MONASH University
Information Technology

FIT3031 INFORMATION & NETWORK SECURITY

Lecture 2:

Symmetric Encryption Techniques

Unit Structure: Lecture Topics

- ✓ OSI security architecture
  - **common security standards and protocols for network security applications**
  - **common information risks and requirements**
- ✓ operation of private key encryption techniques
- operation of public encryption techniques
- concepts and techniques for digital signatures, authentication and non-repudiation
- security threats of web servers, and their possible countermeasures
- Wireless Security Issues
- security threats of email systems and their possible countermeasures
- IP security
- intrusion detection techniques for security purpose
- risk of malicious software, virus and worm threats, and countermeasures
- firewall deployment and configuration to enhance protection of information assets
- network management protocol for security purpose

Review of Last Lecture

Key points from the last lecture:

- **Security of Information stored or in transit during transmission is becoming increasingly important in internetworked environment**
- **Survey by UK Trade and Industry in 2012 revealed that 93% Large organizations & 76% small businesses suffered security breaches**
  - **This causes financial as well as productivity loss**
- **OSI security architecture provides a useful framework to systematically define the requirements and approaches to satisfy those requirements**
- **OSI Security Architecture focuses on three aspects of information security** :
  - **security attacks**
  - **security services**
  - **security mechanisms**
- **There are mainly two types of security attack**
  - **Passive attack – eavesdrop or monitoring, do not attempt to change data**
  - **Active attack – attempt to modify data or create false data**
- **X.800 defines security services into 6 major categories: Confidentiality, Authentication, Integrity, Non-repudiation, Access control and Availability**
- **Security mechanism are used to implement security services**

LN2 : Outline

- **Role of cryptography in achieving security**
- **Private key (symmetric) encryption principles**
- **Symmetric encryption algorithms**
- **Cipher block modes of operation**
- **Stream Ciphers**
- **Key Distribution**

Cryptography

- Derived from the Greek word *Crypto* which means *hidden*
- Cryptography refers to the study of mathematical techniques to achieve secure communication
- Very important to protect data in transit
  - communication channels and IP protocol are insecure
- Cryptography is a major technique behind information & network security and a means to achieve
  - authentication
  - data integrity
  - confidentiality
  - digital signature
  - privacy

Cryptographic Techniques

- **Main cryptographic techniques**
  - **Encryption**
    - Symmetric encryption
    - Asymmetric encryption
  - **Hash function**
- **Can also be used to protect data in file storage**

Symmetric - Asymmetric encryption names

Symmetric	Asymmetric
Shared-secret encryption	Public key encryption
Single-key encryption	Dual-Key encryption
Secret-key encryption	Dual-key pair
One key encryption	Public Key Cryptography
Private-key encryption	
Shared Key encryption	
conventional encryption	

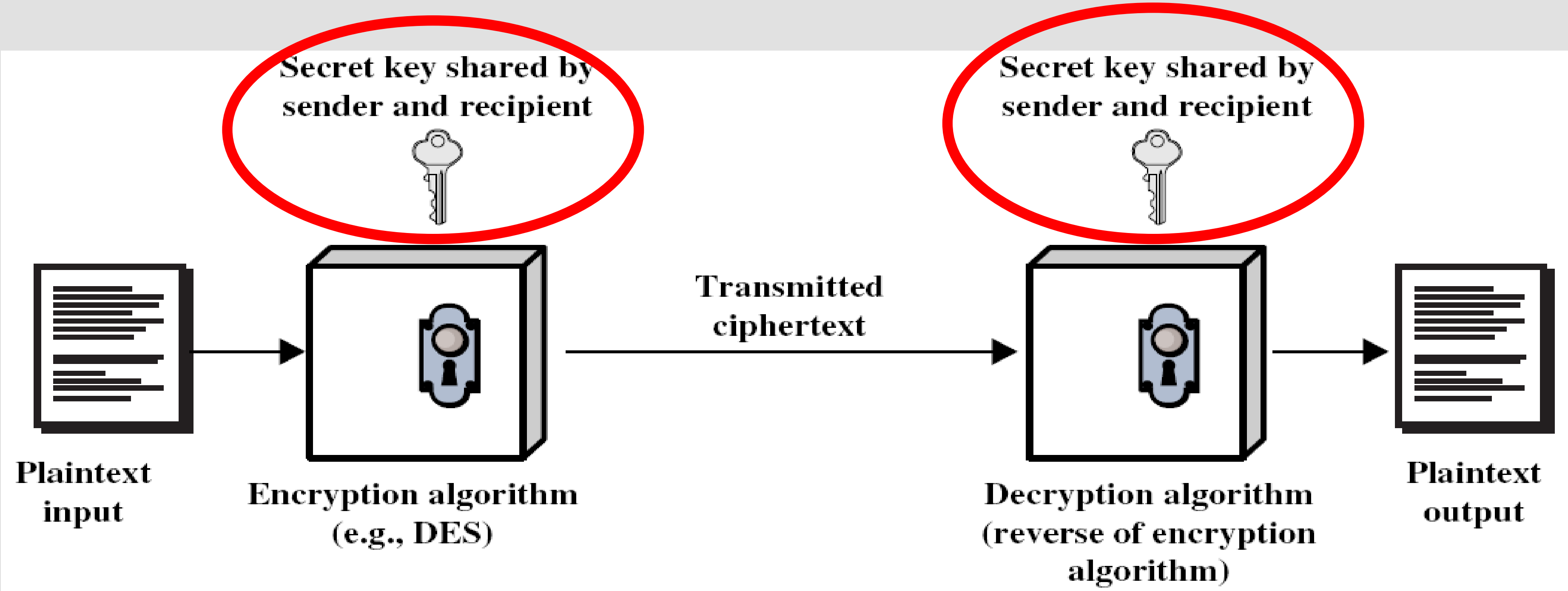
Encryption & Decryption

- ***Encryption*** **is the transformation of data into a form that is almost impossible to read without the *appropriate knowledge***
- ***Decryption*** **is the reverse of encryption - the transformation of encrypted data back into an intelligible form**
- **A *key* or *key pair* (appropriate knowledge) is necessary for encryption and decryption**
  - **A key is a digital object**

Symmetric Encryption

- **A symmetric encryption scheme has five components:**
  - **plaintext**
  - **encryption algorithm**
  - **a secret key**
  - **ciphertext**
  - **decryption algorithm**
- **sender and recipient share a common key**
- **Security depends on the secrecy of the key,**
- **not the secrecy of the algorithm**

Symmetric Encryption Model



A simplified model of symmetric encryption

MONASH University
Information Technology

Symmetric Encryption components

- **Five components:**
  - plaintext - the original message
  - ciphertext – the transformed (coded) message
  - encryption algorithm - algorithm that transforms plaintext to ciphertext
  - a secret key - information used by the algorithm and known only to sender/receiver
  - decryption algorithm – encryption algorithm in reverse; retrieve original message from coded message

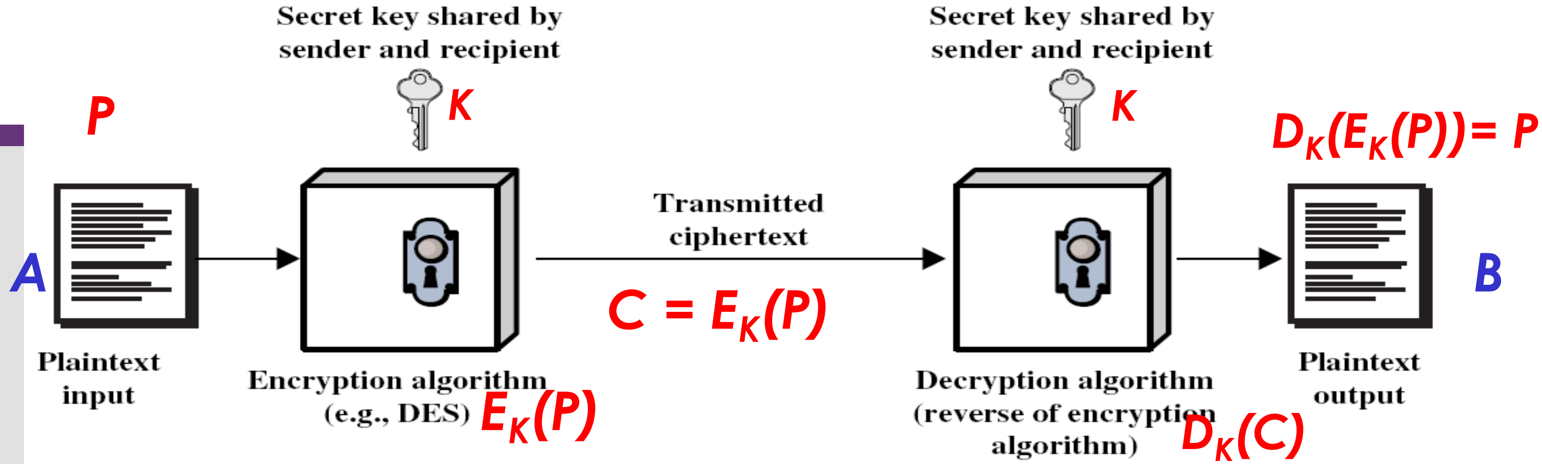
Encryption Example

- **Original message:**
  - This is a demonstration of symmetric encryption
- **Encryption key (128 bit):**
  - Qwopryoo58gj10pe
- **Encrypted message:**

  #£¤âÆ.zþ¸T(K»ùç¼LN:ÑÅQÔ\aÔïÏîL#P4ã

- **Decrypted message**
  - This is a demonstration of symmetric encryption
- **You may try these simple demos, linked to day-2 section**
- **http://www.tools4noobs.com/online_tools/ecrypt/**
- **http://www.tools4noobs.com/online_tools/decrypt/**
- **http://www.tools4noobs.com/online_tools/hash/**

Symmetric Cryptography - Principle



- A shares a secret key ***K*** with **B**.
  - The key may be distributed by a key distribution center (KDC).
- A encrypts a plaintext message ***P*** by applying an ***encryption function E*** and the key ***K*** to create a ciphertext
  - $C = E_K(P)$
- A sends the ciphertext ***C*** to **B** over a communication channel. Even if anybody other than **B** gets hold of a copy of ***C***, he can't decrypt the message unless he has ***K***.
- If the key distribution is perfect or **B** does not accidentally or intentionally disclose it, then nobody other than **B** is supposed to have the key ***K***.
- On receiving ***C***, **B** applies ***decryption function D*** and key ***K*** on ***C*** to recover the plaintext message *P*:
  - $D_K(C) = D_K(E_K(P)) = P$

Cipher

- **Two types:**
  - **Block cipher:**
    - process one input block at a time
    - produce one output block for each input block
  - **Stream cipher:**
    - process input element continuously
    - produces one element at a time

