

# 정보보호

(5111041)

# 20장

대칭 암호화와 메시지 기  
밀성

# 대칭 암호화

- 또한 다음과 같이 불림:
  - 관용 암호화
  - 비밀키 또는 단일 키 암호화
- 1970년대 공개키 암호화 이전의 유일한 대안
  - 현재에도 가장 널리 사용
- 다섯 가지 구성요소:
  - 평문
  - 암호화 알고리즘
  - 비밀키
  - 암호문
  - 복호화 알고리즘



# 암호 기법

다음의 세 가지 측면에 따라 분류됨:

평문에서  
암호문으로의 전화에  
사용되는 연산 유형

- 대체 (substitution)– 각 평문 요소들이 또 다른 평문 요소로 맵핑 됨
- 전치 (Permutation)– 평문에 있는 요들이 재 배열됨

사용되는 키  
숫자 (number)

- 전송자와 수신자가 동일한 키 사용 – 대칭적
- 전송자와 수신자가 각기 다른 키를 사용 – 비대칭적

평문이 처리되는 방식

- 블록 암호 – 프로세스는 한번에 하나의 블록을 입력
- 스트림 암호 – 프로세스는 요소들을 연속적으로 입력

# 계산상 안전한 암호화 스키마 (Computationally Secure Encryption Schemes)

- 다음의 경우 암호화가 계산상으로 안전:
  - 암호를 푸는 비용이 정보의 가치보다 높은 경우
  - 암호를 푸는데 요구되는 시간이 정보의 이용 주기 보다 많은 경우
- 대체로 암호를 푸는데 요구되는 노력(effort)의 정도를 측정하는 것은 어려움
- 전수공격(brute-force )에 대한 time/cost 측정 가능

# Feistel 암호 구조 → DES 암호화 알고리즘

Feistel is a German American cryptographer

[https://en.wikipedia.org/wiki/Horst\\_Feistel](https://en.wikipedia.org/wiki/Horst_Feistel)

[https://en.wikipedia.org/wiki/Feistel\\_cipher](https://en.wikipedia.org/wiki/Feistel_cipher)

참고링크 :

[https://bpsecblog.wordpress.com/2016/11/29/amalmot\\_5/](https://bpsecblog.wordpress.com/2016/11/29/amalmot_5/)

