

Lecture 2: Symmetric Cryptography

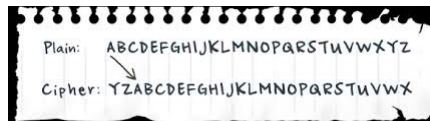
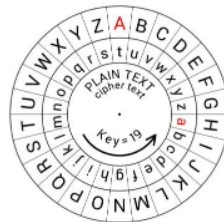
Network Security – 2025
Professor Anh T. Pham

Cryptography or Cipher

- ❖ What is the first thing you might think of?
 - Email encryption?
 - Secure connection to a shopping website?
 - Smart payment?
- ❖ Cryptography is however a rather old business



Scytale of Sparta (~500BC)



Caesar Cipher Wheel (~50BC)

Contents

- ❖ Cryptography
 - History, terms
- ❖ Cryptography systems
- ❖ Encryption Model
- ❖ Feistel Cipher Structure
- ❖ Block Ciphers and Standard
 - Data Encryption Standard
 - Triple DES
 - Advanced Encryption Standard
- ❖ Stream Cipher
 - RC4
- ❖ Modes of Operations

Part 2

N02: Network Security

2

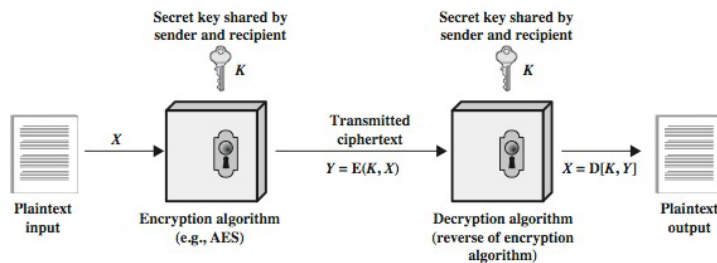
Cryptosystems

Cryptosystems can be characterized by:

- ❖ Type of encryption operations
 - **Substitution**: each element in the plaintext (bit, letter, group of bits or letters) is mapped into another element
 - **Transposition**: elements in the plaintext are re-arranged
 - **Product**: multiple stages of substitutions and transpositions
- ❖ Number of keys
 - Single-key or private
 - Two-key or public
- ❖ The way in which plaintext is processed
 - Block cipher processes the input one block of elements at a time,
 - Stream cipher processes the input elements continuously, producing output one element at a time, as it goes along

Symmetric Cryptosystem Model

One single secret key, shared between Sender and Recipient



Mathematically we have:

$$Y = E(K, X)$$

$$X = D(K, Y)$$

$E(K, X)$ = encryption of X using key K

$D(K, Y)$ = decryption of Y using key K

Components

1. **Plaintext:** This is the original message or data that is fed into the algorithm as input.
2. **Encryption algorithm (cipher):** The encryption algorithm performs various substitutions and transformations on the plaintext.
3. **Secret key:** The secret key is also input to the algorithm. The exact substitutions and transformations performed by the algorithm depend on the key.
4. **Ciphertext:** This is 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.
5. **Decryption algorithm (decipher):** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the same secret key and produces the original plaintext.

Requirements

- ❖ A **strong encryption algorithm**, such that
 - **Encryption algorithm is known** by opponent, and that
 - The opponent should be **unable to decrypt ciphertext** or **discover the key** even if he/she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext.
- ❖ Sender and receiver must have **obtained copies of the secret key** in a secure fashion and must keep the key secure.
 - If someone can discover the key and knows the algorithm, all communication using this key is readable.
 - Security of symmetric encryption **depends on the secrecy of the key**, not the secrecy of the algorithm.

Cryptanalysis

- ❖ **Terminology**
 - **Cryptography** - study of encryption principles/methods
 - **Cryptanalysis (codebreaking)** - study of principles/ methods of deciphering ciphertext without knowing key
 - **Cryptology** - field of both cryptography and cryptanalysis
- ❖ **Cryptanalysis: the process of attempting to discover the plaintext or key**
 - The strategy depends on the **nature of the encryption scheme** and
 - The information available to the cryptanalyst

Type of Cryptanalytic Attacks

Type of Attack	Known to Cryptanalyst
Ciphertext only	<ul style="list-style-type: none"> • Encryption algorithm • Ciphertext to be decoded
Known plaintext	<ul style="list-style-type: none"> • Encryption algorithm • Ciphertext to be decoded • One or more plaintext–ciphertext pairs formed with the secret key
Chosen plaintext	<ul style="list-style-type: none"> • Encryption algorithm • Ciphertext to be decoded • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen ciphertext	<ul style="list-style-type: none"> • Encryption algorithm • Ciphertext to be decoded • Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen text	<ul style="list-style-type: none"> • Encryption algorithm • Ciphertext to be decoded • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key • Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

Part 2

N02: Network Security

9

Cryptanalysis (Attack)

- ❖ The most difficult problem is when there is ciphertext only
 - Assuming that the encryption algorithm is known (sometimes, it is NOT)
- ❖ One possible attack is to **brute-force approach**: try **every possible key** on a piece of ciphertext until an intelligible translation into plaintext is obtained
- ❖ **How long does it take, on average?**

Part 2

N02: Network Security

10

Time Required for Exhaustive Search (BF Attack)

Key Size (bits)	Number of Alternative Keys	Time required at 1 decryption/μs	Time required at 10 ⁶ decryptions/μs
32	$2^{32} = 4.3 \times 10^9$	$2^{31} \mu s = 35.8 \text{ minutes}$	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55} \mu s = 1142 \text{ years}$	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127} \mu s = 5.4 \times 10^{24} \text{ years}$	$5.4 \times 10^{18} \text{ years}$
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167} \mu s = 5.9 \times 10^{36} \text{ years}$	$5.9 \times 10^{30} \text{ years}$
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26} \mu s = 6.4 \times 10^{12} \text{ years}$	$6.4 \times 10^6 \text{ years}$

On average, half of all possible keys must be tried to achieve success

Secured System: What it mean?

