DEBRE MARKOS UNIVERSITY

BURIE CAMPUS

DEPARTMENT OF COMPUTER SCIENCE

Computer Security

By:

Amare W.

Chapter 3: Cryptography and Encryption Techniques

3.1 Basic cryptographic terms

- It is the science concerned with data communication and storage in secure and usually secret form.
- It is the technology which is applied in implementing computer security.
- Cryptology is the branch of science that deals with secret communications.
- Cryptography—the name means secret writing—is probably the strongest defense in the arsenal of computer security protection.
- Well-disguised data cannot easily be read, modified, or fabricated.

- It is the study of Cryptosystems.
  - Cryptosystems are the techniques for ensuring the secrecy and/or authenticity of information.
- It has two main branches: cryptography and cryptanalysis.
  - Cryptography is the study of designing the techniques of cryptosystem. It designs the secret codes and ciphers.
  - Cryptanalysis deals with defeating such techniques, to recover information. It deals with "breaking" and reading secret codes and ciphers.
  - A **cryptanalyst** studies encryption and encrypted messages, hoping to find the hidden meanings.

Encryption and Decryption

- **Encryption:** Process of encoding (enciphering) a message so that its meaning is not obvious.The process by which plaintext is converted into ciphertext.
- The original form of a message is known as plaintext, and the encrypted form is called ciphertext. The transformation of the plaintext under the control of the key into a cipher (also called ciphertext).
- **Decryption:** Process of decoding (deciphering or transforming) an encrypted message to its original form. Recovering plaintext from the ciphertext.
- The inverse operation, by which a legitimate receiver recovers the concealed information from the cipher using the key.

- Symmetric Encryption Scheme (Cryptography) has five ingredients:
  1. Plaintext
  2. Encryption algorithm
  3. Secret Key
  4. Ciphertext
  5. Decryption algorithm

- **Plaintext (P):** original form of message (source info.)
- **Cipher text (C):** encrypted message
- Encryption algorithm: **E**
- Decryption algorithm: **D**
- Secret Key: **K**

Description:

- A sender S wanting 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

Notation:

- **Given**
  - P=Plaintext
  - C=CipherText
- $C = E_K (P)$ Encryption
- $P = D_K (C)$ Decryption
- A cryptosystem involves a set of rules for how to encrypt the plaintext and decrypt the ciphertext.
- The encryption and decryption rules, called algorithms, often use a device called a key, denoted by K, so that the resulting ciphertext depends on the original plaintext message, the algorithm, and the key value.
- **Secret key (K): The secret information known only to the transmitter and the receiver which is used to secure the plaintext.**

- Security depends on the secrecy of the key, not the secrecy of the algorithm.

Principle of Encryption

- Very hard (impossible) to find out the message without knowing the key
- Very easy (and fast) to find out the message knowing the key
- The two types of attack on an encryption algorithm are:

1) **Cryptanalysis**, based on properties of the encryption algorithm, and

2) **Brute-force**, which involves trying all possible keys.

Types of Cryptosystems

- There are two cryptographic systems

1. Symmetric cryptosystem

- Also called secret key encipherment or secret/private key cryptosystems.
- It is a form of cryptosystem in which encryption and decryption are performed using the same key.
- Transforms plaintext into ciphertext using a secret key and an encryption algorithm. Using the same key and a decryption algorithm, the plaintext is recovered from the ciphertext.
- The key has to be kept secret, and has to be communicated using a secure channel.

2. Asymmetric cryptosystem

- Also called public-key cryptosystem
  - Keys for encryption and decryption are different but form a unique pair
  - Only one of the keys need to be private while the other can be public
- to send a secure message to the receiver, the sender first encrypts the message by using the receiver ‘s public key.
- to decrypt the message, the receiver uses his own private key.

Classical Encryption Techniques

- They are traditional symmetric cryptosystems
- They are simple cryptosystems

i. Substitution techniques: map plaintext elements (characters, bits) in to cipher text elements.

ii. Transposition techniques: systematically transpose the positions of plaintext elements (rearrange their orders).

Substitution ciphers

- A substitution cipher is one in which the letters of plaintext are replaced by other letters or by numbers or symbols.
- Substitution ciphers can be categorized as either
  - a. Monoalphabetic ciphers
  - b. Polyalphabetic ciphers

I. Monoalphabetic ciphers

- In this case, a character (or symbol) in the plaintext is always changed to the same character (or symbol) in the ciphertext regardless of its position or text.
  - For example, if the algorithm says that letter A in the plaintext is changed to letter D, every letter A is changed to letter D.
  - The relationship in between plaintext and ciphertext is one-to-one.

Cont'd

- Example: The following example shows a plaintext and the corresponding ciphertext.
- We use the lowercase characters to show the plaintext and uppercase characters to show the ciphertexts.
- The cipher is monoalphbetic because both the l's are encrypted as O's:

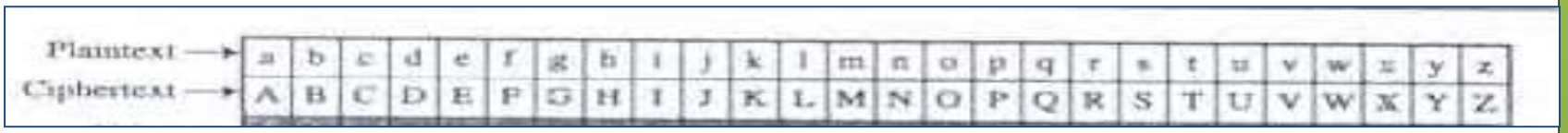
Plaintext: hello Ciphertext: KHOOR

Cont'd

- The group of monoalphabetic ciphers includes:
  - i. Additive ciphers or Caesar ciphers
  - ii. Multiplicative ciphers
  - iii. Affine ciphers

i. Additive ciphers:

- This is the simplest monoalphabetic cipher.
- We assume that the plaintext contains the lowercase characters (a to z) and the ciphertext contain the upper text characters (A to Z) as follows:



- **the Figure, each character is assigned an integer in Z_{26} .**
- **The secret key is also an integer in Z_{26}**
- **The encryption algorithm adds the key to the plaintext characters and the decryption algorithm subtracts the key from the ciphertext characters.**
- Then the algorithm can be expressed as follows:

For each plaintext letter *P*, substitute the ciphertext letter *C*:

(*a* mod *n* is the remainder when *a* is divided by *n*.

3/2/2018

E.g. 11 mod 7 = 4)

- **The encryption algorithm is**

  $C = E(k, P) = (P + k) \bmod 26$; where k takes a value in the range 1 to 25.

- **The decryption algorithm is simply**

  $P = D(k, C) = (C - k) \bmod 26$; where k takes a value in the range 1 to 25.