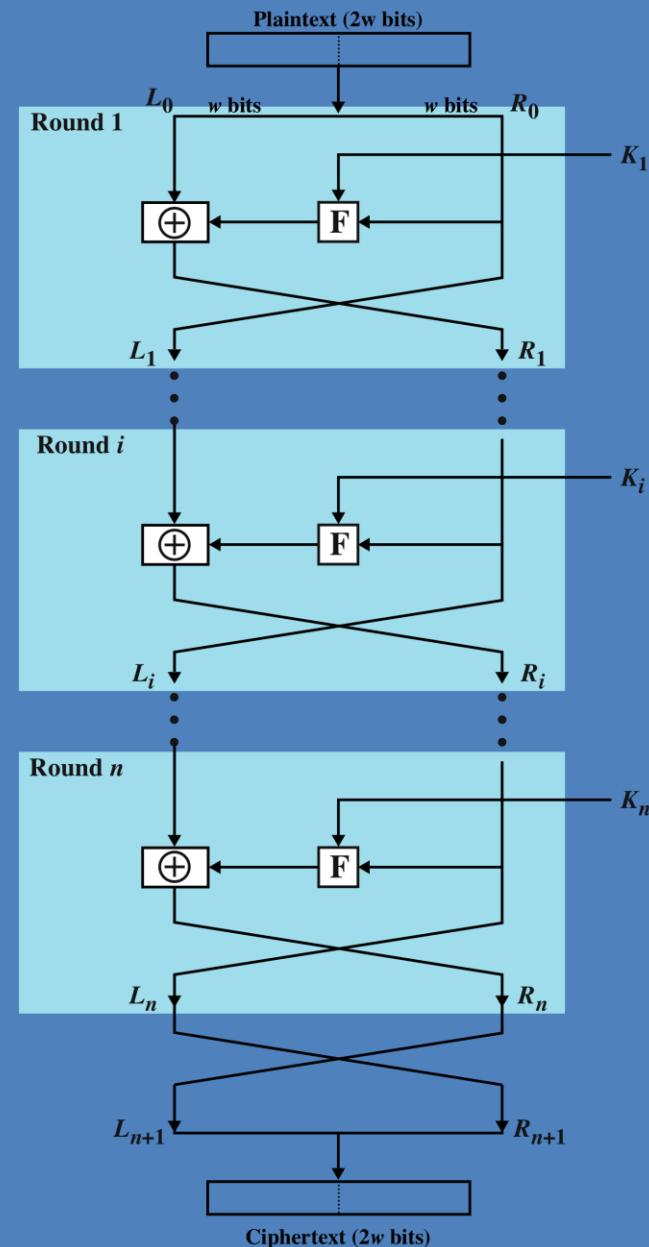


Figure 20.1 Classical Feistel Network

# DES의 암호화 복호화

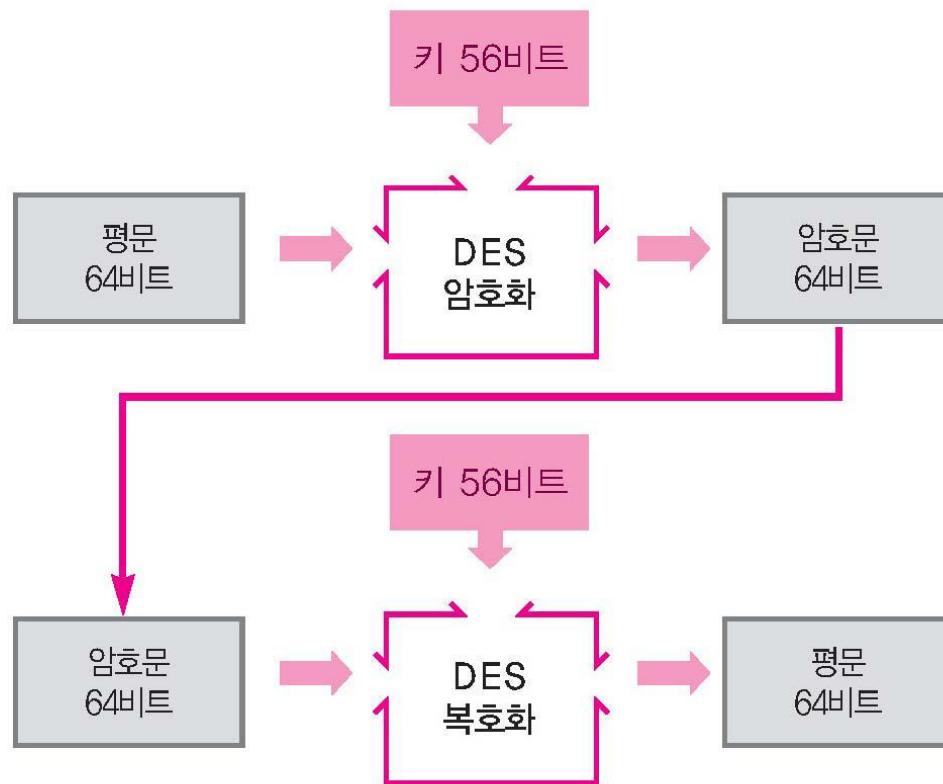


그림 3-8 DES의 암호화 · 복호화

# Feistel 구조의 한 라운드

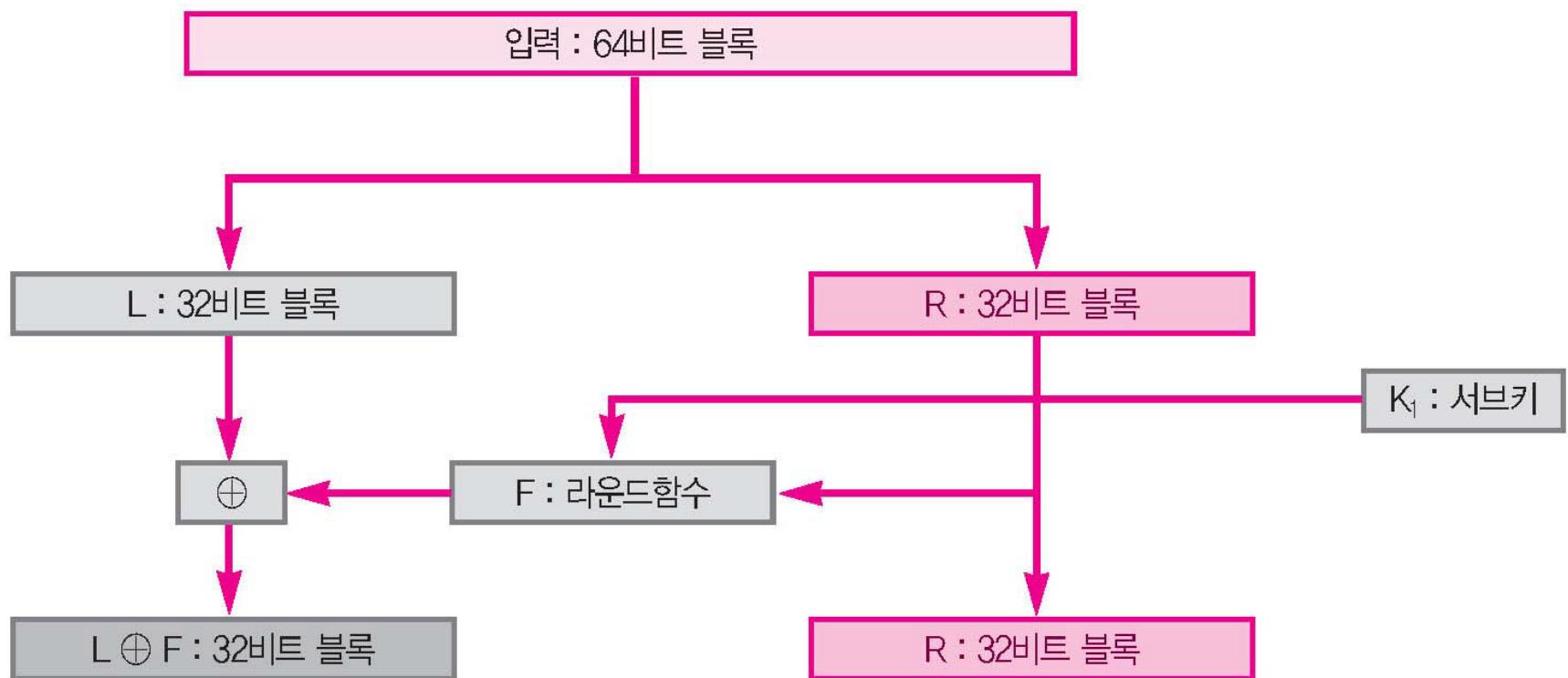


그림 3-9 페이스텔 네트워크 1 라운드

# 3 라운드 Feistel 구조에 의한 암호화 예시

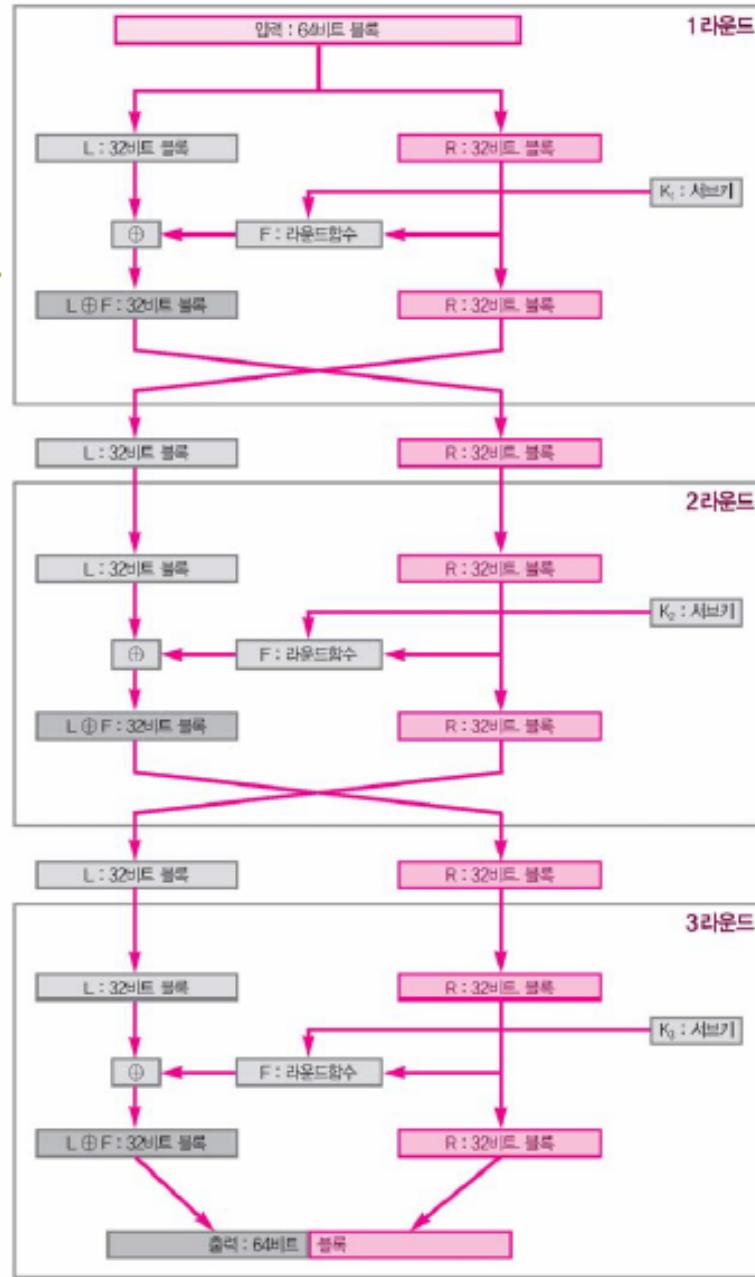


그림 3-10 퍼이스텔 네트워크의 암호화(3라운드)

# 동일한 보 조키로 2회 반복

→ 원래로  
돌아감(복  
호)

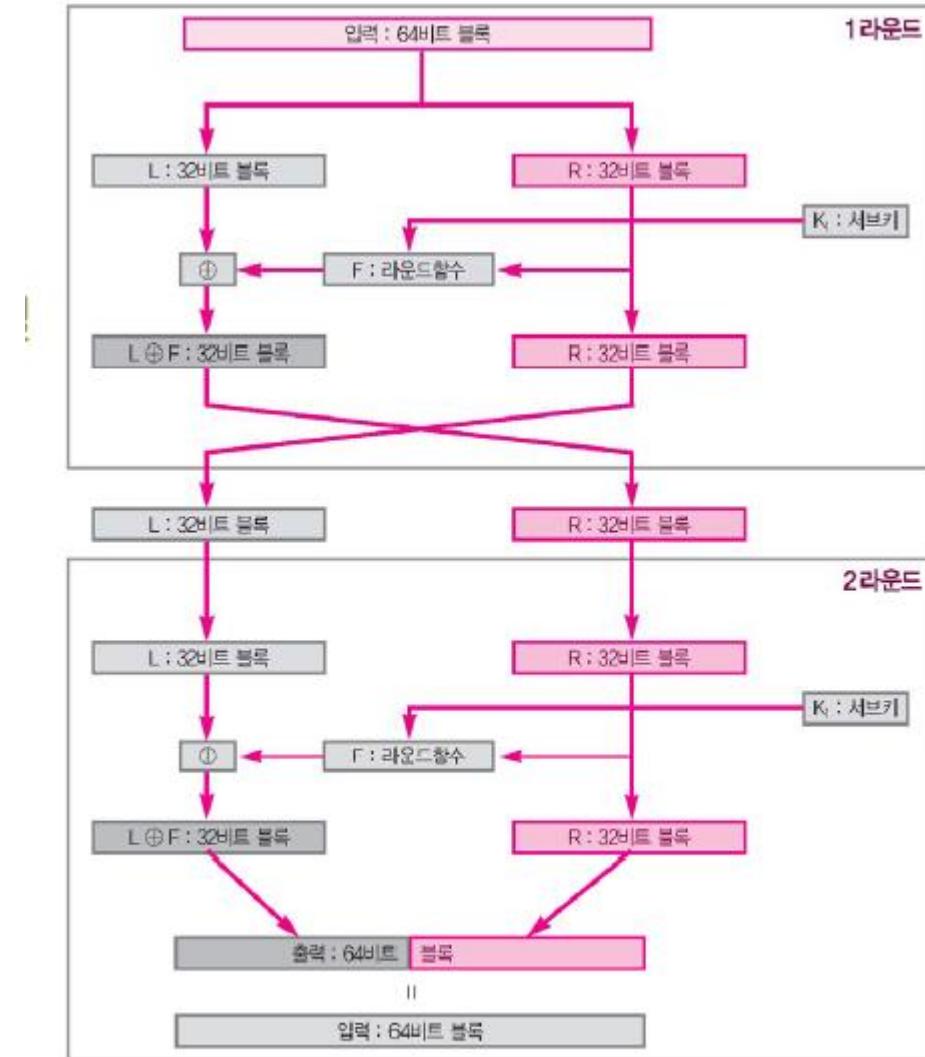


그림 3-11 같은 서브 키로 퍼스트 네트워크를 2회 통과시키면 원래로 돌아간다

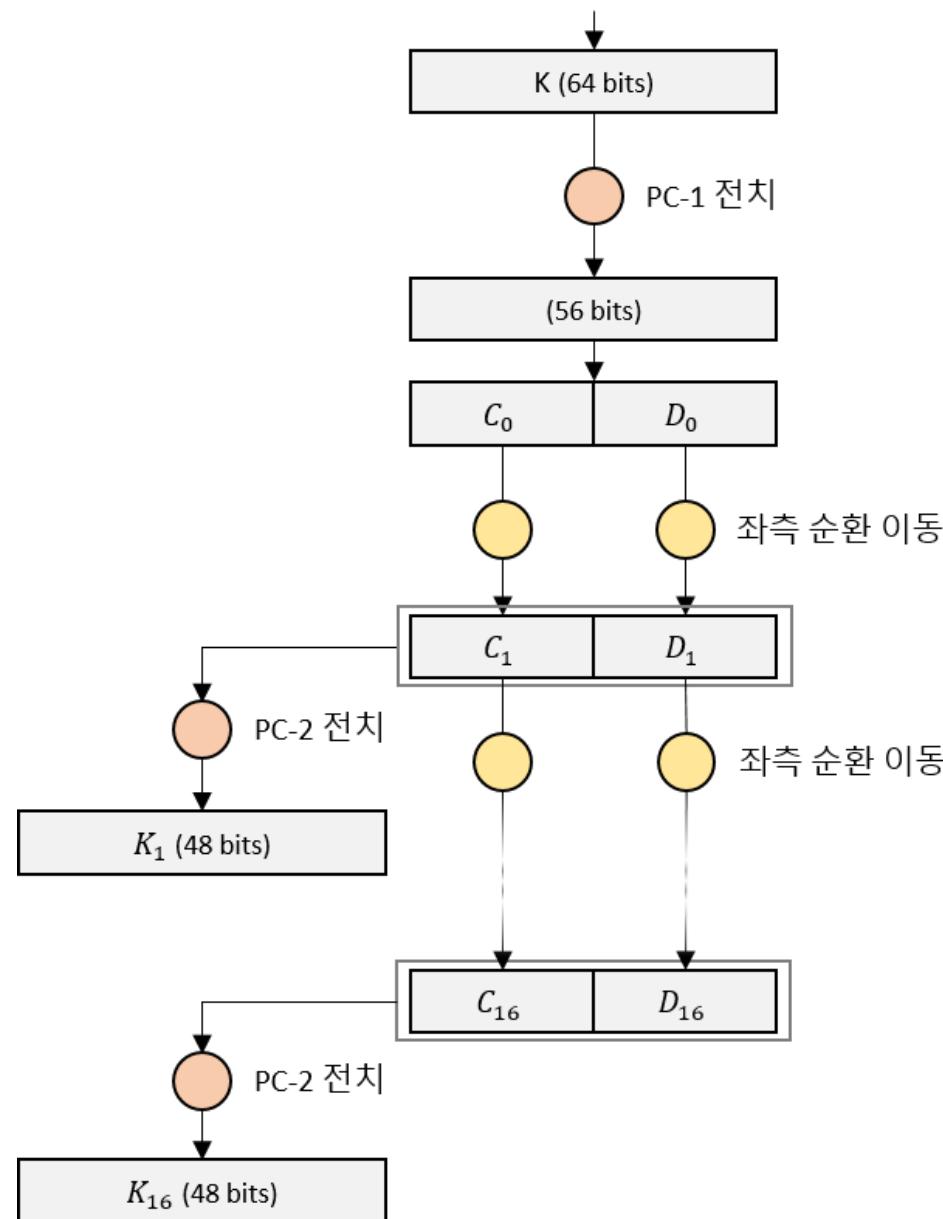
# Feistel 구조의 성질

- 원하는 만큼 라운드 수를 늘릴 수 있다.
  - DES는 16라운드로 구성
  - 각 라운드에서 각기 다른 보조키를 사용
- 라운드 함수 F에 다양한 함수를 사용할 수 있다.
  - DES에 적용된 라운드 함수 학습
- 암호화와 복호화를 완전히 동일한 구조로 수행한다.

# Feistel 보조키 생 성 예시

## [보조키 만들기]

56 bits 키 + 7비트마다 에러 정정 비트



# Feistel 보조키 생성 과정

- Error 정정비트
  - 0100111 0011011 1101110 1000110 0101011 1000001  
1000010 1100100
  - 01001110 00110110 11011101 10001101 01010110  
10000010 10000100 11001001
- Permutation Choice 1 (PC-1 전치)
  - Ex) 57번째 비트를 첫 번째 자리로, 49번째 비트를 두 번째 자리로 위치시키기라는 뜻

PC-1

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

# Feistel 보조키 생성 과정

- Permutation Choice 1 (PC-1 전치)
  - Left 1110110 0100101 0100000 0100001  
Right 0011001 1010111 1110000 1010110
- 좌측 순환 이동 (Left circular shift)
  - Left sequence에 적용 예
  - Before 1110110 0100101 0100000 0100001  
After 1101100 1001010 1000000 1000011

Rotations	
Round number	Number of left rotations
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