- ❖ **Unconditionally secure**
 - No matter how much ciphertext is available, how much time and computing ability an opponent has, it is impossible for him/her to decrypt the ciphertext
 - Basically, **no such algorithm, except a scheme known as “one-time pad”**
- ❖ **Computationally secure**
 - 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

Part 2

N02: Network Security

11

Part 2

N02: Network Security

12

Classical Cipher

❖ Substitution technique

- Is one in which the letters of plaintext are replaced by other letters or by numbers or symbols
- If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns

❖ Caesar cipher

- Simplest and earliest known use of a *substitution cipher* - used by Julius Caesar
- Involves replacing each letter of the alphabet with the letter standing *three places* further down the alphabet

Example

❖ The key

```
plain:  a b c d e f g h i j k l m n o p q r s t u v w x y z
cipher: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

Plaintext : meet me after the toga party

Ciphertext : PHHW PH DIWHU WKH WRJD SDUWB

Questions

❖ In case of Caesar cipher (for the English alphabet), if we use **Brute Force attack**, how many key do we have to check?

- How many characters in the alphabet?
- What is the permutation of that alphabet?
- How many keys can we have?
- How long does it take if we have decryption speed of **1 decryption/μs**

❖ What if we can guess 1 character?

- 2 character?
- 3 character?
- 5 character?
- 8 character?

Cryptanalysis (Attack): Actually

❖ To supplement the brute-force approach, **some degree of knowledge** about the expected plaintext is needed, and some means of automatically distinguishing plaintext from garble is also needed

❖ Relies on the nature of the algorithm, plus perhaps some knowledge of the **general characteristics of the plaintext** or **even some sample** plaintext (ciphertext pairs)

❖ Exploits the characteristics of the algorithm to attempt to deduce a **specific plaintext** or to **deduce the key being used**

We will learn more on this in Lab. Exercise 2

More on Classical Cipher: Lab 2 Lecture

Block Cipher

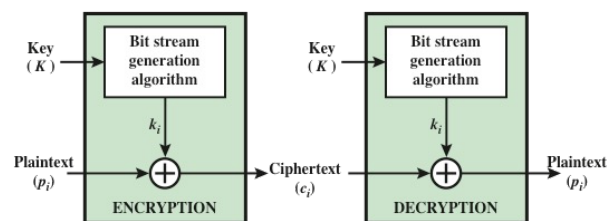
Part 2 N02: Network Security

18

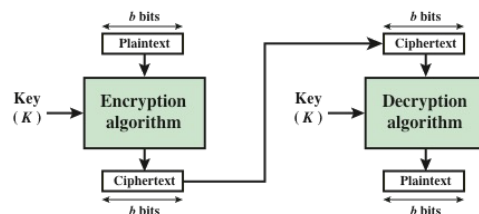
Block vs. Stream Cipher

5% Question

- **Stream cipher:** encrypt a digital data stream one bit or one byte at a time.
- **Block cipher:** a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length



(a) Stream Cipher Using Algorithmic Bit Stream Generator



(b) Block Cipher

Figure 4.1 Stream Cipher and Block Cipher

Block Ciphers

❖ Feistel Cipher Structure

❖ Block Cipher Standards

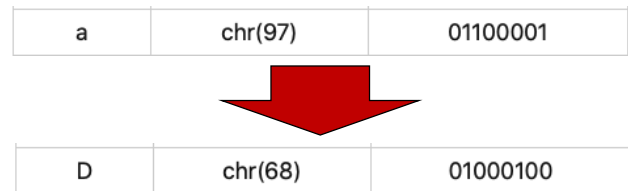
- Data Encryption Standard (DES)
- Triple DES (3DES)
- Advanced Encryption Standard (AES)

Feistel Cipher: Motivation

- ❖ A block cipher operates on a **plaintext block of n bits** to produce a **ciphertext**

- Example?

plain: a b c d e f g h i j k l m n o p q r s t u v w x y z
cipher: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C



Feistel Cipher: Motivation (cont.)

