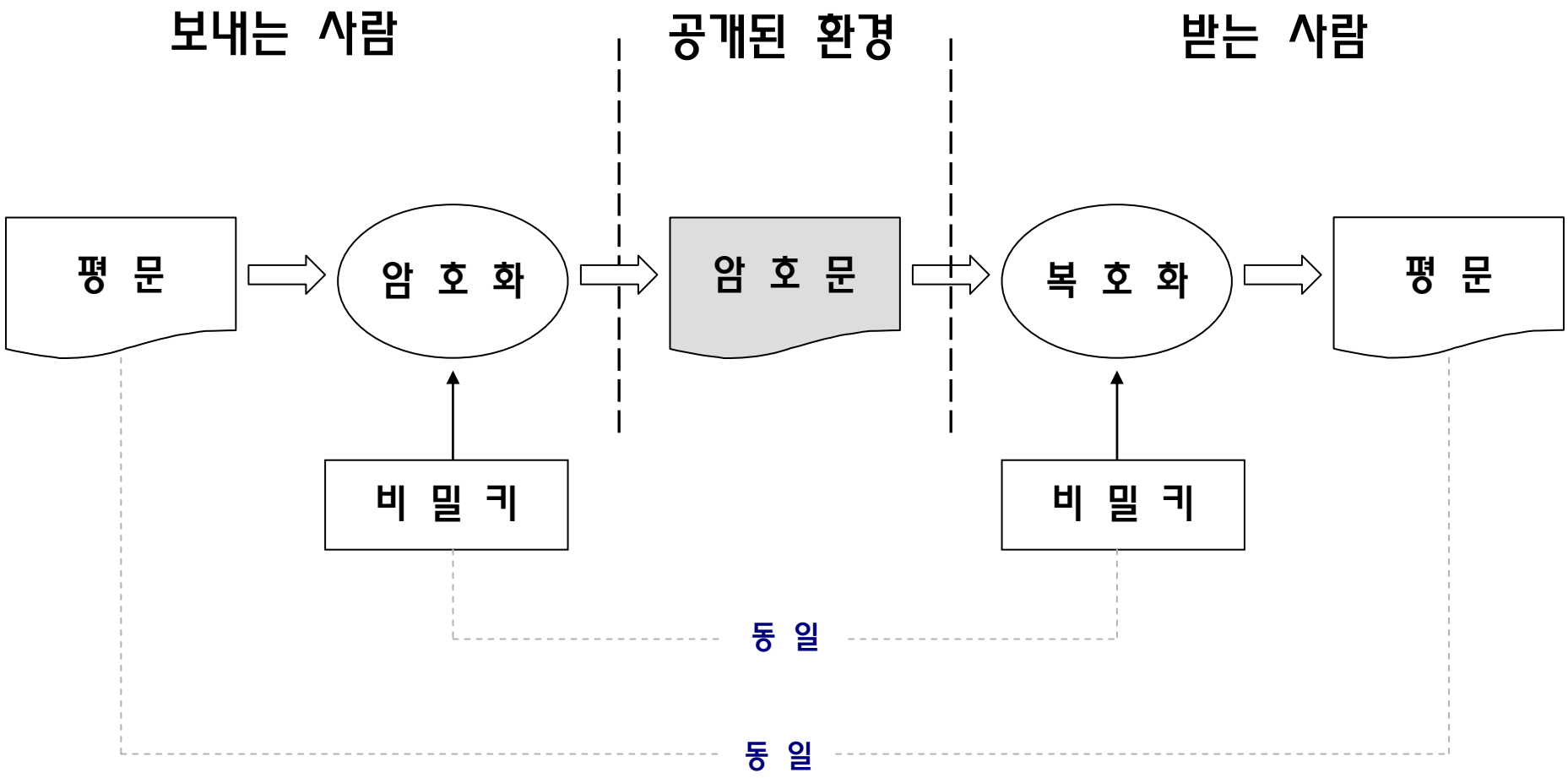
제 3 장 대칭키 암호 알고리즘

컴퓨터 시스템 보안

목 차

- DES
- DES 알고리즘의 변형
- 기타 블록 암호
- 블록 암호 모드
- OpenSSL을 이용한 블록 암호화 예제

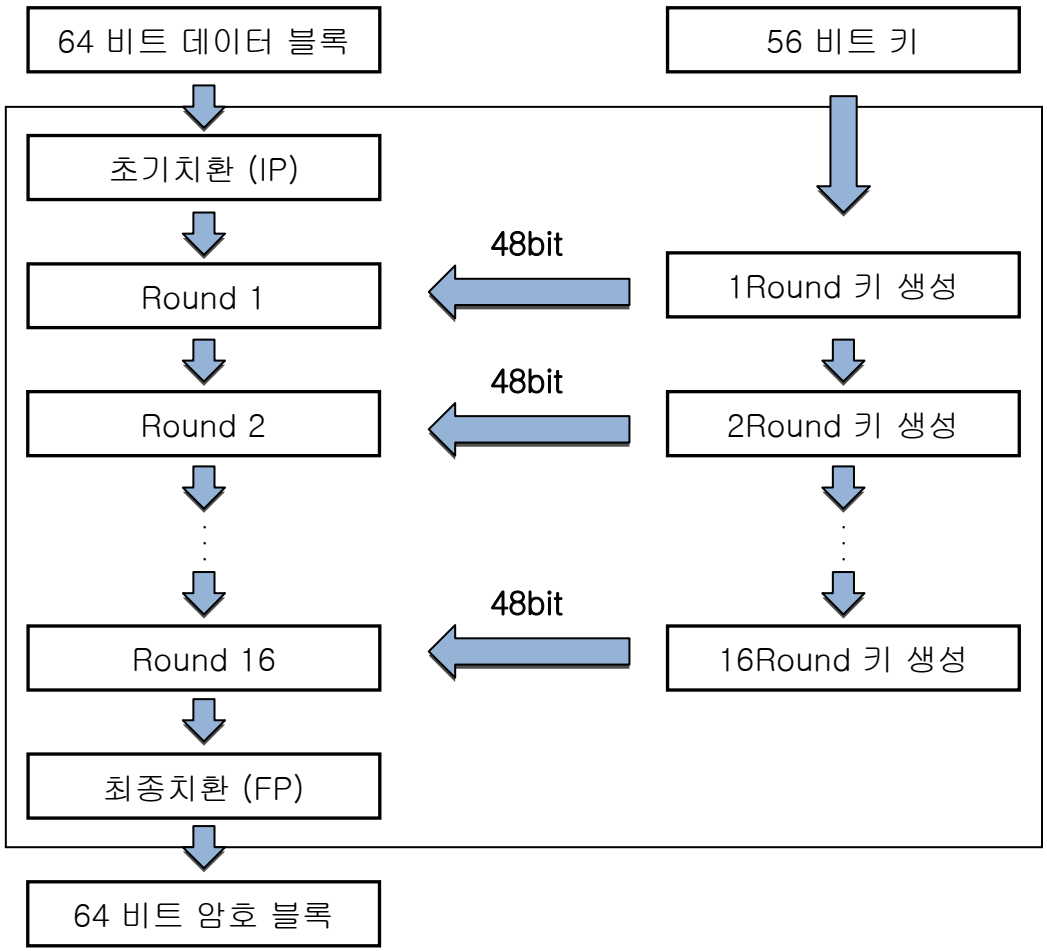
대칭키 암호 알고리즘의 구조



DES의 역사 및 사양

- 대표적인 대칭키/비밀키 암호 알고리즘
- 1972년 미국 NBS에서 암호 알고리즘 공모
- 1974년 IBM의 Lucifer 알고리즘 선정
- 1976년 미 연방정부에서 사용 결정
- 64 비트 크기의 평문을 암호화하여 동일 크기의 암호문 생성
- 56 비트 크기의 비밀키를 사용
- 16회의 라운드 적용
- 각 라운드마다 서브비밀키를 사용
- 서브비밀키는 56 비트 비밀키로부터 생성

DES 알고리즘의 구조


64 비트 데이터 블록
56 비트 키
초기치환 (IP)
48bit
Round 1
1Round 키 생성
48bit
Round 2
2Round 키 생성
48bit
Round 16
16Round 키 생성
최종치환 (FP)
64 비트 암호 블록