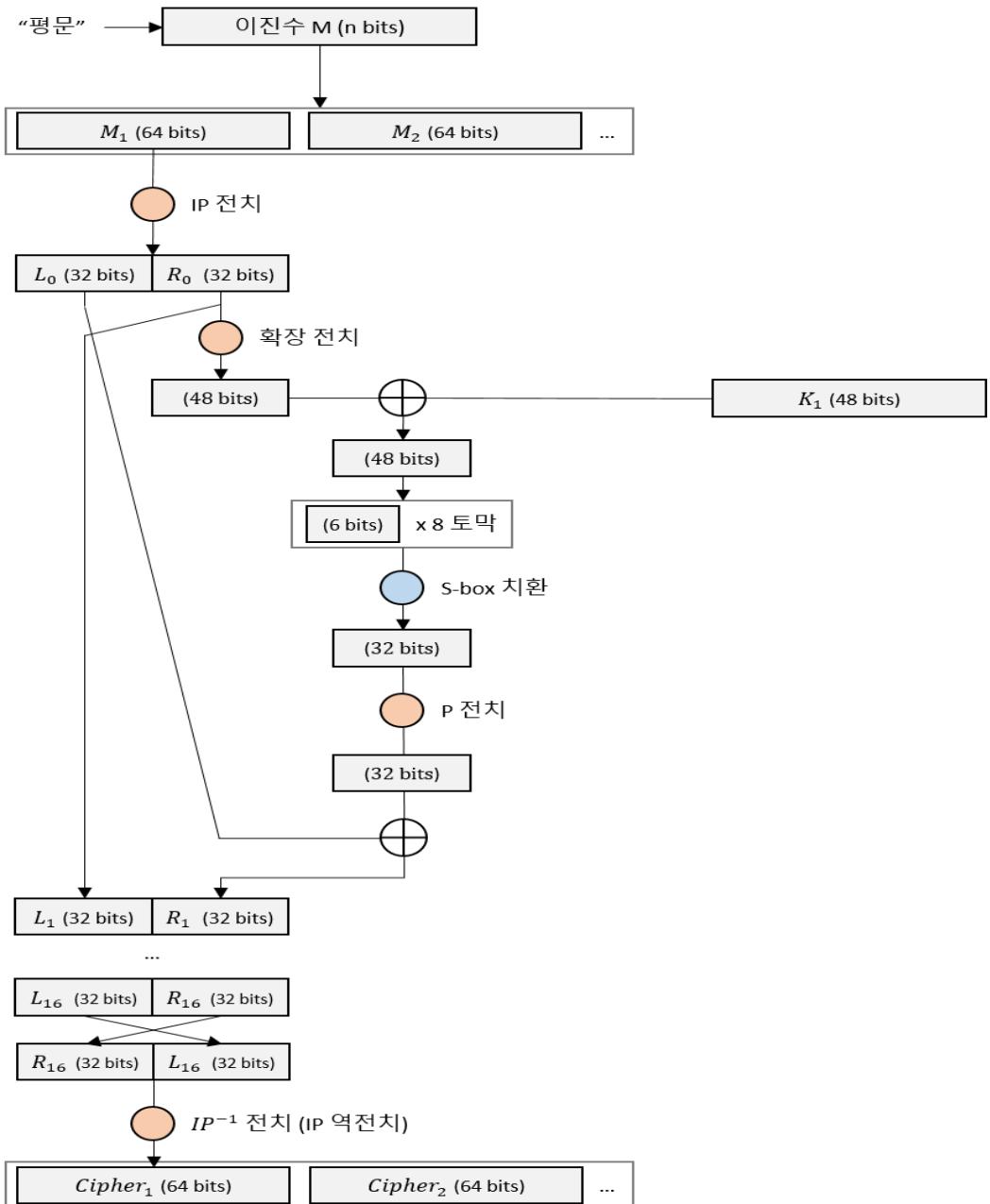
# Feistel 보조키 생성 과정

- Permutation Choice 2 (PC-2 전치)
  - 16 라운드 반복을 통해 16개의 보조키 생성

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

## [암호화 하기]

# Feistel 라운드 함수 예시



# Feistel 라운드 함수

- A. Initial Permutation (IP)
  - 64bit input → 64bit output
  - 순서만 변경

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	

- B. Left(32bit) 와 Right(32bit)로 분리

# Feistel 라운드 함수

- C. R0( Right 32bit sequence)에 대해 Expansion Permutation
  - 확장순열
  - 32bit input  $\rightarrow$  48bit output
    - $\rightarrow$  48bit? Key의 길이와 같음!!!
- D. 보조키와 XOR 연산을 수행
  - 48bit input  $\rightarrow$  48bit output



그림 3-3 XOR은 그림을 마스크한다

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

# S-Box

- E. S-Box 치환
  - 6bit \* 8 토막 분리 후  
Down selection
    - 6bit  $\rightarrow$  4bit  
(최종: 48bit  $\rightarrow$  32bit)
  - 1, 6번째 bit
    - Row를 선택
  - 2,3,4,5번째 bit
    - Column을 선택
  - S1 Example
    - 첫 번째 토막,
    - 011011  $\rightarrow$  5  $\rightarrow$  0101

<https://en.wikipedia.org/wiki/S-box>

S-boxes																
$S_1$	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0yyyy1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1yyyy0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1yyyy1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
$S_2$	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
0yyyy1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
1yyyy0	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
1yyyy1	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
$S_3$	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
0yyyy1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
1yyyy0	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1yyyy1	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
$S_4$	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
0yyyy1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
1yyyy0	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
1yyyy1	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
$S_5$	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
0yyyy1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
1yyyy0	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
1yyyy1	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
$S_6$	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
0yyyy1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
1yyyy0	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
1yyyy1	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
$S_7$	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
0yyyy1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1yyyy0	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
1yyyy1	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
$S_8$	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
0yyyy1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
1yyyy0	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
1yyyy1	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

# Feistel 라운드 함수

- F. Last permutation
  - 32bit → 32bit

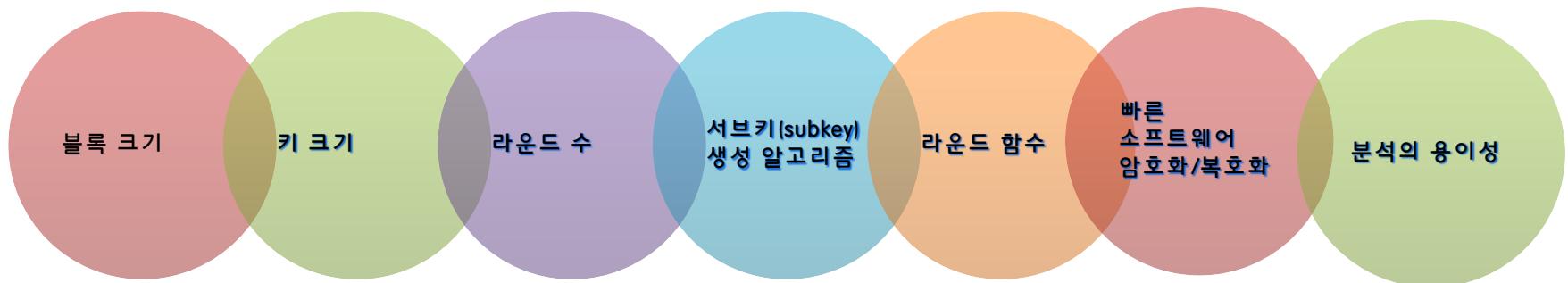
P	16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10	
2	8	24	14	32	27	3	9	
19	13	30	6	22	11	4	25	

- G. F의 결과(32bit) 를 L0(32bit)와 XOR
  - 32bit → 32bit
- H. 위의 결과를 R1에 대입, L1=R0을 대입하여 64bit 재배열

B~H 의 과정을 16라운드 반복 후,  
A의 inverse 과정을 수행하면 encryption  
된 64bit가 최종 생성!!

# 블록 암호 구조 (Block Cipher Structure)

- 대칭 블록 암호는 다음을 구성:
  - 일련의 라운드(round)
  - 라운드
  - 키에 의해 제어되는 대체(substitutions)와 순열(permutations)
- 파라미터 및 설계 요소:



- 가장 널리 사용되는 암호화 스키마
- 1977년 미 연방 표준국에 의해 채택됨
  - 현재 NIST
- FIPS PUB 46
- DEA(Data Encryption Algorithm) 참조



DES  
(Data  
Encryption  
Standard)

## 트리플 DES (3DES)

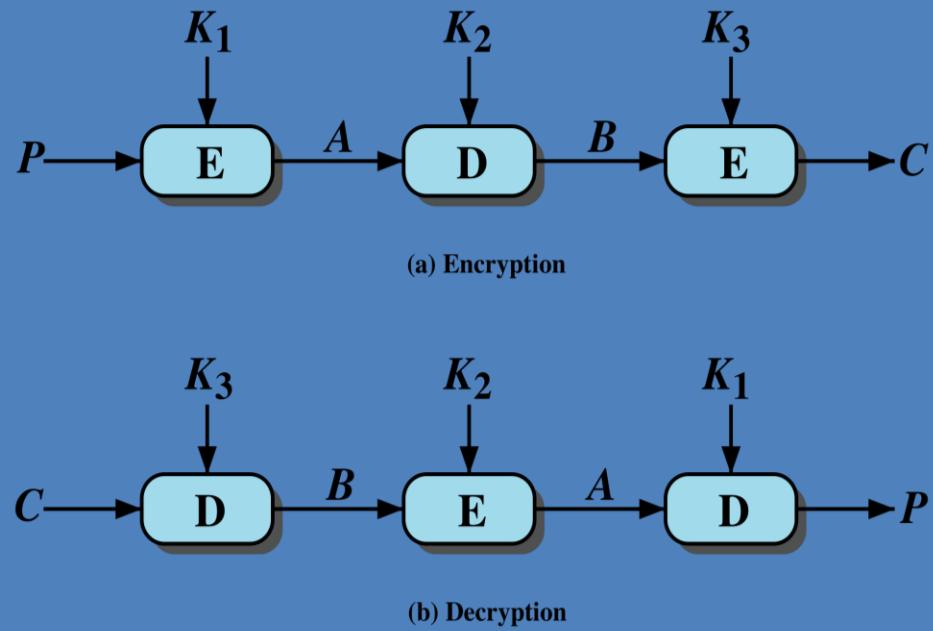


Figure 20.2 Triple DES

- 처음 금융 어플리케이션에서 사용됨
- 1999년 DES FIPS PUB 46-3로 표준
- 3개의 키와 3개의 DES 실행:  
$$C = E(K_3, D(K_2, E(K_1, P)))$$
- 복호화는 역방향 키를 가지고 동일한 방식으로 운영
- 2단계에서의 복호화하는 본래 DES 사용자와의 호환성 제공
- 효과적인 168 비트 키 길이, 속도가 느리지만 안전

# AES (Advanced Encryption Standard)

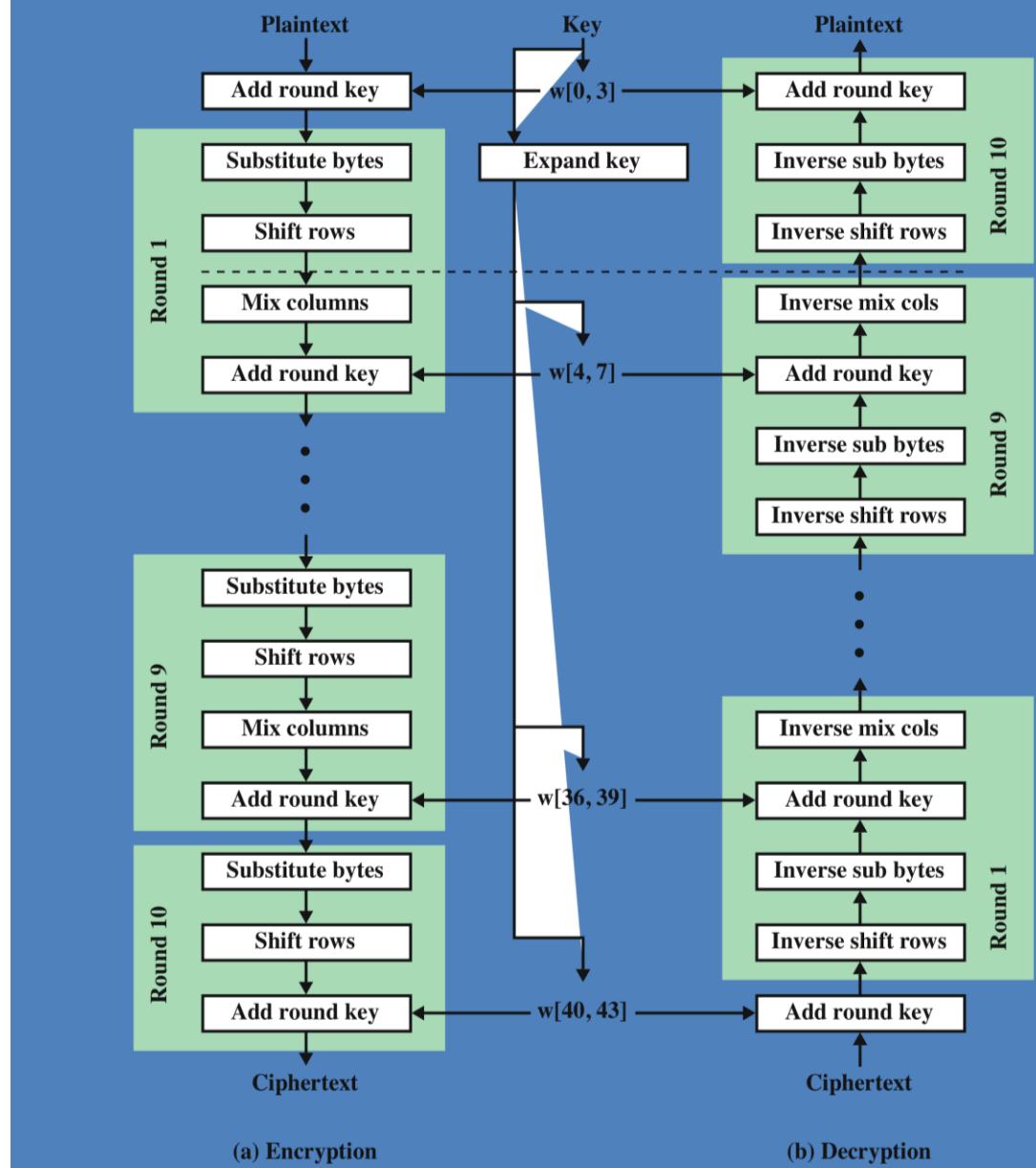


Figure 20.3 AES Encryption and Decryption

# ASE 암호 알고리즘

- 블록 길이 : 128 bit
- 키 길이 : 128, 192, 256 bit (보통 128 bit 사용)
- Feistel 구조가 아님
- 알고리즘 순서
  - 128 input → 8bit \* 16토막으로 분리
  - Add round key 수행 후,
  - AES 암호화 라운드 10회 수행
    - AES 암호화 라운드 : Substitute bytes, Shift rows, Mix columns, Add round key 순서로 구성
  - Add round key 단계에서만 Key가 사용됨
    - 각 단계마다 다른 라운드 키를 입력키로부터 확장해서 사용함

