakable: there is no such thing as an unbreakable code
      - There are codes that have not yet been broken
      - There are codes that are very hard to break
      - But when someone claims that their method is "completely unbreakable", be suspicious

- Certified: There is no recognized certification process for encryption methods. Therefore, any “certification” the company has is totally worthless
- Inexperienced people: A company is marketing a new encryption method
  - What is the experience of the people working with it?
  - Does the cryptographer have a background in maths, encryption, or algorithms?
  - If not, has s/he submitted the method to experts in peer-reviewed journals?
  - Or, is s/he at least willing to disclose how the method works so that it can be fairly judged?
- Some claim that you should only use widely known methods such as Blowfish and PGP (Pretty Good Privacy – to be briefly covered in Chapter 4 - Network Security Concepts and Mechanisms)

- Older is better
  - In Cryptography, Older is better
  - It is usually unwise to use the “latest thing” in encryption for the simple reason that it is unproven
  - An older encryption method, provided it has not yet been broken, is usually a better choice because it has been subjected to years of examination by experts and to cracking attempts by both experts and less honorably motivated individuals