2026/04/30, 2026 보안 기초 세미나

암호학과 네트워크 보안

- 5장 현대 대칭-키 암호 소개,
- 8장 현대 대칭-키 암호를 이용한 암호화 기법 -

김 민 식 (ms6127@naver.com)

세종대학교 프로토콜공학연구실

목 차

- 보충
- 현대 대칭-키 암호 소개
- 현대 대칭-키 암호를 이용한 암호화 기법

암호 수학

- $GF(2^n)$ 체
  - 다항식(Polynomial) 표현 (1/2)
    - 컴퓨터 이용 곱셈
      - 나눗셈 연산 때문에 두 다항식을 곱하기 위한 프로그램 개발 시 효율성 문제가 존재하여 이를 해결하기 위해 줄여진 다항식에 x를 반복적으로 곱하는 알고리즘을 사용
      - 예제 4.22
        - $(x^8 + x^4 + x^3 + x + 1)$을 기약다항식으로 갖는 $GF(2^8)$ 체에서 $P_1 = (x^5 + x^2 + x)$와 $P_2 = (x^7 + x^4 + x^3 + x^2 + x)$의 곱셈

거듭제곱	연산	새로운 결과	약분
$x^0 \otimes P_2$		$x^7 + x^4 + x^3 + x^2 + x$	NO
$x^1 \otimes P_2$	$x \otimes (x^7 + x^4 + x^3 + x^2 + x)$	$x^5 + x^2 + x + 1$	YES
$x^2 \otimes P_2$	$x \otimes (x^5 + x^2 + x + 1)$	$x^6 + x^3 + x^2 + x$	NO
$x^3 \otimes P_2$	$x \otimes (x^6 + x^3 + x^2 + x)$	$x^7 + x^4 + x^3 + x^2$	NO
$x^4 \otimes P_2$	$x \otimes (x^7 + x^4 + x^3 + x^2)$	$x^5 + x + 1$	YES
$x^5 \otimes P_2$	$x \otimes (x^5 + x + 1)$	$x^6 + x^2 + x$	NO
$P_1 \times P_2 = (x^6 + x^2 + x) + (x^6 + x^3 + x^2 + x) + (x^5 + x^2 + x + 1) = x^5 + x^3 + x^2 + x + 1$			

- P_2에 x를 반복적으로 곱해 필요한 중간 값을 생성함
- 차수가 8 이상이면 기약다항식으로 약분함
- P_1에 포함된 항에 해당하는 결과만 XOR하여 최종 결과를 구함

암호 수학

- $GF(2^n)$ 체
  - 다항식(Polynomial) 표현 (2/2)
    - 컴퓨터 이용 곱셈
      - 비트 알고리즘
        1. 이전 결과의 최상위 비트가 0이라면 이전 결과를 왼쪽으로 한 비트 이동
        2. 이전 결과의 최상위 비트가 1이면
           - 왼쪽으로 한 비트 이동
           - 최상위 비트를 제외하고 모듈로 다항식과 XOR 연산
      - 예제 4.23
        - $P_1 = 000100110, P_2 = 10011110$, modulus=100011010이다(8비트 형식)

거듭제곱	왼쪽 이동 연산	XOR 연산
$x^0 \otimes P_2$		10011110
$x^1 \otimes P_2$	00111100	$(00111100) \oplus (00011010) = \mathbf{00100111}$
$x^2 \otimes P_2$	01001110	01001110
$x^3 \otimes P_2$	10011100	10011100
$x^4 \otimes P_2$	00111000	$(00111000) \oplus (00011010) = \mathbf{00100011}$
$x^5 \otimes P_2$	01000110	**01000110**
$P_1 \otimes P_2 = (\mathbf{00100111}) \oplus (\mathbf{00100011}) \oplus (\mathbf{01000110}) = 00101111$		

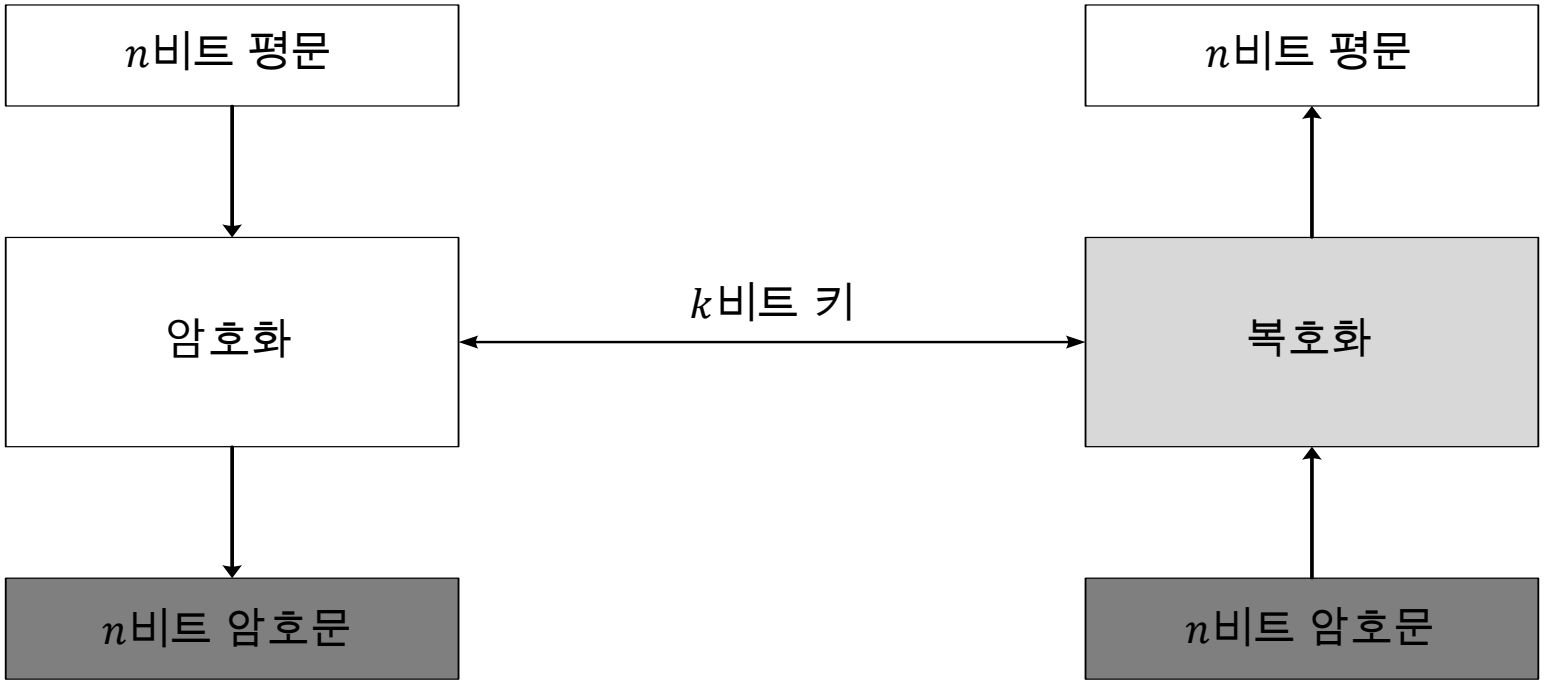
- x를 곱하는 것은 비트열을 왼쪽으로 한 칸 이동하는 것과 같음
- 왼쪽 이동 시 오른쪽 끝에는 0이 채워짐
- 맨 앞 비트가 1이면 x^8 항이 생기므로 $GF(2^8)$의 범위를 넘음
- 이 경우 앞의 1을 제외한 8비트에 modulus의 하위 8비트를 XOR하여 약분함
- P_1에서 1인 비트 위치에 해당하는 결과만 선택하여 XOR함

목 차

- 보충
- 현대 대칭-키 암호 소개
- 현대 대칭-키 암호를 이용한 암호화 기법

현대 대칭-키 암호 소개

- 대칭 키 현대 블록 암호
  - 정의
    - 키를 이용하여 n-비트 평문 블록을 암호화 하거나 n-비트 암호문 블록을 복호화 하는 방식



현대 대칭-키 암호 소개

- 대칭 키 현대 블록 암호

$|M|$:메시지 M의 길이
$|pad|$:블록 길이에 맞추기 위해 추가하는 패딩

  - 특징
    - 암호화와 복호화에 동일한 비밀키를 사용함
    - 메시지는 n-비트 블록 크기에 맞추어 길이를 조정하여 처리함
      - 메시지의 길이가 n-비트보다 짧으면, n-비트를 맞추기 위해 패딩(Padding) 을 추가함
      - 메시지의 길이가 n-비트보다 길면, 메시지를 n-비트 블록 단위로 분할하여 처리함
      - 일반적으로 블록의 길이는 64, 128, 256, 512비트가 사용됨
    - 예제 5.1

100개의 문자를 8비트 ASCII 코드를 사용하고 64비트 블록 암호로 암호화하기를 원한다면, 100개의 문자로 구성된 메시지는 몇 비트를 덧붙여야 하는가?