# AES 라운드 조

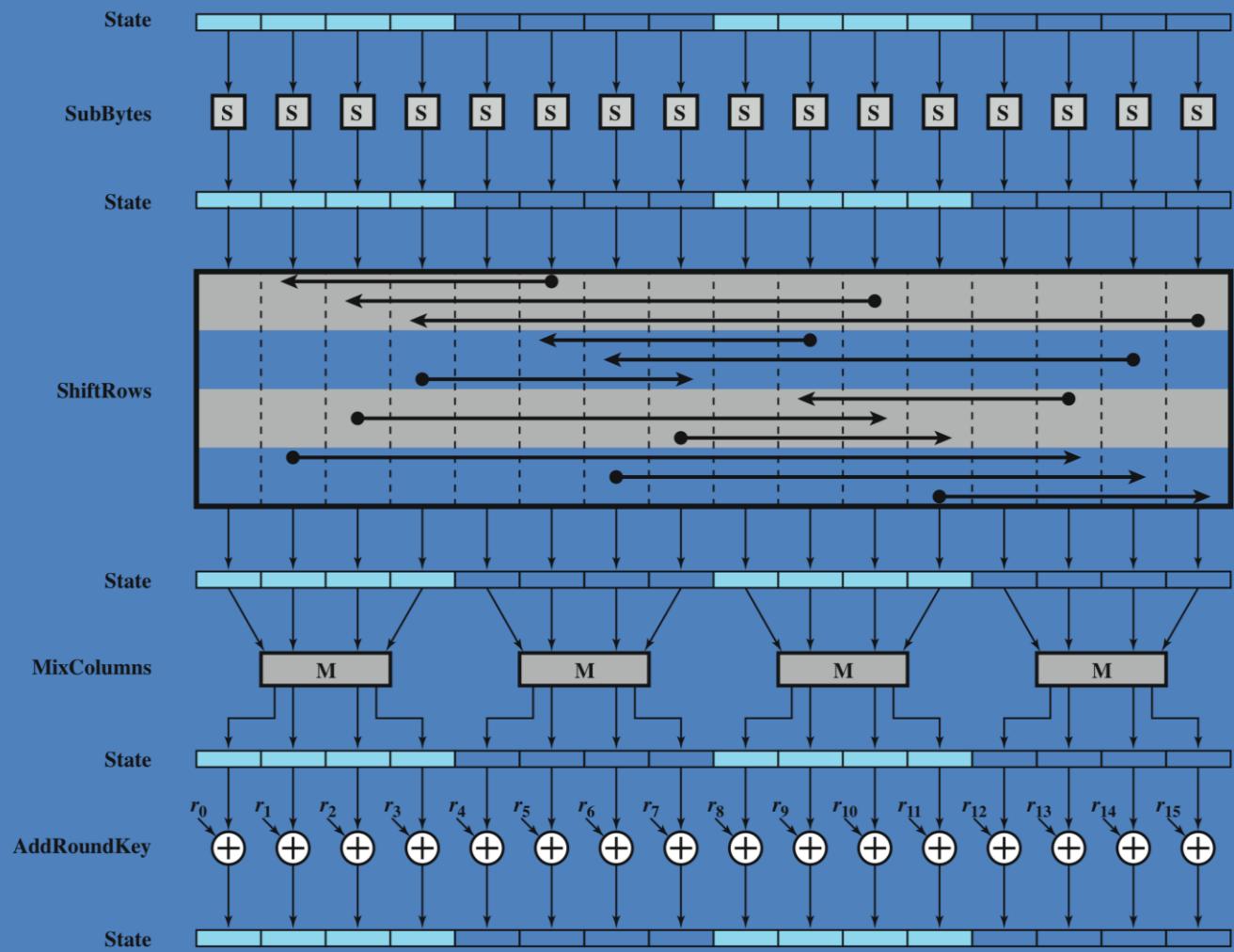


Figure 20.4 AES Encryption Round

# Table 20.2

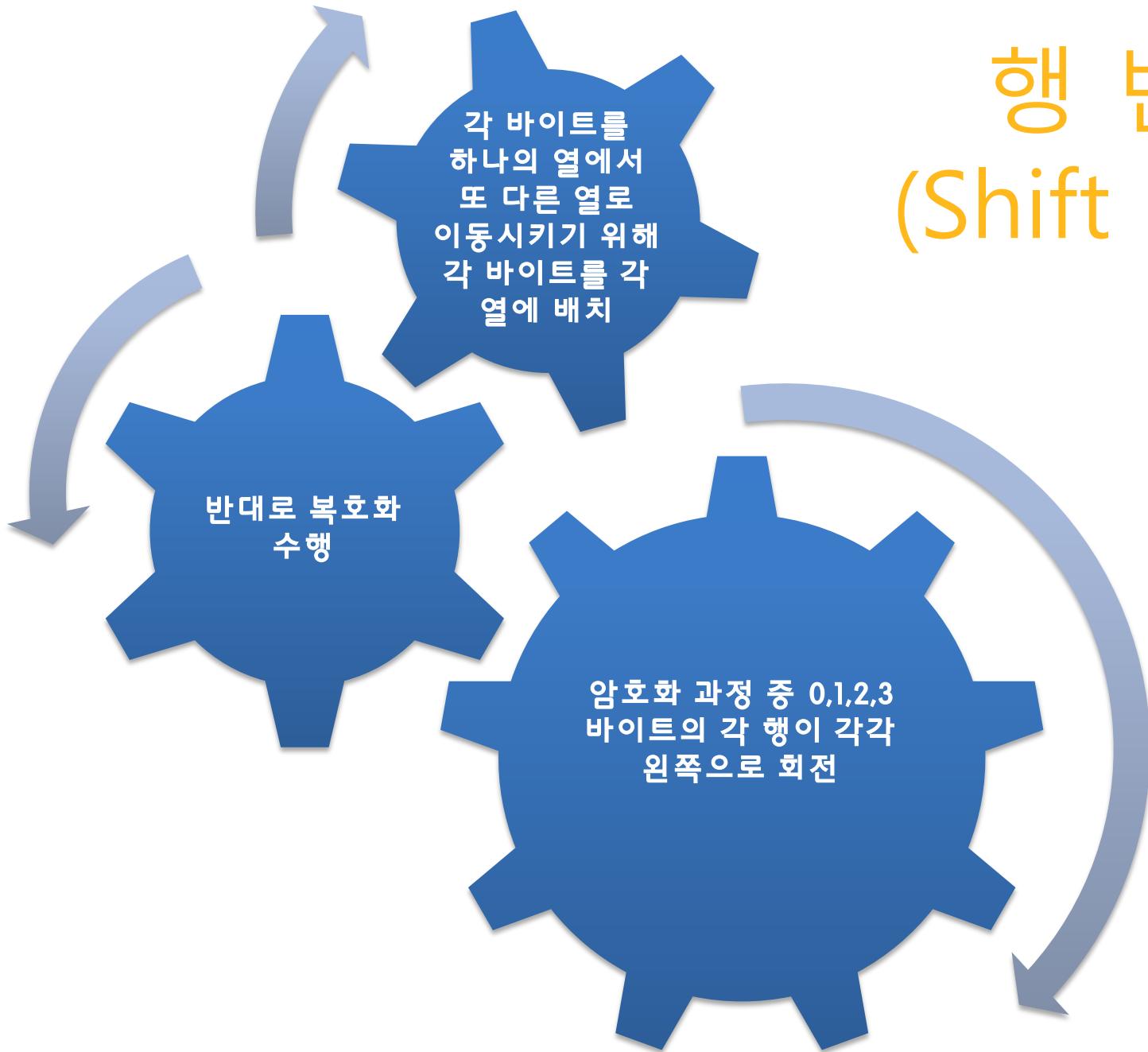
## (a) S-box

	<b>y</b>																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

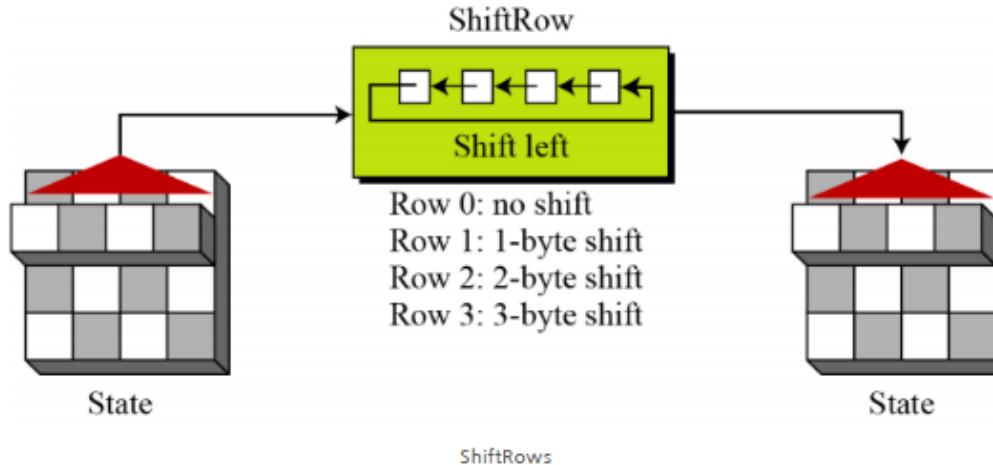
Table 20.2  
(b) 역 S-box

	<b>y</b>																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

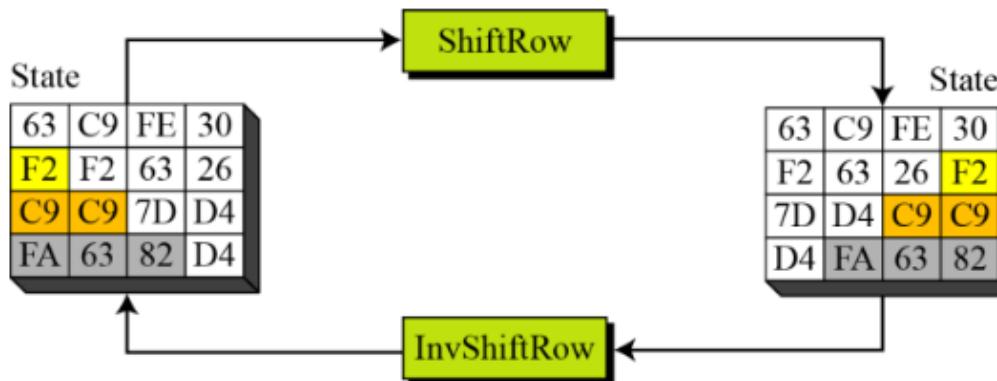
# 행 변환 (Shift Rows)



# 행 변환 (Shift Rows)



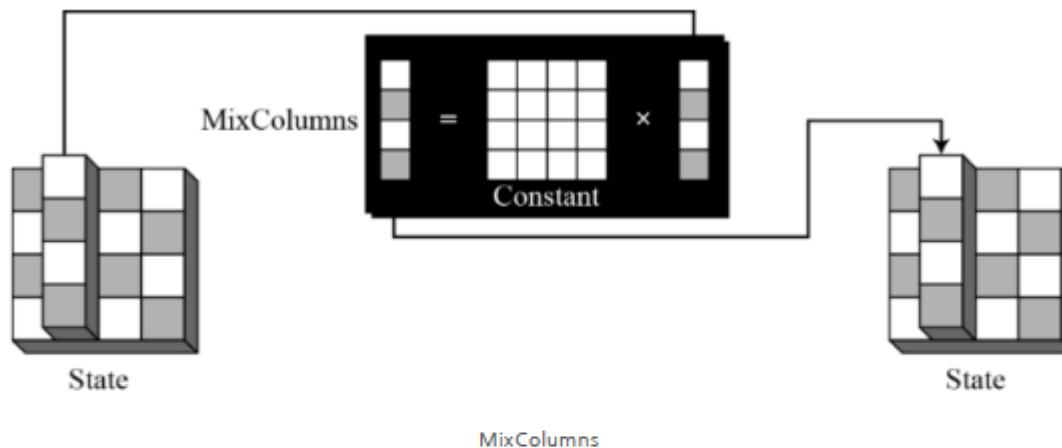
- 첫 번째 행은 Shift 되지 않는다.
- 두 번째 행은 한 자리 왼쪽으로 Shift 된다.
- 세 번째 행은 두 자리 왼쪽으로 Shift 된다.
- 마지막 행은 세 자리 왼쪽으로 Shift 된다.