Example: Use the additive cipher with **key =15** to encrypt the message "**hello**".

Soln. : We apply the encryption algorithm to the plaintext character by character

Plaintext : h → 07	Encryption: (07+15)mod 26	Ciphertext: 22→ W
Plaintext : e → 04	Encryption: (04+15)mod 26	Ciphertext: 19 → T
Plaintext : l → 11	Encryption: (11+15)mod 26	Ciphertext: 00 → A
Plaintext : l → 11	Encryption: (11+15)mod 26	Ciphertext: 00 → A
Plaintext : o → 14	Encryption: (14+15)mod 26	Ciphertext: 03 → D

So, the result is "WTAAD"

Note: By using the reverse decrypt algorithm, we can now decrypt the ciphertext "WTAAD".

- Additive cipher is also called as shift cipher. The reason is that the encryption algorithm can be interpreted as "shift key character down" and the decryption algorithm can be interpreted as "shift key character up".

- Three important characteristics of this problem enabled us to use a brute-force cryptanalysis:

1. The encryption and decryption algorithms are known.
2. There are only 25 keys to try.
3. The language of the plaintext is known and easily recognizable.

- Julius Caesar used an additive cipher to communicate with his officers. For this reason, this cipher is also sometimes called as caesar cipher.
- Caesar Cipher: The earliest known example of a substitution cipher in which each character of a message is replaced by a character three position down in the alphabet. For example:
  - Plaintext: are you ready
  - Ciphertext: duh brx uhdgb

ii. Multiplicative cipher:

- In this cipher, the encryption algorithm specifies the multiplication of the plaintext by the key and the decryption algorithm specifies the division of the ciphertext by the key.
- Since operations are in Z_{26}, decryption here means multiplying by the multiplicative inverse of the key.
  - The general multiplicative encryption algorithm is

    $C = E(k, P) = (P * k) \bmod 26$; where k takes on a value in the range 1 to 25.
  - The general multiplicative decryption algorithm is

    $P = D(k, C) = (C * k^{-1}) \bmod 26$; where k takes on a value in the range 1 to 25.

- Example: Use the multiplicative cipher with key =7 to encrypt the message "hello".;

Soln.: We apply the encryption algorithm to the plaintext character by character

Plaintext :h → 07	Encryption: (07*07)mod 26	Ciphertext: 23→ X
Plaintext : e → 04	Encryption: (04*07)mod 26	Ciphertext: 02 →C
Plaintext : l → 11	Encryption: (11*07)mod 26	Ciphertext: 25 → Z
Plaintext : l → 11	Encryption: (11*07)mod 26	Ciphertext: 25→ Z
Plaintext : o → 14	Encryption: (14*07)mod 26	Ciphertext: 20 → U

So, the result is "XCZZU"

Note: By using the reverse decrypt algorithm, we can now decrypt the ciphertext "XCZZU". (Use $K^{-1} = 15$)

iii. Affine cipher:

- **Affine cipher is a combination of additive and multiplicative ciphers with a pair of keys.**
- **The first key is used with multiplicative cipher and the second key is used with the additive cipher.**
- **Affine cipher is actually two ciphers applied one after the other.**
- **In the affine cipher, the relations between the plaintext and ciphertext are**

$C = E(k_1, k_2; P) = (P * k_1 + k_2) \bmod 26$ and $P = D(C; k_1, k_2) = ((C - k_2) * k_1^{-1}) \bmod 26$

- **Example:** Use the affine cipher with key pair (7, 2) to encrypt the message "hello".
- **Soln. :** We use 7 for the multiplicative key and 2 for the additive key.
- We apply the encryption algorithm to the plaintext character by character

Plaintext : h → 07	Encryption: (07*07+2)mod 26	Ciphertext: 25 → Z
Plaintext : e → 04	Encryption: (04*07+2)mod 26	Ciphertext: 04 → E
Plaintext : l → 11	Encryption: (11*07+2)mod 26	Ciphertext: 01 → B
Plaintext : l → 11	Encryption: (11*07+2)mod 26	Ciphertext: 01 → B
Plaintext : o → 14	Encryption: (14*07+2)mod 26	Ciphertext: 22→ W

So, the result is "ZEBBW".

Note: By using the reverse decrypt algorithm, we can now decrypt the ciphertext "ZEBBW".

- Because additive, multiplicative and affine ciphers have small key domain, they are vulnerable to brute-force attack.
- A brute-force attack involves trying every possible key until an intelligible translation of the ciphertext into plaintext is obtained. On average, half of all possible keys must be tried to achieve success.
- After the sender and the receiver agreed Ki a single key, that key is used to encrypt each letter in the plaintext or decrypt each letter in the ciphertext.
- A better solution is to create a mapping in between each letter of the plaintext and each letter of the ciphertext.

II Polyalphabetic cipher

- In this kind of cipher, each occurrence of character may have a different substitute. The relationship between the characters in the plaintext and the characters in ciphertext is one-to-many.
- For example, ‘a’ could be enciphered as ‘D’ in the beginning, but as ‘N’ in the middle. Polyalphabetic ciphers have the advantage of hiding the letter frequency of the underlying language.
- To create a polyalphabetic cipher, we need to make each ciphertext character dependent on both the plaintext character and the position of the plaintext character.
- We need to have a key stream k= (k_1, k_2, k_3,…) in which k_i is used to encipher the i^{th} character in the plaintext to create the i^{th} character in the ciphertext.

Cont'd

- The group of polyalphabetic ciphers includes:

1. Autokey cipher

- In this cipher, the key is a stream of sub keys, in which each sub key is used to encrypt the corresponding character in the plaintext. The first sub key is a predetermined value agreed upon by the sender and the receiver.
- The second sub key is the value of first plaintext character (between 0 and 25). The third subkey is the value of second plaintext character and so on.
- The name of the cipher autokey implies that the sub keys are automatically generated from the plaintext cipher characters during the encryption process.

$P=P_1P_2P_3\ldots$ $C=C_1C_2\ C_3\ldots$ $k=k_1P_1P_2\ldots$

Encryption: $C_i = (P_i + k_i) \bmod 26$

Decryption: $P_i = (C_i - k_i) \bmod 26$

Cont'd

Example: *Encrypt the plaintext "attack is today" using the initial key value* $k_1 = 12$.

Sol.: Here enciphering is done character by character.

Each character in the plaintext is first replaced by its integer value as shown in the figure. The first sub-key is added to create the first ciphertext character.

The rest of the key is created as the plaintext characters are read.