Cryptanalysis

- **An attacker may try different types of attacks on the encrypted message to discover plaintext or secret**
- **A brute-force attacker tries every possible key**
- **On an average a successful break would require half the number of all possible keys tried**
- **It is possible to try 1 million keys per μsec.**
  - requires only ten hours to break a 56-bit symmetric encryption
- **Higher length key does not necessarily make an encryption more secure**
  - also depends on the encryption algorithm

Cryptanalytic Attacks

- **ciphertext only**
  - only know algorithm & ciphertext, is statistical, know or can identify plaintext
- **known plaintext**
  - know/suspect algorithm, one or more plaintext – ciphertext pairs
- **chosen plaintext**
  - Algorithm, select plaintext and obtain ciphertext
- **chosen ciphertext**
  - Algorithm, select ciphertext and obtain plaintext
- **chosen text**
  - Algorithm, select plaintext or ciphertext to encrypt/decrypt

Exhaustive Key Search / Brute Force Attack

- **Always possible to simply try every key**
- **Most basic attack, proportional to key size**
- **Assume either know / recognize plaintext**

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

MONASH University
Information Technology

Feistel Cipher

- **Virtually all *conventional block encryption* algorithms, including DES (Data Encryption Standard) have a structure first described by Horst Feistel of IBM in 1973**
- **The realization of a Feistel Network depends on the choice of the parameters used and design features**

Feistel Cipher Structure

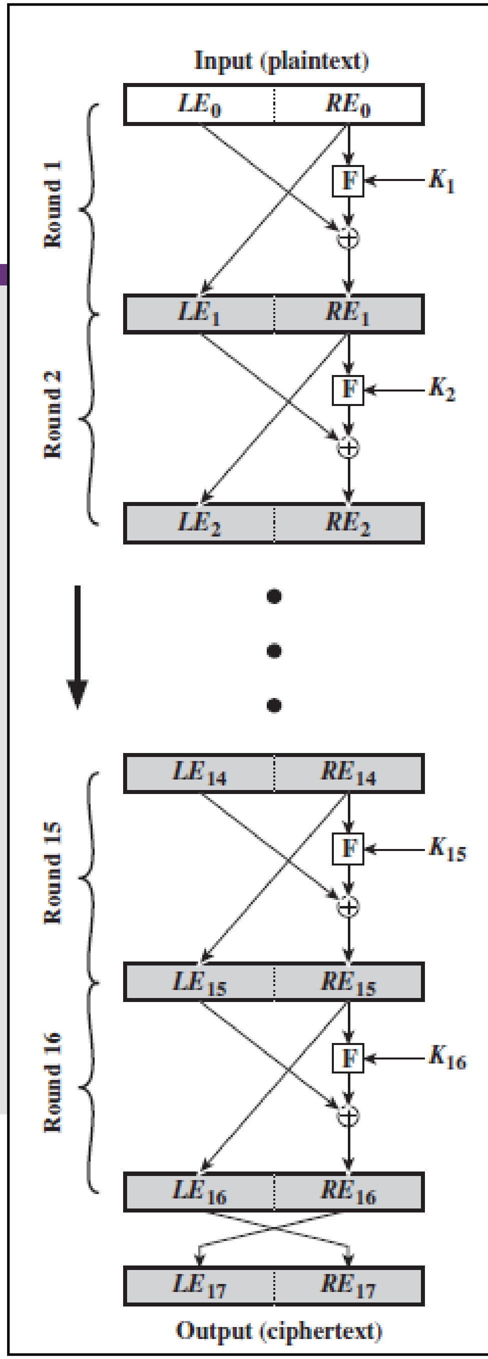
- **Fiestel Cipher operation (see figure in next slide):**
  - Split the plaintext block into two equal pieces, (L0, R0)
  - ***n*** rounds of processing for each half before combining to produce ciphertext
  - For each round $i = 1,2,\ldots,n$, compute

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

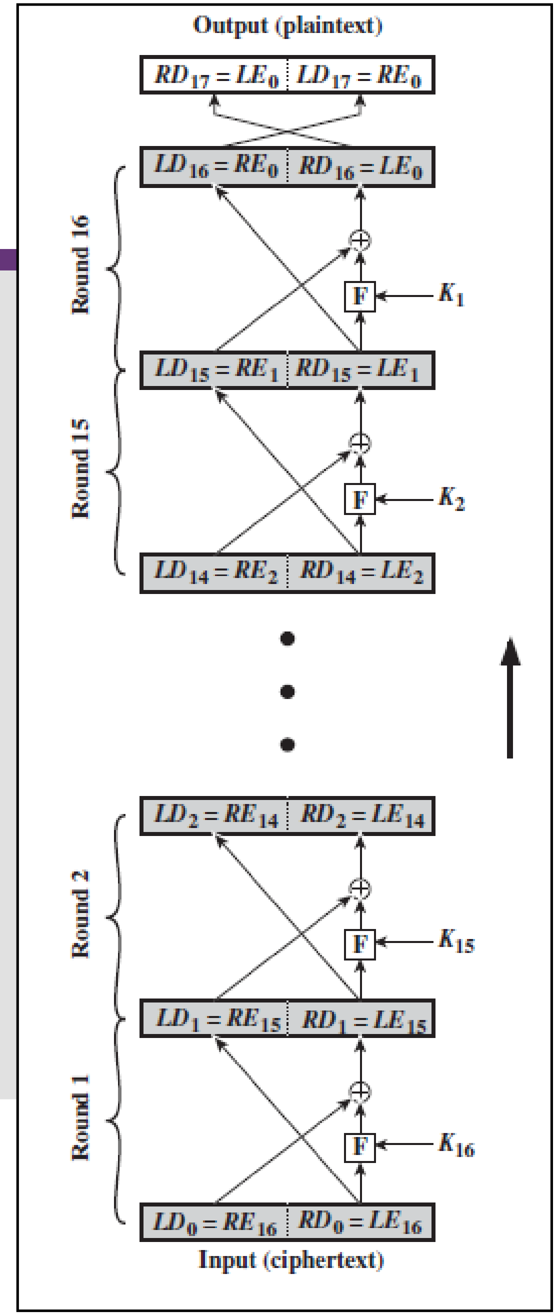
  - where ***f*** is the round function and K_i is the sub-key
  - After $\boldsymbol{n}^{th}$ round the ciphertext is (L_n, R_n).
  - Regardless of the function ***f***, decryption is accomplished via

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

Classical Feistel Network



Feistel Encryption (16 rounds)



Feistel Decryption (16 rounds)

Feistel Cipher Parameters

- **Fiestel Cipher depends on:**
  - **Block size:** larger block sizes mean greater security
  - **Key Size:** larger key size means greater security
  - **Number of rounds:** multiple rounds offer increasing security, typically 16 rounds
  - **Subkey generation algorithm:** greater complexity will lead to greater difficulty of cryptanalysis.
  - **Round function:** greater complexity means greater resistance

Symmetric Encryption Algorithms

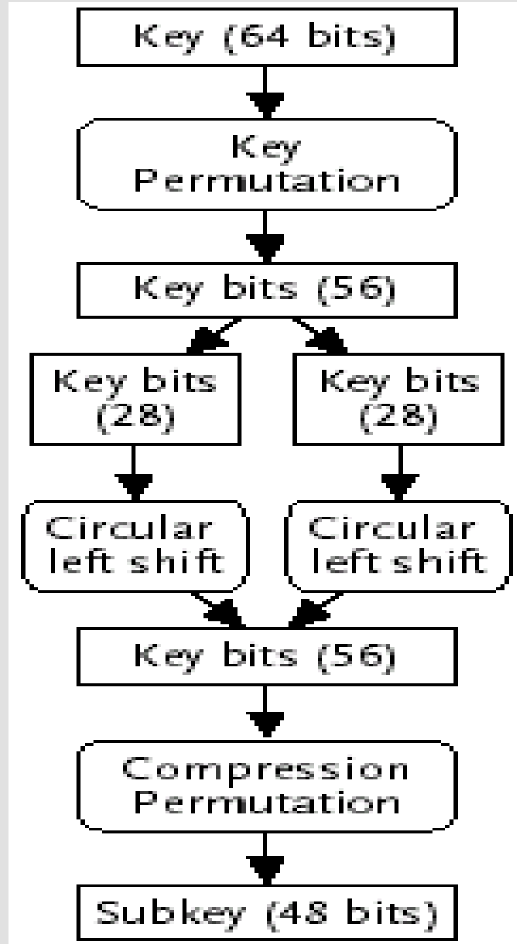
- **Data Encryption Standard (DES)**
  - The most widely used encryption scheme
  - The algorithm is referred to as the Data Encryption Algorithm (DEA)
  - DES is a block cipher
    - processed in 64-bit blocks
    - 56-bits key
      - 8 parity bits are stripped off from the full 64-bit key (8 character)
    - 16 subkeys created for 16 rounds
  - **Concern**: Proved insecure in today's fast processing power

Substitution & Permutation

- **Substitution**
  - a binary word is replaced by some other binary word
  - also known as **S-box**
  - impractical to build 64-bit blocks
    - multiple S-boxes of smaller blocks are used
  - Example: for an input ’011001’ to an S-box, the output may be ’1001’
- **Permutation:**
  - A binary word has its bits reordered (permute)
  - Also known as **P-box**
  - Example: 1^{st} bit may become 7^{th} bit, 2^{nd} bit 12^{th} bit and so on