# Mix Columns와 Add Key

- 열변환(mix columns)
  - 각 열마다 개별적으로 수행
  - 각 열의 전체 4바이트 함수의 새로운 값에 각 바이트를 맵핑
  - 맵핑은 유한체(finite fields) 방정식 사용
  - 각 열마다 바이트를 잘 섞어주기 위함
- 라운드 키 추가(add round key)
  - 확장된 키 비트단위로 state를 XOR 수행
  - 라운드 키 확장의 복잡성과 다른 단계의 AES로 보안

# Mix Columns Transformation



현재 행렬에서 칼럼을 가져와서 미리 정의된 Constant matrix에 곱하게 된다.

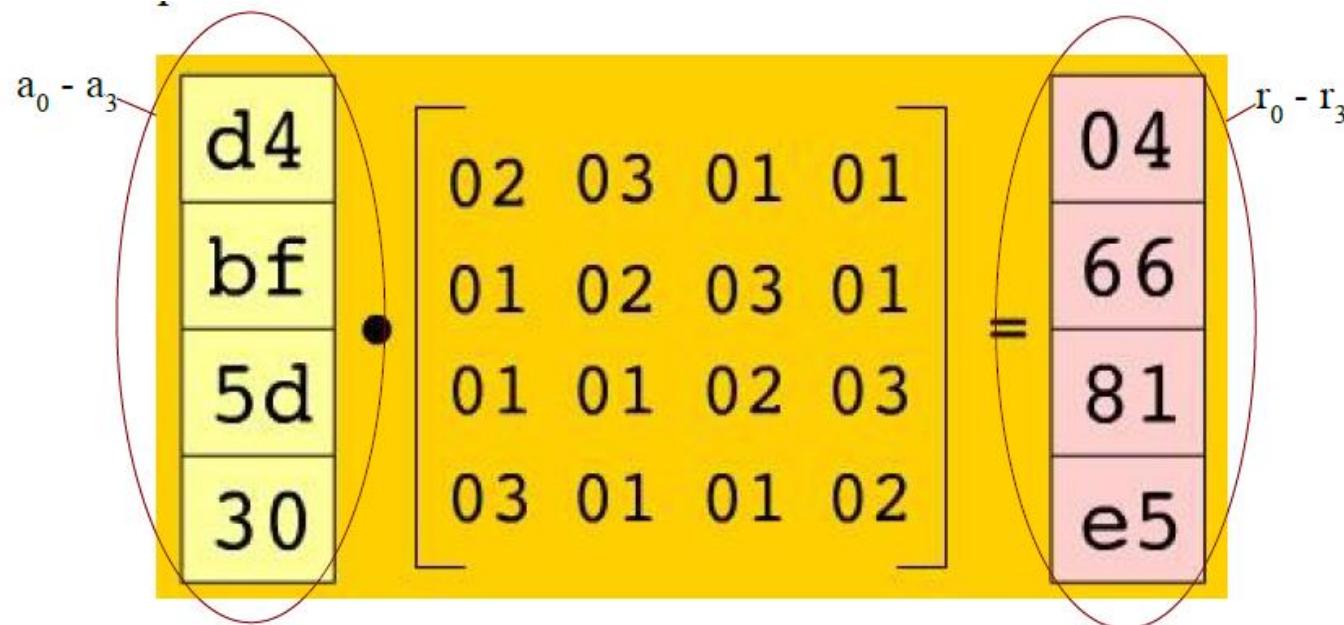
$$\begin{array}{l} ax + by + cz + dt \\ ex + fy + gz + ht \\ ix + jy + kz + lt \\ mx + ny + oz + pt \end{array} \rightarrow \boxed{\begin{array}{c} \text{New matrix} \\ \text{Old matrix} \end{array}} = \boxed{\begin{array}{c} \text{Constant matrix} \\ \text{Old matrix} \end{array}} \times \boxed{\begin{array}{c} \text{Old matrix} \\ \text{Old matrix} \end{array}}$$

The diagram shows a mapping from a set of linear combinations of variables (x, y, z, t) to a New matrix, which is then multiplied by a Constant matrix to produce an Old matrix.

# Mix Columns Transformation

- AES MixColumns Transformation

$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$



# Inverse MixColumns Transformation

## InverseMixColumns [ [edit](#) ]

---

The MixColumns operation has the following inverse (numbers are decimal):

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

Or:

$$b_0 = 14 \bullet d_0 \oplus 11 \bullet d_1 \oplus 13 \bullet d_2 \oplus 9 \bullet d_3$$

$$b_1 = 9 \bullet d_0 \oplus 14 \bullet d_1 \oplus 11 \bullet d_2 \oplus 13 \bullet d_3$$

$$b_2 = 13 \bullet d_0 \oplus 9 \bullet d_1 \oplus 14 \bullet d_2 \oplus 11 \bullet d_3$$

$$b_3 = 11 \bullet d_0 \oplus 13 \bullet d_1 \oplus 9 \bullet d_2 \oplus 14 \bullet d_3$$

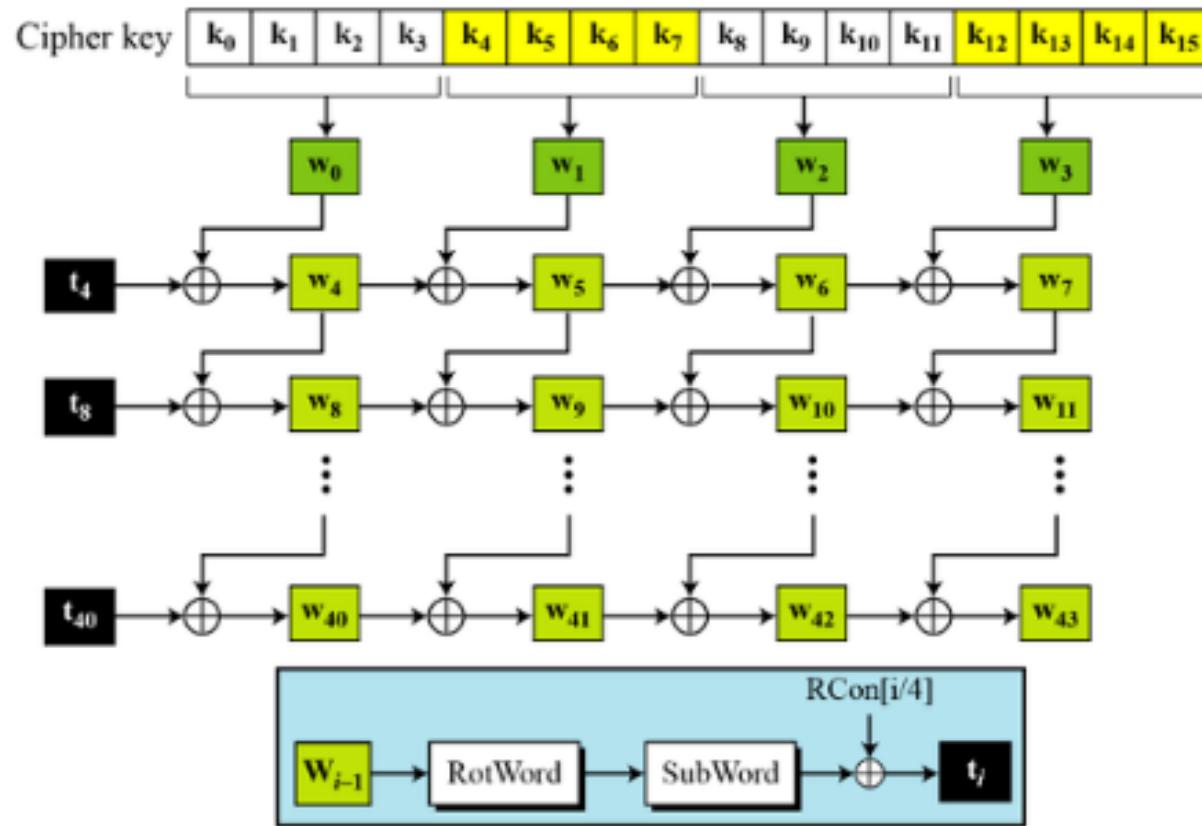
# Inverse MixColumns Transformation

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \xleftarrow{\text{Inverse}} \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

$C$     $C^{-1}$

MixColumns Constant matrix

# Subkey Generation, Key Schedule



Making of  $t_i$  (temporary) words  $i = 4 N_r$

$Rcon[i] = [x^{i-1}, \{00\}, \{00\}, \{00\}]$   
 $Rcon[1] = [x^0, \{00\}, \{00\}, \{00\}] = [\{01\}, \{00\}, \{00\}, \{00\}] = 01000000$   
 $Rcon[2] = [x^1, \{00\}, \{00\}, \{00\}] = 02000000$   
 $Rcon[3] = [x^2, \{00\}, \{00\}, \{00\}] = 04000000$   
 $Rcon[4] = [x^3, \{00\}, \{00\}, \{00\}] = 08000000$   
 $Rcon[5] = [x^4, \{00\}, \{00\}, \{00\}] = 10000000$   
 $Rcon[6] = [x^5, \{00\}, \{00\}, \{00\}] = 20000000$   
 $Rcon[7] = [x^6, \{00\}, \{00\}, \{00\}] = 40000000$   
 $Rcon[8] = [x^7, \{00\}, \{00\}, \{00\}] = 80000000$   
 $Rcon[9] = [x^8, \{00\}, \{00\}, \{00\}] = [x^7 \bullet x, \{00\}, \{00\}, \{00\}] = 1b000000$   
 $x^7 \bullet x = xtime(x^7) = xtime(80) = \{leftshift(80)\} \oplus \{1b\} = 1b$   
 $Rcon[10] = [x^9, \{00\}, \{00\}, \{00\}] = [x^8 \bullet x, \{00\}, \{00\}, \{00\}] = 36000000$   
 $Rcon[11] = [x^{10}, \{00\}, \{00\}, \{00\}] = [x^9 \bullet x, \{00\}, \{00\}, \{00\}] = 6c000000$   
 $Rcon[12] = [x^{11}, \{00\}, \{00\}, \{00\}] = [x^{10} \bullet x, \{00\}, \{00\}, \{00\}] = d8000000$   
 $Rcon[13] = [x^{12}, \{00\}, \{00\}, \{00\}] = [x^{11} \bullet x, \{00\}, \{00\}, \{00\}] = ab000000$   
 $x^{11} \bullet x = xtime(x^{11}) = xtime(d8) = \{leftshift(d8)\} \oplus \{1b\} = ab$   
 Round Constant