- ❖ **Block of n bits** → there are 2^n possible different plaintext blocks
- ❖ Reversibility (i.e., for decryption to be possible): each plaintext must produce a **unique ciphertext block**
- ❖ Example, $n = 2$

Reversible Mapping	
Plaintext	Ciphertext
00	11
01	10
10	00
11	01

- ❖ How many possible mappings (considering it is reversible)?

5% Question

Reversible vs. Irreversible

- ❖ Example of Irreversible Mapping

Irreversible Mapping	
Plaintext	Ciphertext
00	11
01	10
10	01
11	01

- ❖ Why this one is NOT reversible?

Another Example with $n = 4$

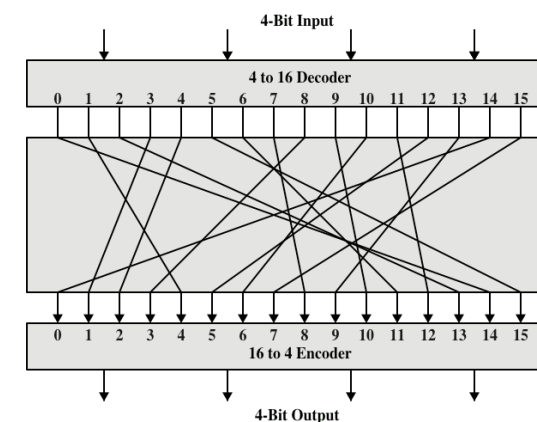


Figure 4.2 General n -bit- n -bit Block Substitution (shown with $n = 4$)

Encryption and Decryption Table

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

What is the Problem?

- ❖ Previously mentioned Block Cipher is called “Ideal Block Cipher” → it allows the maximum number of possible encryption mapping between plaintext and ciphertext
- ❖ How long the key for n-bit block cipher?
- ❖ We need key to determine the **mapping** from 2^n possible different plaintext blocks to 2^n possible different cipher blocks

→ This requires $n \times 2^n$ bits for the key!!!

5% Question

- ❖ Example: in the previous n=4 code, encoder needs to maintain (store) the following key

Encryption key

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

How big is the key?

4 (one ciphertext) × 16 (row) = 64 bits

- ❖ What is the key length for 64-bit block?

5% Question

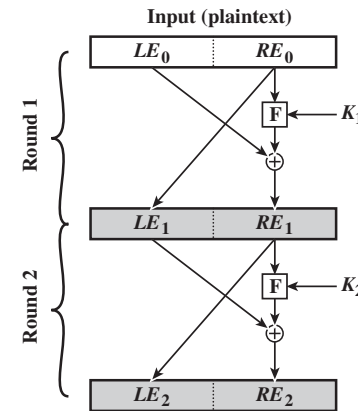
Solution: Feistel Cipher

- ❖ Horst Feistel (of IBM in 1973) proposed the use of a cipher that **alternates substitutions and permutations**
 - **Substitution:** Each plaintext element or group of elements is uniquely **replaced by** a corresponding ciphertext element or group of elements.
 - **Permutation:** A sequence of plaintext elements is replaced by a permutation of that sequence (no elements are added or deleted or replaced in the sequence, rather **the order in which the elements appear in the sequence is changed**)
- ❖ Many symmetric block encryption algorithms, **including DES**, have a structure following this principle

Solution: Feistel Cipher (2)

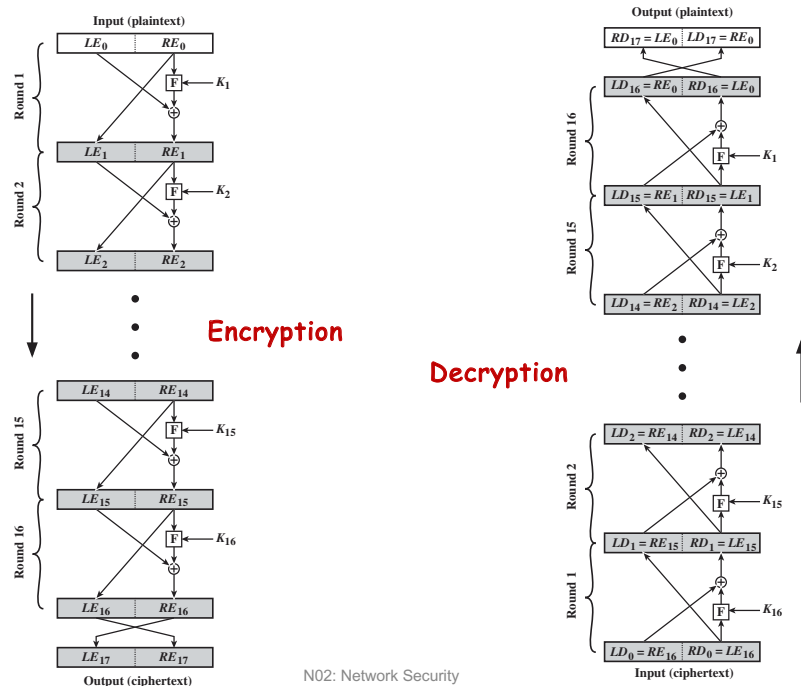
- ❖ This in fact approximate the ideal block cipher by utilizing the concept of a **product cipher** which is the execution of **two or more simple ciphers in sequence**
 - in such a way that the final result or product is **cryptographically stronger** than any of the component ciphers.
- ❖ Develop a block cipher with a **key length of k bits** and a **block length of n bits**, allowing a total of 2^k possible transformations, rather than the $2^n!$ transformations available with the ideal block cipher

Feistel Cipher Structure

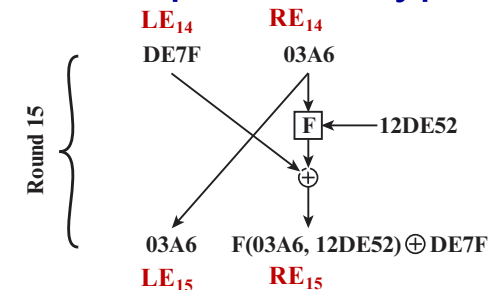


- ❖ Partitions input block (plaintext) into **two halves**
 - Process through multiple rounds: the **same structure but different key** (K_1 and K_2)
 - Each round performs a **substitution** (XOR) on left data half based on **round function (F)** of right half & subkey (*same structure for each round*)
 - Then have **permutation** swapping halves