- $|M| + |pad| = 0 \; mod \; 64$
- $|pad| = -800 \; mod \; 64$
- $32 \; mod \; 64$

현대 대칭-키 암호 소개

- 대칭 키 현대 블록 암호 종류
  - 대치(Substitution)암호
    - 정의
      - 평문의 각 문자, 기호 또는 비트를 다른 문자, 기호 또는 비트 값으로 대체하여 암호문을 만드는 방식임
    - 특징
      - 문자의 위치나 순서는 그대로 유지되고, 값이 변경됨
      - 같은 평문 요소는 같은 규칙에 따라 같은 값으로 변경됨
      - 비트 단위로 보면 값 자체가 다른 값으로 바뀌므로 정보 표현이 달라짐

현대 대칭-키 암호 소개

- 대치암호 또는 전치암호
  - 전치(Transposition)암호
    - 정의
      - 평문의 문자,기호 또는 비트의 값은 바꾸지 않고 위치나 순서만 재배열하여 암호문을 만드는 방식임
    - 특징
      - 문자나 비트의 값은 그대로 유지되고, 위치나 순서만 변경됨
      - 평문과 암호문에서 이진수 0과 1의 개수는 동일하게 유지됨

현대 대칭-키 암호 소개

- 대치암호 또는 전치암호
  - 예제 5.2

64비트 블록 암호라 가정하자 (n=64) 만약 암호문에서 1의 개수가 10이라면, 공격자가 중간에 가로챈 암호문으로부터 평문을 도출해 내기 위해서 필요한 시행착오(trial-and-error test)는 다음 두 가지 경우 각각 몇 번인가?

a.암호가 대치암호로 설계된 경우
b.암호가 전치암호로 설계된 경우

a. 대치암호의 경우 1의 개수를 알 수 없으므로 공격자는 64비트 블록에 대하여 2^{64}번의 시행착오가 필요
b. 전치암호의 경우, 1의 개수가 유지되므로 공격자는 1이 10개인 평문만 조사 필요
   따라서 1에서 10개인 조합의 수는 $\frac{64!}{(10!)(54!)} = 151{,}473{,}214{,}816$

현대 대칭-키 암호 소개

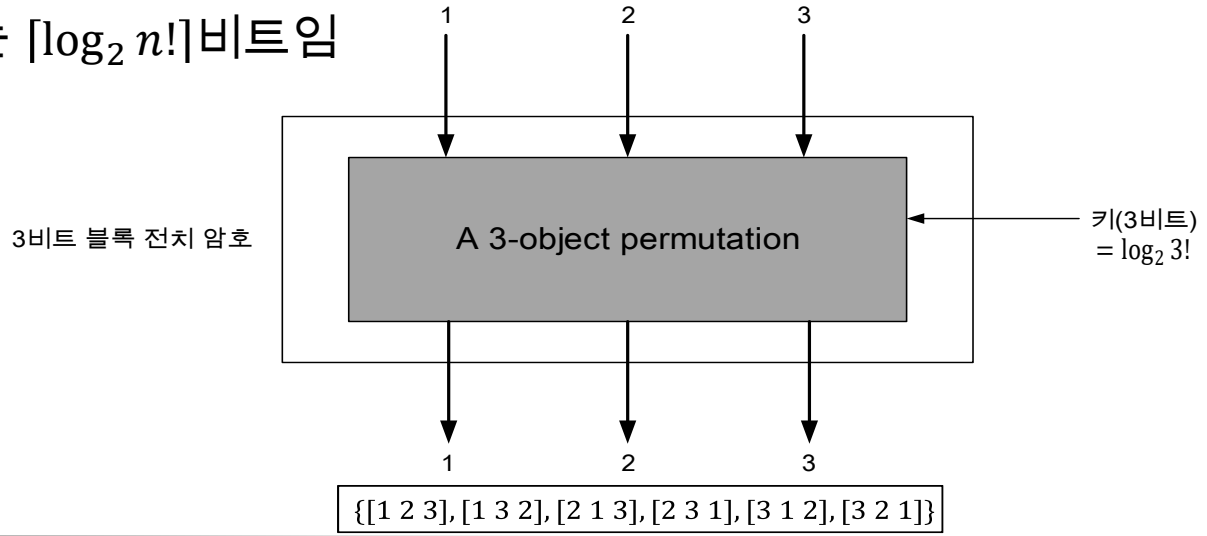
- 치환 군으로서의 블록 암호
  - 치환 군(Permutation Group)
    - 정의
      - 어떤 집합 위에서 정의된 치환들의 모임이 함수의 합성 연산에 대해 군(group)을 이루는 것
    - 특징
      - 전치 암호와 대치 암호는 모두 입력 집합 위의 일대일 대응이므로 치환에 해당함
      - 전체 길이 키 암호를 여러 번 합성하더라도 결과는 하나의 치환으로 표현됨

현대 대칭-키 암호 소개

- 치환 군으로서의 블록 암호
  - 전체 길이 키 암호(Full-size Key Cipher)(1/4)
    - 정의
      - 블록 암호에서 가능한 모든 치환(또는 순열)을 각각 하나의 키로 표현할 수 있을 만큼 키 길이가 충분한 암호 방식
    - 종류
      - 전체 길이 키 전치 블록 암호(Full-size Key Transposition Block Ciphers)
      - 전체 길이 키 대치 블록 암호(Full-size Key Substitution Block Ciphers)

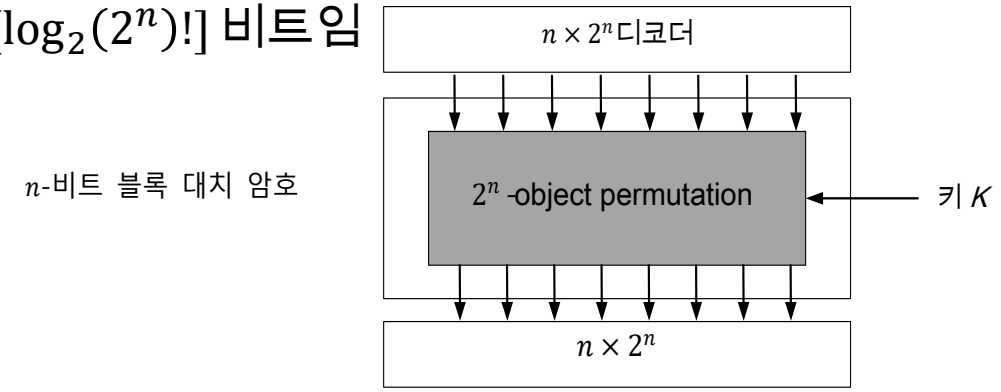
현대 대칭-키 암호 소개

- 치환 군으로서의 블록 암호
  - 전체 길이 키 암호(Full-size Key Cipher)(2/4)
    - 전체 길이 키 전치 블록 암호(Full-size Key Transposition Block Ciphers)
      - 정의
        - 가능한 모든 $n!$가지 전치를 키로 표현할 수 있는 전치 블록 암호
      - 특징
        - 각 비트의 값은 대체하지 않고 위치만 재배열함
        - n-비트 블록의 가능한 전치 경우의 수는 $n!$개임
        - 전치암호의 키 길이는 $[\log_2 n!]$비트임