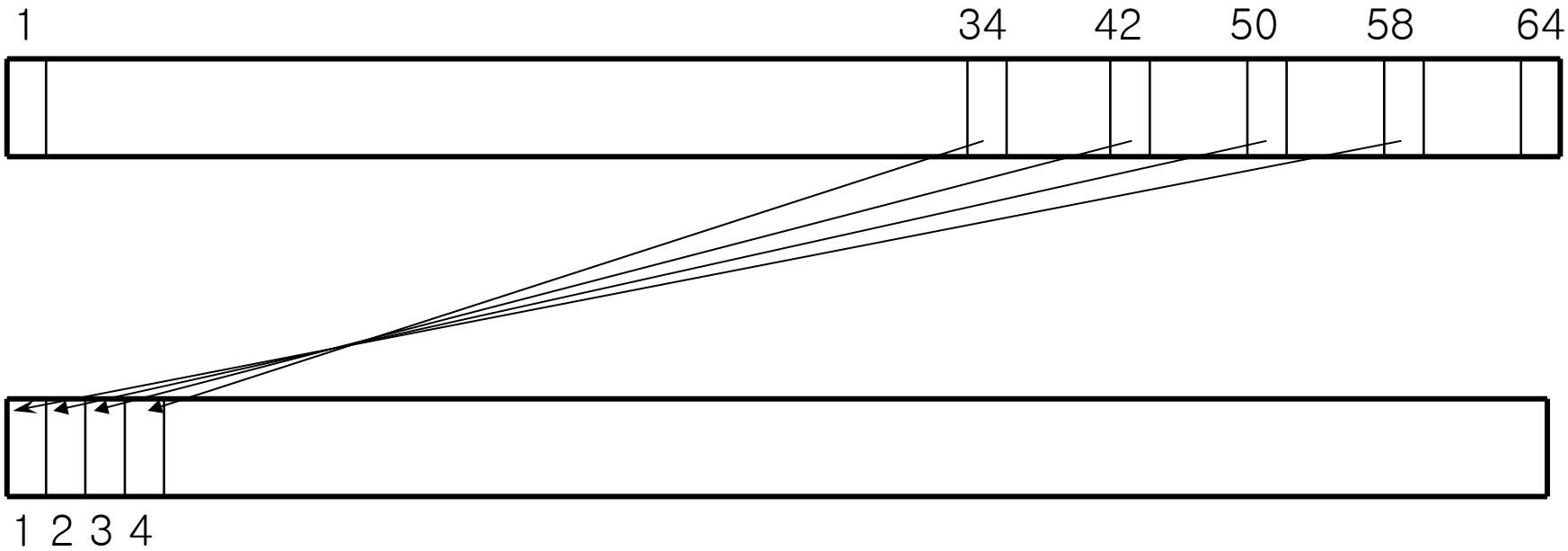
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

IP (초기치환)

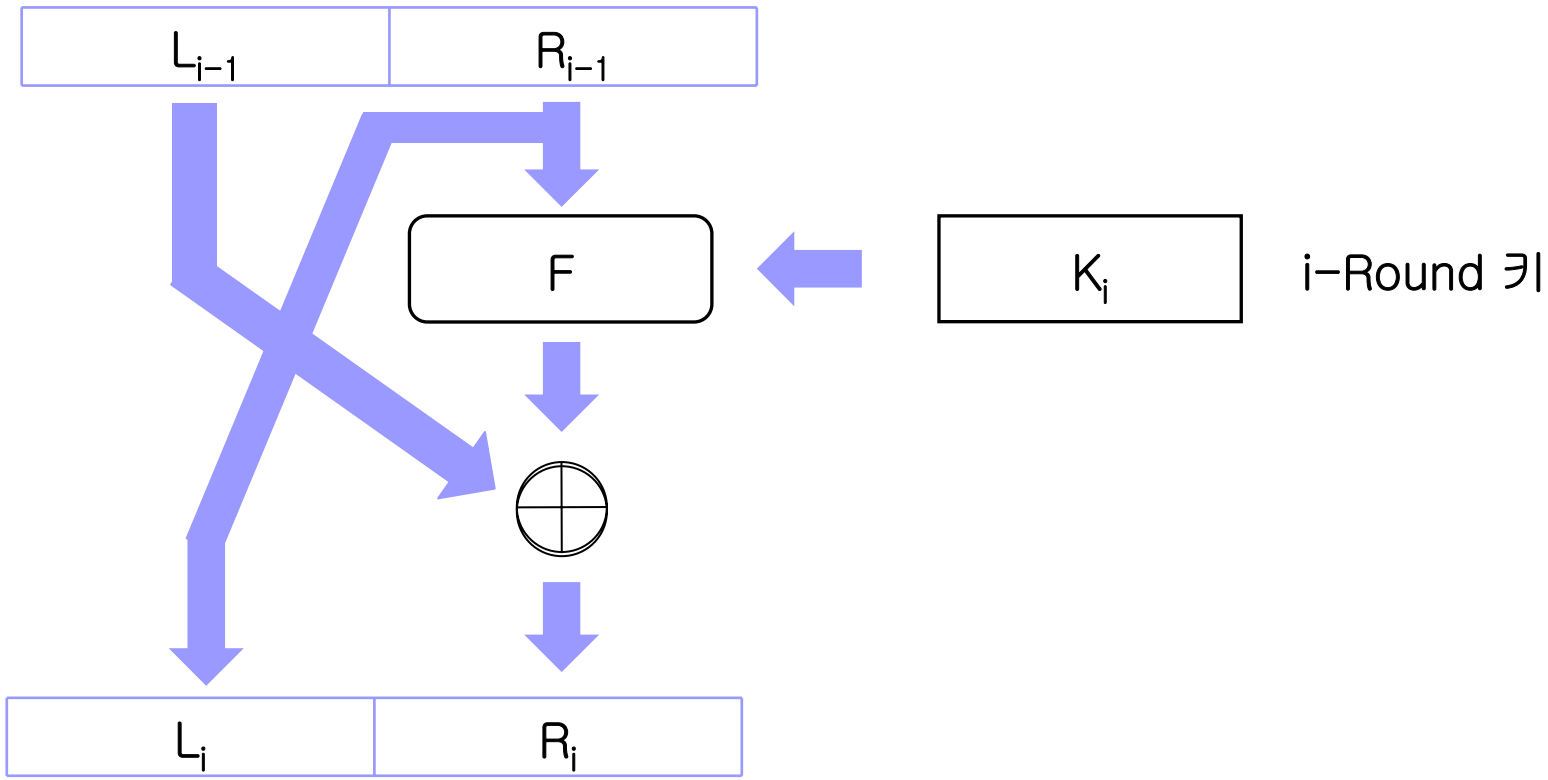
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

FP (최종치환)