Plaintext:	a	t	t	a	c	k	i	s	t	o	d	a	y
P's Values:	00	19	19	00	02	10	08	18	19	14	03	00	24
Key stream:	12	00	19	19	00	02	10	08	18	19	14	03	00
C's Values:	12	19	12	19	02	12	18	00	11	7	17	03	24
Ciphertext:	M	T	M	T	C	M	S	A	L	H	R	D	Y

We note that the cipher is polyalphabetic because the three occurrences of "a" in the plaintext are encrypted differently. The three occurrences of the "t" are enciphered differently.

2. Playfair cipher

- The secret key in this cipher is made of 25 alphabetic letters arranged in 5×5 matrix.
- Different arrangements of the letters of the matrix can create many different secret keys. One of the possible arrangements has been shown in the following figure:

Secret Key =

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

- Before encryption, if the two letters in a pair are same, a bogus letter is inserted in between to separate them.
- After inserting bogus letters, if the number of characters in the plaintext is odd, one extra bogus character is added at the end to make the number of characters even.

Cont'd

❖ The cipher uses three rules for encryption:

i. If the two letters in a pair is located in the same row of the secret key, the corresponding encrypted character for each letter is the next letter to the right in the same row (with wrapping to the beginning of the row if the plaintext letter is the last character in the row)

ii. If the two letters in a pair are located in the same column of the secret key, the corresponding encrypted character for each letter is the letter beneath it in the same column (with wrapping to the beginning of the column if the plaintext letter is the last character in the column).

iii. If the two letters in a pair are not in the same row or column of the secret, the corresponding encrypted character for each letter is a letter that is in its own row but in the same column as the other letter.

Cont'd

- In Playfair cipher, the key is a stream of subkeys in which the subkeys are created two at a time.
- The encryption algorithm takes a pair of characters from plaintext & creates a pair of ciphertext by following the above-mentioned rules
- We can say that the key stream depends on the position of the characters of the plaintext.
  - $P=P_1P_2P_3.....\quad C=C_1C_2C_3\ldots\quad k=((k_1, k_2), (k_3, k_4),\ldots)$

    Encryption : $C_i = k_i$ Decryption : $P_i = k_i$

Example*:* Encrypt the plaintext '**attack**' by using the key in the above matrix.

Sol. : When we group the letters in two-character pair, we get "at, ta, ck".

at -> BP ta -> PB ck->FP

Ciphertext: BPPBFP

3. Vegenere cipher

- This cipher was designed by Blaise de Vegenere, a sixteenth century French mathematician.
- A Vegenere cipher uses a different strategy to create the key stream.
- The key stream is a repetition of an initial secret key stream of length m, where we have $1 \leq m \leq 26$.
- The cipher can be described as follows where $(k_1, k_2, \ldots, k_m)$ is the initial secret key agreed by the sender and the receiver.

❖ $P=P_1P_2\ldots$ $C= C_1C_2\ldots$ $K=[(k_1, k_2, \ldots, k_m), (k_1, k_2, \ldots, k_m), \ldots.]$

Encryption: $C_i = (P_i + k_i) \bmod 26$

Decryption: $P_i = (C_i - k_i) \bmod 26$

Cont'd

- The difference between the Vegenere cipher and the other two polyalphabetic ciphers is that the Vegenere key stream does not depend on the plaintext characters; it depends only on the position of the character in the plaintext.
- In other words, the key stream can be created without knowing what the plaintext is.

Example: Encrypt the message "she is listening" using the 6-character keyword "PASCAL".

Sol.: The initial key stream is (15, 0, 18, 2, 0, 11). The key stream is the repetition of this initial key stream.

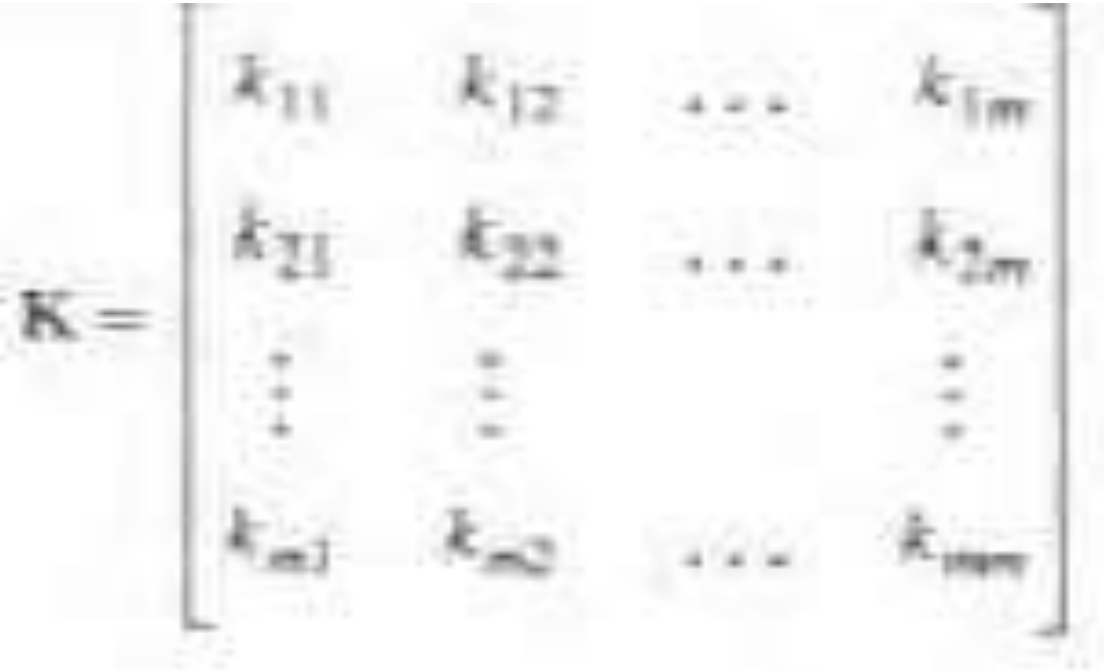
Plaintext:	s	h	e	i	s	l	i	s	t	e	n	i	n	g
P's values:	18	07	04	08	18	11	08	18	19	04	13	08	13	06
Key stream:	*15*	*00*	*18*	*02*	*00*	*11*	*15*	*00*	*18*	*02*	*00*	*11*	*15*	*00*
C's values:	07	07	22	10	18	22	23	18	11	6	13	19	02	06
Ciphertext:	H	H	W	K	S	W	X	S	L	G	N	T	C	G

4. Hill Cipher

- Another interesting polyalphabetic cipher is the Hill cipher, developed by the mathematician Lester Hill in 1929.
- Unlike other polyalphabetic cipher, the plaintext is divided into equal-size blocks.
- The blocks are encrypted one at a time in such a way that each character in the block contributes to the encryption of the other character in the block.
- For this reason, the hill cipher belongs to a category of ciphers called Block ciphers.
- Playfair cipher is also a block cipher of size 2.
- The other ciphers we studied so far belong to the category called stream ciphers.
- In the Hill cipher, the key is a square matrix of size m×m in which m is the size of the block.