현대 대칭-키 암호 소개

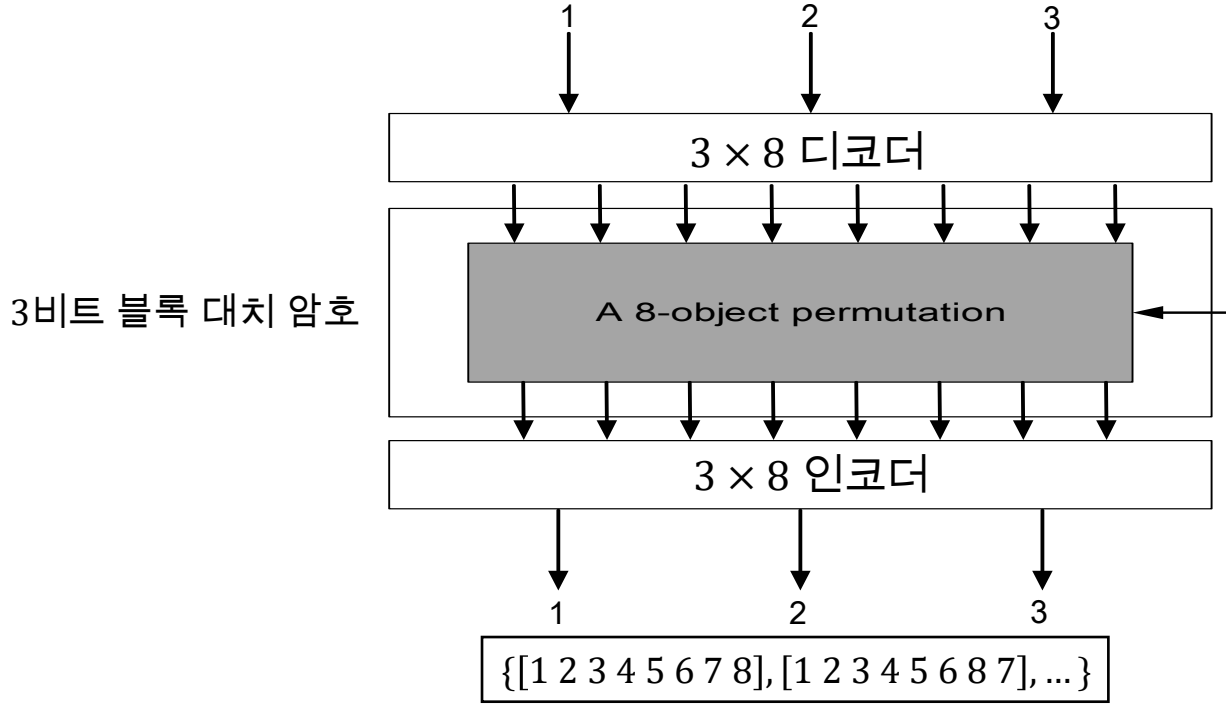
- 치환 군으로서의 블록 암호
  - 전체 길이 키 암호(Full-size Key Cipher)(3/4)
    - 전체 길이 키 대치 블록 암호(Full-size Key Substitution Block Ciphers)
      - 정의
        - n-비트 입력 블록 전체를 다른 n-비트 출력 블록으로 대체하는 블록 암호임
      - 특징
        - 입력 블록 값을 다른 블록 값으로 대체함
        - 단순한 비트 재배열만으로는 직접 표현할 수 없음
          - 입력 값을 디코딩(Decoding) 출력 값을 인코딩(Encoding) 하는 방식으로 모델화 가능함
        - $n-$비트 입력 값은 0부터 $2^n - 1$까지의 정수로 볼 수 있음
        - 대치 암호에 대하여 키 길이는 $[\log_2(2^n)!]$ 비트임



현대 대칭-키 암호 소개

- 치환 군으로서의 블록 암호
  - 전체 길이 키 암호(Full-size Key Cipher)(4/4)
    - 전체 길이 키 대치 블록 암호 (Full-size Key Substitution Block Ciphers)

블록 길이가 3비트인 3비트 블록 대치 암호의 모델과 치환표의 집합을 보이시오



- 3 비트 입력 평문은 0과 7 사이의 정수 값을 가짐
- 입력 값은 한 비트만 1의 값을 갖는 8비트 스트링으로 디코딩 됨
- 3비트 대치 블록 암호의 치환표 집합의 원소 개수 $8! = 40{,}320$개
- 키 길이 $[\log_2 40320] = 16$비트
- 16비트 키는 $65{,}536$개의 서로 다른 대응을 정의 할 수 있지만 $40{,}320$개의 키만 사용됨

현대 대칭-키 암호 소개

- 치환 군으로서의 블록 암호
  - 부분 길이 키 암호(Reduced-key Cipher)
    - 정의
      - 블록 길이에서 가능한 모든 치환 중, 키 길이로 표현 가능한 일부만 사용하는 암호
    - 특징
      - 전체 길이 키 암호보다 키 길이가 훨씬 짧음
      - 실제 암호 시스템에서 사용 가능한 현실적인 방식임
      - 대치 블록 암호에서는 전체 길이 키가 너무 커서 부분 길이 키 방식이 필요함
        - e.g., DES는 64비트 블록 암호이지만, 가능한 모든 치환을 표현하지 않고 56비트 키만 사용함

현대 대칭-키 암호 소개

- 치환 군으로서의 블록 암호
  - 키 없는 암호(1/3)
    - 정의
      - 키에 의해 동작이 달라지지 않고, 고정된 규칙으로만 동작하는 암호
    - 종류
      - 키가 없는 전치 암호
      - 키가 없는 대치 암호

현대 대칭-키 암호 소개

- 치환 군으로서의 블록 암호
  - 키 없는 암호(2/3)
    - 키가 없는 전치 암호
      - 정의
        - 값은 바꾸지 않고 위치만 재배열하며, 키가 없거나 키가 고정된 전치 암호
      - 특징
        - 비트나 문자의 값은 바꾸지 않고 위치만 재배열함
        - 키가 없거나, 키가 고정되어 있음
        - 하드웨어로 구현하면 미리 고정된 내장 배선(prewired) 전치 암호로 볼 수 있음
        - 소프트웨어로 구현하면 고정된 전치 규칙을 표로 나타낼 수 있음
        - 현대 블록 암호에서는 이러한 전치 요소가 P-박스(P-box) 의 기초가 됨

현대 대칭-키 암호 소개

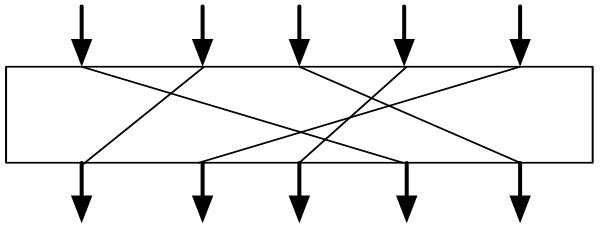
- 치환 군으로서의 블록 암호
  - 키 없는 암호(3/3)
    - 키가 없는 대치 암호
      - 정의
        - 입력 값을 사전에 정의된 규칙에 따라 다른 출력 값으로 대응시키는, 키가 없거나 고정된 대치 암호임
      - 특징
        - 입력 값을 다른 출력 값으로 대체함
        - 키가 없거나, 키가 고정되어 있음
        - 값 자체를 바꾸므로 전치 암호처럼 단순한 위치 재배열과는 다름
        - 사전에 정의된 대응 테이블 또는 수학적 함수로 표현할 수 있음
        - 현대 블록 암호에서는 이러한 대치 요소가 S-박스(S-box) 의 기초가 됨