치환 테이블의 의미

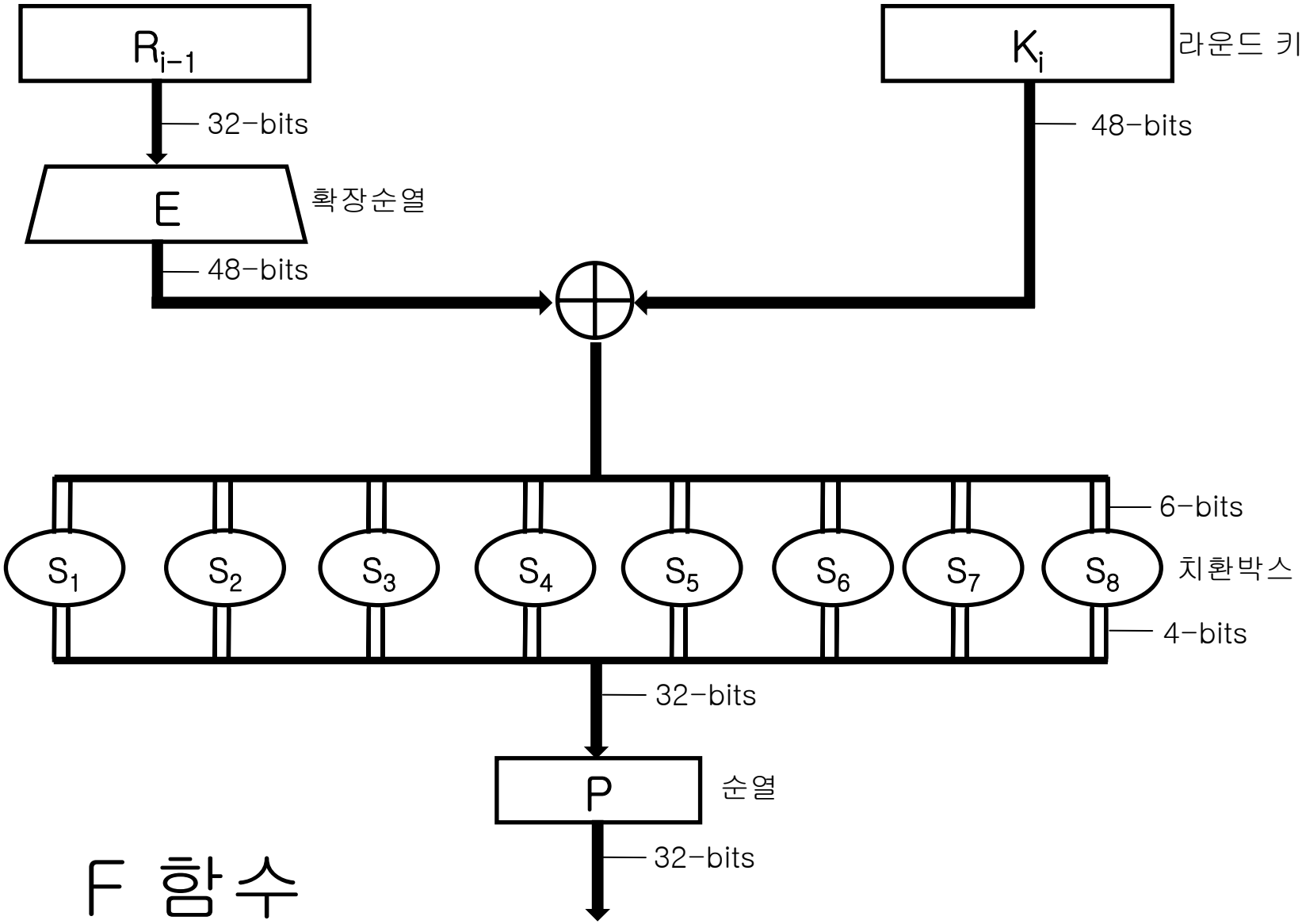


라운드 구조



라운드 구조의 수식과 역 수식

- i-번째 라운드
  - $L_i = R_{i-1}$
  - $R_i = F(R_{i-1}, K_i) \oplus L_{i-1}$
- i-번째 라운드의 역 수식
  - $R_{i-1} = L_i$
  - $L_{i-1} = R_i \oplus F(L_i, K_i)$


R_{i-1}
32-bits
E
확장순열
48-bits
K_i
라운드 키
48-bits
6-bits
S_1
S_2
S_3
S_4
S_5
S_6
S_7
S_8
치환박스
4-bits
32-bits
P
순열
32-bits

F 함수

확장순열 (E)

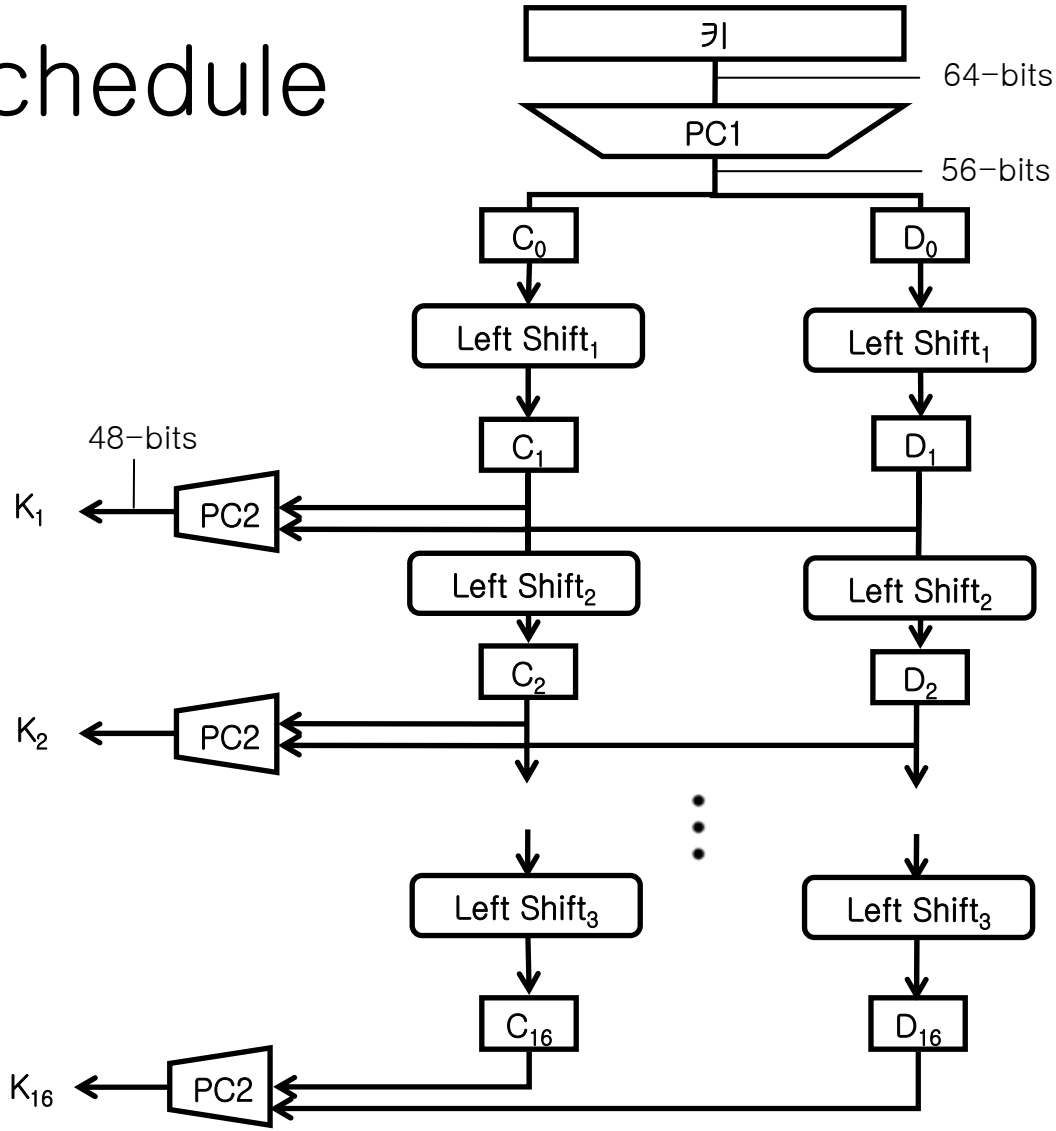
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

S-box (S1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	$S_2S_3S_4S_5$
00	14	14	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
01	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	
10	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
11	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	

S_1S_6

Key schedule



Round 별 Shift 개수

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

PC1과 PC2

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

PC1

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

PC2

DES의 안전성

- 키의 개수 : 2^{56} 개 (약 7.2×10^{16})
- 브룻포스 공격
  - 하나의 키 시도에 1μsec 소요 → 평균 1100년
  - 전용회로 12,000개 사용 → 평균 1달
  - 그리드 컴퓨팅, 클라우드 컴퓨팅 등장
- 차분 해독, 선형 해독
  - 충분한 평문, 암호문 쌍 필요

DES 알고리즘의 변형

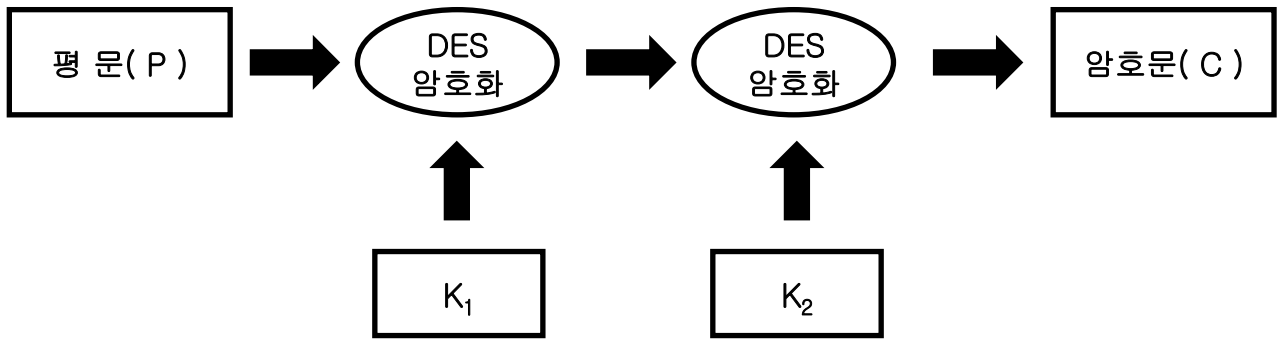
- 이중 DES (2DES)
- 삼중 DES (3DES, triple DES)