Cont'd

- If we call the key matrix K, each element of the matrix is $k_{i,j}$ as shown in the following Figure:



- To encrypt one block of the plaintext, let us call the m characters in the plaintext block as P_1, P_2,...., P_m. Then the corresponding characters in the ciphertext block are C_1, C_2, …, C_m.

Cont'd

- Then we have

$$C_1 = P_1 k_{11} + P_2 k_{21} + \cdots + P_m k_{m1}$$
$$C_2 = P_1 k_{12} + P_2 k_{22} + \cdots + P_m k_{m2}$$
$$\cdots$$
$$C_m = P_1 k_{1m} + P_2 k_{2m} + \cdots + P_m k_{mm}$$

- The equation shows that each ciphertext character such as C_i depends on all plaintext characters in the block $(P_1, P_2, \ldots, P_m)$.
- Decryption requires using the inverse of the matrix K. The inverse K^{-1} of a matrix K.

Cont'd

Example: Encrypt the plaintext "code is ready" by Hill cipher

- **Sol.:** In this case, the plaintext "code is ready" can make a 3×4 matrix when adding extra bogus character (e.g. "z") to the block and removing the spaces.
- The receiver can decrypt the message using the inverse of the key matrix. Encryption and decryption are shown in the following Figure.
- The ciphertext is "OHKNIHGKLISS"

$$\begin{bmatrix} 14 & 07 & 10 & 13 \\ 08 & 07 & 06 & 11 \\ 11 & 08 & 18 & 18 \end{bmatrix} = \begin{bmatrix} 02 & 14 & 03 & 04 \\ 08 & 18 & 17 & 04 \\ 00 & 03 & 24 & 25 \end{bmatrix} \begin{bmatrix} 09 & 07 & 11 & 13 \\ 04 & 07 & 05 & 06 \\ 02 & 21 & 14 & 09 \\ 03 & 23 & 21 & 08 \end{bmatrix}$$

C = P × K

a. Encryption

$$\begin{bmatrix} 02 & 14 & 03 & 04 \\ 08 & 18 & 17 & 04 \\ 00 & 03 & 24 & 25 \end{bmatrix} = \begin{bmatrix} 14 & 07 & 10 & 13 \\ 08 & 07 & 06 & 11 \\ 11 & 08 & 18 & 18 \end{bmatrix} \begin{bmatrix} 02 & 15 & 22 & 03 \\ 15 & 00 & 19 & 03 \\ 09 & 09 & 03 & 11 \\ 17 & 00 & 04 & 07 \end{bmatrix}$$

P = C × K^{-1}

b. Decryption

Reading Assignment

Go back to your Applied Mathematics course and read about for computing Inverse of a matrix to decrypt ciphertext in the hill cipher technique.

Rotor cipher

Enigma Machine cipher

Transposition Techniques

- Systematically transpose the positions of plaintext elements (rearrange their orders). A transposition cipher does not substitute one symbol for another, instead it changes the location of the symbols.
- A symbol in the first position of the plaintext may appear in the tenth position of the ciphertext. A symbol in the eighth position in the plaintext may appear in the first position of the ciphertext.
- In the other words, a transposition cipher reorders (transposes) the symbols. This group of ciphers include:
    1. Keyless transposition ciphers
    2. Keyed transposition ciphers

1. Keyless Transposition ciphers

- The simple transposition ciphers are keyless. There are two methods for permutation of characters .
- In the first method, the text is written into a table column by column and then transmitted row by row.
- In the second method, the text is written into the table row by row and then transmitted column by column. Example: Rail fence cipher
- In this cipher the plaintext is arranged in two lines as a zigzag pattern (which means column by column); the ciphertext is created by reading the pattern row by row.

Cont'd

- For example, to send the message " meet me at the park" to the receiver, the sender writes

- He then creates the ciphertext " MEMATEAKETETHPR" by sending the first row followed by the second row
- The receiver receives the ciphertext and divides it in half (in this case the second half has one less character)
- The first half forms the first row; the second half the second row. The receiver reads the result in zigzag.

Cont'd

2. Keyed Transposition cipher:

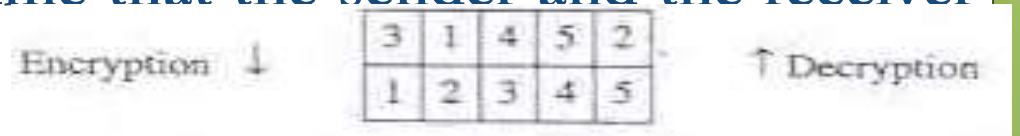
- The keyless ciphers permutes the characters by using writing plaintext in one way (row by row , for example) and reading it in another way (column by column , for example).
- The permutation is done on the whole plaintext to create the whole ciphertext.
- Another method is to divide the plain text into groups of predetermined size, called blocks, and then use a key to permute the characters in each block separately.

Cont'd

Example: The sender needs to send the message" enemy attacks tonight"

- In this case, both agreed to divide the text into groups of five characters and then permute the characters in each group.
- The following show the grouping after adding a bogus character at the end to make the last group the same size as the others.

  enemy attac kston ightz
- The key used for encryption and decryption is a permutation key, which shows how the character are permuted.
- For this message, assume that the sender and the receiver used the following key



Cont'd

- The third character in the plaintext block becomes the first character in the ciphertext block; the first character in the plaintext block becomes the second character in the ciphertext block; and so on. The permutation yields

  EEMYNTAACTTKONSHITZG

- The receiver divides the ciphertext into 5-character groups and , using the key in the reverse order, finds the plaintext.

Symmetric key cryptography

Block Ciphers

- A symmetric encryption algorithms in which a large block of plaintext bits (typically 64) is transformed as a whole into a ciphertext block of the same length.
- Block ciphers operate on blocks of plaintext and ciphertext- usually of 64 bits but sometimes longer.
- The groups of bits are called blocks. For modern computer algorithms, a typical block size is 64 bits.
- Block ciphers operate on blocks of a message and apply the encryption algorithm to an entire message block at the same time.

Cont'd

- With a block cipher, the same plaintext block will always encrypt to the same ciphertext block, using the same key.
- A block cipher is an encryption/decryption scheme in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. It may be viewed as a simple substitution cipher with large character size.
- The function is parameterized by a k-bit key K, taking values from a subset K (the key space) of the set of all k-bit vectors V_k.
- *n*-bit block cipher takes *n* bit plaintext and produces *n* bit ciphertext.

3/2/2018

2^n possible different plaintext blocks (inputs) will be there.