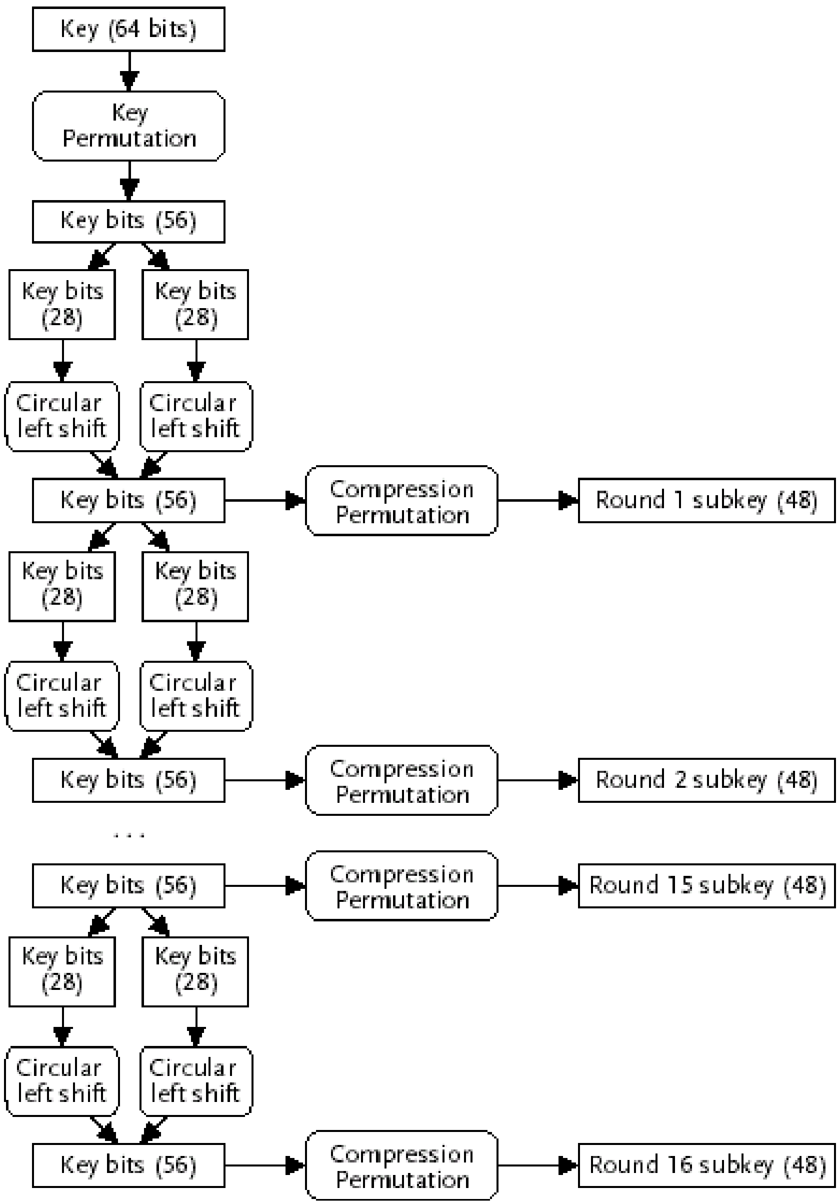
DES Subkey Generation - round # 1

- **drop 8 parity bits**
  - effective key size 56 bits
- **permute the bits and divide into two 28-bits**
- **rotate the bits left by single bit**
- **permute and extract 48 bits as a subkey**



MONASH University
Information Technology

DES Subkey Generation

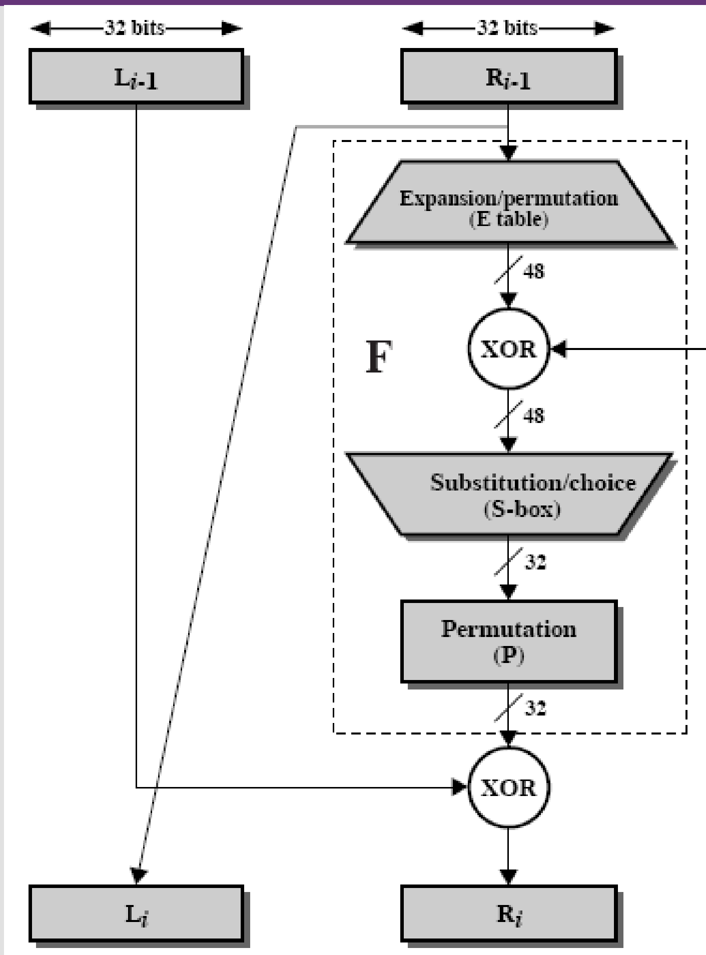


- **One bit shift – for round 1,2,9 and 16**
- **Two bit shift for the remaining rounds 3,4,5,6,7,8,10, …..15**

DES Round

- **Each of the sixteen rounds takes a 64-bit block as input and produces a 64-bit block as output**
- **The output from the initial permutation is the input to round one**
- **Round one's output is the input to round two**
- **Round two's output is the input to round three**
- **…**
- **The output from round sixteen is the 64-bit block of ciphertext**

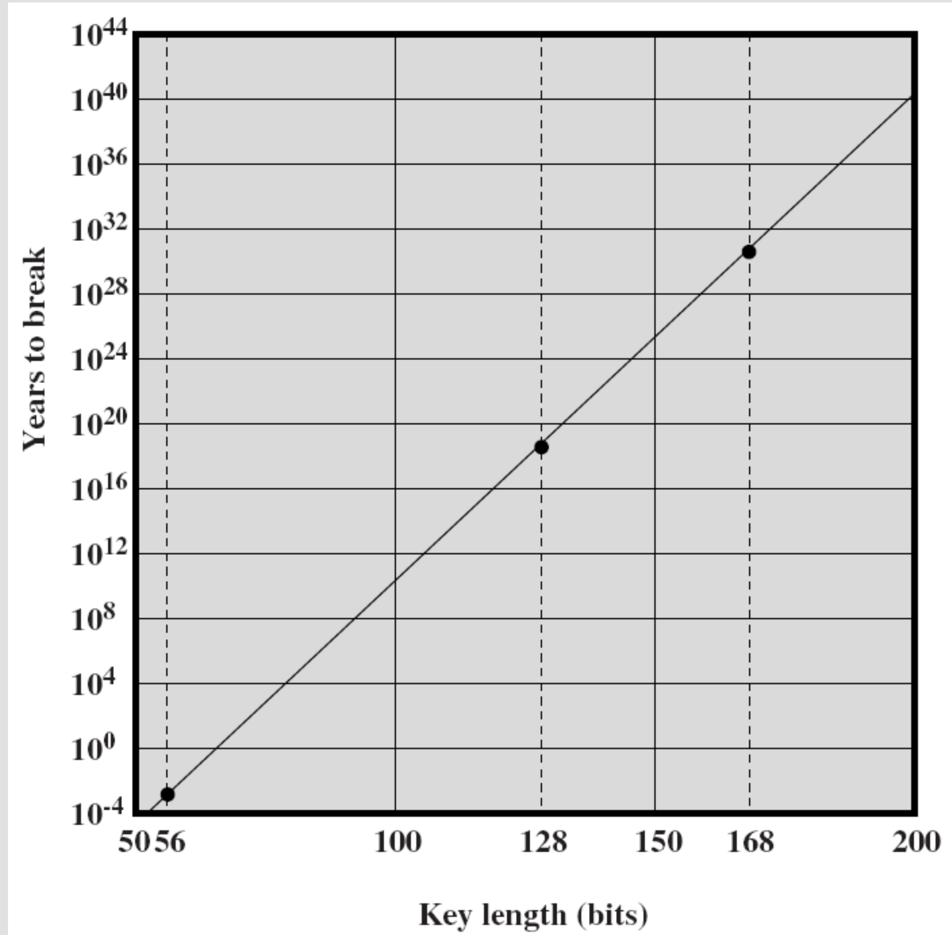
Single DES Round



- Similar to Fiestel Cipher structure
- 64-bit plaintext is divided into two 32-bit blocks (L &R)
- L_i is the unchanged R_{i-1} (previous round)
- R_{i-1} goes through F function
  - E table-expanded to 48bits and permuted
  - 48 bits XORed with subkey K_i
  - Substitution produces 32-bit
    - 8 S-boxes
    - each takes 6 bits and produces 4 bits
    - transformation is defined by substitution tables
    - different substitution table for each S-box
  - Permutes the output of S-box
- R_i is L_{i-1} XORed with permuted output

Encryption Algorithms (DES…):Time to break a code

Time to break a code (10^6 decryptions/μs)



Encryption Algorithms: Triple DES

- **Apply DES algorithm three times**
- **Use three keys and three executions of the DES algorithm (encrypt-decrypt-encrypt)**

$$C = E_{K3}[\ D_{K2}[\ E_{K1}[P]\ ]\ ]$$

C = ciphertext

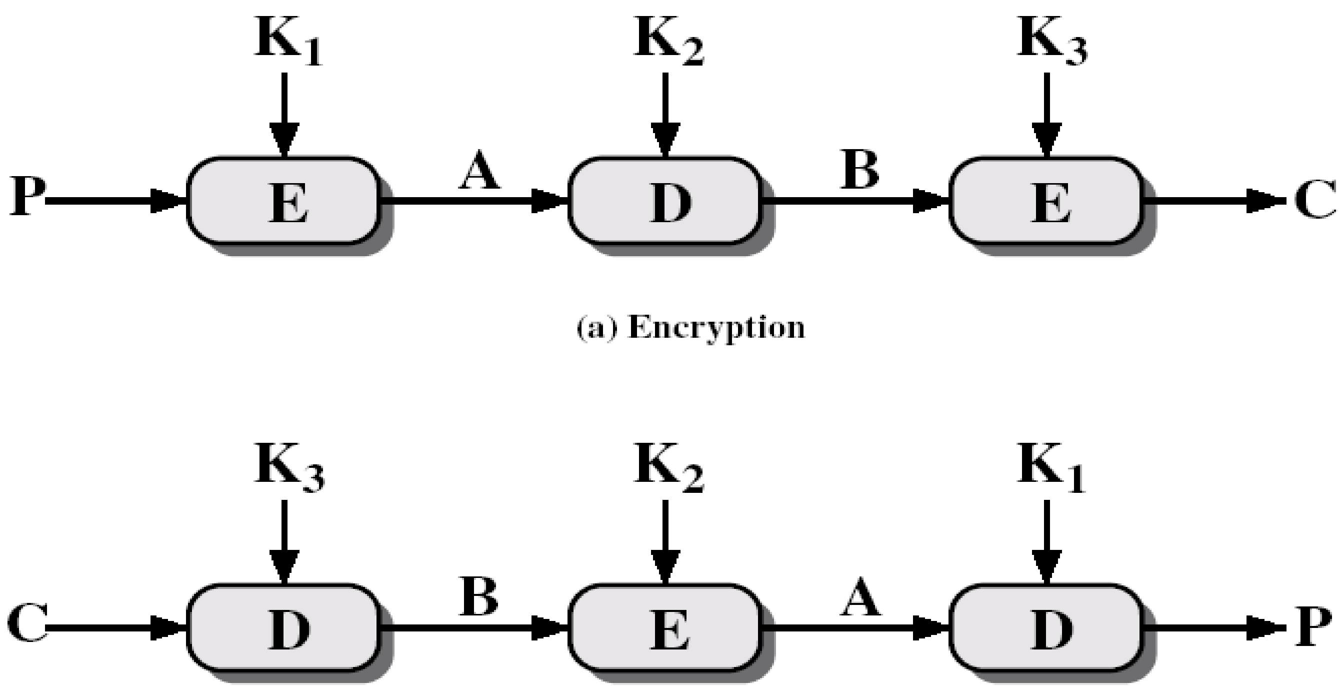
P = Plaintext

$E_K[X]$ = encryption of X using key K

$D_K[Y]$ = decryption of Y using key K

- **Effective key length of 168 bits**
- **Decryption: same operation with keys reversed**