2DES

- 두 개의 비밀키 K_1, K_2 사용
- 암호화방식

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

- C: 암호문, E: DES 암호화, P: 평문



2DES에 대한 공격

- 평문 P와 그에 대한 암호문 C를 구함

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

- 양변을 복호화시킴

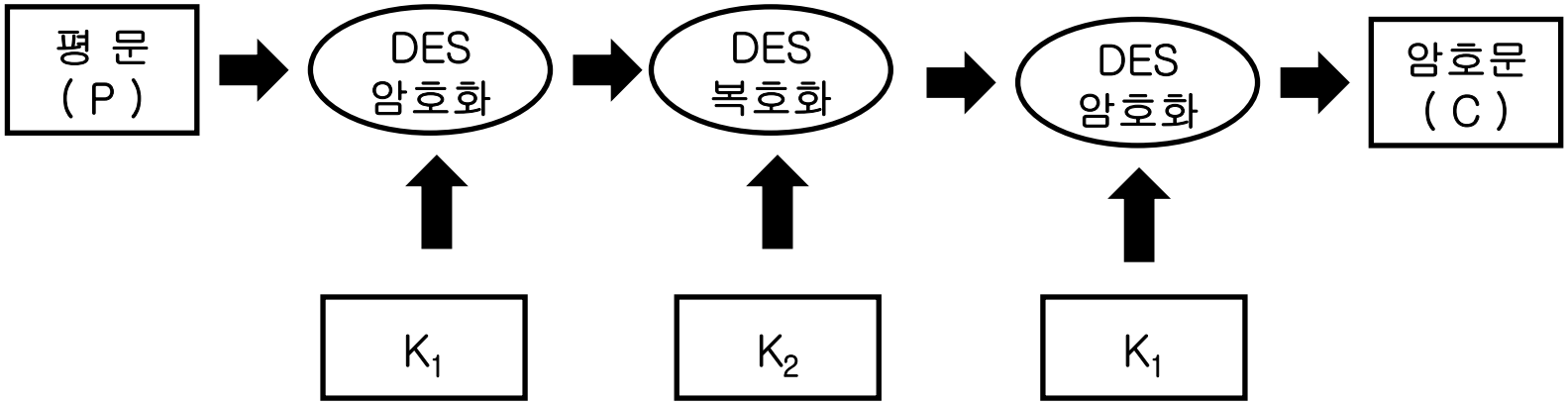
$$D(C, K_2) = E(P, K_1)$$

- 가능한 모든 키 K_2에 대해 왼변을 구하여 (2^{56}) 배열에 저장
- 오른변에 K_1을 하나씩 적용해 배열에서 같은 값을 찾음 (평균 2^{55})
- 필요한 기억공간은 약 10^8 Gbytes

3DES

- 삼중DES, triple DES
- 3개의 키를 쓰는 방법 (EEE 방식)

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



- 키의 크기 : 168 = 56 x 3 비트
- 2개의 키를 사용하는 경우에 비해 공격 시간이 월등히 많이 걸리지 않는다.

3DES (계속)

- 2개의 키를 쓰는 방법 (EDE 방식)

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

- 키의 크기 : 112 = 56 x 2 비트
- 상대방이 단일 DES만 제공하는 경우
  - 키를 1개만 사용하면 상대방과 호환성 유지

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

기타 블록 암호

- IDEA
- Blowfish
- AES

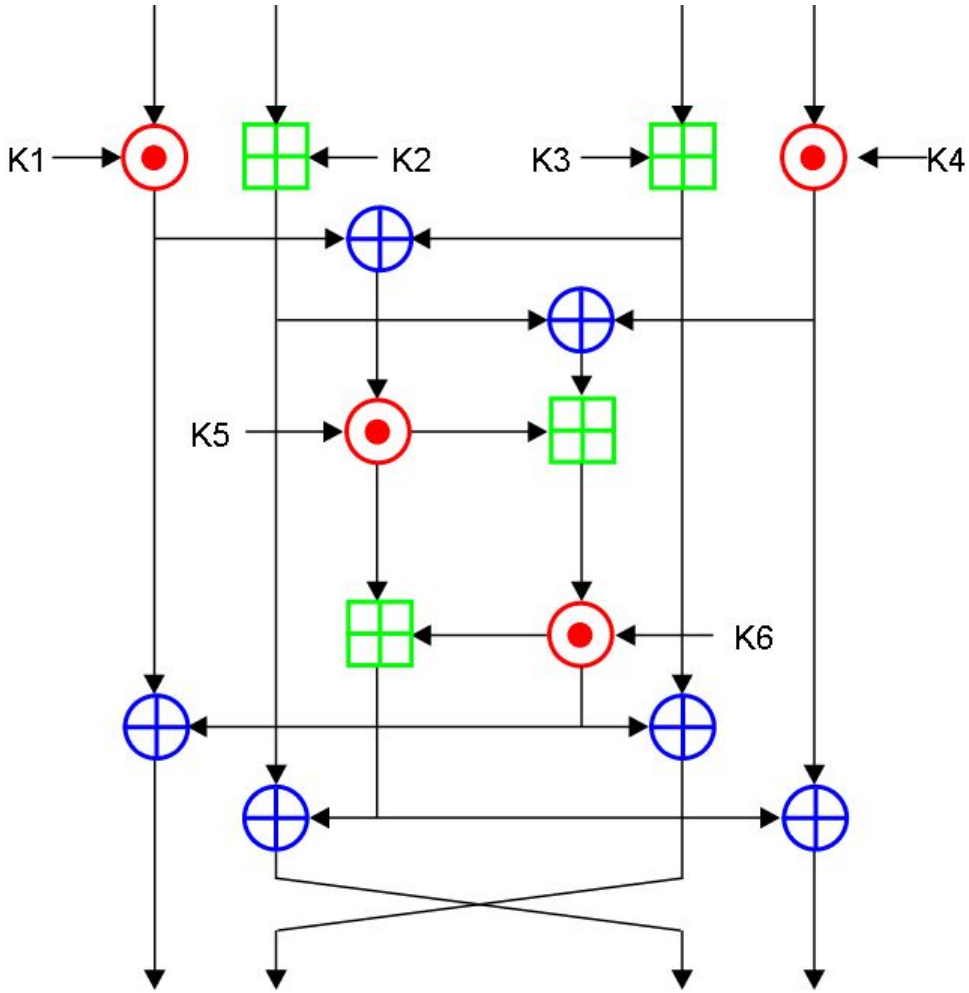
IDEA

- International Data Encryption Algorithm
- Zuejia Lai와 James Massey에 의해 1990년에 PES (Proposed Encryption Standard)로 개발됨
- 1992년에 IDEA로 개명되어 공개됨
- 특허로 인해 상용은 로열티를 지급해야 함, 비상용은 자유로운 사용 가능
- 64 비트 크기의 평문을 암호문으로 바꿈
- 128 비트 크기의 비밀키를 사용
- 8.5회의 라운드 적용

IDEA (계속)

- 8.5회의 라운드
- 각 라운드마다 6개의 16 비트 크기의 서브키 (sub-key) 사용
- 각 라운드에는 16 비트 입력 두 개를 한 개의 16 비트 출력으로 만드는 3 종류의 연산 사용
  - Bitwise xor : $z = x \oplus y$
  - Addition modulo 2^{16} : $z = (a + b) \% 2^{16}$
  - Modified multiplication : $z = (a * b) \% 2^{16} + 1$
    - 0이 입력이면 2^{16}으로 바꾸어 계산함