Widely used block Ciphers

Some of the block cipher algorithms widely used today are the following:

i. DES (Data Encryption Standard)

ii. Triple DES

iii. AES (Advanced Encryption Standard)

Block Ciphers: Data Encryption Standard (DES)

- The Data Encryption Standard, known as DES, is a simple block cipher developed way back in the 1970s. The design is based on the Lucifer cipher, a Feistel cipher developed by IBM.
- It is the most widely used encryption scheme. The plaintext is 64-bits in length, and the key is 56-bits in length.
- There are 16 rounds of processing. From the original 56-bit key, 16 subkeys are generated, one of which is used for each round.
- With a key length of 56 bits, there are 2^{56} possible keys, which is approximately 7.2 X 10^{16} keys. Thus, on the face of it, a brute-force attack appears impractical.

Block Ciphers: DES

- *Outline of the DES Algorithm:* DES operates on two inputs to the encryption function:
  - a 64-bit block of plaintext to be encrypted and the 56-bit key k.
- Note: Actually, the function expects a 64-bit key as input. However, only 56 of these bits are ever used; the other 8 bits (8^{th}, 16^{th}, 24^{th},…,64^{th}) can be used as parity bits or simply set arbitrarily.
- The processing of the plaintext proceeds in four phases: In first phase, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the *permuted input*. This permuted input is then broken into a right half and a left half, each 32-bits long.

Cont'd

- Second phase consists of 16 rounds of an identical operation, called the function F, in which data are combined with the key.
- This phase consists of sixteen rounds of the same function, which involves both permutation and substitution functions. In each round (see Figure):
- The key bits are shifted, and then 48 bits are selected from the 56 bits of the key.
- The right half of the data is expanded to 48 bits via an expansion permutation, combined with 48 bits of a shifted and permuted key via an XOR, sent through 8 S-boxes producing 32 new bits, and permuted again.

Cont'd

- The output of Function F is then combined with the left half via another XOR.
- The result of these operations becomes the new right half; the old right half becomes the new left half. These operations are repeated 16 times, making 16 rounds of DES.
- In the third phase, the output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key.
- The left and right halves of the output are swapped to produce the preoutput.

Cont'd

- In the final phase, the *preoutput* is passed through an inverse permutation (IP^{-1}) of the initial permutation function, to produce the 64-bit ciphertext.
- The right-hand portion of the following Figure shows the way in which the 56-bit key is used.
- Initially, the 64 bit key is passed through a permutation function, then 8 bits ($k_8, k_{16}, k_{24}, \ldots, k_{64}$) of K are discarded.
- Then, for each of the 16 rounds, a 48 bit subkey (K_i) is produced by the combination of a left circular shift and a permutation from the 56 bit key.
- The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.