Encryption Algorithms: Triple DES…



(a) Encryption

(b) Decryption

Other Encryption Algorithms

- International Data Encryption Algorithm (IDEA)
  - Was developed to replace DES
  - 128-bit key, 64-bit block cipher
  - different round function and subkey generation from DES
  - Used in PGP
- Blowfish
  - DES-like algorithm: 64-bit cipher, 16 rounds
  - variable key length upto 448: 128-bit common
  - easy to implement, high execution speed, low memory requirement

Other Encryption Algorithms…

- **RC5**
  - suitable for hardware and software
  - Fast and simple
  - adaptable to processors of different word lengths
  - variable number of rounds
  - variable-length key
  - low memory requirement
  - high security
  - data-dependent rotations

Other Encryption Algorithms…

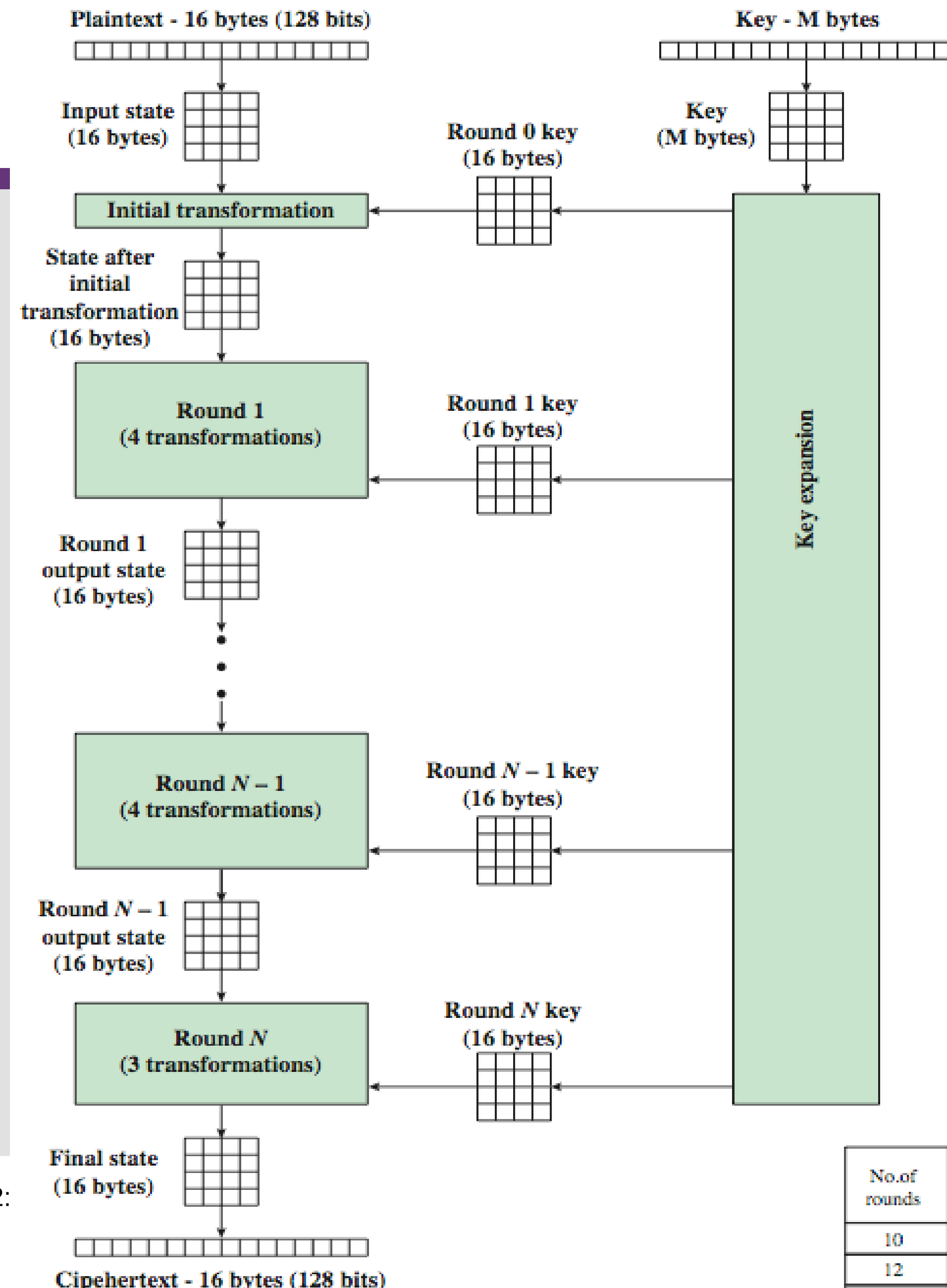
- **Triple DES enjoyed widespread use for few years, but has severe drawbacks:**
  - sluggish in software
  - use 64-bit block size. A larger length is desirable for efficiency and security
- **In 1997, US National Institute of Standards and Technology (NIST) officially endorsed Advanced Encryption Standard (AES) to replace DES based on:**
  - security, computational efficiency, memory requirements, hardware and software suitability, flexibility

Encryption Algorithms (AES)

- **Developed by Joan Daemen and Vincent Rijmen from Belgium – also known as Rijndael algorithm**
- **The algorithm uses**
  - 128-bit block cipher with three different key length: 128, 192 and 256 bits & support 10 to 14 rounds
  - Design simplicity / flexibility / efficient / fast for both implementations in software and hardware, and code compactness on many CPUs
  - It works fast even on small devices such as smart phones, smart cards etc.
  - AES provides more security due to larger block size and longer keys.
  - AES uses 128 bit fixed block size and works with 128, 192 and 256 bit keys.
  - offers lots of flexibility: AES in general is flexible enough to work with key and block size of any multiple of 32 bit with minimum of128 bits and maximum of 256 bits.
  - Security: Resistance to power analysis and other implementation attacks

AES Encryption Process

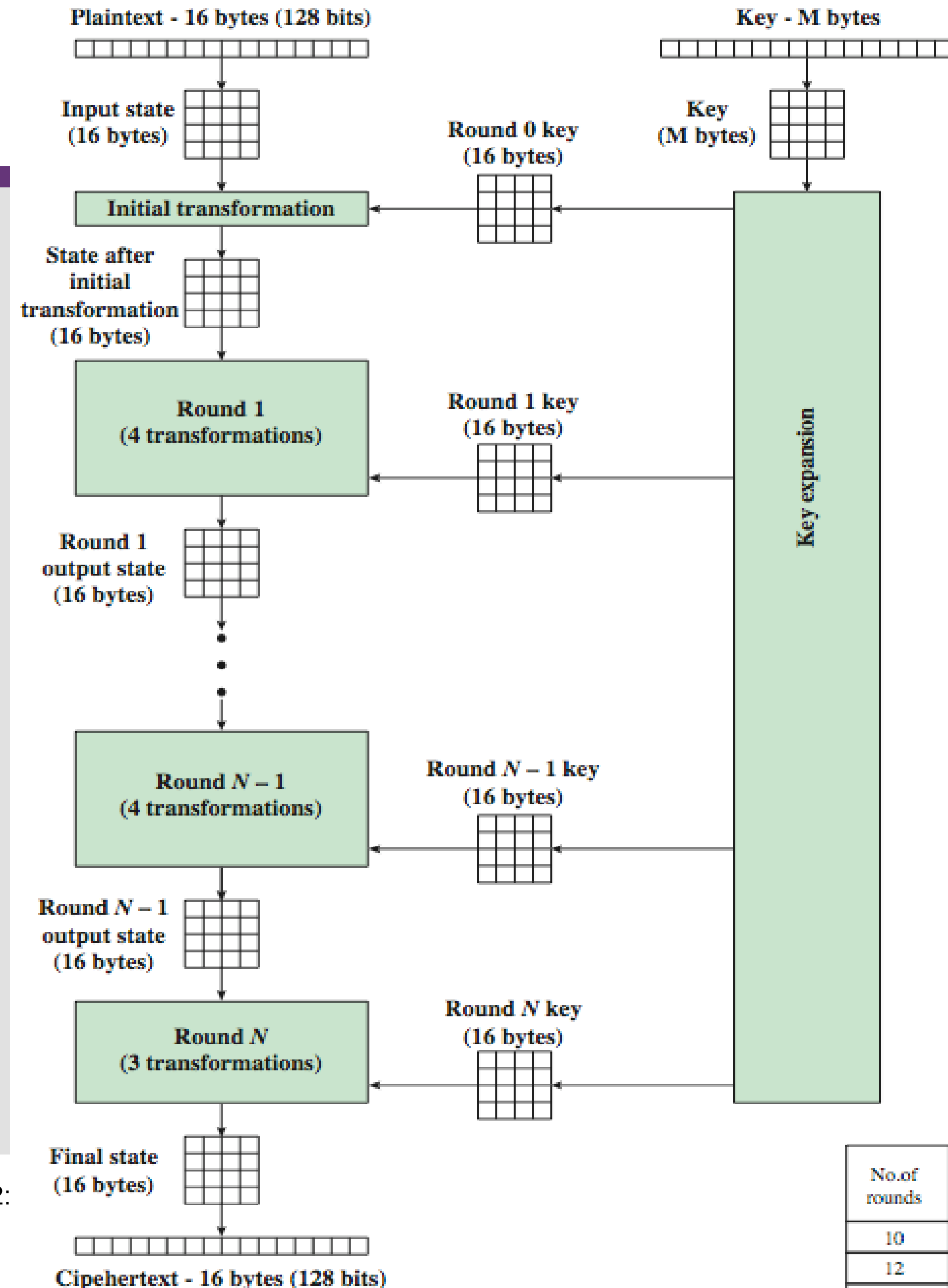
- The cipher consists of N rounds, where the number of rounds depends on the key length: 10 rounds for a 16-byte key; 12 rounds for a 24-byte key; and 14 rounds for a 32-byte key.
- The first N – 1 rounds consist of four distinct transformation functions: SubBytes, ShiftRows, MixColumns, and AddRoundKey
- The final round N contains only 3 transformation functions: SubBytes, ShiftRows, and AddRoundKey
- Each transformation takes one or more 4 x 4 matrices as input and produces a 4 x 4 matrix as output.
- key expansion function generates N + 1 round keys, each of which is a distinct 4 x 4 matrix