현대 대칭-키 암호 소개

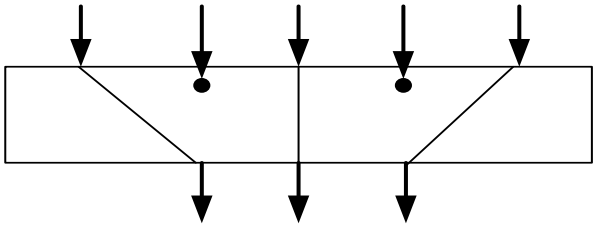
- 현대 블록 암호
  - 정의
    - 고정된 길이의 평문 블록을 입력받아, 키를 이용하여 같은 길이의 암호문 블록으로 변환하는 암호 방식임
  - 구성요소
    - 대치 및 전치 요소
      - P-박스, S-박스
    - 비트 조작 요소
      - XOR, 순환 이동, 스왑, 분할과 결합
    - 반복 구조
      - 라운드, 합성암호
    - 설계 원리
      - 확산, 혼돈

현대 대칭-키 암호 소개

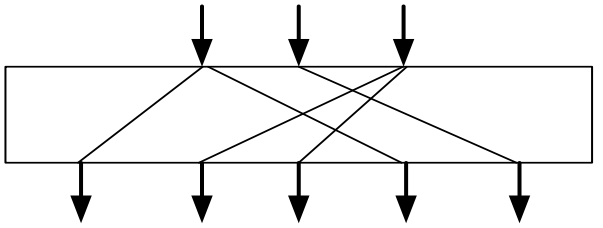
- 현대 블록 암호의 구성요소
  - P-박스(P-box, Permutation Box) (1/6)
    - 정의
      - 입력 비트의 위치를 재배열하여 출력하는 전치 함수임
    - 종류
      - 단순(Straight) P-박스
      - 축소(Compression) P-박스
      - 확장(Expansion) P-박스



단순 P-박스



축소 P-박스



확장 P-박스

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - P-박스(P-box, Permutation Box) (2/6)
    - 단순 P-박스
      - 정의
        - n비트를 입력하여 n비트를 출력하는 전치 함수
      - 특징
        - 입력 비트 수와 출력 비트 수가 같음
        - 비트의 순서만 바뀌고 개수는 유지됨
        - 가능한 대응 수는 $n!$개임
        - 역함수가 존재하므로 복호화가 가능함

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - P-박스(P-box, Permutation Box) (3/6)
    - 축소 P-박스
      - 정의
        - $(n > m), n$비트를 입력하여 m비트를 출력하는 전치함수 $(n > m)$
      - 특징
        - 일부 입력 비트는 출력에서 제거되므로 비트 수가 줄어듦
        - 역함수가 존재하지 않으므로 복호화가 불가능함
        - 다음 단계로 전달되는 비트 수를 줄일 때 사용됨

현대 대칭-키 암호 소개

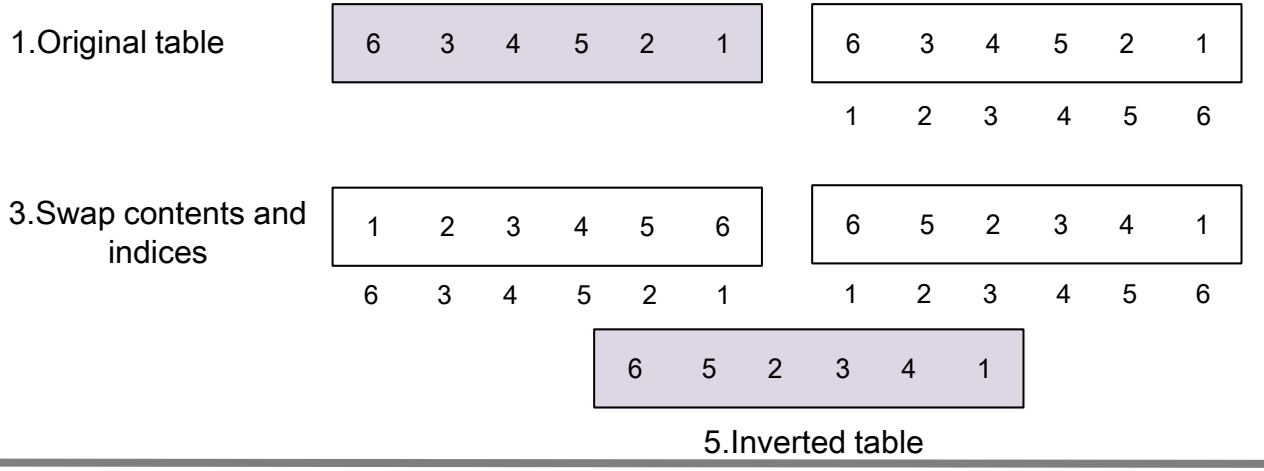
- 현대 블록 암호의 구성요소
  - P-박스(P-box, Permutation Box) (4/6)
    - 확장 P-박스
      - 정의
        - $(n < m), n$비트를 입력하여 m비트를 출력하는 전치함수
      - 특징
        - 하나의 입력 비트가 여러 출력 비트에 연결될 수 있음
        - 비트 수가 늘어남
        - 역함수가 존재하지 않으므로 복호화가 불가능함
        - 다음 단계에서 사용할 비트 수를 늘릴 때 사용됨

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - P-박스(P-box, Permutation Box) (5/6)
    - P-박스 종류에 따른 보안성 비교
      - 확장 P-박스 > 축소 P-박스 > 단순 P-박스
        - 확장 P-박스는 한 입력 비트가 여러 출력 위치로 복제되어 확산 효과가 가장 크고, 역함수가 존재하지 않아 공격자가 출력에서 입력을 복원할 수 없기 때문에 보안성이 가장 높음
        - 축소 P-박스는 일부 입력 비트가 버려져 역함수가 존재하지 않아 공격자가 버려진 비트를 복원할 수 없지만, 한 비트가 여러 위치로 퍼지지 않아 확산 효과는 확장 P-박스보다 작기 때문에 중간 수준의 보안성을 가짐
        - 단순 P-박스는 일대일 대응이라 역함수가 항상 존재하고, 비트 위치만 재배열되어 1의 개수와 값이 그대로 유지되므로 공격자가 단순 분석만으로 입력을 복원할 수 있기 때문에 보안성이 가장 낮음

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - P-박스(P-box, Permutation Box) (6/6)
    - 역함수의 존재성
      - 역함수가 존재하는 경우
        - 일대일 대응이므로 역함수가 존재함
          - e.g., 단순 P-박스
      - 역함수가 존재하지 않는 경우
        - 일대일 대응이 아니므로 역함수가 존재하지 않음
          - e.g., 축소 P-박스, 확장 P-박스
      - 예제 5.7



- 원래 P-박스 표는 각 입력 비트가 이동할 출력 위치를 나타냄
- 복호화에서는 이동된 비트를 원래 위치로 되돌려야 하므로 역치환표가 필요함
- 내용값과 인덱스를 교환한 뒤 인덱스 기준으로 정렬하면 역치환표를 얻을 수 있음
- 예제의 역치환표는 $[6, 5, 2, 3, 4, 1]$임

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - S-박스(S-box, Substitution Box) (1/8)
    - 정의
      - n-비트를 입력받아 m-비트로 대치하여 출력하는 치환 함수임
    - 종류
      - 선형(Linear) S-박스
      - 비선형(Nonlinear) S-박스

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - S-박스(S-box, Substitution Box) (2/8)
    - 선형 S-박스
      - 정의
        - 입력 비트와 출력 비트 사이의 관계가 선형적으로 표현되는 치환 함수
      - 특징
        - 입력과 출력의 관계가 선형 함수로 나타남
        - 구조가 단순하고 계산이 쉬움
        - 예측이 쉬워서 보안성이 낮음
        - 현대 블록 암호에서는 단독으로 잘 쓰이지 않음