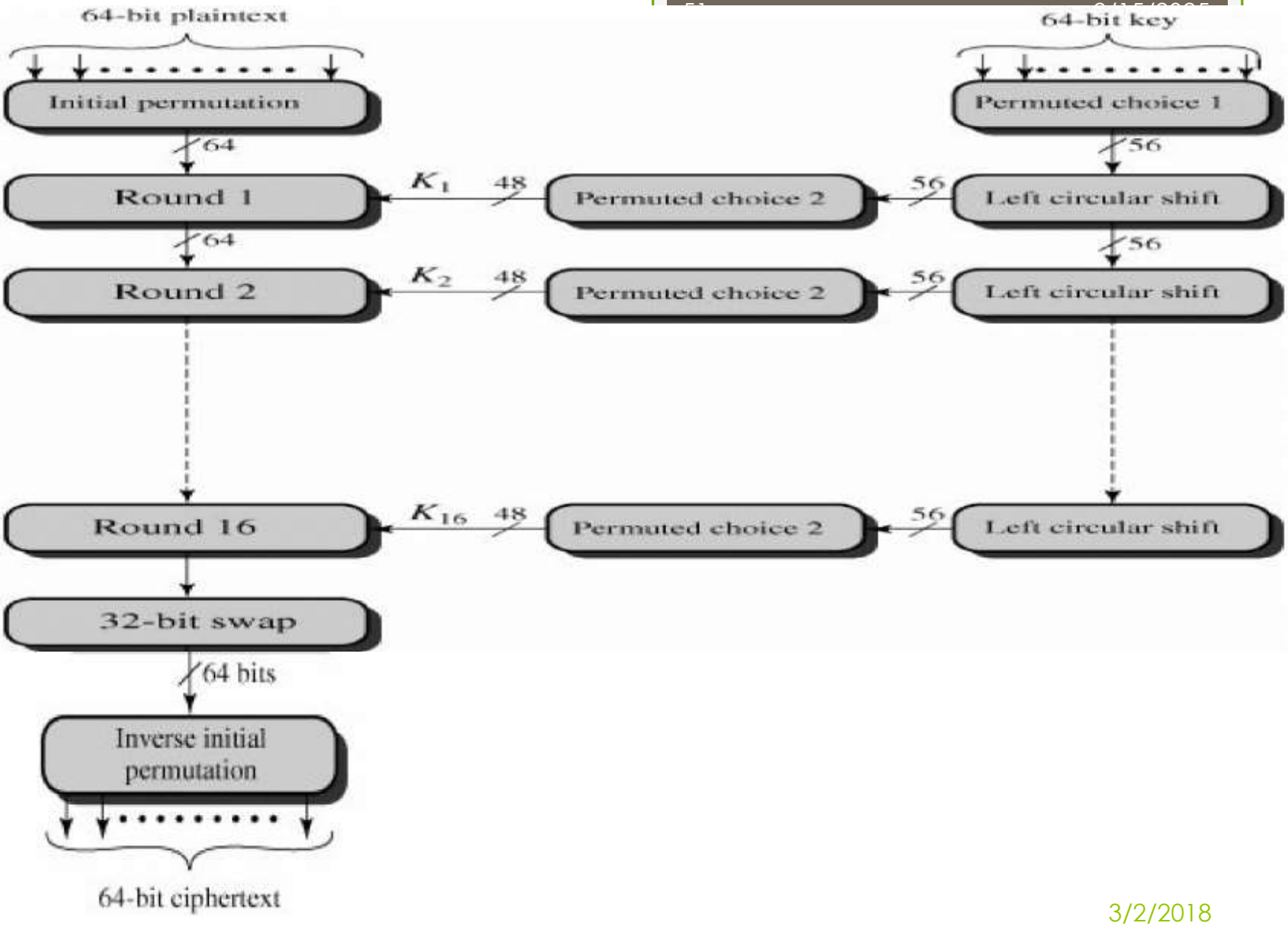


Figure: General Depiction of DES Encryption Algorithm

Cont'd

- *Outline of the DES Algorithm*

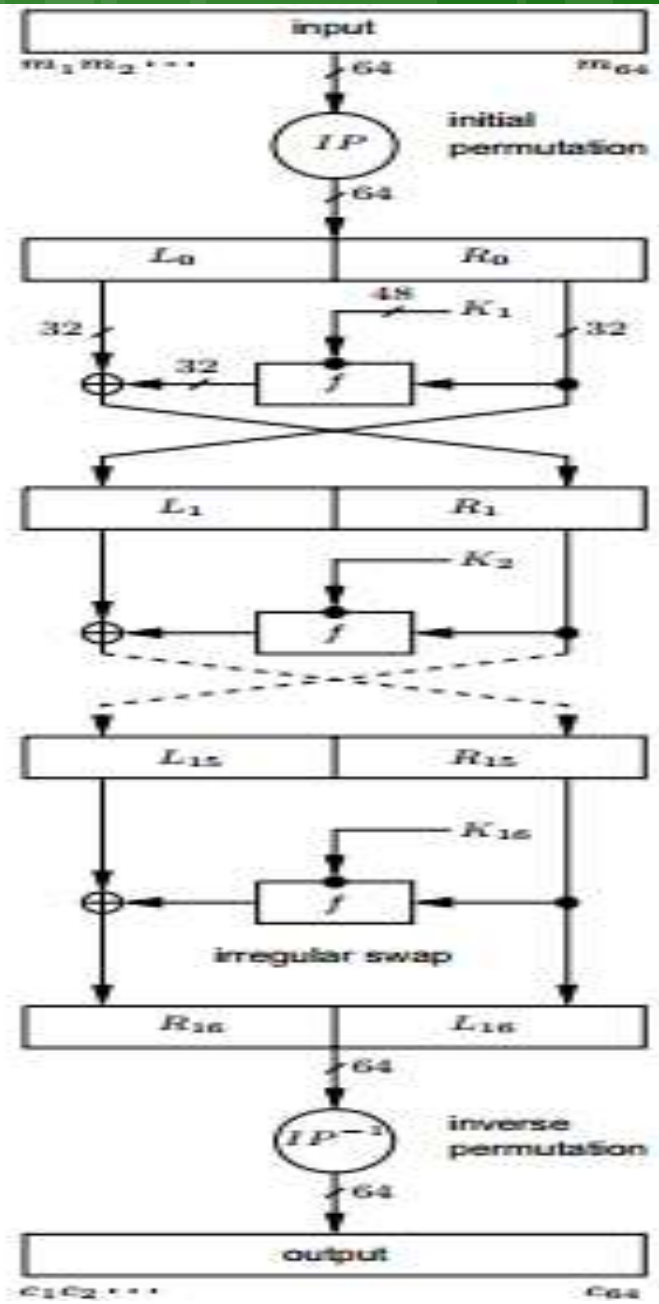
- With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher, as shown in the following Figure.

- For each *round* $i = 1, 2, \ldots, 16$, new left and right halves are computed according to the rule

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

where K_i is the *subkey* for round i.

Des computation path



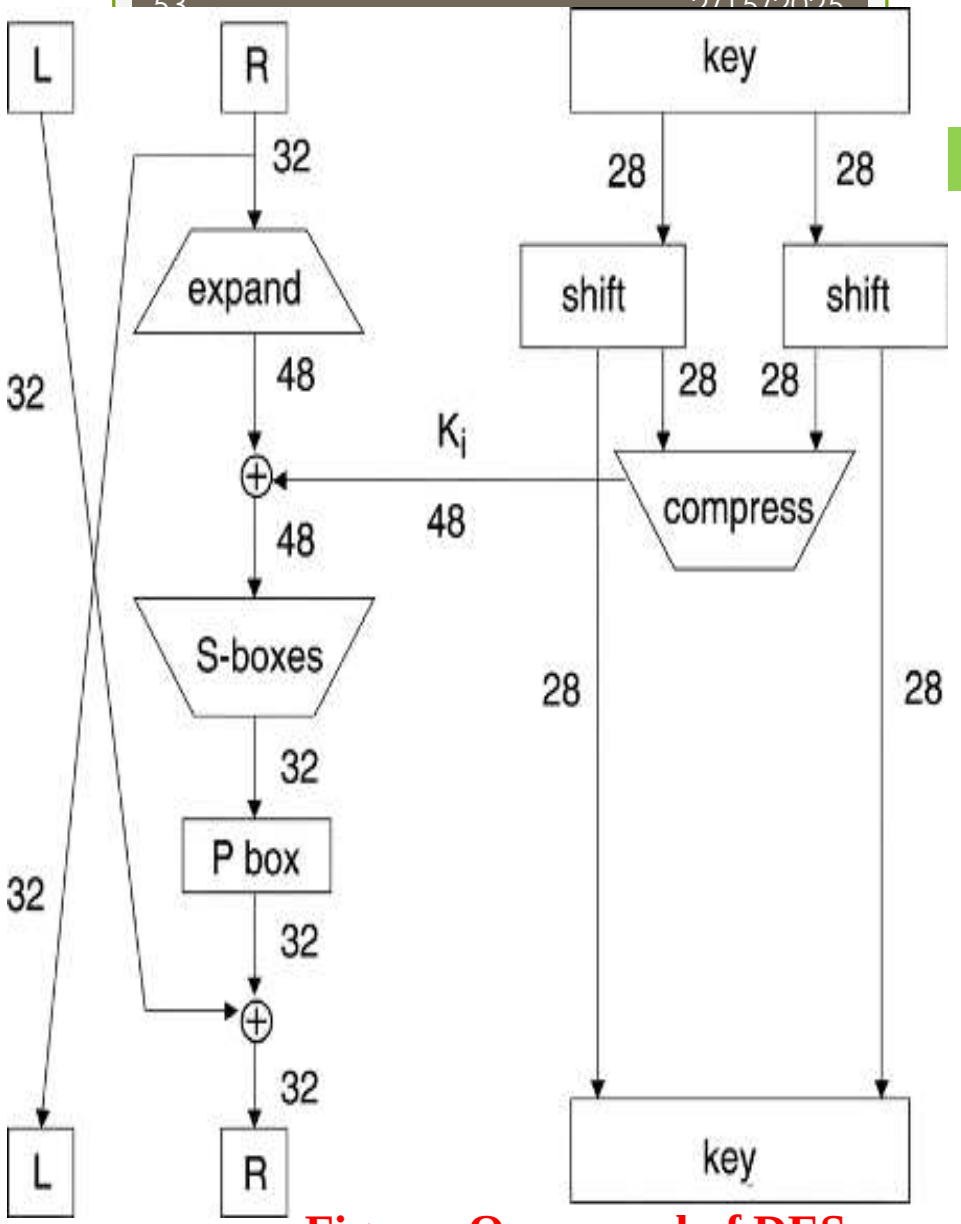


Figure: One round of DES

Block Ciphers: DES

- Unscrambling the previous diagram, we see that the DES round function F can be written as follows.

$$F(R_{i-1},K_i) = \text{P-box}(\text{S-boxes}(\text{Expand}(R_{i-1}) \oplus K_i))$$

- Since the DES block size is 64 bits, each L_i and R_i is 32 bits. The new left half is simply the old right half.
- The round function F is the composition of the expansion permutation, addition of subkey, S-boxes, and P-box.
- The expansion permutation expands its input from 32 to 48 bits (all bits are used once; some are used twice), and the 48 bit subkey is XORed with the result.
- The S-boxes then compress these 48 bits down to 32 bits before the result is passed through the P-box. The P-box output is then XORed with the old left half to obtain the new right half.