IDEA의 각 round 구조



from Wikipedia

Blowfish

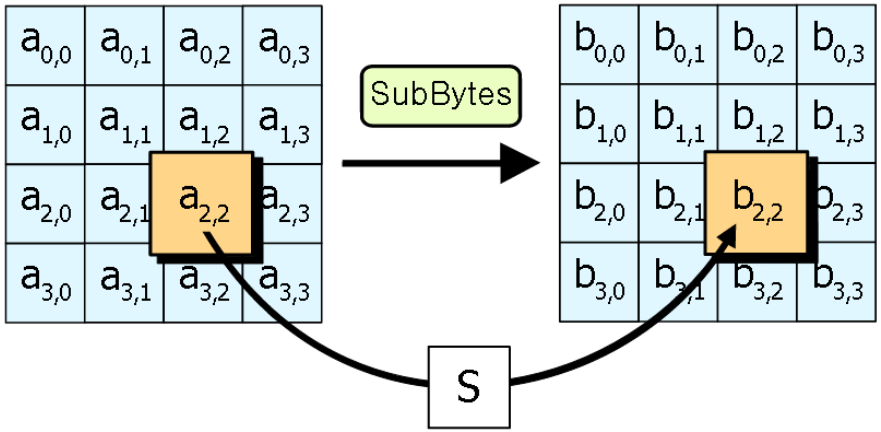
- 1993년에 Bruce Schneier가 개발
- 퍼블릭 도메인 (public domain)으로 사용
- 64 비트 평문을 32 ~ 448 비트 크기의 비밀키로 암호화
- 16회의 라운드 수행
- 각 라운드마다 키 관련 순열과 데이터 관련 치환 수행
- 32 비트 마이크로프로세서에 적합하도록 설계
- 5 Kbyte 이내의 메모리 사용

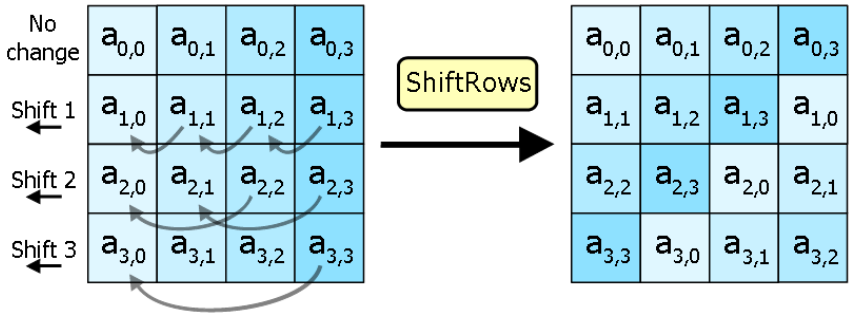
AES

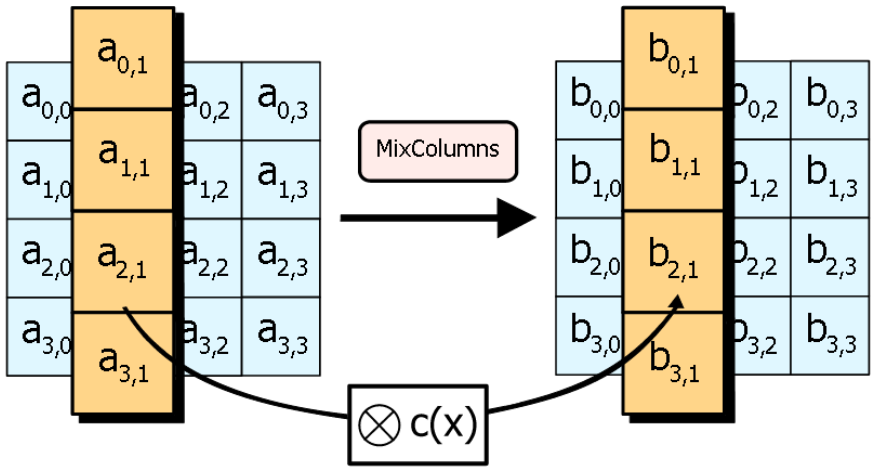
- Advanced Encryption Standard
- 2001년에 NIST에서 발표하여, 2002년 5월에 유효화된 미국 표준암호임
- 5개의 후보알고리즘들 중에서 Rijndael을 기반으로 설계
- 128 비트 크기의 평문을 128, 192, 또는 256 비트 크기의 비밀키로 암호화
- 각 키 크기에 대해 10, 12, 14회의 라운드 적용

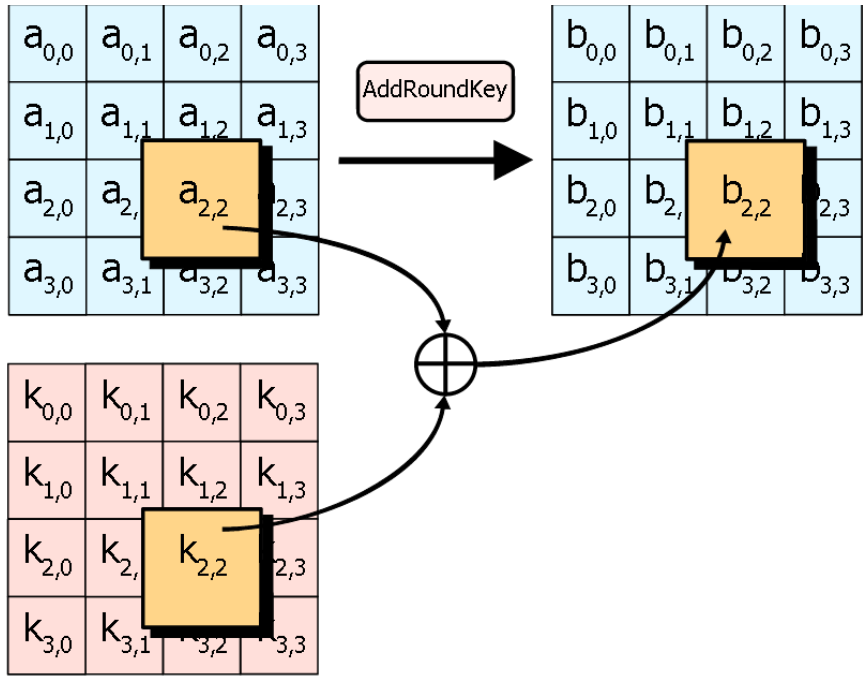
AES의 라운드

- 16 바이트의 입력을 16개의 1 바이트 단위로 잘라 4×4 행렬에 저장
- 다음의 4 단계 작업을 수행
  - SubBytes : 행렬의 각 원소를 치환
  - ShiftRows : 행렬의 각 행을 왼쪽으로 회전
  - Mix Columns : 행렬의 각 열에 대해 연산
  - AddRoundKey : 행렬의 각 요소를 키 행렬과 xor 연산


SubBytes
S


No change
Shift 1
Shift 2
Shift 3
ShiftRows


MixColumns
c(x)