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - S-박스(S-box, Substitution Box) (3/8)
    - 예제 5.8

$$y_1 = f_1(x_1, x_2, \cdots, x_n)$$
$$y_2 = f_2(x_1, x_2, \cdots, x_n)$$
$$y_m = f_m(x_1, x_2, \cdots, x_n)$$

$\longrightarrow$

$$y_1 = a_{1,1}\, x_1 \oplus a_{1,2}, x_2 \oplus \cdots a_{1,n} x_n$$
$$y_2 = a_{2,1}\, x_1 \oplus a_{2,2}, x_2 \oplus \cdots a_{2,n} x_n$$
$$y_m = a_{m,1}\, x_1 \oplus a_{m,2}\, x_2 \oplus \cdots a_{m,n} x_n$$

3비트를 입력받아 2비트를 출력하는 S-박스가 다음을 만족한다

$$y_1 = x_1 \oplus x_2 \oplus x_3 \qquad y_2 = x_1$$

- $a_{1,1} = a_{1,2} = a_{1,3} = a_{2,1} = 1$ 이고 $a_{2,2} = a_{2,3} = 0$이기 때문에, 위의 S-박스는 선형임
- 아래의 행렬식처럼 표현할 수 있음
- $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - S-박스(S-box, Substitution Box) (4/8)
    - 비선형 S-박스
      - 정의
        - 입력 비트와 출력 비트 사이의 관계가 선형적으로 표현되지 않는 치환 함수
      - 특징
        - 입력과 출력의 관계가 비선형 함수로 구성됨
        - 출력 값을 쉽게 예측하기 어려움
        - 암호 해석을 어렵게 만들어 보안성을 높임
        - 현대 블록 암호에서 혼돈(confusion) 을 제공하는 핵심 요소임

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - S-박스(S-box, Substitution Box) (5/8)
    - 예제 5.9

$$y_1 = f_1(x_1, x_2, \cdots, x_n)$$
$$y_2 = f_2(x_1, x_2, \cdots, x_n)$$
$$y_m = f_m(x_1, x_2, \cdots, x_n)$$

$\longrightarrow$

$$y_1 = a_{1,1}\, x_1 \oplus a_{1,2}, x_2 \oplus \cdots a_{1,n} x_n$$
$$y_2 = a_{2,1}\, x_1 \oplus a_{2,2}, x_2 \oplus \cdots a_{2,n} x_n$$
$$y_m = a_{m,1}\, x_1 \oplus a_{m,2}\, x_2 \oplus \cdots a_{m,n} x_n$$

3비트를 입력받아 2비트를 출력하는 S-박스가 다음을 만족한다

$$y_1 = (x_1)^3 + x_2 \qquad y_2 = (x_1)^2 + x_1 x_2 + x_3$$

- 해당 예제에서 곱셈과 덧셈은 $GF(2)$에서 정의됨
- 입력 값과 출력 값 사이에 선형 관계식이 존재하지 않으므로 위의 S-박스는 비선형임

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - S-박스(S-box, Substitution Box) (6/8)
    - 예제 5.10

다음의 표는 4×2 길이의 S-박스에 대한 입출력 관계식을 정의한다. 입력 값의 왼쪽 최상위 비트는 행을 정의하고 입력 값의 오른쪽 하위 두 비트는 열을 정의한다. 두 개의 출력비트는 선택된 행과 열이 교차되는 성분의 값이다.

최상위 비트 ↓

	00	01	10	11
0	00	10	01	11
1	10	00	11	01

← 오른쪽 비트

출력 비트

- 입력이 010 이면 01을 출력함
- 입력이 101 이면 00을 출력함

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - S-박스(S-box, Substitution Box) (7/8)
    - 역함수의 존재성
      - 역함수가 존재하는 경우
        - 입력 비트 수와 출력 비트 수가 같고, 각 입력이 서로 다른 출력에 일대일로 대응할 때 역함수가 존재함
      - 역함수가 존재하지 않는 경우
        - 입력과 출력 비트 수가 다른 경우
          - $n \neq m$ 이면 역함수가 존재하지 않음
        - 입력과 출력 비트 수가 같아도 일대일 대응이 아닌 경우
          - $n = m$ 이어도 서로 다른 두 입력이 같은 출력으로 가면 역함수 없음

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - S-박스(S-box, Substitution Box) (8/8)
    - 역함수의 존재성
      - 예제 5.11

아래 그림은 역함수가 존재하는 S-박스의 예제를 보여준다. 표 중 하나는 암호 알고리즘에서 사용되며, 다른 하나는 복호 알고리즘에서 사용된다. 각 표에서 입력 값의 왼쪽 최상위 비트는 행을 정의하고 다음 두 비트는 열을 정의한다. 출력은 입력 값의 행과 열이 교차되는 값이다.

3 비트

	00	01	10	11
0	011	101	111	100
1	000	010	001	110

암호화에 사용되는 표

- 입력이 110 이면 001을 출력함
- 입력이 001 이면 101을 출력함

3 비트

3 비트

	00	01	10	11
0	100	110	101	000
1	011	001	111	010

복호화에 사용되는 표

- 입력이 001 이면 110을 출력함
- 입력이 101 이면 001을 출력함

3 비트

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - 배타적 논리합(XOR, Exclusive-Or) (1/3)
    - 정의
      - 두 입력 비트가 서로 다를 때 1을, 같을 때 0을 출력하는 논리 연산
    - 특징
      - 두 비트가 다르면 1, 같으면 0
      - 비트 단위로 쉽게 연산할 수 있음
      - 자기 자신과 다시 XOR하면 원래 값으로 복원 가능함

XOR 진리표 →

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

현대 대칭-키 암호 소개

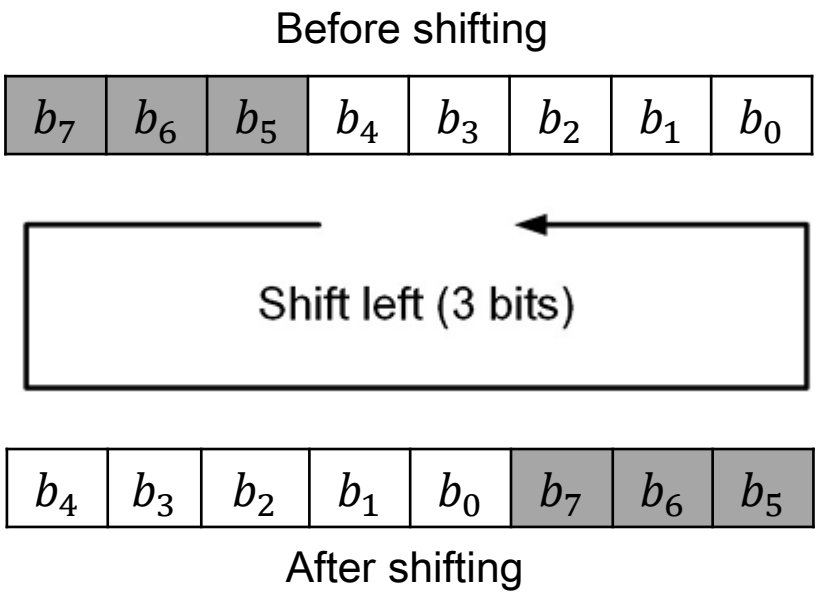
- 현대 블록 암호의 구성요소
  - 배타적 논리합(XOR, Exclusive-Or) (2/3)
    - 성질
      - 닫힘
        - 두 개의 n비트 워드에 대해 XOR 연산을 수행한 결과도 다시 n비트 워드가 됨
      - 결합 법칙
        - $x \oplus (y \oplus z) = (x \oplus y) \oplus z$
      - 교환 법칙
        - $x \oplus y = y \oplus x$
      - 항등원의 존재성
        - $x \oplus 0 = x$
      - 역원의 존재성
        - $x \oplus x = 0$