$if(i \bmod 4=0) \quad w_i = t \oplus w_{i-4}$   
 $else \quad w_i = w_{i-1} \oplus w_{i-4}$

$t = SubWord(RotWord(w_{i-1})) \oplus RCon_{i/4}$

AES Key Expansion

# 스트림 암호

- 프로세스가 인자들(elements)을 연속적으로 입력
- 의사 난수(pseudorandom) 비트 생성기로 키 입력
  - 숫자와 같은 랜덤 스트림 생성
  - 입력 키 없이는 예측 불가능
  - 평문 바이트로 XOR 키스트림이 출력
- 훨씬 빠르고 비교적 단순한 코드
- 설계 고려사항:
  - 암호화 시퀀스의 주기가 커야 함
  - 키 스트림은 랜덤 숫자의 속성(properties)과 유사
  - 충분한 길이의 키 사용

# Table 20.3

**대칭 암호화 속도 비교 / 특정 컨디션에서 상대적 비교**

Cipher	Key Length	Speed (Mbps)
DES	56	21
3DES	168	10
AES	128	61
RC4	Variable	113

*Source:* <http://www.cryptopp.com/benchmarks.html>

# RC4(Rivest Cipher 4) 알고리즘

---

- 알고리즘 동작 원리

- 셔플링 기법이 동작의 기본 원리임
- 송신자는 비밀 키를 이용하여 256 바이트의 수를 뒤섞음
- 그런 후, 특정한 두 지점을 선택하여  $t$ 를 계산한 후,  $S[t]$  값을 사용하여 평문의 1바이트씩 XOR 연산하여 암호화 수행

# RC4(Rivest Cipher 4) 알고리즘

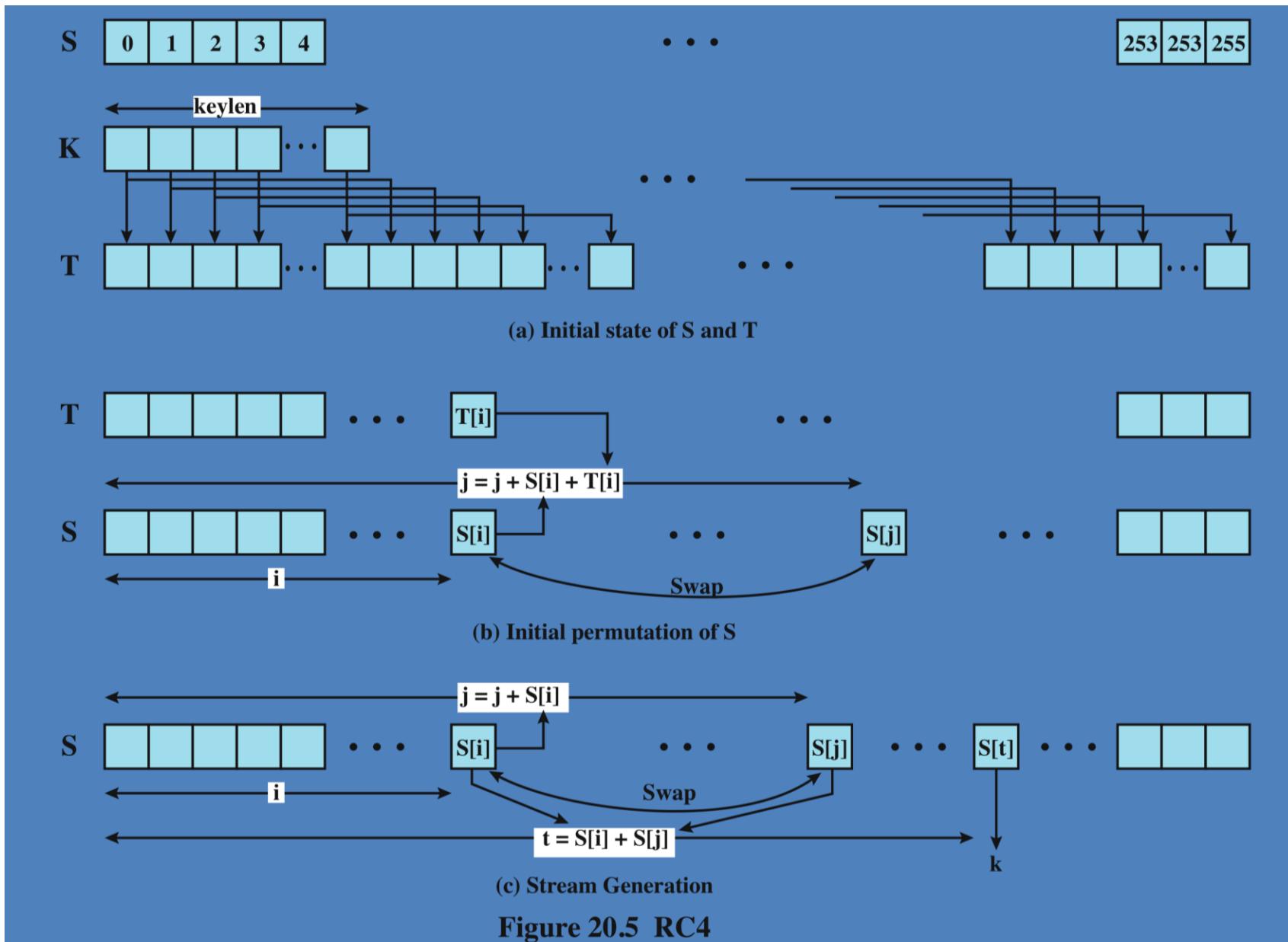


Figure 20.5 RC4

# RC4(Rivest Cipher 4) 알고리즘

- S초기화

For  $i = 0$  to 255

$S[i] = i$

$T[i] = \text{key}[i \ (\text{mod } N)]$

$j = 0$

for  $i = 0$  to 255

$j = (j + S[i] + T[i]) \ (\text{mod } 256)$

swap(  $S[i], S[j]$  )

# RC4(Rivest Cipher 4) 알고리즘

- Stream 생성

i=j=0

for counter = 0 to plaintext

i = ( i+1) mod 256

j = (j +S[i]) mod 256

Swap(S[i],S[j])

t=(S[i] + S[j]) mod 256

keystreamByte = S[t]

plaintext[counter] ^=S[keystreamByte]



20장

대칭 암호화와 메시지 기  
밀성 #2

# 운용 모드 (작동 모드)

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	<ul style="list-style-type: none"><li>Secure transmission of single values (e.g., an encryption key)</li></ul>
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	<ul style="list-style-type: none"><li>General-purpose block-oriented transmission</li><li>Authentication</li></ul>
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.	<ul style="list-style-type: none"><li>General-purpose stream-oriented transmission</li><li>Authentication</li></ul>
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding DES output.	<ul style="list-style-type: none"><li>Stream-oriented transmission over noisy channel (e.g., satellite communication)</li></ul>
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	<ul style="list-style-type: none"><li>General-purpose block-oriented transmission</li><li>Useful for high-speed requirements</li></ul>

# ECB(Electronic Codebook)

- 가장 단순한 모드
- 평문은  $b$  비트로 처리되며, 각 블록은 동일한 키를 사용하여 암호화 됨
- “codebook”은 각 평문 블록은 고유의 암호문 값을 가지기 때문에 사용
  - 반복적인 평문 모양에 대해서 동일한 암호문이 출력되기 때문에 장문의 메시지에는 안전하지 않음
- 보안 결함을 극복하기 위해 반복되는 구간에서 다른 암호문 블록을 생성하도록 하는 기법이 필요

# CBC(Cipher Block Chaining)

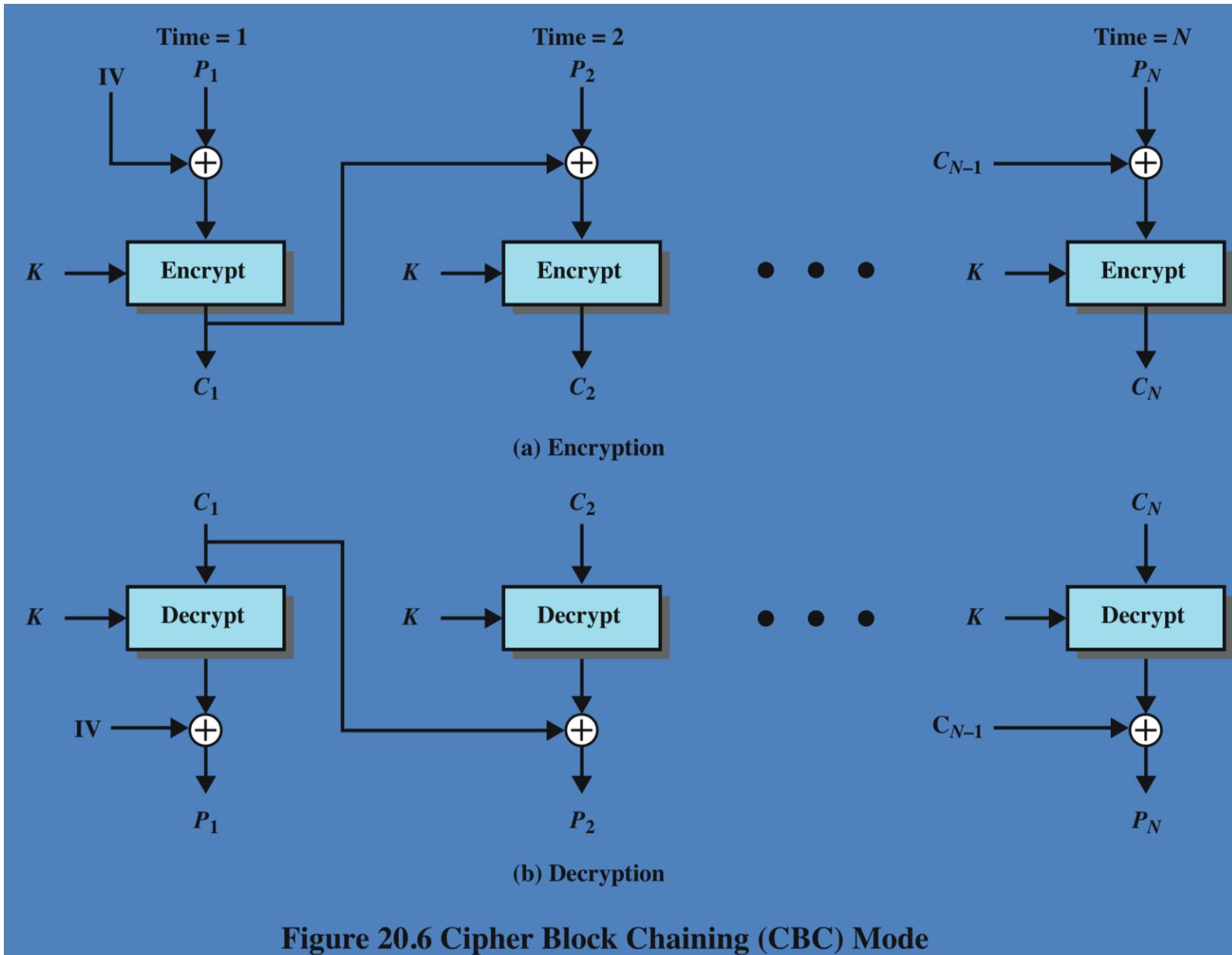


Figure 20.6 Cipher Block Chaining (CBC) Mode

# CBC(Cipher Block Chaining)

---

- 각 블록 간 연관성을 제공
  - 블록 간 체인화
- 같은 원문 데이터가 다른 암호문으로 encryption 됨
  - 기밀성 향상
- 복호화도 동일한 과정