F is not required to be a reversible function, even, it is OK, taking a limiting case, in which F produces a constant output (e.g., all ones)

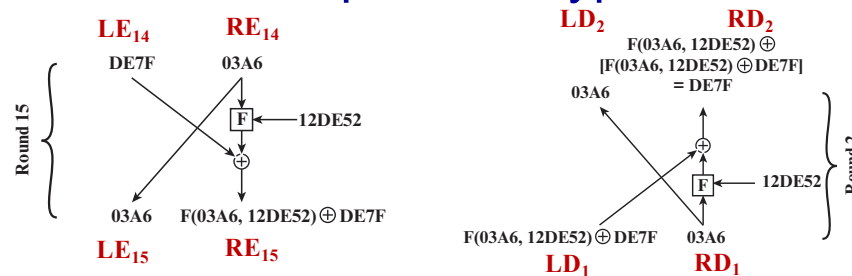


Feistel: Example - Encryption



- ❖ 32-bit input: DE7F 03A6
 - $LE_{14} = \text{DE7F}$
 - $RE_{14} = \text{03A6}$
- ❖ 16-bit key, $K_{15} = \text{12DE52}$
- ❖ After encryption we have
 - $LE_{15} = \text{03A6}$
 - $RE_{15} = F(\text{03A6}, \text{12DE52}) \oplus \text{DE7F}$

Feistel: Example - Decryption



- ❖ The decryption is inverse process of encryption
 - Output of Round 16 (encryption) → the ciphertext
 - Input of Round 1 (decryption) is the ciphertext
- ❖ To confirm the Feistel works, we need to prove that
 - When the input of round 2 (decryption) = the output of round 15 encryption, then
 - The output of round 2 = the input of round 15

Part 2

N02: Network Security

33

Feistel: Example - Decryption

❖ Input (ciphertext)

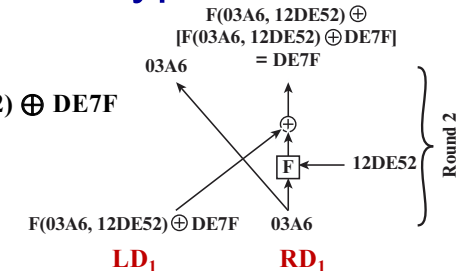
- $LD_1 = RE_{15} = F(03A6, 12DE52) \oplus DE7F$
- $RD_1 = LE_{15} = 03A6$

❖ Output of round 2

- $LD_2 = 03A6 = RE_{14}$
- $RD_2 = F(03A6, 12DE52) \oplus [F(03A6, 12DE52) \oplus DE7F]$
 $= DE7F = LE_{14}$

Here we use the following XOR properties

$$\begin{aligned}
 [A \oplus B] \oplus C &= A \oplus [B \oplus C] \\
 D \oplus D &= 0 \\
 E \oplus 0 &= E
 \end{aligned}$$



Part 2

N02: Network Security

34

Design Elements

- ❖ **Block size:** *increasing size improves security*, but slows cipher
- ❖ **Key size:** *increasing size improves security*, makes exhaustive key searching harder, but may slow cipher
- ❖ **Number of rounds:** *increasing number improves security*, but slows cipher
- ❖ **Subkey generation algorithm:** *greater complexity can make analysis harder*, but slows cipher
- ❖ **Round function (F):** *greater complexity can make analysis harder*, but slows cipher
- ❖ **Fast software en/decryption:** more recent concern for practical use
- ❖ **Ease of analysis:** for easier validation & testing of strength

Part 2

N02: Network Security

35

Data Encryption Standard (DES)

- ❖ Issued in 1977 by the National Bureau of Standards (now NIST) as Federal Information Processing Standard 46
- ❖ Was the most widely used encryption scheme until the introduction of the Advanced Encryption Standard (AES) in 2001
- ❖ Algorithm itself is referred to as the Data Encryption Algorithm (DEA)
 - Data are encrypted in 64-bit blocks using a 56-bit key
 - The algorithm transforms 64-bit input in a series of rounds into a 64-bit output
 - The same rounds, with the same key, are used to reverse the encryption
 - There are 16 rounds of processing. From the original 56-bit key, 16 subkeys are generated, one of which is used for each round

Part 2

N02: Network Security

36

Strength of DES

❖ Algorithm

- **most-studied** encryption algorithm in existence, no one has so far succeeded in discovering a fatal weakness in DES

❖ Key size (56-bit)

- With a key length of 56 bits, there are 2^{56} possible keys, which is approximately 7.2×10^{16} keys \rightarrow a brute-force attack appears impractical