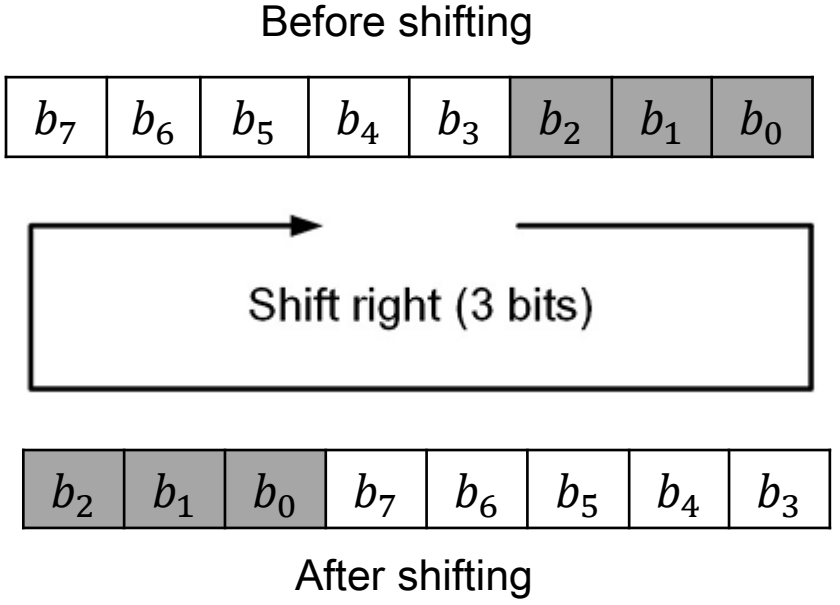
현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - 배타적 논리합(XOR, Exclusive-Or) (3/3)
    - 보수(Complement)
      - 정의
        - 한 워드에서 각 비트를 반대 값으로 바꾸는 단일 연산임
      - 특징
        - 0은 1로, 1은 0으로 바뀜
        - 모든 비트가 1인 값과 XOR하면 보수를 얻을 수 있음
      - 성질
        - $x \oplus \overline{x} = (11 \ldots 1),\ x \oplus (11 \ldots 1) = \overline{x}$ ($\overline{x}$는 x의 보수)
    - 역(Inverse)
      - 정의
        - 어떤 연산의 결과를 다시 원래 값으로 되돌리는 연산임
      - 특징
        - 역이 존재하면 원래 값 복원이 가능함
        - XOR 연산은 자기 자신이 역임

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - 순환 이동 연산(Circular Shift Operation) (1/2)
    - 정의
      - n비트 워드의 각 비트를 왼쪽이나 오른쪽으로 k비트만큼 이동시키는 연산임



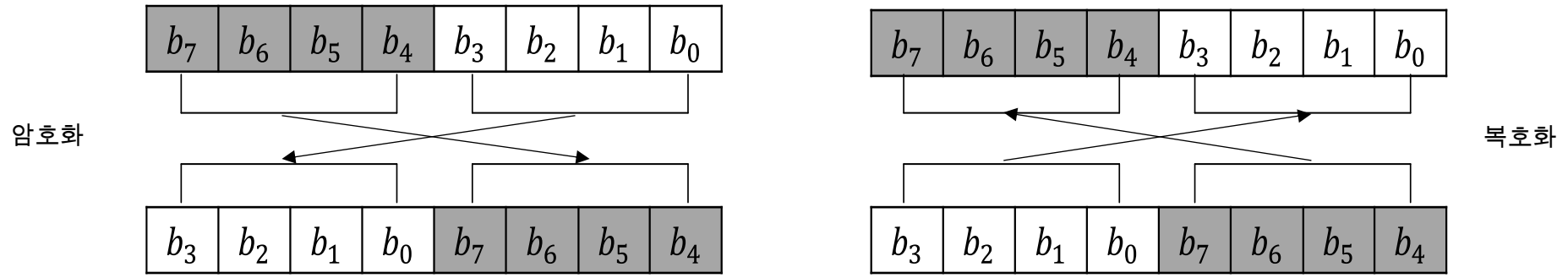


현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - 순환 이동 연산(Circular Shift Operation) (2/2)
    - 특징
      - 현대 블록 암호에 많이 사용되는 구성요소임
      - 왼쪽 순환 이동과 오른쪽 순환 이동 두 가지가 있음
      - 워드의 비트를 섞음으로써 기존의 패턴을 숨기는 데 활용됨
      - 이동하는 위치 수 k는 키로 사용할 수도 있지만,
        보통은 키를 사용하지 않고 k값을 고정하여 사전에 정의함
    - 성질
      - 이동되는 양은 모듈로 n
        - $k = 0$ 이거나 $k = n$이면 원래 값과 같음
        - $k > n$ 이면 실제로는 $k \bmod n$만큼 이동한 것과 같음
      - 합성 연산을 정의하면 순환 이동 연산 집합은 군이 됨

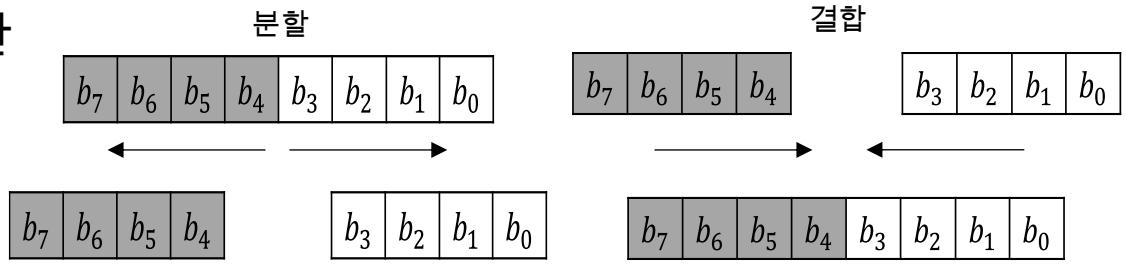
현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - 스왑 연산(Swap Operation)
    - 정의
      - n비트 워드의 앞 절반과 뒤 절반을 서로 교환하는 연산
    - 특징
      - $k = n/2$인 순환 이동의 특별한 경우임
      - n이 짝수일 때만 가능
        - 왼쪽으로 이동되는 $n/2$비트와 오른쪽으로 이동되는 $n/2$비트가 같아야 하기 때문임
      - 자기 자신이 역연산임
      - 암호화와 복호화에 같은 스왑 연산을 사용할 수 있음



현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - 분할과 결합
    - 분할 연산(Split)
      - 정의
        - n비트 워드를 두 개의 같은 길이의 워드로 나누는 연산임
    - 결합 연산(Combine)
      - 정의
        - 두 개의 같은 길이의 워드를 연결하여 n비트 워드로 만드는 연산임
    - 공통 특징
      - 분할과 결합은 서로 역관계임
      - 서로를 상쇄하기 위해 한 쌍으로 사용됨
      - 암호화와 복호화 과정에서 서로 대응됨
        - e.g., $n = 8$인 경우의 연산