# CFB(Cipher Feedback)

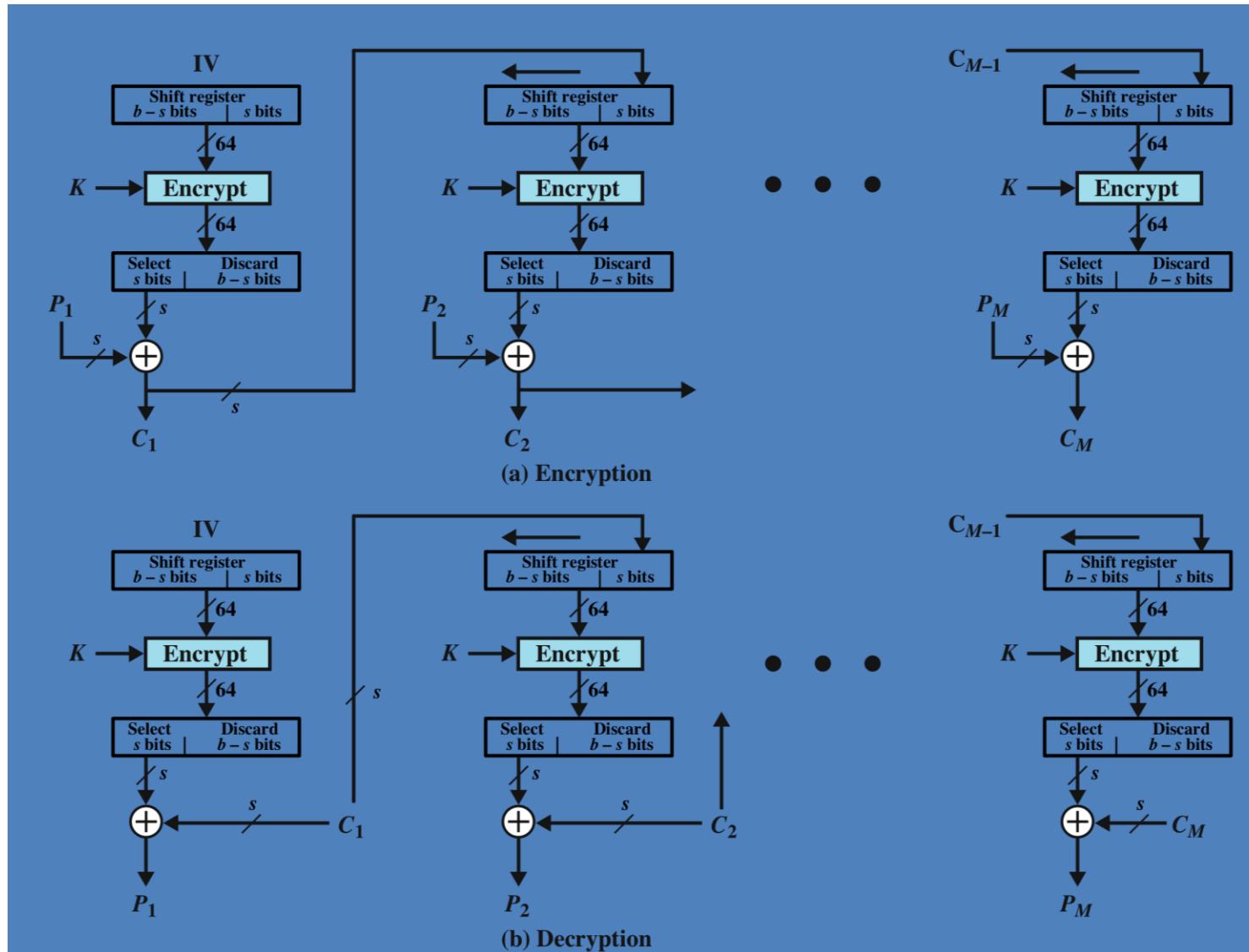
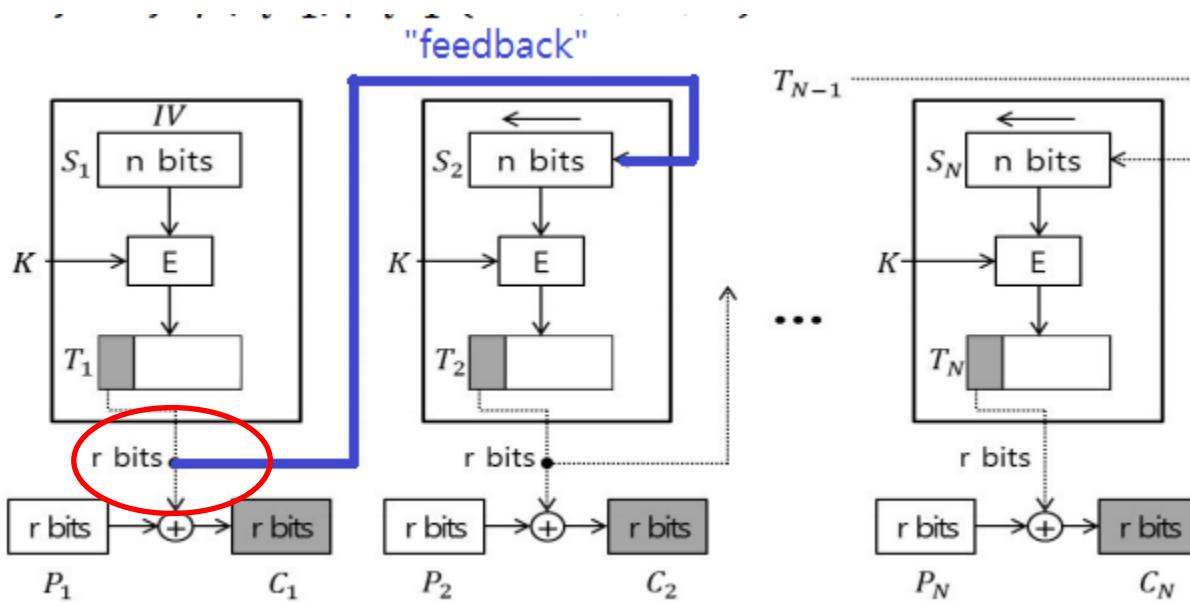


Figure 20.7 *s*-bit Cipher Feedback (CFB) Mode

# CFB(Cipher Feedback)

- 블록 암호화 기법을 응용하여 스트림 암호화  
    ● 원문의 1바이트씩 처리
- CFB(Cipher Feedback) vs. OFB(output Feedback)  
    ● Block encryption 블록의 위치에 따라 결정
- 복호화도 동일한 과정

# OFB(output Feedback)

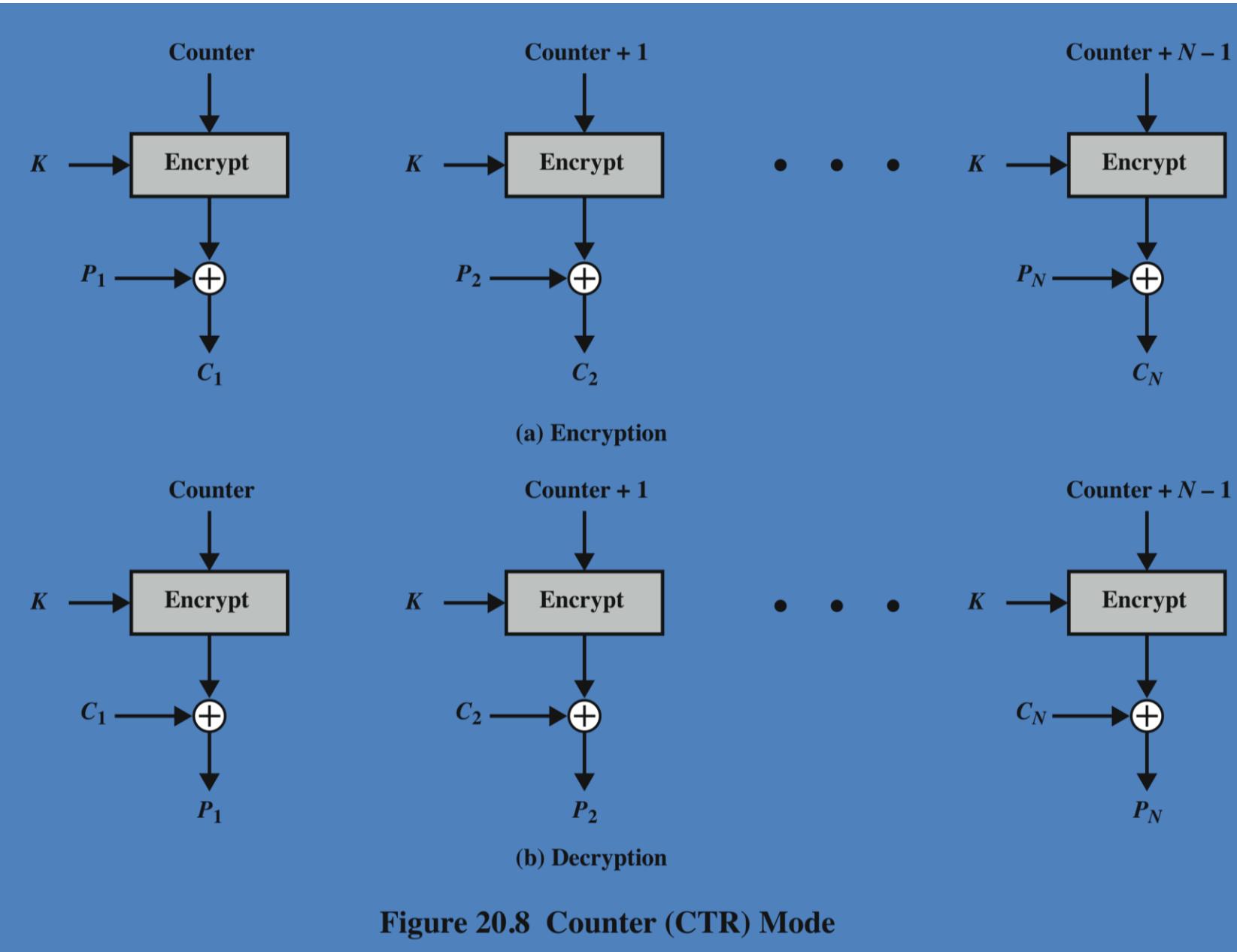


- CFB와 다른점?

- Error propagation?

- Preprocessing?

# CTR(Counter)



# CTR(Counter)

- 원리
  - 카운터 값을 encryption 하여 원문과 XOR
- 장점
  - 하드웨어/소프트웨어 효율성
    - 앞의 chaining 모드와는 달리 병렬 프로세스 가능
    - 앞의 블록에 대한 처리가 끝나기 전에 다음 블록에 대한 처리를 할 수 있음
  - Preprocessing
  - 랜덤 접근
  - 안전성
  - 간단함