AddRoundKey

블록 암호 모드

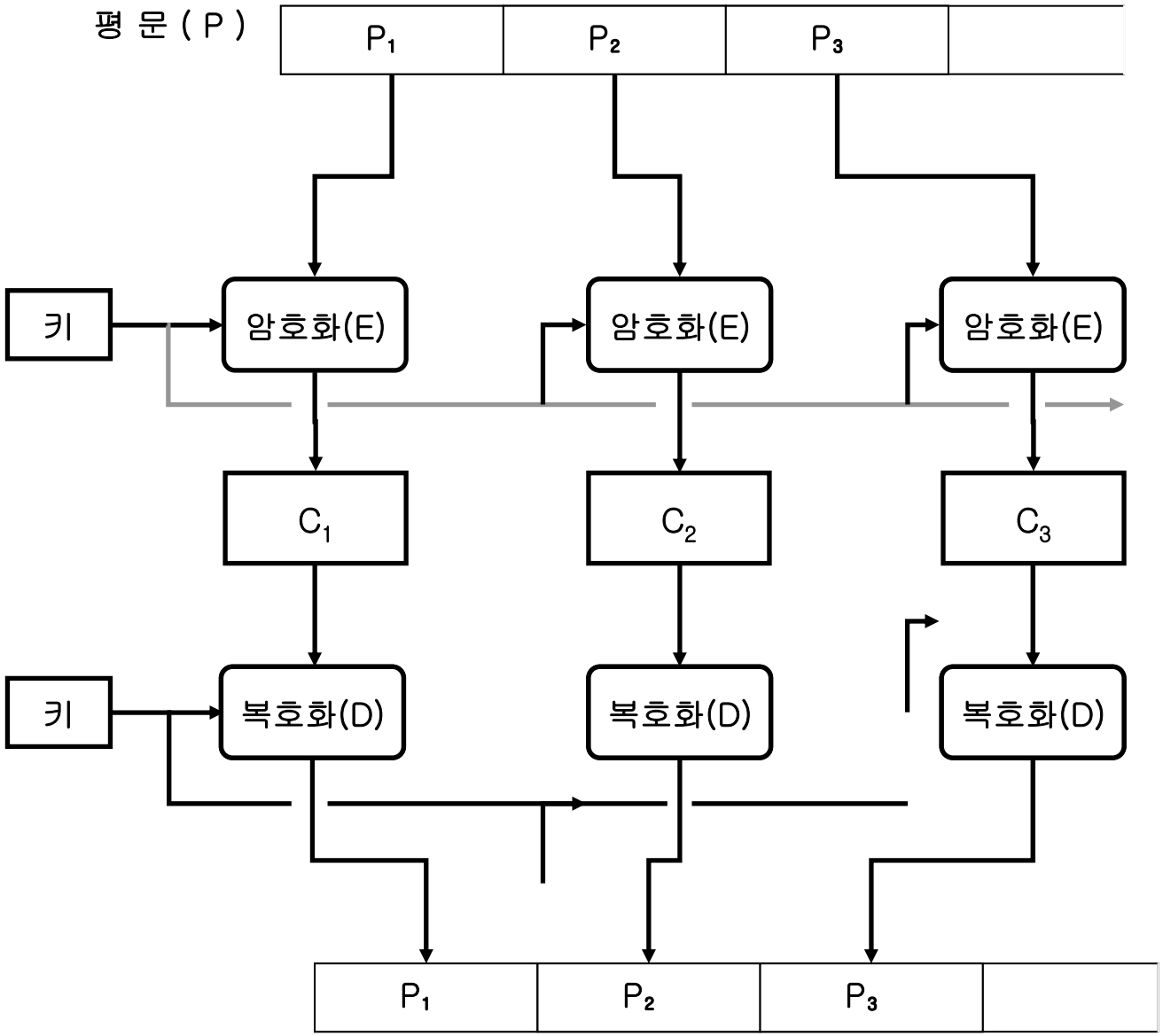
- 블록 암호 알고리즘들은 일정한 크기의 평문을 암호화함
- 평문 크기 < 암호 알고리즘의 입력 크기
  - 나머지를 패딩으로 채움
    - 패딩 : 0과 같은 값으로 블록을 채우는 작업
- 평문 크기 > 암호 알고리즘의 입력 크기
  - 평문을 암호 알고리즘의 입력 크기로 나눔
- 각 암호문은 동일한 비밀키를 사용함
- 블록 암호 모드는 임의 크기의 평문을 암호화하여 보내는 방법
  - 블록들 간의 관련이 있을 수 있음

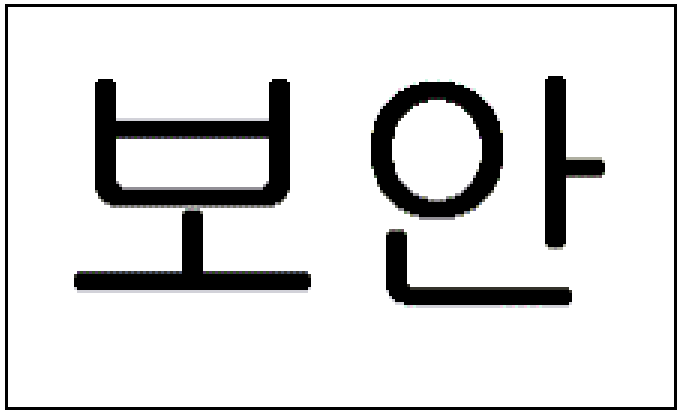
블록 암호 모드

- ECB mode
- CBC mode
- CFB mode
- OFB mode

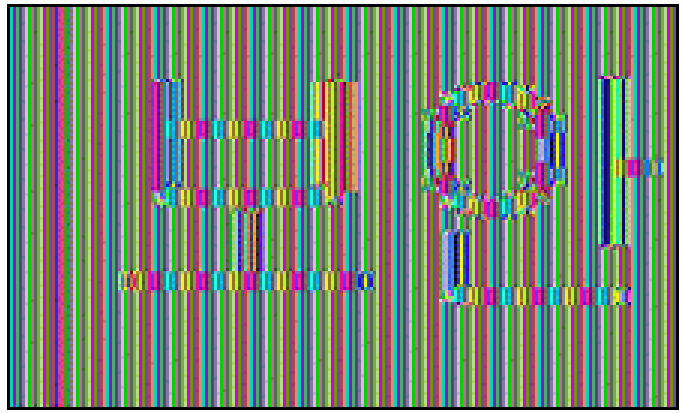
ECB mode

- 각 암호문 블록이 독립적임
- 가정
  - 평문이 P_1, P_2, P_3, … 으로 분할
  - 암호문이 C_1, C_2, C_3, … 으로 생성
  - 비밀키는 K
- 각 암호문은 $C_i = E(P_i, K)$과 같이 생성
- 동일한 평문은 동일한 암호문을 생성
  - 암호문으로부터 평문의 윤곽을 짐작할 수 있음
  - 오류가 확산되지는 않음


평 문 (P)
P1
P2
P3
키
암호화(E)
암호화(E)
암호화(E)
C1
C2
C3
키
복호화(D)
복호화(D)
복호화(D)
P1
P2
P3



(a)

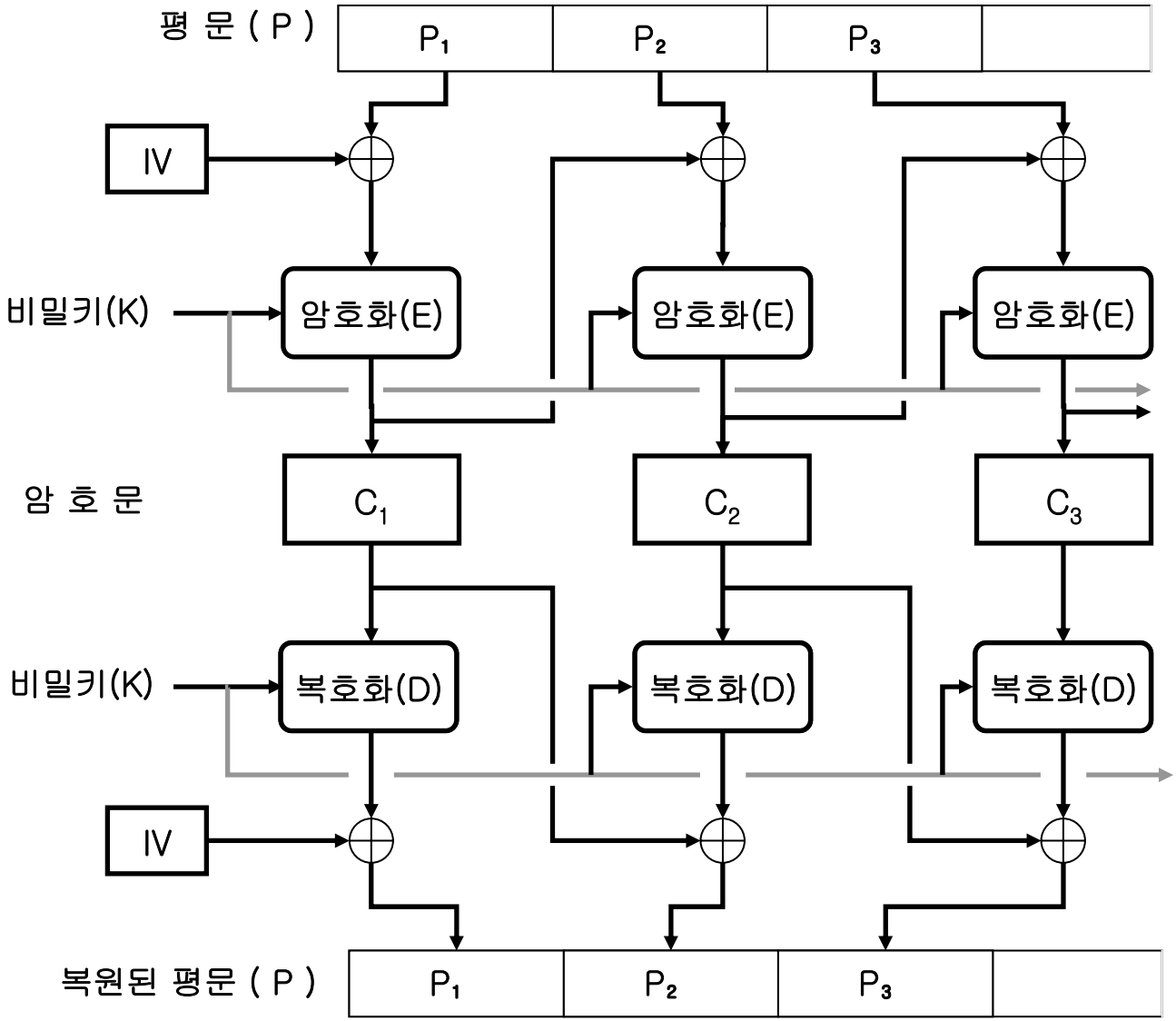


(b)

(c)

CBC 모드

- Cipher Block Chaining mode
- 암호화 : $C_i = E(P_i \oplus C_{i-1}, K)$
- 복호화 : $P_i = D(C_i, K) \oplus C_{i-1}$
  - 평문 : $P_1, P_2, P_3, \cdots$
  - 암호문 : $C_1, C_2, C_3, \cdots$
  - $C_0 = IV$
  - IV : 보내는 쪽과 받는 쪽이 모두 아는 값
    - 공개되지만 매번 다른 값이 바람직하다.