현대 대칭-키 암호 소개

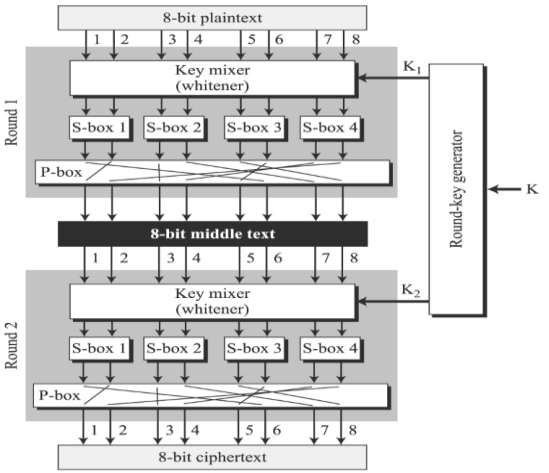
- 현대 블록 암호의 구성요소
  - 합성 암호(Product Cipher) (1/13)
    - 정의
      - 대치, 전치 등의 구성요소를 결합한 복합적인 암호임
    - 확산과 혼돈
      - 확산(Diffusion)
        - 암호문과 평문 사이의 관계를 숨기기 위해 사용함
        - 암호문에 대한 통계 테스트를 통하여 평문을 찾기 어렵게 하기위해 사용함
        - 평문의 한 비트가 바뀌면 암호문의 여러 비트가 바뀔 수 있음
      - 혼돈(Confusion)
        - 암호문과 키의 관계를 숨기기 위해 사용함
        - 암호문을 이용하여 키를 찾기 어렵게 하기 위해 사용함
        - 키의 한 비트가 변하면 암호문의 여러 비트가 바뀔 수 있음

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - 합성 암호(Product Cipher) (2/13)
    - 라운드(Round) (1/5)
      - 정의
        - 반복적으로 사용되는 합성 암호의 한 단위를 라운드라고 함
      - 특징
        - S-박스, P-박스 및 기타 구성요소의 결합으로 구성됨
        - 확산과 혼돈을 얻기 위해 반복 수행됨
        - 각 라운드마다 서로 다른 키를 사용할 수 있음
        - 라운드 수행 중의 상태 값을 중간 상태 값이라 함
        - 실제 합성 암호는 여러 라운드를 반복하여 구성됨

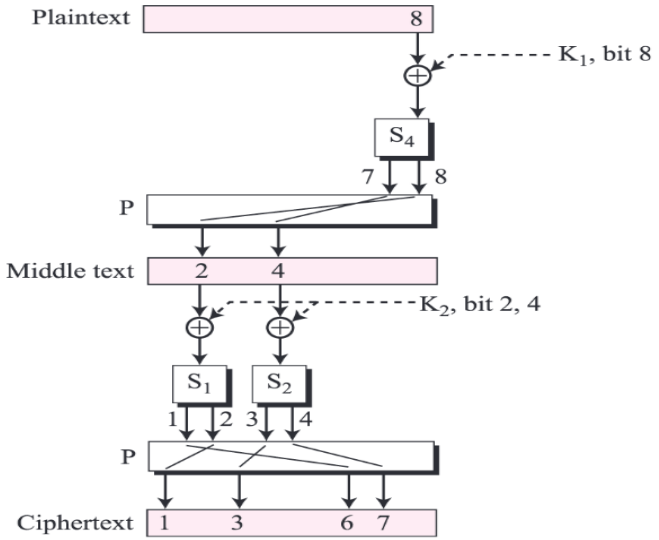
현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - 합성 암호(Product Cipher) (3/13)
    - 라운드(Round) (2/5)
      - 라운드 수행 과정
        1. 8비트 문자열은 문자열의 정보를 숨기기 위하여 키와 섞임
           - 일반적으로 8비트 키와 8비트 문자열의 배타적 논리합이 수행됨
        2. 1의 결과 값은 2비트씩 분할하고, 각 2비트 값은 4개의 S-박스에 입력됨
           - 이 변환과정에서 각 2비트 값은 S-박스의 구조에 따라 변함
        3. 각 비트를 치환하기 위하여 S-박스의 출력 값은 P-박스를 통과함
           - 다음 라운드에서 각 박스는 서로 다른 입력 값을 받음



현대 대칭-키 암호 소개

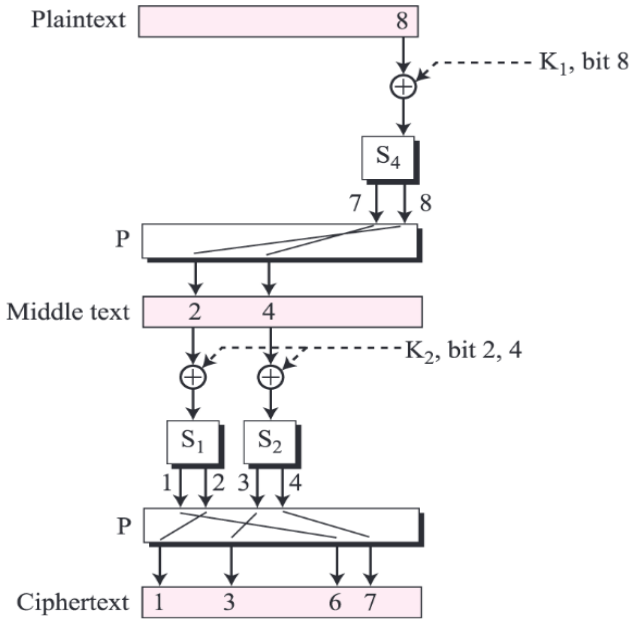
- 현대 블록 암호의 구성요소
  - 합성 암호(Product Cipher) (4/13)
    - 라운드(Round) (3/5)
      - 확산
        - 합성 암호에서 평문의 한 비트가 여러 단계를 거치면서 암호문의 여러 비트에 영향을 미치는 과정
          - e.g., 8번째 비트 값은 1,3,6,7 번째 비트에 영향을 미침



- 평문의 8번째 비트가 S-박스와 P-박스를 지나면서 중간 값의 여러 비트로 퍼지고, 최종적으로 암호문의 1, 3, 6, 7번째 비트에 영향을 미침

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - 합성 암호(Product Cipher) (5/13)
    - 라운드(Round) (4/5)
      - 혼돈
        - 암호문의 특정 비트들이 키의 여러 비트에 의해 영향을 받는 과정
          - e.g., 1,3,6,7번째 비트는 키에서 3개의 비트에 의해 영향을 받음



- 평문 비트와 키 비트가 XOR되어 중간값을 생성함
- 중간값은 S-박스와 P-박스를 거치며 위치와 값이 섞임
- 이후 다른 라운드 키와 다시 XOR되어 키의 영향이 더해짐
- 결과적으로 암호문의 특정 비트가 여러 키 비트의 영향을 받게 됨

현대 대칭-키 암호 소개

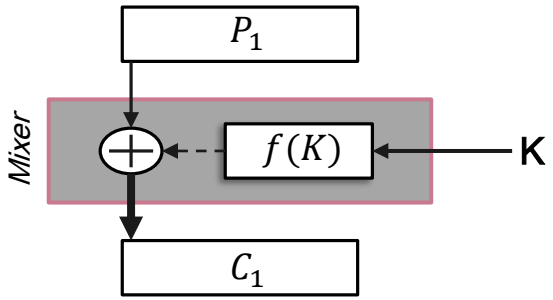
- 현대 블록 암호의 구성요소
  - 합성 암호(Product Cipher) (6/13)
    - 라운드(Round) (5/5)
      - 실제 암호의 활용
        - 확산과 혼돈 효과를 높이기 위해 실제 암호에서는 더 큰 데이터 블록, 더 많은 S-박스, 더 많은 라운드를 사용함
        - 암호문이 더욱 난수열처럼 보이게 됨
        - 평문과 암호문 사이의 관계를 더 알아내기 어려워짐
        - 라운드 수가 증가할수록 암호문과 키의 관계를 찾기 어려워짐

현대 대칭-키 암호 소개

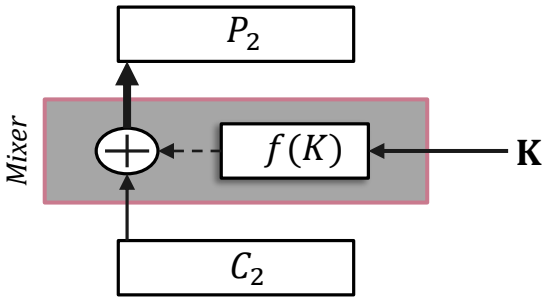
- 현대 블록 암호의 구성요소
  - 합성 암호(Product Cipher) (7/13)
    - 종류
      - Feistel 암호
        - 정의
          - 평문을 좌우로 분할하여 라운드 함수와 XOR을 반복 적용하는 합성 암호
            - e.g., DES 암호
        - 특징
          - 역함수가 존재하지 않는 구성요소도 사용할 수 있음
      - non-Feistel 암호
        - 정의
          - 평문 전체에 역연산 가능한 구성요소를 순차 적용하는 합성 암호
            - e.g., AES 암호
        - 특징
          - 역함수가 존재하는 구성요소만 사용함