No.of rounds	Key Length (bytes)
10	16
12	24
14	32

AES Encryption Process

- The cipher consists of N rounds, where the number of rounds depends on the key length: <u>10 rounds for a 16-byte key; 12 rounds for a 24-byte key; and 14 rounds for a 32-byte key</u>.
- The first N – 1 rounds consist of four distinct transformation functions: SubBytes, ShiftRows, MixColumns, and AddRoundKey
- The final round N contains only 3 transformation functions: SubBytes, ShiftRows, and AddRoundKey
- Each transformation takes one or more 4 x 4 matrices as input and produces a 4 x 4 matrix as output.
- key expansion function generates N + 1 round keys, each of which is a distinct 4 x 4 matrix



No.of rounds	Key Length (bytes)
10	16
12	24
14	32

AES Structure

- **data block of 4 columns of 4 bytes is state**
- **key is expanded to array of words (44/52/60)**
- **Has 9/11/13 identical rounds in which state undergoes:**
  - **byte substitution** (1 S-box used on every byte)
  - **shift rows** (simple permutation)
    - > 1st row-no_shift, 2nd row -1Byte left_circular_shift, 3rd row - 2Byte, 4th row – 3Byte))
  - **mix columns** (subs using matrix multiply of groups)
  - **add round key** (XOR state with the expanded key words)
  - **view the whole operation as alternating** XOR key & scramble data bytes
- **initial XOR key material & incomplete last round (10th / 12th / 14th)**
- **with fast XOR & table lookup implementation**

AES Encryption / Decryption Process

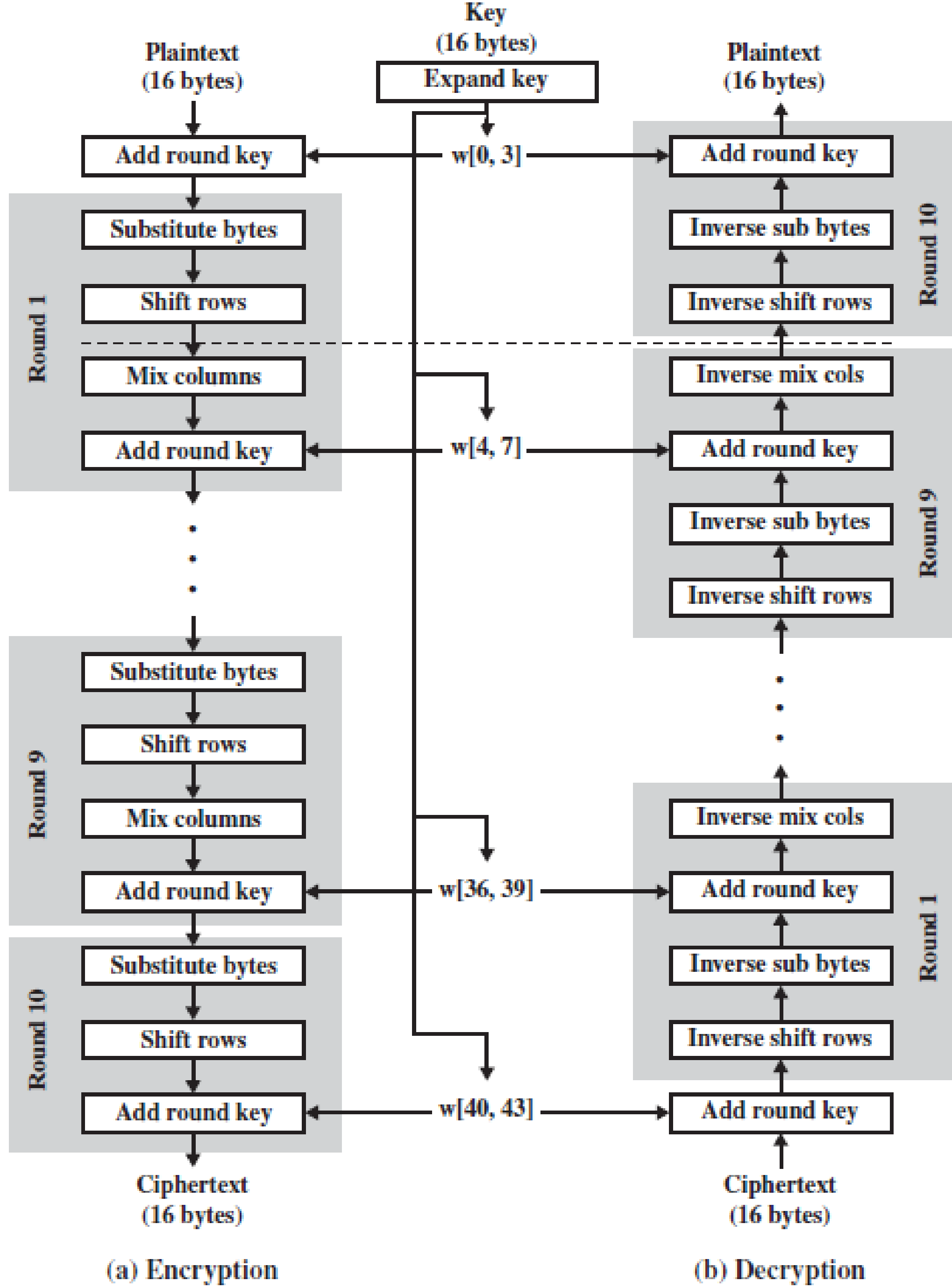
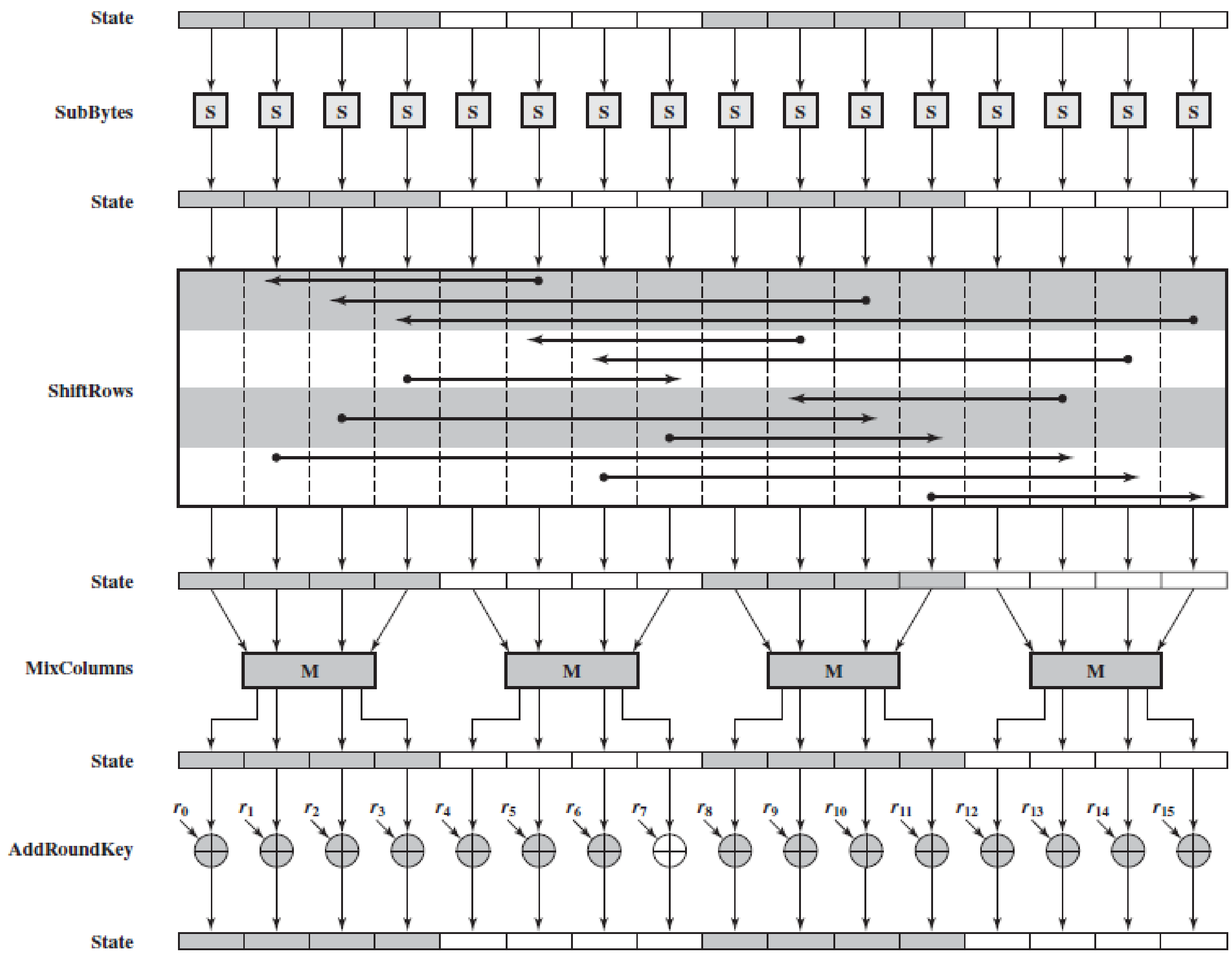


Figure 2.5 AES Encryption and Decryption

AES single Round



MONASH University
Information Technology

Modes of Operation

- **block ciphers encrypt fixed size blocks**
  - e.g. DES encrypts 64-bit blocks with 56-bit key
- **need some way to en/decrypt arbitrary amounts of data in practice**
- **NIST SP 800-38A defines 5 modes of operations**
- **have block and stream modes**
- **to cover a wide variety of applications**
- **can be used with any block cipher**