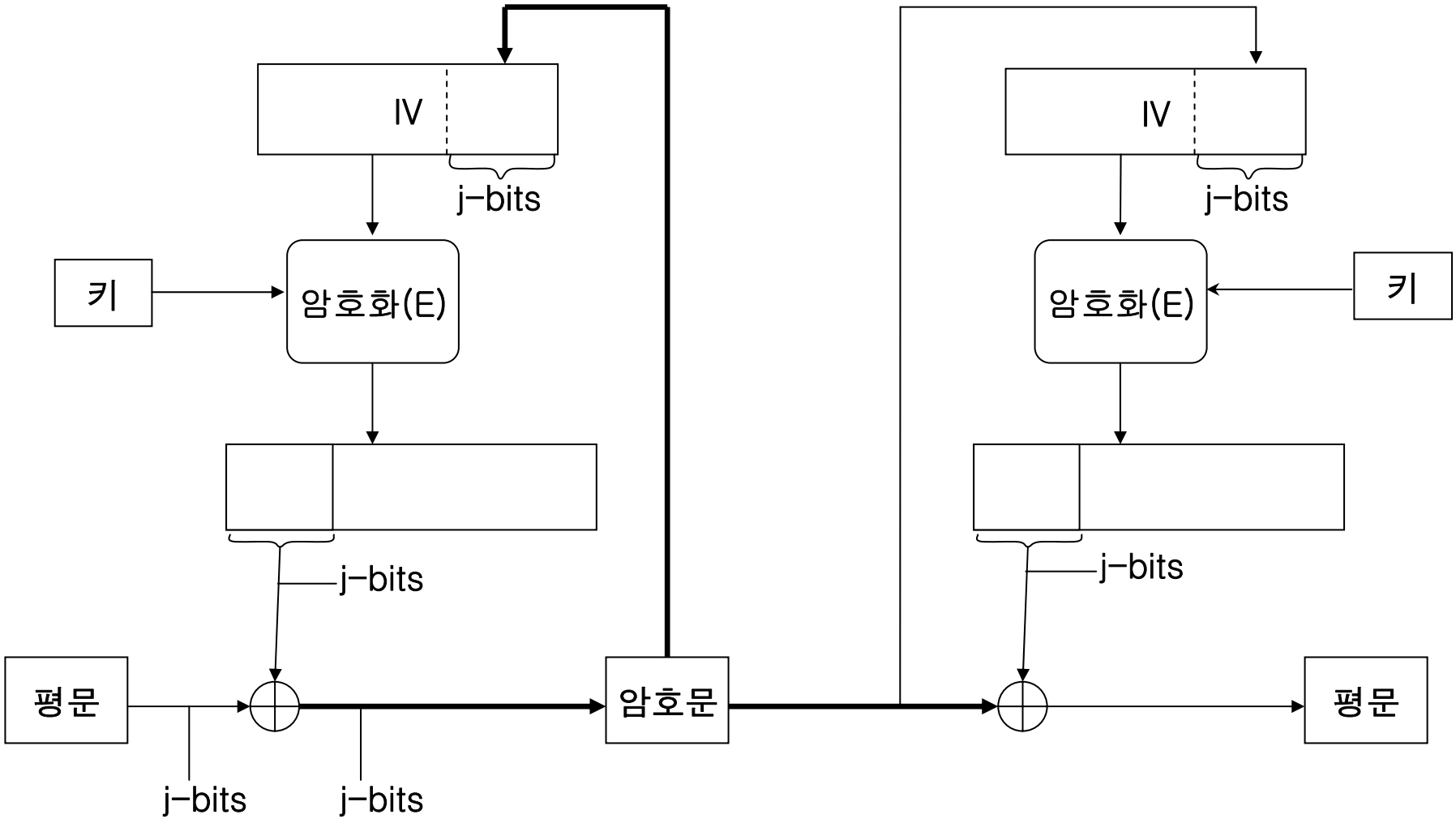

평 문 (P)
P1
P2
P3
IV
비밀키(K)
암호화(E)
암호화(E)
암호화(E)
암 호 문
C1
C2
C3
비밀키(K)
복호화(D)
복호화(D)
복호화(D)
IV
복원된 평문 (P)
P1
P2
P3

CBC 모드의 오류 확산

- 암호문 C_i가 전송 중 오류로 C'_i로 바뀐 경우
- 복호화된 $P'_i = D(C'_i, K) \oplus C_{i-1}$
- 다음 블록 $P'_{i+1} = D(C_{i+1}, K) \oplus C'_i$