 - one DES encryption per microsecond \rightarrow more than 1,000 year
 - However, with recent development, cheaper hardware \rightarrow less than 3 days (DES cracker, price: 250,000 USD in 1998)

- ❖ Note: the analyst must be able to recognize plaintext as plaintext \rightarrow more difficult in cases of **compressed or numerical data**

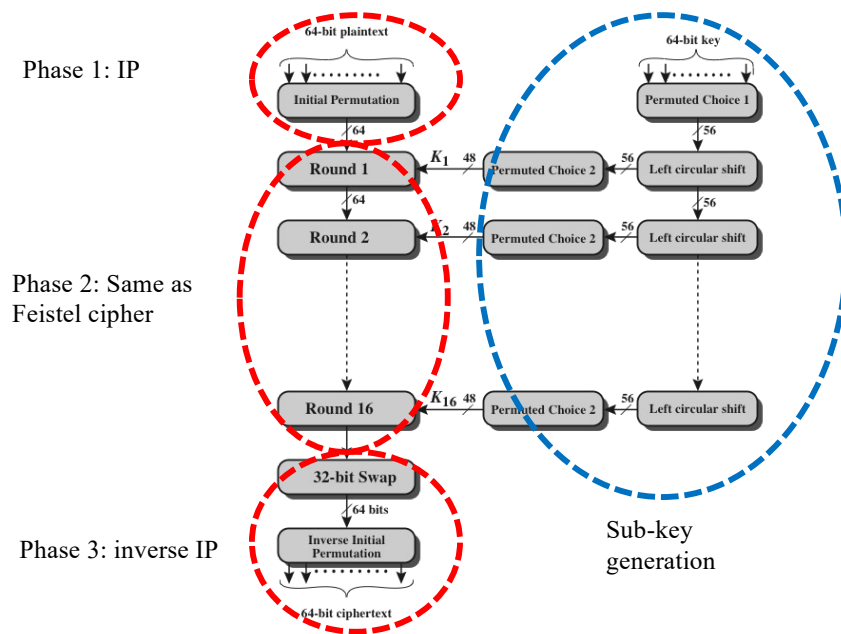


Figure 4.5 General Depiction of DES Encryption Algorithm
N02: Network Security

Part 2

37

Part 2

N02: Network Security

38

Practical BF Attacks (56-bit)

- ❖ In 1977, Diffie and Hellman proposed a machine costing an estimated US\$20 million which could find a DES key (56-bit) in a single day
- ❖ By 1993, Wiener had proposed a key-search machine costing US\$1 million which would find a key within 7 hours
- ❖ In 1997, 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
- ❖ In 1998, the EFF built Deep Crack (named in reference to IBM's Deep Blue chess computer) for less than \$250,000. In response to DES Challenge II-2, on July 15, 1998, Deep Crack decrypted a DES-encrypted message after only 56 hours of work, winning \$10,000.

How to Improve DES Strength?

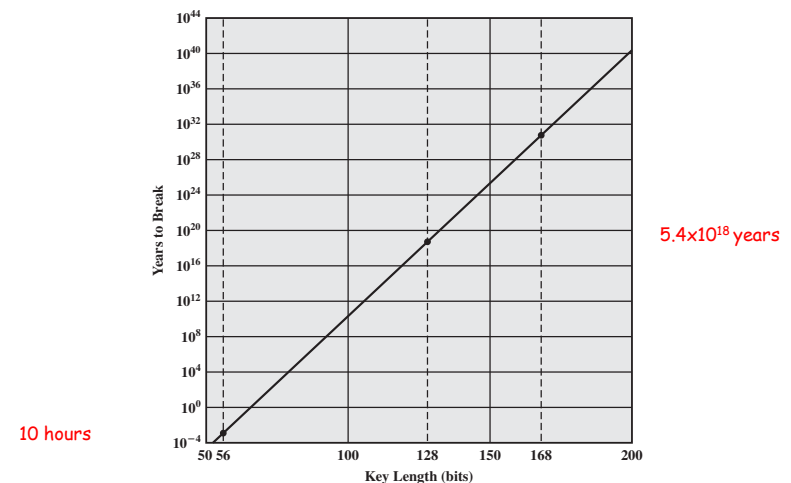


Figure 2.3 Time to Break a Code (assuming 10^6 decryptions/ μ s)

Part 2

N02: Network Security

39

Part 2

N02: Network Security

40

Multiple Encryption & DES

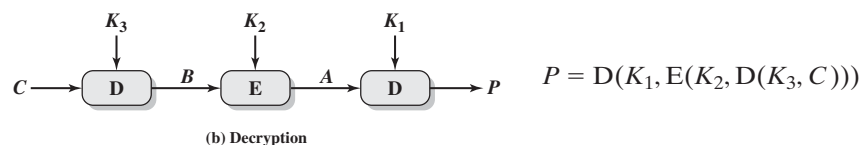
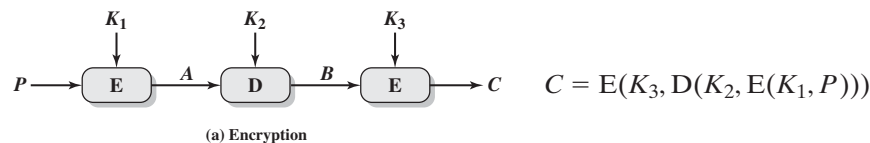
- ❖ Clear a replacement for DES was needed
 - theoretical attacks that can break it
 - demonstrated exhaustive key search attacks
- ❖ AES is a new cipher alternative
- ❖ Prior to this alternative was to use multiple encryption with DES implementations
- ❖ Triple-DES is the chosen form

Triple-DES

- ❖ Triple DES (3DES) was first standardized for use in financial applications in ANSI standard X9.17 in 1985
- ❖ 3DES was incorporated as part of the Data Encryption Standard in 1999 with the publication of FIPS 46-3
- ❖ No current known practical attacks
 - several proposed impractical attacks might become basis of future attacks

Triple-DES: Encryption & Decryption

- ❖ 3DES uses three keys and three executions of the DES algorithm



$E[K, X]$ = encryption of X using key K
 $D[K, Y]$ = decryption of Y using key K

Triple-DES: Discussion

- ❖ With three distinct keys, 3DES has an effective key length of **168 bits**.
 - **5.8×10^{29} years to for BF attack**
- ❖ FIPS 46-3 also allows for the use of two keys, with $K_1 = K_3$; this provides for a key length of 112 bits.
- ❖ It is easy to see that 3DES is a formidable algorithm
 - the underlying cryptographic algorithm is DEA, 3DES can claim the same resistance to cryptanalysis based on the algorithm as is claimed for DEA
 - 168-bit key length, brute-force attacks are effectively impossible

AES: Advanced Encryption Standard

- ❖ Symmetric-key NIST standard, replaced DES (Nov 2001)
- ❖ Processes data in 128-bit blocks
- ❖ 128, 192, or 256-bit keys
- ❖ Brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

Block Ciphers: Mode of Operations

Modes of Operation