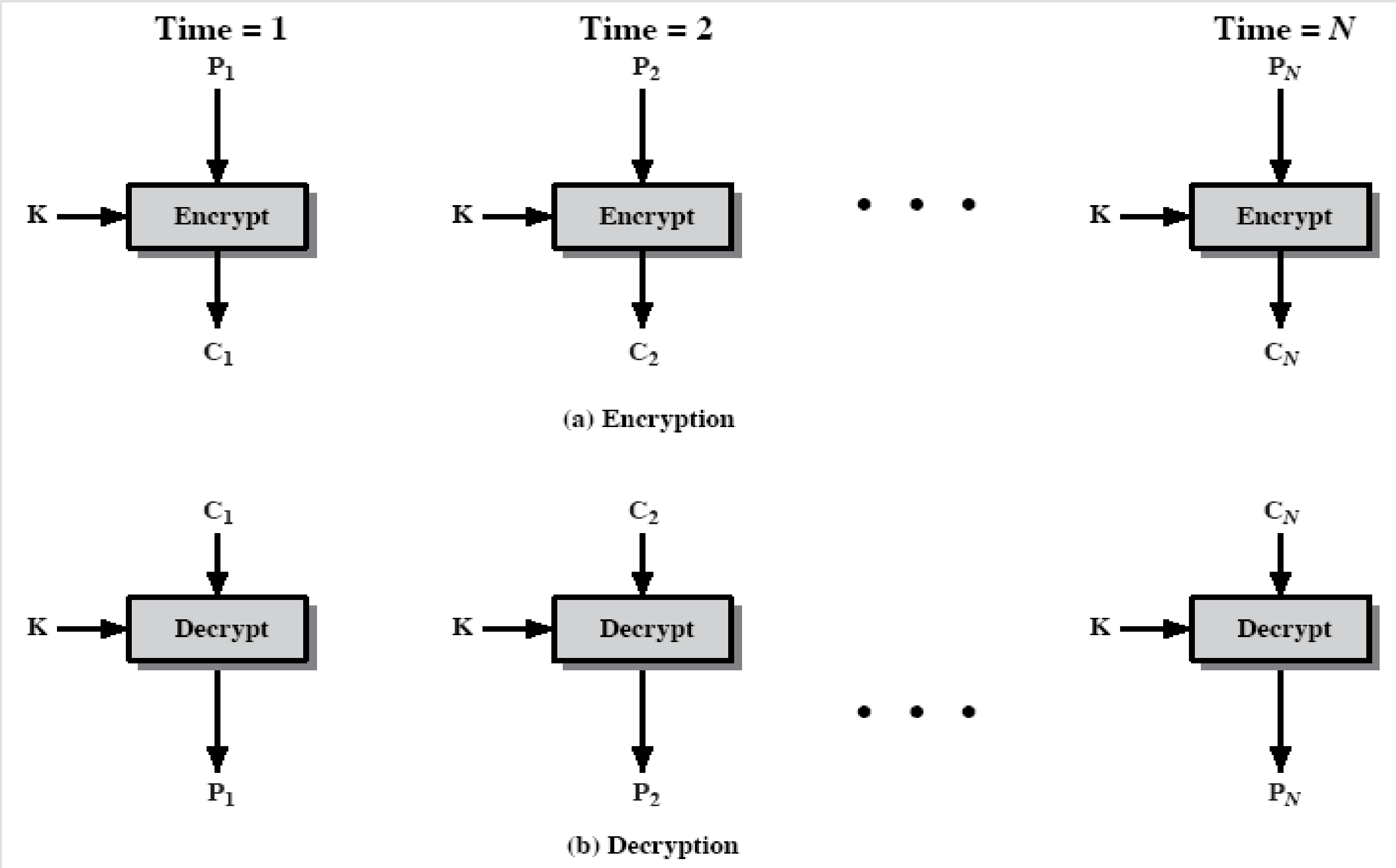
Cipher Block Modes of Operations

- **Five common modes in which plaintext, secret key and / or ciphertext are combined to encrypt/decrypt arbitrary size blocks**
  - Electronic Codebook (ECB) mode
  - Cipher Block Chaining (CBC) mode
  - Cipher Feedback (CFB) mode
  - Output Feedback (OFB) mode
  - Counter (CTR) mode

ECB (Electronic Codebook) mode

- **Simplest mode of operation**
- **Each block (64bits) is processed at a time**
  - input data is padded out to an integer number of block size
  - each block is encrypted and decrypted independently
  - lost data blocks do not affect decryption of other blocks
  - error is not propagated, limited to single block
  - All blocks are encrypted with the same key
- **Concern:**
  - same plaintext produces same ciphertext
    - does not hide pattern
    - if message contains repetitive elements, these elements can be identified
    - traffic analysis is possible

ECB mode…



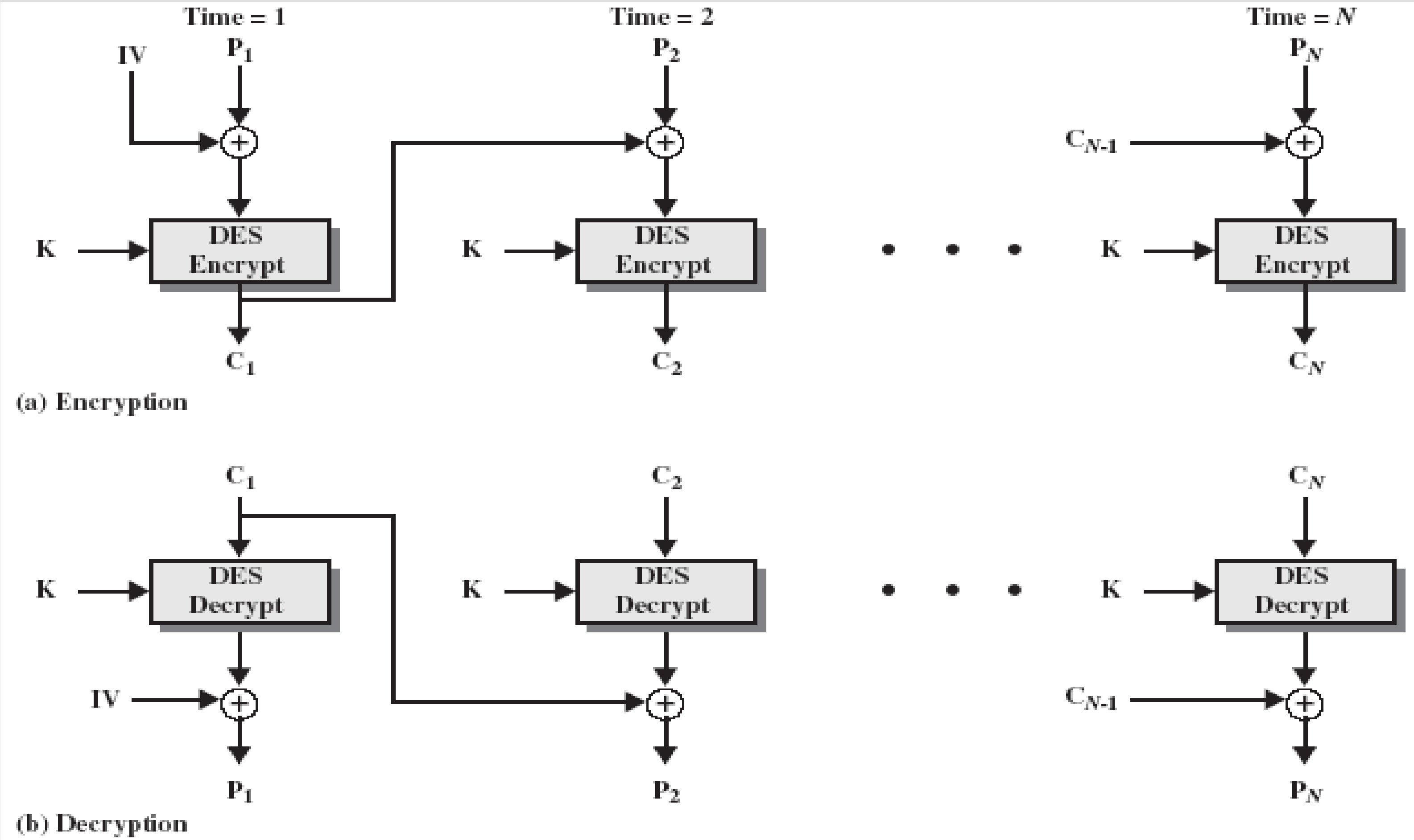
MONASH University
Information Technology

CBC (Cipher Block Chaining) mode

- **Most commonly used mode**
- **Each plaintext block is XORed with the previous ciphertext block and encrypted (see next slide)**
  - first plaintext block is XORed with an initialization vector (IV)
  - same encryption key is used for each block
- **More secure**
  - effectively scrambles plaintext prior to each encryption step
  - repetitive patterns are not exposed
  - different IV for different messages with the same key

- **Disadvantage:**
  - **encryption of a data block becomes dependent on all the blocks prior to it**
  - **a lost block of data will prevent decoding of the next block of data**

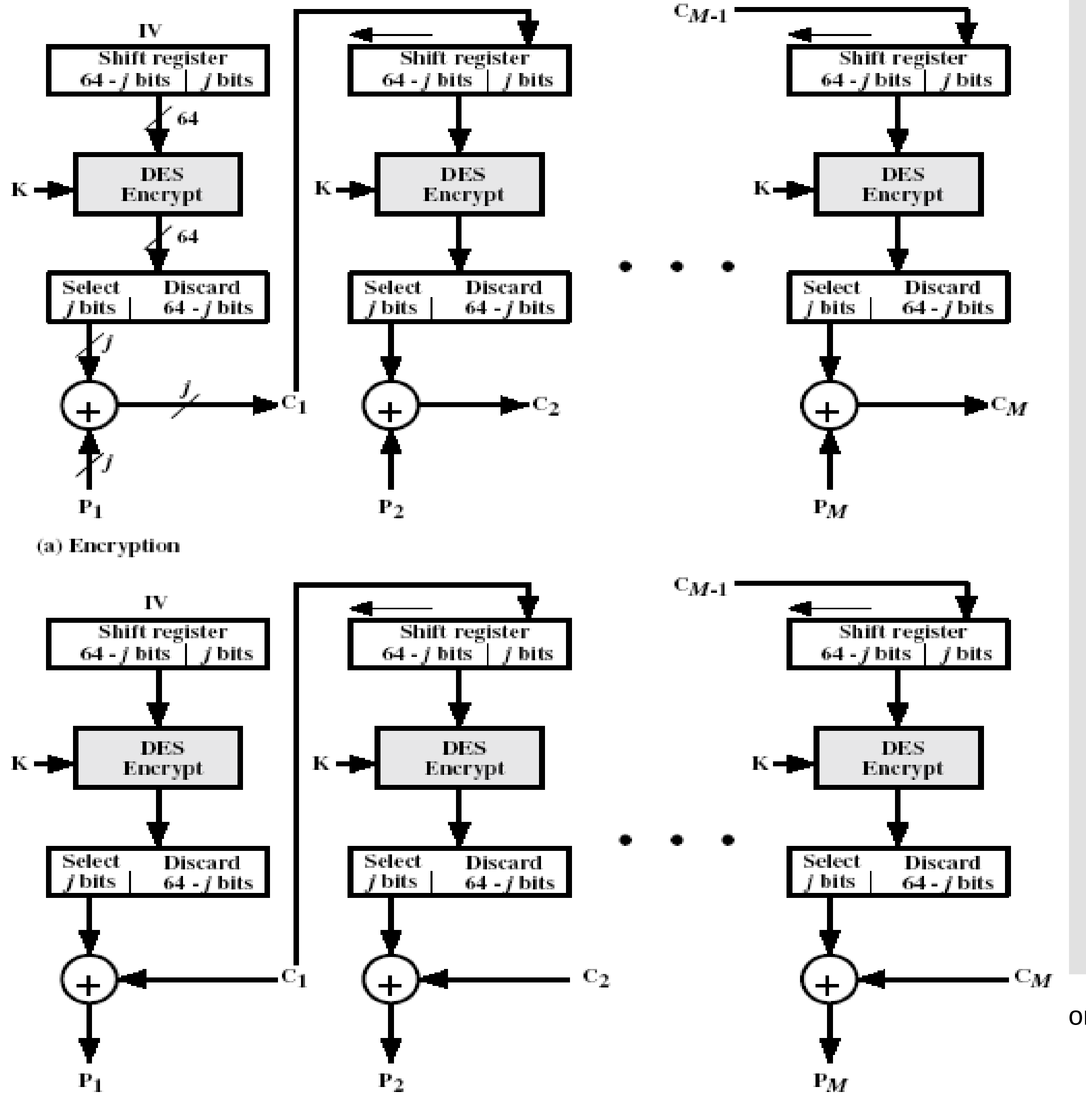
CBC (Cipher Block Chaining) mode



CFB (Cipher feedback) mode

- **Message is treated as a stream of bits**
- **Stream cipher does not require any padding**
- **Added to the output of the block cipher**
- **Result is fed-back for next stage (hence the name)**
- **standard allows any number of bits (1,8 or 64 or whatever) to be fed-back**
  - denoted by CFB-1, CFB-8, CFB-64 etc.
- **It is most efficient to use all 64 bits (CFB-64)**
  - $C_i = P_i \text{ XOR } DES_{K1}(C_{i-1})$
  - $C_{-1} = IV$
- **uses: stream data encryption, authentication**