Block Ciphers: DES: DES FUNCTIONS IN DETAIL

Now let us describe each of the components of F.

1) The Initial Permutation

- The initial permutation and its inverse are defined by the following tables.
- These tables, like all the other tables in this chapter, should be read left to right, top to bottom.
- For example, the initial permutation moves bit 58 of the plaintext to bit position 1, bit 50 to bit position 2, bit 42 to bit position 3, and so forth.

Cont'd

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

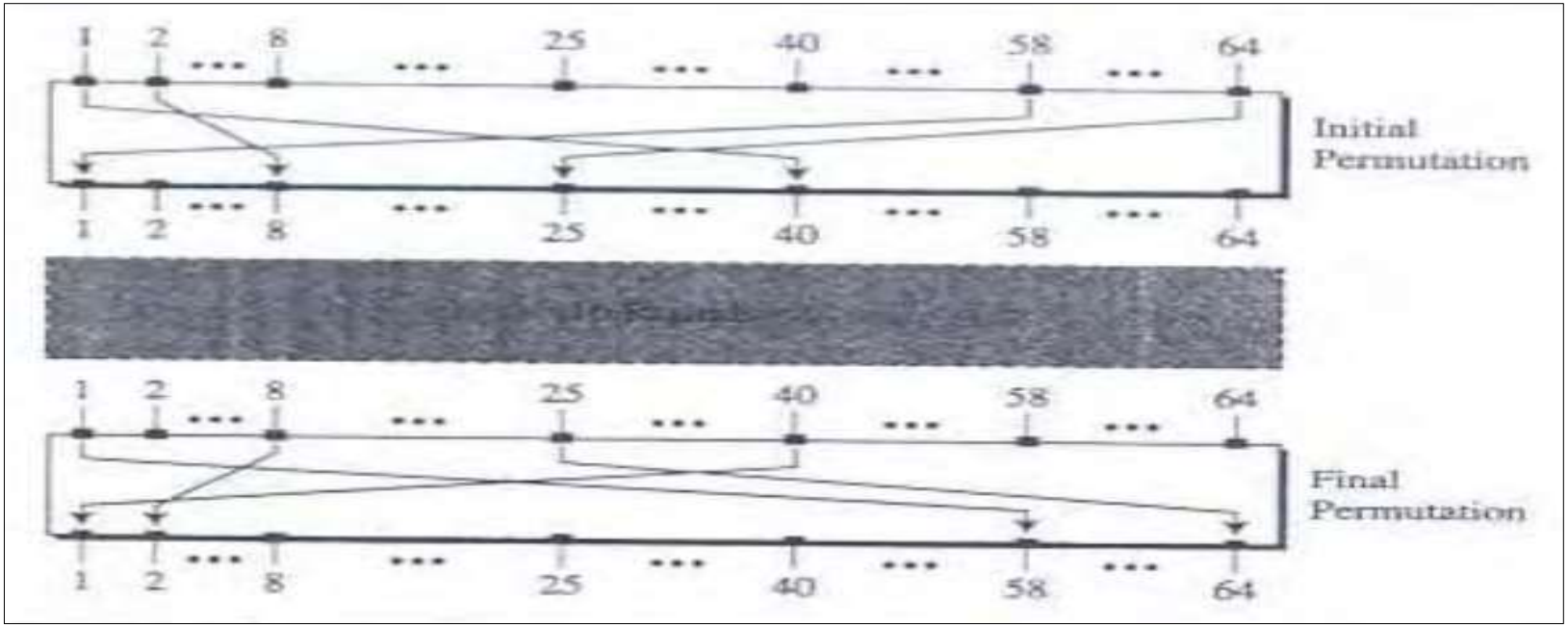


Figure: Initial and Final permutations in DES

Cont'd

The tables are to be interpreted as follows.

- The input to a table consists of 64 bits numbered from 1 to 64.
- The 64 entries in the permutation table contain a permutation of the numbers from 1 to 64.
- Each entry in the permutation table indicates the position of a numbered input bit in the output, which also consists of 64 bits.

➢ To see that these two permutation functions are indeed the inverse of each other, consider the following 64-bit input M, where M_i is a binary digit:

M1	M2	M3	M4	M5	M6	M7	M8
M9	M10	M11	M12	M13	M14	M15	M16
M17	M18	M19	M20	M21	M22	M23	M24
M25	M26	M27	M28	M29	M30	M31	M32
M33	M34	M35	M36	M37	M38	M39	M40
M41	M42	M43	M44	M45	M46	M47	M48
M49	M50	M51	M52	M53	M54	M55	M56
M57	M58	M59	M60	M61	M62	M63	M64

Cont'd

Then the permutation X= IP(M) is as follows:

M_{58}	M_{50}	M_{42}	M_{34}	M_{26}	M_{18}	M_{10}	M_2
M_{60}	M_{52}	M_{44}	M_{36}	M_{28}	M_{20}	M_{12}	M_4
M_{62}	M_{54}	M_{46}	M_{38}	M_{30}	M_{22}	M_{14}	M_6
M_{64}	M_{56}	M_{48}	M_{40}	M_{32}	M_{24}	M_{16}	M_8
M_{57}	M_{49}	M_{41}	M_{33}	M_{25}	M_{17}	M_9	M_1
M_{59}	M_{51}	M_{43}	M_{35}	M_{27}	M_{19}	M_{11}	M_3
M_{61}	M_{53}	M_{45}	M_{37}	M_{29}	M_{21}	M_{13}	M_5
M_{63}	M_{55}	M_{47}	M_{39}	M_{31}	M_{23}	M_{15}	M_7

- If we then take the inverse permutation
  $Y= IP^{-1}(X) = IP^{-1}(IP(M))$,

it can be seen that the original ordering of the bits is restored

Cont'd

2) The Key Generation/Transformation

- Returning back to Figures of general depiction and one round of DES, we see that a 64-bit key is used as input to the algorithm.
- The bits of the key are numbered from 1 through 64; every eighth bit is ignored, as indicated in the following table.