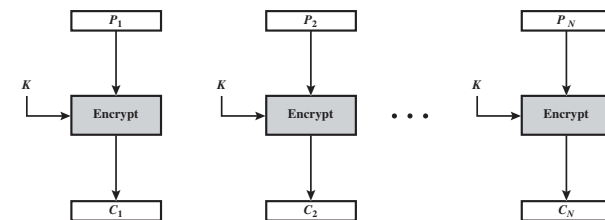
- ❖ A symmetric block cipher processes one block of data at a time
 - In the case of DES and 3DES, **the block length is $b = 64$ bits**
 - For AES, **the block length is $b = 128$**
 - For longer amounts of plaintext, it is necessary to **break the plaintext into b -bit blocks**, padding the last block if necessary
- ❖ **Five modes** of operation have been defined by NIST
 - Intended to cover virtually all of the possible applications of encryption for which a block cipher could be used
 - Intended for use with any symmetric block cipher, including triple DES and AES

1. Electronic Codebook (ECB)
 - Each block of plaintext bits is **encoded independently** using the same key
2. Cipher Block Chaining (CBC)
 - The input to the encryption algorithm is the **XOR of the next block of plaintext and the preceding block of ciphertext**
3. Cipher Feedback (CFB)
 - Input is processed **s bits at a time**. Preceding ciphertext is used as input to the encryption algorithm to **produce pseudorandom output**, which is XORed with plaintext to produce next unit of ciphertext.
4. Output Feedback (OFB)
 - Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used
5. Counter (CTR)
 - Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.

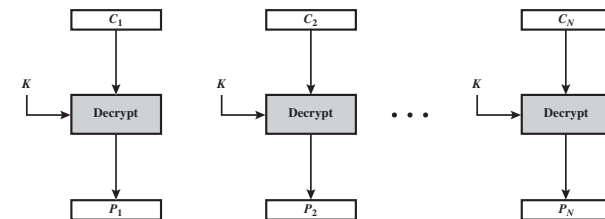
Electronic Codebook Book (ECB)

- ❖ Plaintext is handled b bits at a time and each block of plaintext is encrypted using the same key
- ❖ The term “codebook” is used because, for a given key, there is a **unique ciphertext for every b -bit block of plaintext**
 - One can imagine a gigantic codebook in which there is an entry for every possible b -bit plaintext pattern showing its corresponding ciphertext
- ❖ With ECB, if the same b -bit block of plaintext appears more than once in the message, it always produces the same ciphertext
 - Because of this, for lengthy messages, **the ECB mode may not be secure**
 - If the message is **highly structured**, it may be possible for a cryptanalyst to exploit these regularities

ECB Mode Operation



(a) Encryption

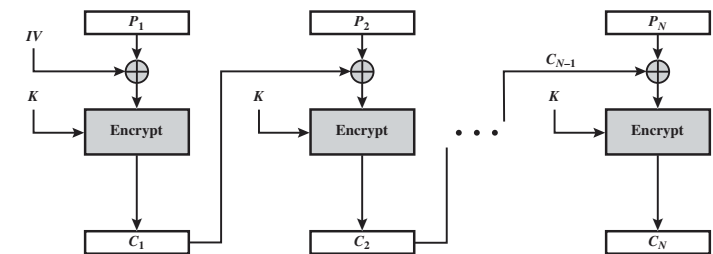


(b) Decryption

Cipher Block Chaining (CBC)

- ❖ To overcome the security deficiencies of ECB, we would like a technique in which the same plaintext block, if repeated, produces different ciphertext blocks.
- ❖ Cipher Block Chaining (CBC)
 - Message is broken into blocks
 - Linked together in encryption operation
 - Each previous cipher blocks is chained with current plaintext block, hence name
- ❖ **We will work with this scheme in Lab Ex. 2.**

Cipher Block Chaining (CBC)



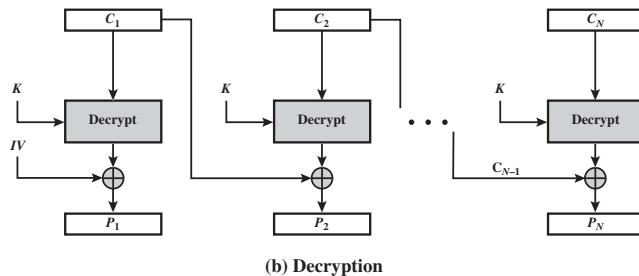
(a) Encryption

$$C_1 = E(K, [P_1 \oplus IV])$$

$$C_j = E(K, [P_j \oplus C_{j-1}]) \quad j = 2, \dots, N$$

- **The IV must be known to both the sender and receiver but be unpredictable by a third party.**
- For maximum security, the IV should be protected against unauthorized changes.

Cipher Block Chaining (CBC)



$$P_1 = D(K, C_1) \oplus IV$$

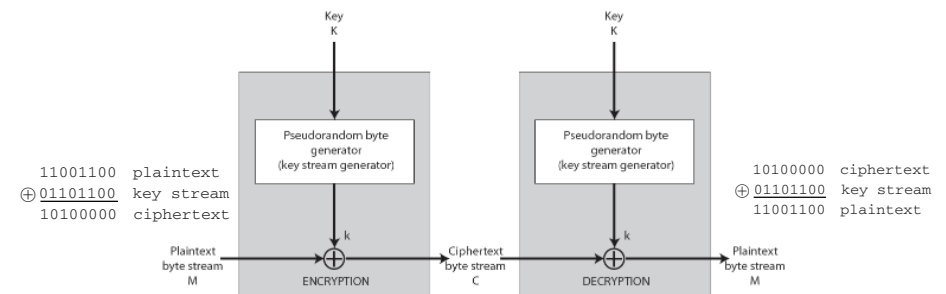
$$P_j = D(K, C_j) \oplus C_{j-1} \quad j = 2, \dots, N$$

Stream Cipher & RC4