CFB (Cipher feedback) mode



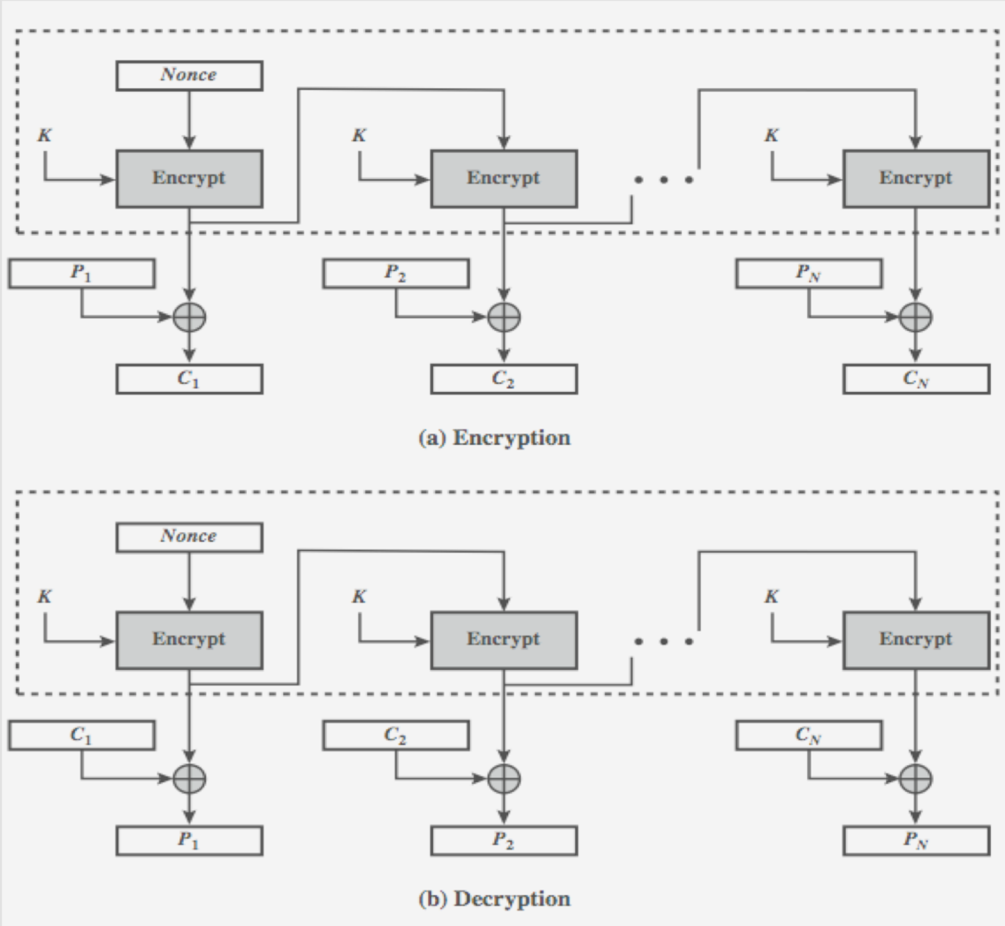
(a) Encryption

(b) Decryption

OFB (Output feedback) mode

- **Message is treated as a stream of bits**
- **Similar to CFB, but**
  - quantity XORed with each plaintext block is generated independently of both the plaintext and ciphertext
  - output of cipher block is added to message
- **Feedback is independent of message**
- **uses: stream encryption over noisy channels**

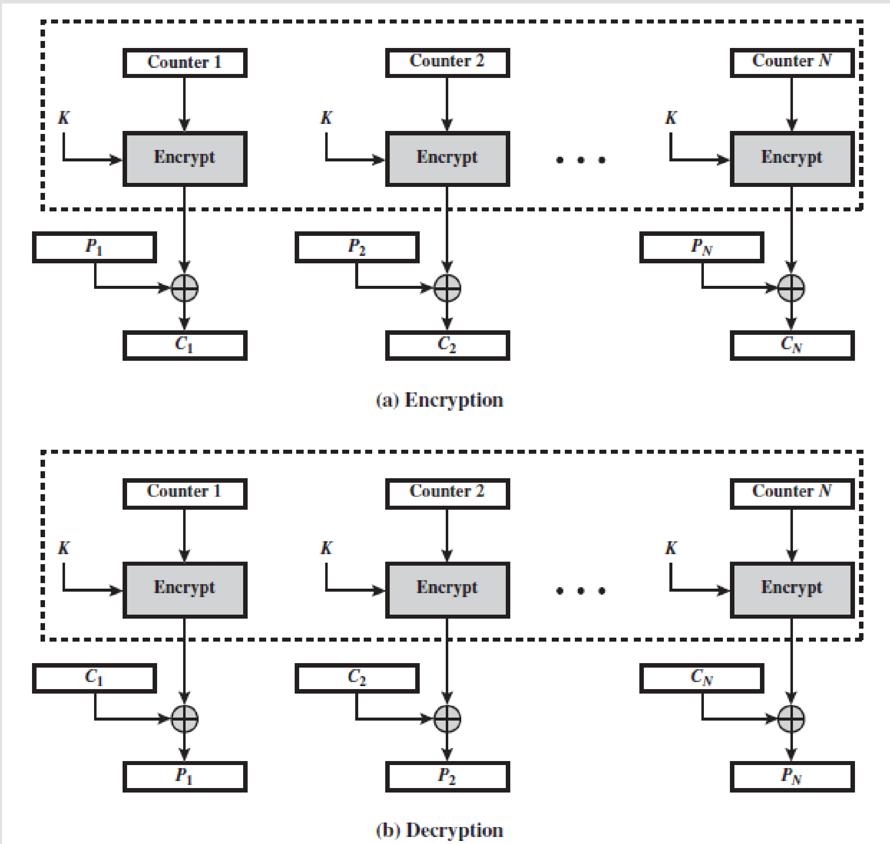
OFB (Output feedback) mode

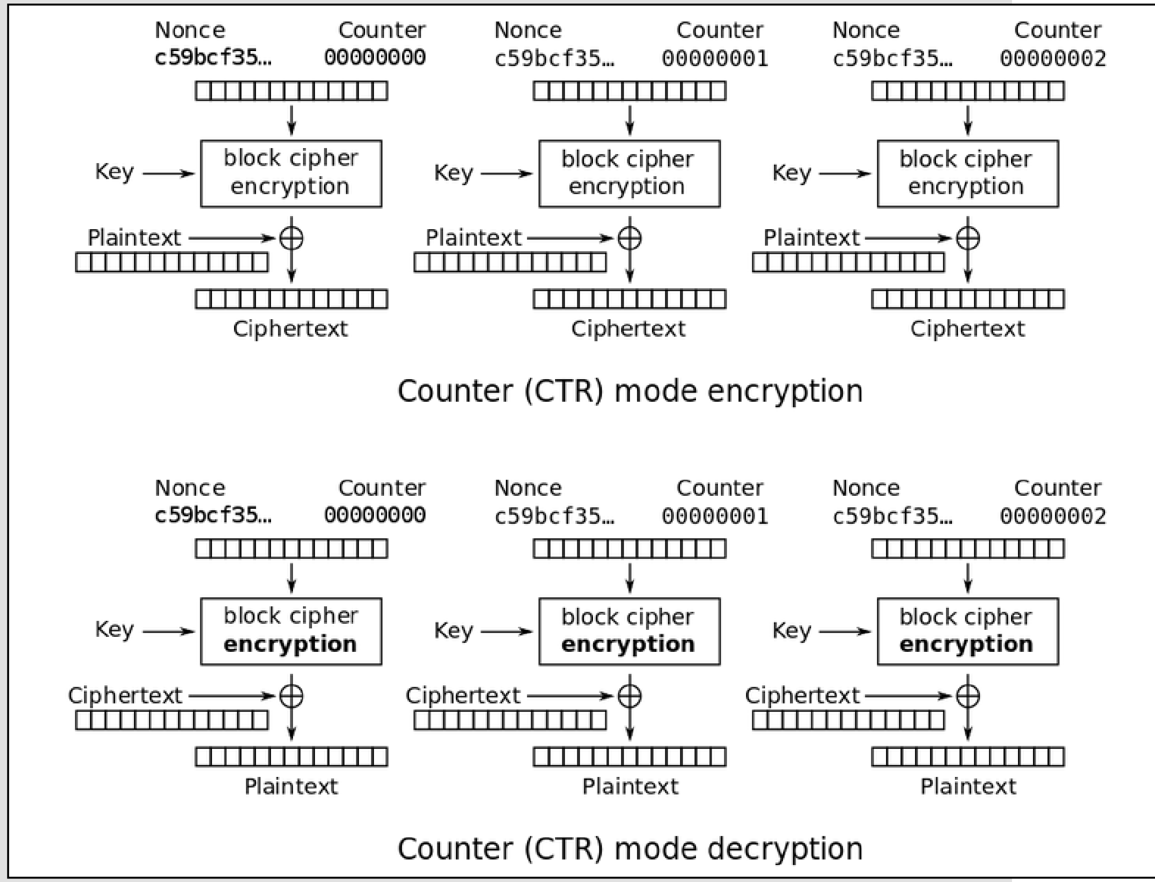


Counter (CTR)

- **a "new" mode, though proposed early on**
- **similar to OFB but encrypts using counter value rather than any feedback value**
- **must have a different counter value for every plaintext block (never reused)**
  - $O_i = E_K(i)$
  - $C_i = P_i \text{ XOR } O_i$
- **uses: high-speed network encryptions**

Counter (CTR)





MONASH University
Information Technology

Advantages and Limitations of CTR

- **efficiency**
  - can do parallel encryptions in h/w or s/w
  - can pre-process in advance of need
  - good for bursty high speed links
- **random access to encrypted data blocks**
- **provable security (good as other modes)**
- **but must ensure never reuse counter values, otherwise could break (cf OFB)**