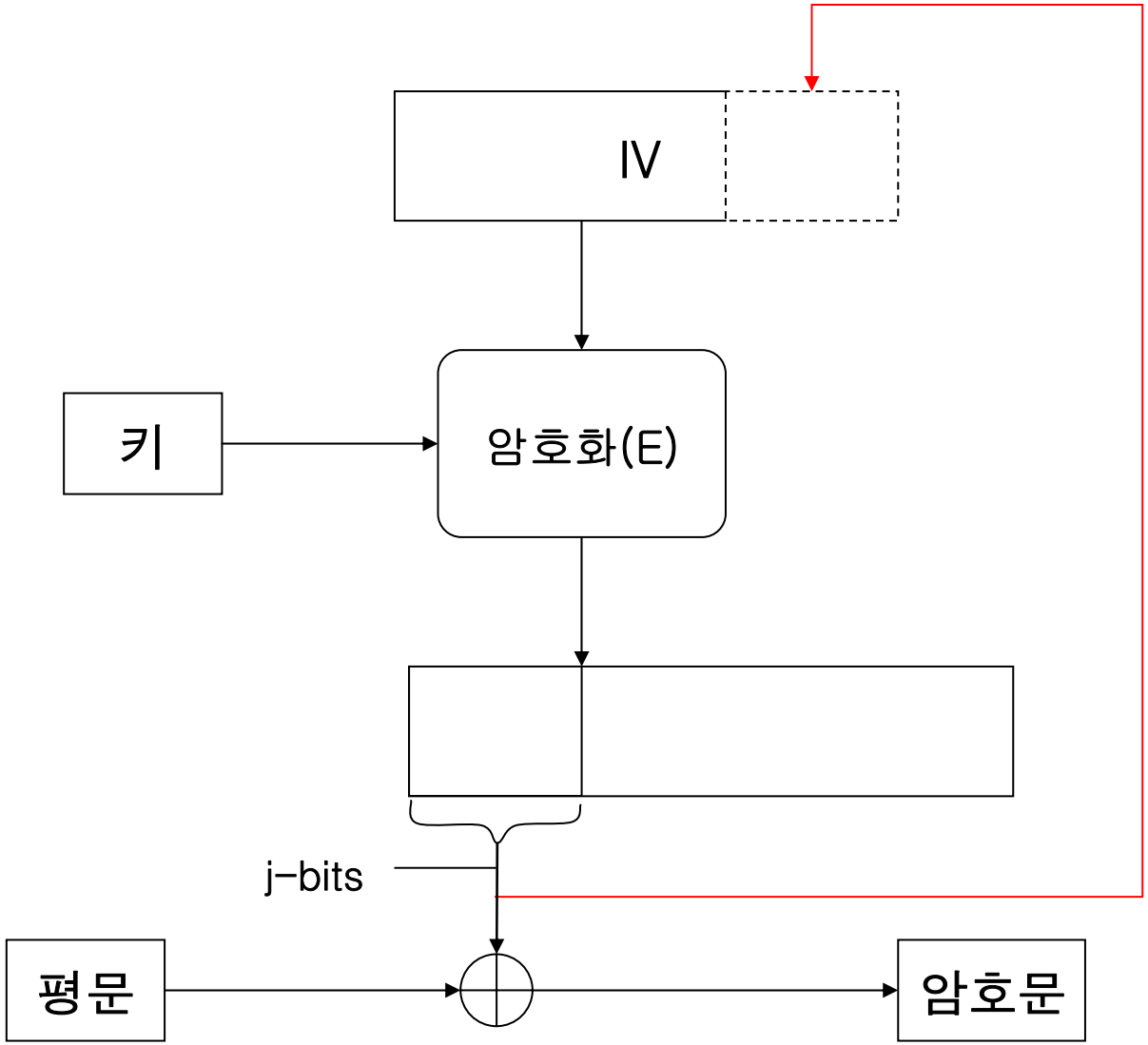
CFB mode

- Cipher FeedBack : 블록 암호를 이용하여 스트림 암호 구현
  - 난수 스트림 생성
- 암호문 $C = P \oplus L(E(IV, K), j)$
  - L(a,j) : 블록 a의 왼편 j 비트를 출력
- $IV_{new} = LS(IV, j) \mid C$
  - LS(a,j) : 블록 a를 왼편으로 j 비트 이동, 왼편의 j 비트는 0으로 채움
- 장점
  - 복호화를 지원하지 않는 암호화 알고리즘 사용 가능
- 단점
  - 오류의 확산
  - 오류 암호문은 IV에 포함되어 없어질 때까지 잘못된 복호화 수행


IV
j-bits
키
암호화(E)
j-bits
평문
j-bits
j-bits
암호문
IV
j-bits
암호화(E)
키
j-bits
평문

OFB mode

- Output FeedBack mode
- 암호문 대신 난수를 다음 난수 생성에 사용
  - IV에 xor 처리된 암호문 대신 생성된 난수가 들어감
- 암호문 $C = P \oplus L(E(IV, K), j)$
- $IV_{new} = LS(IV, j) \mid L(E(IV, K), j)$


IV
키
암호화(E)
j-bits
평문
암호문

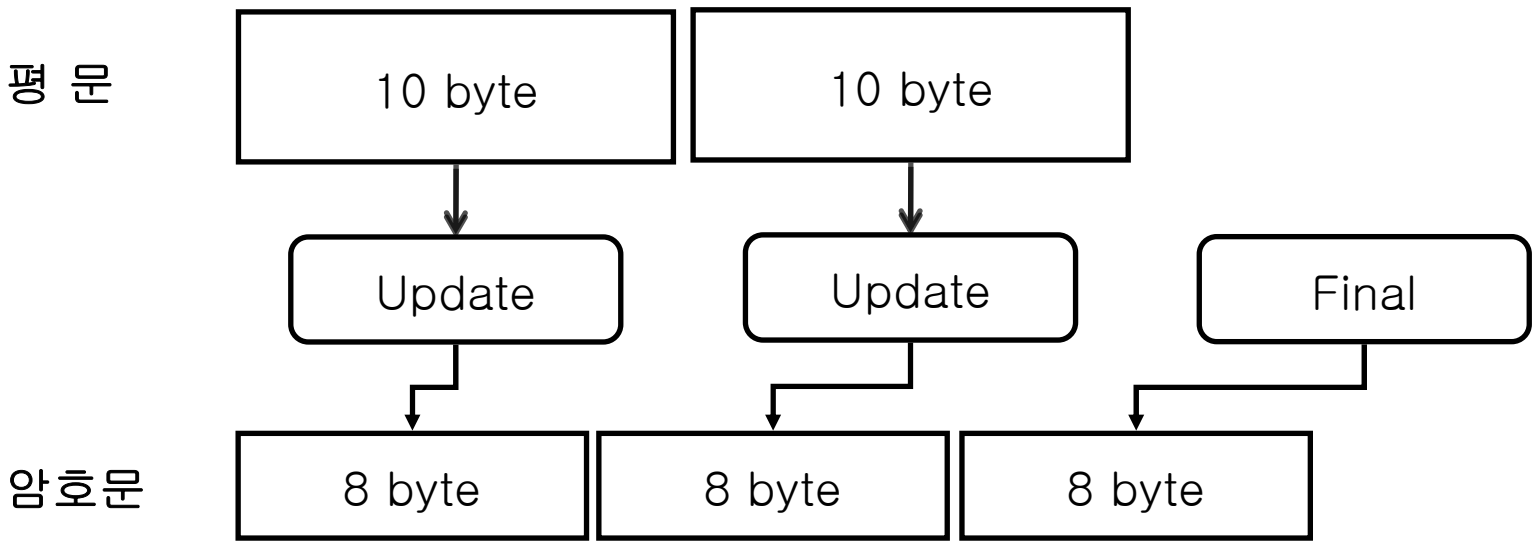
OpenSSL을 이용한 암·복호화

- OpenSSL 암호화 개요
  - 8 종류의 암호 알고리즘 (11 항목)
  - 4 종류의 암호 모드 제공
  - EVP 라이브러리를 통해 임의의 크기 메시지를 암호화
  - CTX : 암호화 수행 자료 구조
  - 암호화 수행 함수
    - EVP_CIPHER_CTX_init,
    - EVP_CipherInit_ex
    - EVP_CipherUpdate
    - EVP_CipherFinal_ex
    - EVP_CIPHER_CTX_cleanup

대칭키 암호화 관련 함수

- EVP_CIPHER_CTX_init : CTX 초기화
- EVP_CipherInit_ex : 암호화 알고리즘, 암호화 모드, 대칭키, IV 입력
- EVP_CipherUpdate : 블록 암호화 또는 복호화 수행
  - 평문 메시지의 크기에 따라 여러 번 호출되는 함수
- EVP_CipherFinal_ex : 암호화 또는 복호화 작업 마무리
  - CTX에 남은 암호문을 출력하는 부분이 포함됨
- EVP_CIPHER_CTX_cleanup : CTX 정리 및 할당한 메모리 복귀

EVP_CipherFinal_ex



EVP_CipherInit_ex

- int EVP_CipherInit_ex(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *type, ENGINE *impl, unsigned char *key, unsigned char *iv, int enc);
  - ctx : 암호화 context
  - type : 암호화 알고리즘 함수
  - impl : 엔진
  - key : 비밀키
  - iv : IV
  - enc : 암호화 또는 복호화 여부 선택, <암호알고리즘>_<ENCRYPT 혹은 DECRYPT>

AES_CBC 암호화 예제 (일부분)

```
EVP_CIPHER_CTX_init(&ctx);
EVP_CipherInit_ex(&ctx, EVP_aes_128_cbc(), NULL, key, iv,
                        AES_ENCRYPT);
while ((in_len = fread(plainbuff, 1, MAXBUFF, ptf)) > 0) {
    ret = EVP_CipherUpdate(&ctx, cipherbuff, &out_len,
                                plainbuff, in_len);
    fwrite(cipherbuff, 1, out_len, ctf);
}
fclose(ptf);
ret = EVP_CipherFinal_ex(&ctx, cipherbuff, &out_len);
fwrite(cipherbuff, 1, out_len, ctf);  fclose(ctf);
EVP_CIPHER_CTX_cleanup(&ctx);
```

프로젝트 P2

- 난이도 : 높음
  - OpenSSL이 제공하는 해시 함수를 OpenSSL 홈페이지의 매뉴얼을 참조하여 사용하여야 함
- 내용
  - 제 1장의 프로젝트에서 구현했던 채팅 프로그램에 암호 알고리즘과 해시 알고리즘을 채용한다.

- 제 1 장의 프로젝트 P1a, P1b, 또는 P1c로부터 확장하며 어떤 것으로 시작했던지 상관없다.
- 비밀키는 서로가 이미 공유하고 있다고 가정한다. 하지만 그 비밀키는 OpenSSL 커멘드라인 명령어를 사용하여 생성한다.
- Sender는 메시지의 해시를 구하고 메시지와 해시 모두를 암호화하여 상대방 Receiver에게 보낸다. 메시지의 해시를 구하는 방법은 EVP_DigestInit() 함수를 참조한다. 필요한 자료구조가 있으면 보낼 메시지에 추가한다.
- Receiver는 암호문을 복호화하고 해시를 메시지의 해시와 비교하여 문제가 없을 경우 화면에 출력한다. 문제가 있으면 에러메시지를 출력하고 계속 진행할지 여부를 묻는다.
- 암호 알고리즘이나 해시 알고리즘은 어떤 것을 선택해도 상관없으나 암호 알고리즘은 CBC 모드를 채용해야 한다.