Stream Cipher Structure

- **Block cipher:** a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length
- **Stream cipher:** encrypt a digital data stream **one bit or one byte** at a time
 - a stream cipher may be designed to operate on **one bit at a time** or on units larger than **a byte at a time**, anyway
- Block ciphers are **FAR more common**, stream ciphers are more appropriate for some applications, such as Internet: browser, web – stream of data.
 - How about email, FTP?
- We will look at basic design of stream ciphers, and examine RC4, the most popular symmetric stream cipher.

Stream Cipher Structure



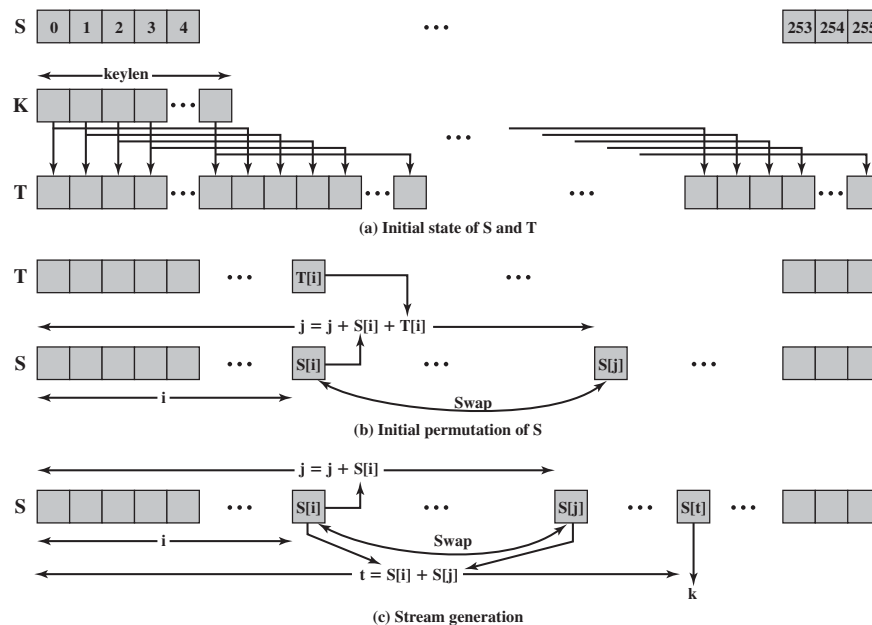
- ❖ A key is input to a pseudorandom bit generator that produces a stream of 8-bit numbers that are **apparently random**
- ❖ The output of the generator, called a **keystream**, is combined one byte at a time with the plaintext stream using the bitwise XOR
- ❖ Decryption requires the use of the same **pseudo-random sequence**

Design Considerations

1. The encryption sequence should have a **large period**, the longer the period of repeat the more difficult it will be to do cryptanalysis.
2. The keystream should approximate the properties of a **true random number stream** as close as possible, the more random-appearing the keystream is, the more randomized the ciphertext is, making cryptanalysis more difficult.
3. To guard against brute-force attacks, the **key needs to be sufficiently long**. The same considerations as apply for block ciphers are valid here. Thus, with current technology, a key length of at least **128 bits is desirable**

The RC4 Algorithm

- ❖ A stream cipher designed in 1987 by Ron Rivest for RSA Security
- ❖ It is a variable key-size stream cipher with byte-oriented operations
- ❖ The algorithm is based on the use of a random permutation
- ❖ Applications
 - Secure Sockets Layer/Transport Layer Security (SSL/TLS) standards that have been defined for communication between Web browsers and servers
 - Wired Equivalent Privacy (WEP) protocol and the newer WiFi Protected Access (WPA) protocol that are part of the IEEE 802.11 wireless LAN standard



RC4: Initialization of S and T

- ❖ A variable-length key of from 1 to 256 bytes (8 to 2048 bits) is used to initialize a 256-byte state vector S
 - $S = S[0], S[1], \dots, S[255]$
 - At all times, S contains a **permutation** of all **8-bit numbers** from 0 through 255
- ❖ To begin, the entries of S are set equal to the values from 0 through 255 in **ascending order**;
 - that is, $S[0] = 0, S[1] = 1, \dots, S[255] = 255$
- ❖ A **temporary vector**, T, is also created from the key K

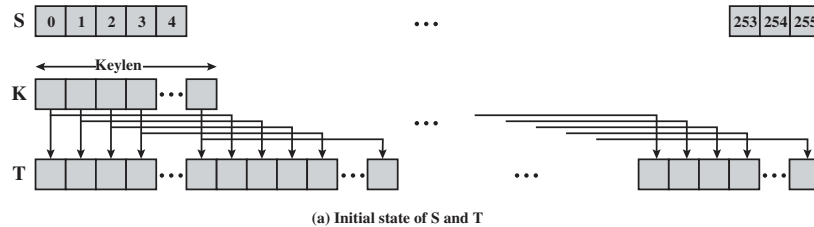
```
/* Initialization */
for i = 0 to 255 do
    S[i] = i;
    T[i] = K[i mod keylen];
```

What if the length of K = 256 ?

RC4: Key Schedule

- ❖ Next, we use T to produce the **initial permutation of S** .
- ❖ This involves starting with $S[0]$ and going through to $S[255]$, and for each $S[i]$, **swapping $S[i]$** with **another byte ($S[j]$) in S** according to a scheme **dictated by $T[i]$** :