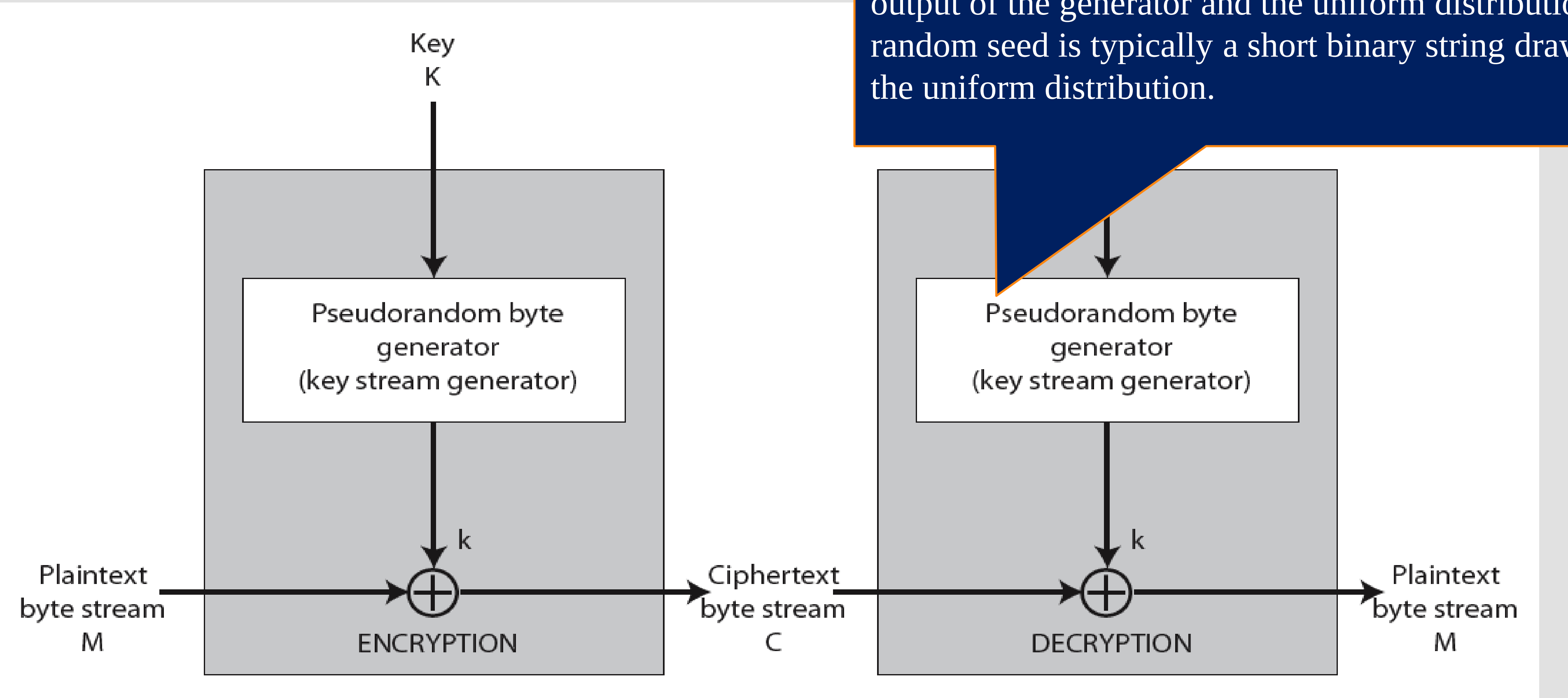
Stream Ciphers

- **process message bit by bit (as a stream)**
- **have a pseudo random** keystream
- **combined (XOR) with plaintext bit by bit**
- **randomness of** stream key **completely destroys statistical properties in message**
  - $C_i = M_i$ XOR $StreamKey_i$
- **but must never reuse stream key**
  - otherwise can recover messages (book cipher)

MONASH University
Information Technology

Stream Cipher Structure

A pseudorandom generator (PRG) is a class of statistical tests is a deterministic procedure that maps a random seed(Key K) to a longer pseudorandom string such that no statistical test in the class can distinguish between the output of the generator and the uniform distribution. The random seed is typically a short binary string drawn from the uniform distribution.



Stream Cipher Properties

- **some design considerations are:**
  - long period with no repetitions
  - statistically random
  - depends on large enough key
  - large linear complexity
- **properly designed, can be as secure as a block cipher with same size key**
- **but usually simpler & faster**
- **Example RC4**

RC4

- **a proprietary cipher owned by RSA DI Management**
- **another Ron Rivest design, simple but effective**
- **variable key size, byte-oriented stream cipher**
- **widely used (web SSL/TLS, wireless WEP)**
- **key forms random permutation of all 8-bit values**
- **uses that permutation to scramble input info processed a byte at a time**

RC4 Key Schedule

- **starts with an array S of numbers: 0..255**
- **use key to well and truly shuffle**
- **S forms internal state of the cipher**

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

```
/* 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]);
```

- RC4 has a programmable secret key of length from 1 to 256 bytes and a state table of 256 bytes
- secret key is used to initialize the state table
- state table is used to generate a pseudo random sequence of bytes and then a pseudo random stream of bits.

RC4 Encryption

- **Encryption: continues shuffling array values**
- **sum of shuffled pair selects "stream key" value from permutation**
- **XOR S[t] with next byte of message to en/decrypt**

```
/* 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];
```

Ci = Mi XOR S[t]

- such bit stream is XORed with the plaintext to produce the ciphertext
- every element (byte) in the state table is swapped at least once

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

MONASH University
Information Technology

RC4 Security

- **claimed secure against known attacks**
  - have some analyses, none practical
- **result is very non-linear**
- **since RC4 is a stream cipher, must** never reuse a key
- **have a concern with WEP, but due to key handling rather than RC4 itself**

Key Distribution

- **Key distribution is an important issue**
  - both parties must exchange the secret key in a secure manner
- **Key exchange**
  - a key could be selected by **A** and physically delivered to **B**
  - a third party could select the key and physically deliver it to **A** and **B**
  - if **A** and **B** have previously used a key, one party could transmit the new key to the other, encrypted using the old key
  - 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**

Key Distribution…

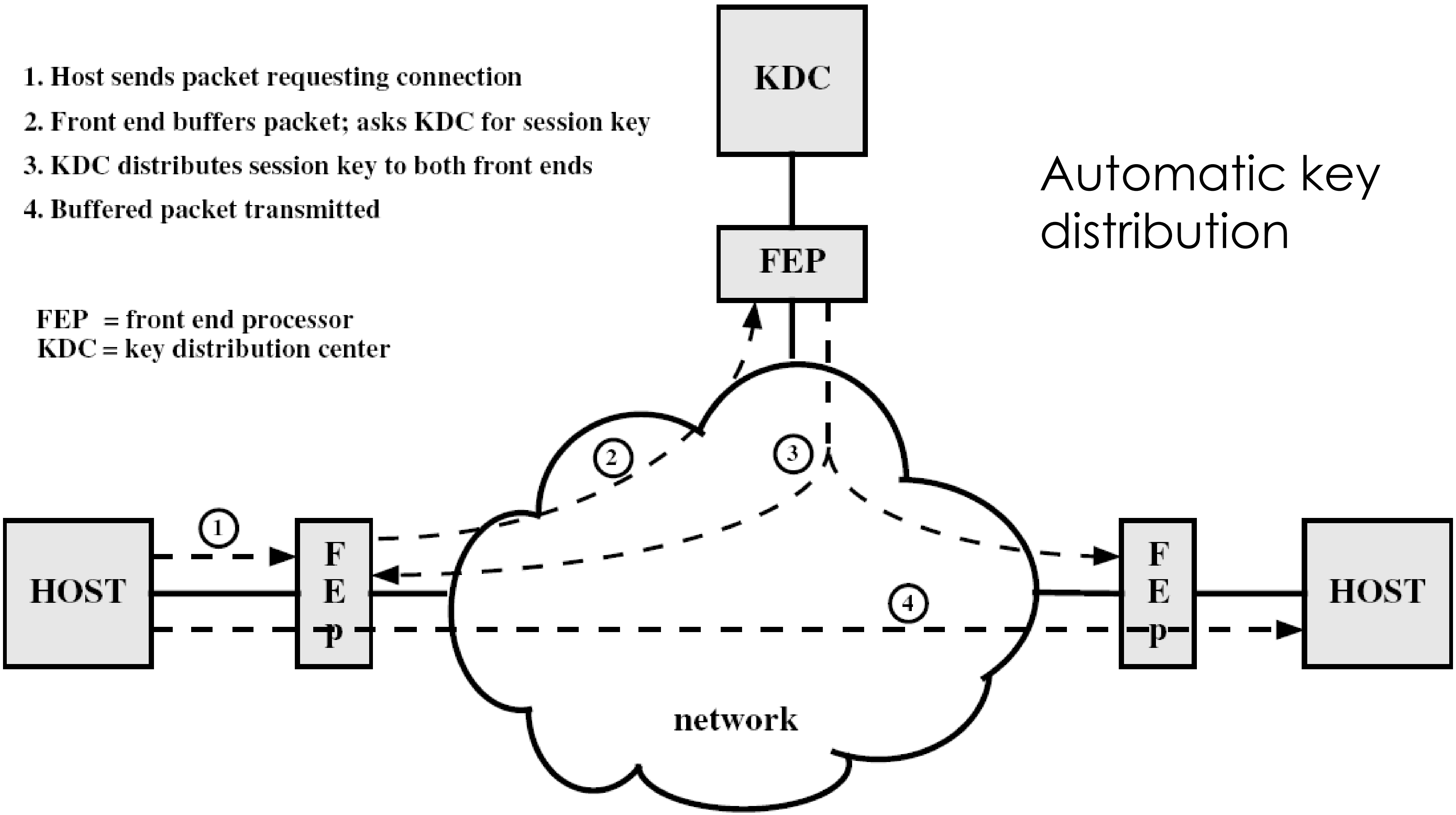
- **For communicating between small number of parties, manual distribution of keys may be feasible**
  - impossible for large number of parties
  - concept of Key Distribution Center (KDC) is more practical
    - determines which systems are allowed to communicate with each other
    - If connection is granted, distribute the session key
  - Front end processor obtains session keys on behalf of host
- **Session key:**
  - data encrypted with a one-time session key
  - destroyed when the session ends
- **Permanent key:**
  - used between entities for the purpose of distributing session keys

Key Distribution….

1. Host sends packet requesting connection
2. Front end buffers packet; asks KDC for session key
3. KDC distributes session key to both front ends
4. Buffered packet transmitted

FEP = front end processor
KDC = key distribution center

Automatic key distribution



Further Reading

- **Study Guide 2**
- **Chapter 2 & Section 4.1 of the textbook: *Network Security Essentials- Application & Standards" by William Stallings* 5th Edition, Prentice Hall, 2013**
- **Additional resources for this week**

- **Acknowledgement: part of the materials presented in the slides was developed with the help of Instructor's Manual and other resources made available by the author of the textbook.**