# 암호화 위치

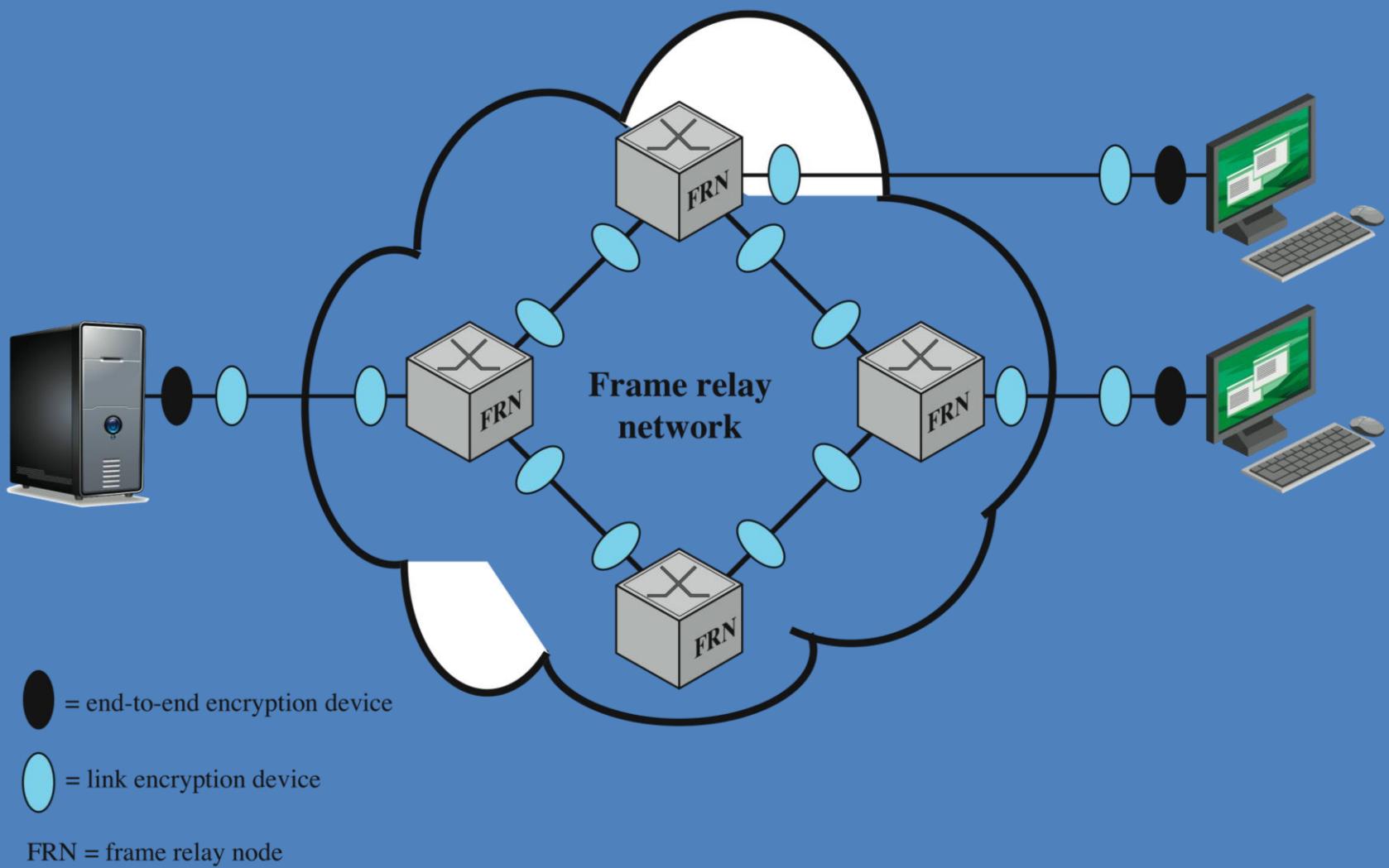


Figure 20.9 Encryption Across a Frame Relay Network

# 암호화 위치

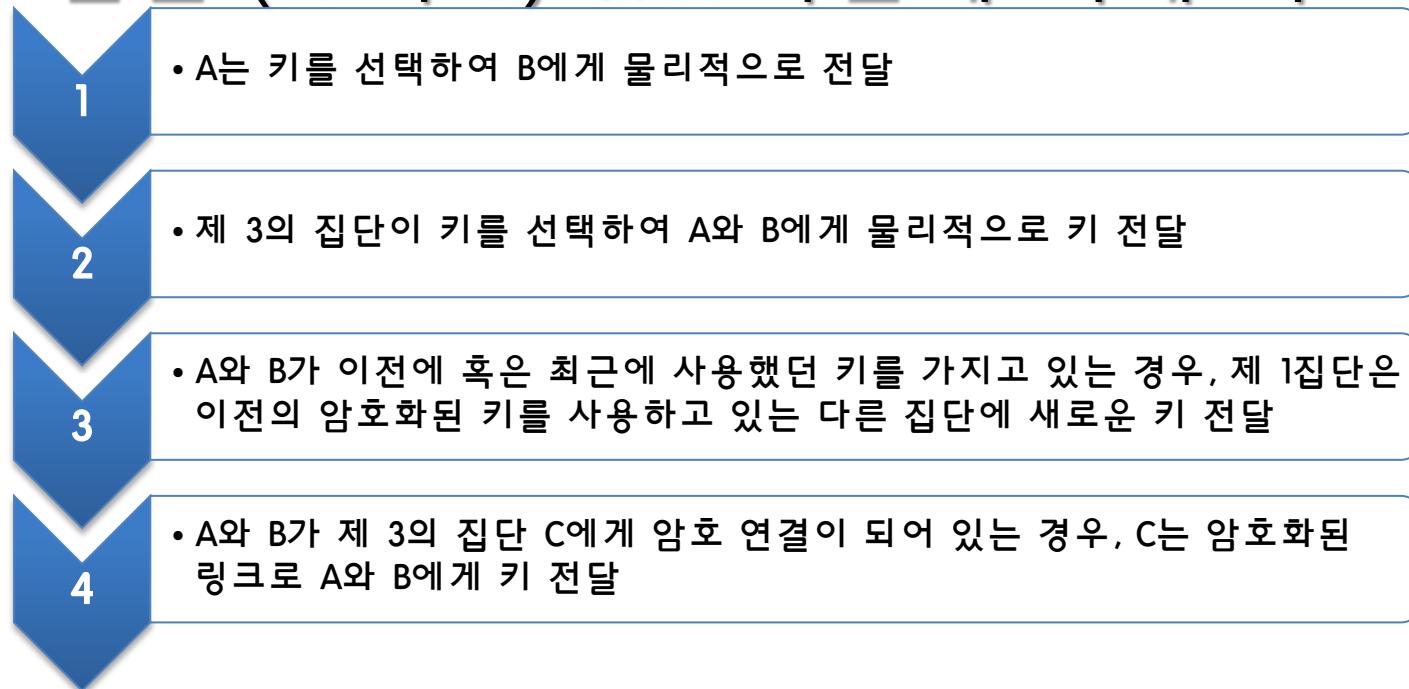
- 링크 암호화(Link encryption)
  - 링크 양 끝 단에 암호화 장치 구비
  - 모든 링크상의 트래픽은 안전
  - 단점
    - 매번 프레임 스위치에 들어갈 때마다 복호화 되어야 함
      - → 경로 지정을 위한 프레임 헤더 속 주소를 읽기 위해
    - 각 스위치에서 취약

# 암호화 위치

- 종단 간 암호화(end-to-end encryption)
  - 두 end-point device에서 암호화 수행
  - 데이터 “만” 암호화된 형태로 네트워크를 가로질러
    - → 중간 릴레이 노드에서 경로 지정을 가능하게 하기 위해
  - 단점
    - 프레임의 전체를 암호화할 수 없다
      - → 중간 릴레이 노드가 헤더를 읽어야 하기 때문
    - 네트워크 내에서 사용자 데이터는 링크간 암호화에 비해 안전하지만(밀성), 트래픽 패턴에 대한 보장은 없음(무결정, 가용성)
- 해결책? 둘 다 사용!

# 키 분배

- 다른 사람에게 키를 보여주지 않고 데이터를 교환하려는 두 집단에게 키를 전달하는 수단
- 두 집단 (A 와 B) can 다음에 의해 키 획득:



# 키 분배

- 방법 1, 2
  - 링크 암호화의 경우에는 좋은 솔루션임
  - 종단 간 암호화에 적용 불가
  - 즉, 네트워크 상에서는 적용 불가
    - 네트워크에선 여러 릴레이 (라우터, 스위치, 혹은 다른 단말)를 거쳐 전달되기 때문
- 방법 3
  - 링크 암호화/종단 간 암호화에 적용 가능
  - 한 키에 대한 접근이 성공한다면 뒤이은 모든 키들 노출
- 방법 4
  - 보안을 위해 일반적으로 적용

# 키 분배

1. Host sends packet requesting connection.
2. Security service buffers packet; asks KDC for session key.
3. KDC distributes session key to both hosts.
4. Buffered packet transmitted.

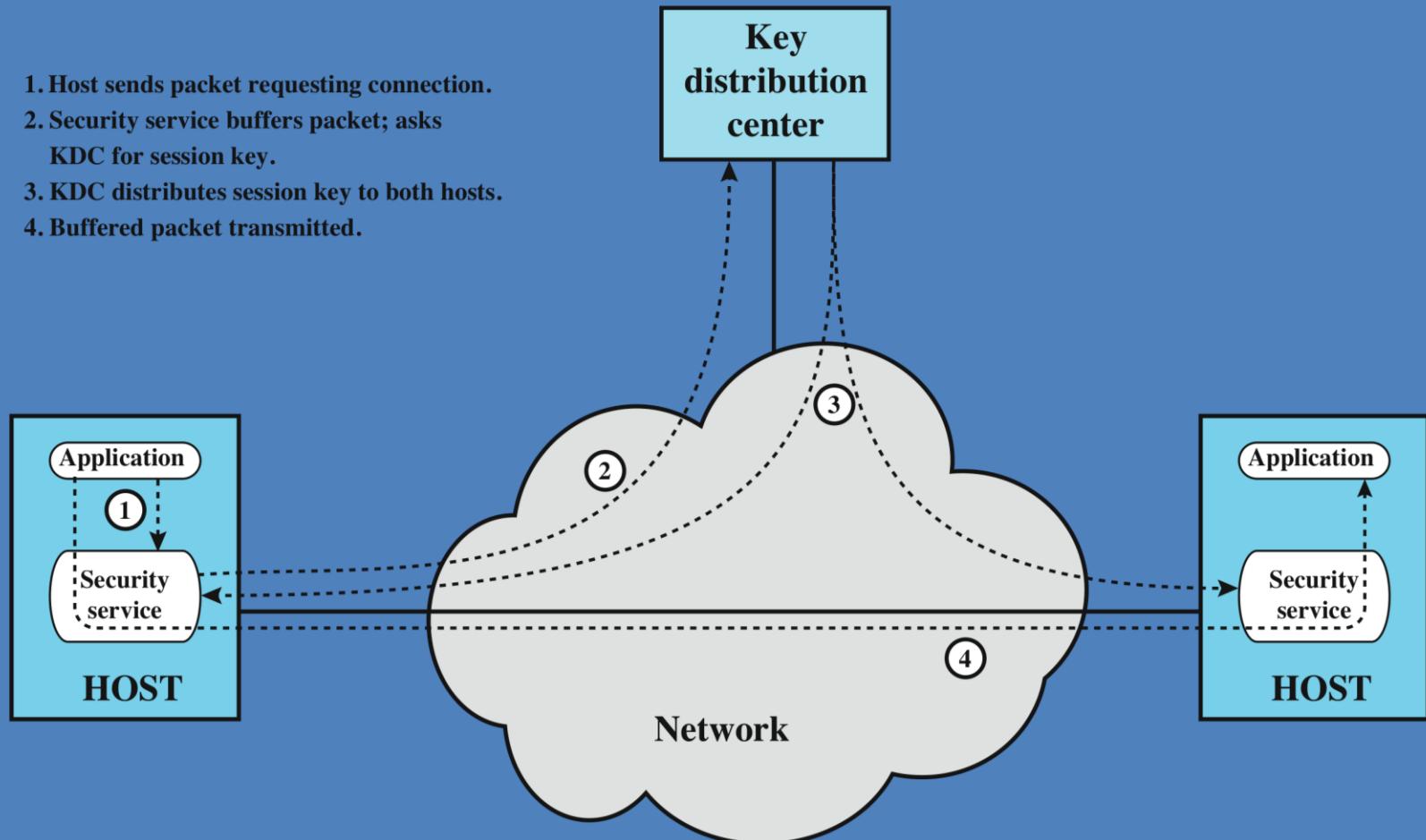


Figure 20.10 Automatic Key Distribution for Connection-Oriented Protocol

# 키 분배

- 두 종류의 키를 식별
  - 세션 키
    - 데이터 전송을 위한 일회용 키
    - 임시적으로 세션이 연결된 동안 생성되었다가 세션이 종료되면 파괴됨
  - 영구키
    - 세션 키를 분배할 목적으로 두 개체간 사용되는 키
    - 공인인증 혹은 서버에서 관리
- 구성 요소
  - 키 분배 센터
    - 두 노드 간 통신을 허용할 것인지 결정,
    - 허용된다면 데이터 전송을 위한 일회용 세션 키를 제공
  - 보안서비스 모듈
    - 단말에 위치, 하나의 프로토콜 계층으로 구성

# 요약

- 대칭 암호화 원리
  - 암호화
  - 복호화
  - Feistel 암호 구조
- DES
  - 트리플 DES
- AES
  - 알고리즘 상세내용
- 키 분배
- 스트림 암호와 RC4
  - 스트림 암호 구조
  - RC4 알고리즘
- 블록 암호 작동 모드
  - ECB(electronic codebook) 모드
  - CBC(cipher block chaining) 모드
  - CFB(cipher feedback) 모드
  - CRT(counter) 모드
- 대칭 암호화 장치 위치