```
/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Swap (S[i], S[j]);
```



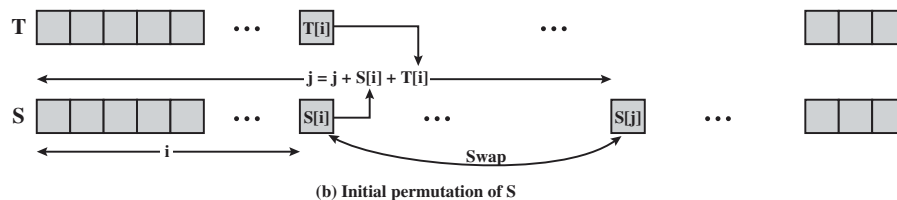
```
/* Initialization */
for i = 0 to 255 do
    S[i] = i;
    T[i] = K[i mod keylen];
```

RC4: Stream Generation

```
/* Stream Generation */
i, j = 0;
while (true)
    i = (i + 1) mod 256;
    j = (j + S[i]) mod 256;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 256;
    k = S[t];
```

- ❖ Stream generation involves cycling through **all the elements of $S[i]$**
- ❖ For each $S[i]$,
 - swapping $S[i]$ with another byte in S according to a scheme dictated by the current configuration of S
- ❖ After $S[255]$ is reached, the process continues, starting over again at $S[0]$

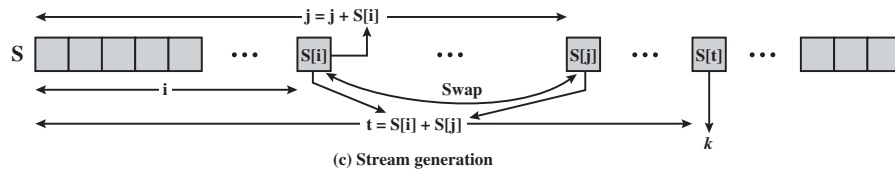
- To encrypt, **XOR the value k with the next byte of plaintext.**
- To decrypt, **XOR the value k with the next byte of ciphertext.**



```
/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Swap (S[i], S[j]);
```

Because the only operation on S is a swap, the only effect is a permutation. **S still contains all the numbers from 0 through 255.**

RC4: Stream Generation



```

/* Stream Generation */
i, j = 0;
while (true)
    i = (i + 1) mod 256;
    j = (j + S[i]) mod 256;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 256;
    k = S[t];

```

Strength of RC4

- ❖ A number of papers have been published analyzing methods of attacking RC4.
 - None of these approaches **is practical against RC4 with a reasonable key length, such as 128 bits**
- ❖ Recently, it is shown that the WEP protocol, intended to provide confidentiality on 802.11 wireless LAN networks, is vulnerable to a particular attack approach
 - the problem is **not with RC4 itself** but the way in which keys are generated for use as input to RC4