Bits included							Bits excluded
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

Table: Bits included and excluded in reducing the key size

Cont'd

The key is first subjected to a permutation governed by a table labeled Permuted Choice One (the following table). This is nothing but permutation of numbers (referring to bit positions) in the left of the previous table.

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,
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

Table: Permuted Choice One

- The resulting 56-bit key is then treated as two 28-bit quantities
- Each halves are separately subjected to a circular left shift, or rotation, of 1 or 2 bits, as governed by the following table.

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

Table: Number of key bits shifted per round

$$r_i = \begin{cases} 1 & \text{if } i \in \{1, 2, 9, 16\} \\ 2 & \text{otherwise.} \end{cases}$$

Cont'd

- These shifted values serve as input to the next round. They also serve as input to Permuted Choice Two, which produces a 48-bit output that serves as input to the function $F(R_{i-1}, K_i)$.
- Because this operation permutes the order of the bits as well as selects a subset of bits (48 bits from 56 bits are selected), it is called a compression permutation.

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

Table: Permuted Choice Two (Compression Permutation)

Cont'd

3) The Expansion Permutation

- This operation expands the right half of the data, Ri, from 32 bits to 48 bits. Because this operation changes the order of the bits as well as repeats certain bits, it is known as an expansion permutation.
- This operation has two purposes: It makes the right half the same size as the key for the XOR operation and it provides a longer result that can be compressed during the substitution operation.

4) The S-Box Substitution

- After the compressed key is XORed with the expanded block, the 48-bit result moves to a substitution operation.
- The substitutions are performed by 8 substitution boxes, or S-boxes. Each S-box has a 6-bit input and a 4-bit output, and there are 8 different S-boxes.
- The 48 bits are divided into eight 6-bit sub-blocks. Each separate block is operated on by a separate S-box: The first block is operated on by S-box 1, the second block is operated on by S-box 2, and so on.

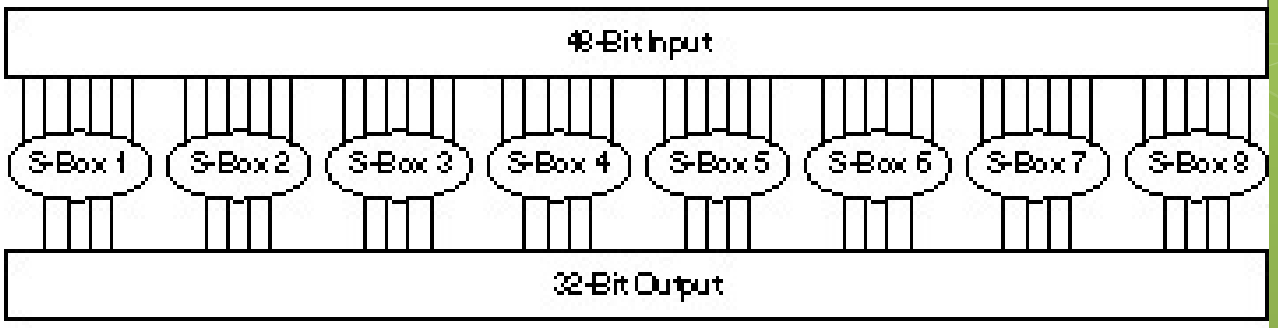


Figure: S-Box Substitution

Cont'd

- Each S-box is a table of 4 rows and 16 columns. Each entry in the box is a 4-bit number. The 6 input bits of the S-box specify under which row and column number to look for the output.
- The following table shows all eight S-boxes

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

The S-Box Substitution

S_5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Cont'd

- The input bits specify an entry in the S-box in a very particular manner. Consider an S-box input of 6 bits, labeled $b_1b_2b_3b_4b_5$ and b_6.
- Bits b_1 and b_6 are combined to form a 2-bit number, from 0 to 3, which corresponds to a row in the table.
- The middle 4 bits, b_2 through b_5 are combined to form a 4-bit number, from 0 to 15, which corresponds to a column in the table.
- For example, assume that in the S1 101101. The first and last bits combine to form 11, which corresponds to row 3 of the S-box.
- The middle 4 bits combine to form 0110, which corresponds to the column 6 of the same S-box.
- The entry under row 3, column 6 of S-box1 is 1. (Remember to count rows and columns from 0 and not from 1). The value 0001 is substituted for 101101.

The S-Box Substitution

	$b_1b_2b_3b_4$															
b_0b5	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

Table: Entry for S-Box one (S-1)

5) The P-Box Permutation

- The 32-bit output of the S-box substitution is permuted according to a P-box.
- This permutation maps each input bit to an output position; no bits are used twice and no bits are ignored. This is called a straight permutation or just a permutation.

6) The Final Permutation

- The final permutation is the inverse of the initial permutation and is described before.
- Note that the left and right halves are not exchanged after the last round of DES; instead the concatenated block $R_{16}L_{16}$ is used as the input to the final permutation.
- There's nothing going on here; exchanging the halves and shifting around the permutation would yield exactly the same result. This is so that the algorithm can be used to both encrypt and decrypt.

Block Ciphers: Triple DES

- A popular variant of DES is triple DES, or 3DES. 3DES was developed in 1999 by IBM – by a team led by Walter Tuchman. 3DES has a 168-bit key and enciphers blocks of 64 bits.
- There are four versions of 3DES.
- The first simply encrypts the plaintext three times, using three different keys: K_1, K_2, and K_3
- It is known as DES-EEE3 mode (the Es indicate that there are three encryption operations, where as the numeral 3 indicates that three different keys are used).
- DES-EEE3 can be expressed using the following notation, where E(K, P) represents the encryption of plaintext P with key K :

3/2/2018

- $E(K_1, E(K_2, E(K_3, P)))$ DES-EEE3 has an effective key length of 168 bits

Block Ciphers: Triple DES

- The second variant (DES-EDE3) also uses three keys but replaces the second encryption operation with a decryption operation:

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

- The third version of 3DES (DES-EEE2) uses only two keys, K_1 and K_2, as follows: $E(K_1, E(K_2, E(K_1, P)))$
- The fourth variant of 3DES (DES-EDE2) also uses two keys but uses a decryption operation in the middle: $E(K_1, D(K_2, E(K_1, P)))$
- Both the third and fourth variants have an effective key length of 112 bits.
- In DES-EDE2, the reason for using decryption as the second step is the backwards compatibility with single DES when it is used with $K_1 = K_2$.
- That is when $K_1 = K_2 = K$ then it collapses to single DES

  $C = E(D(E(P, K), K), K) = E(P, K)$.

Thank you