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - 합성 암호(Product Cipher) (8/13)
    - Feistel 암호 (1/5)
      - 구성요소
        - 자기 자신을 역으로 갖는 함수
        - 역함수가 존재하는 함수
        - 역함수가 존재하지 않는 함수
      - 초기구조
        - 역함수가 존재하지 않는 함수 $f(K)$를 사용
        - 평문과 $f(K)$를 XOR 하여 암호문 생성
        - 같은 $f(K)$와 XOR을 다시 수행하면 복호화 가능
        - 배타적 논리합의 성질 때문에 상쇄됨



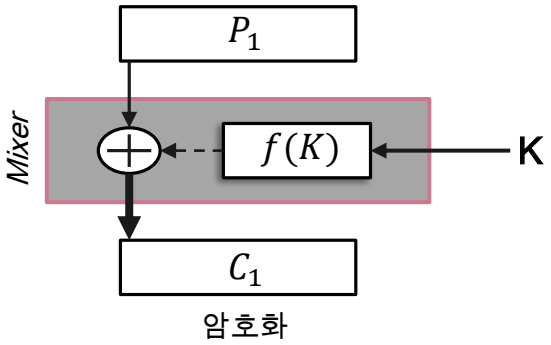
암호화

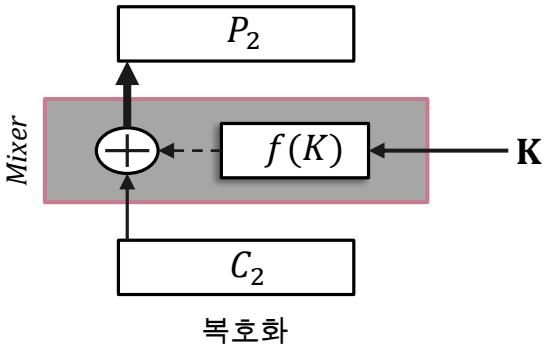


복호화

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - 합성 암호(Product Cipher) (9/13)
    - Feistel 암호 (2/5)
      - 혼합기(Mixer)
        - 키 K에 대한 역함수가 존재하지 않는 함수 $f(K)$와 XOR의 결합
        - 자기 자신을 역함수로 가짐
        - 배타적 논리합의 성질
          - 역의 존재성
          - 항등원의 존재성
        - 암호화: $C_1 = P_1 \oplus f(K)$
        - 복호화: $P_2 = C_2 \oplus f(K) = C_1 \oplus f(K) = P_1 \oplus f(K) \oplus f(K) = P_1 \oplus (00 \ldots 0) = P_1$





현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - 합성 암호(Product Cipher) (10/13)
    - Feistel 암호 (3/5)
      - 예제 5.12

평문과 암호문의 길이가 4비트 이고 키의 길이가 3비트라 하자.
함수는 키의 첫 번째와 세 번째 비트를 입력 받는다.
두 비트는 10진수로 인식하고, 제곱한 뒤,그 결과 값을 4비트 2진수로 출력한다.
만약 평문이 0111이고 키가 101일 때 암호화와 복호화의 결과 값을 보이시오.

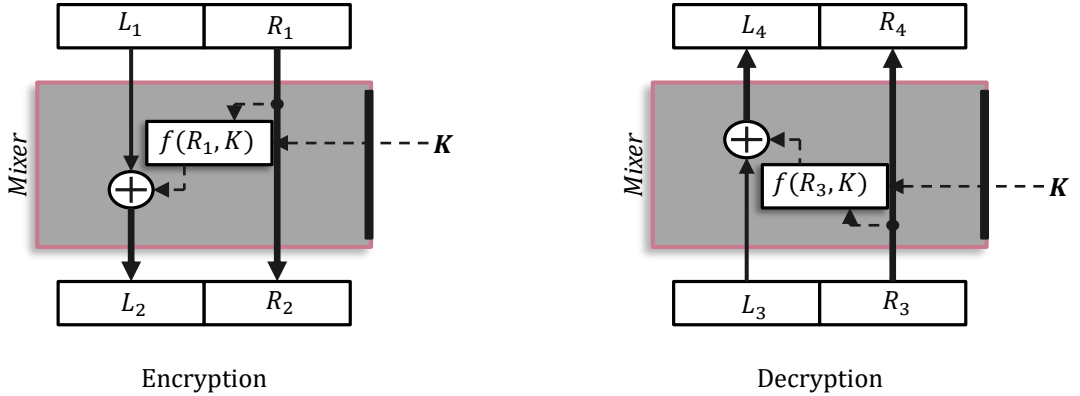
- 키의1, 3 번째 비트를 통해 2진수로 11, 10진수로 3의 값을 얻고 제곱한 값은 9임.
  9는 4비트 2진수로 1001임

- 암호화:$C = P \oplus f(K) = 0111 \oplus 1001 = 1110$
- 복호화:$P = C \oplus f(K) = 1110 \oplus 1001 = 0111$

- XOR 연산은 같은 값을 다시 XOR하면 원래 값으로 돌아오는 성질을 가짐

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - 합성 암호(Product Cipher) (11/13)
    - Feistel 암호 (4/5)
      - 개선 구조
        - 초기 구조의 문제점을 보완함
          - 평문 전체에 $f(K)$를 한 번 XOR하여 평문과 암호문의 관계가 단순하게 남음
        - 평문과 암호문을 왼쪽 블록 L, 오른쪽 블록 R로 분할함
        - 오른쪽 블록을 함수 입력으로 사용하고, 왼쪽 블록은 함수 출력과 XOR 수행함
        - 함수 입력 값은 암호화와 복호화에서 동일해야 함
        - 오른쪽 블록은 입력된 값 그대로 출력됨



Encryption Decryption

- $R_4 = R_3 = R_2 = R_1$
- $L_4 = L_3 \oplus f(R_3, K) = L_2 \oplus f(R_2, K) = L_1 \oplus f(R_1, K) \oplus f(R_1, K) = L_1$

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - 합성 암호(Product Cipher) (12/13)
    - Feistel 암호 (5/5)
      - 최종 구조
        - 개선 방법의 문제점을 보완하여 라운드 수를 증가시킴
          - 한 라운드에서는 한쪽 블록이 그대로 출력되어 충분히 섞이지 않음
        - 각 라운드에 새로운 구성요소인 스와퍼(swapper) 를 추가함
          - 스와퍼는 각 라운드의 왼쪽 블록과 오른쪽 블록을 교환함
        - 암호화 과정에서 바뀐 좌우 블록은 복호화 과정에서 다시 바뀌어 원래 위치로 복원됨
        - 두 개의 라운드 키 $\mathrm{K}_1, \mathrm{K}_2$가 존재하며, 키는 암호화와 복호화에서 역순으로 사용됨
        - 혼합기와 스와퍼가 결합되어 암호 알고리즘과 복호 알고리즘은 서로 역관계를 이룸

        - $L_5 = R_4 \oplus f(L_4, K_2) = R_3 \oplus f(R_2, K_2) = \big(L_2 \oplus f(R_2, K_2)\big) \oplus f(R_2, K_2) = L_2$
        - $R_5 = L_4 = L_3 = R_2$

현대 대칭-키 암호 소개

- 현대 블록 암호의 구성요소
  - 합성 암호(Product Cipher) (13/13)
    - non-Feistel 암호
      - 평문 블록 전체를 대상으로 라운드 연산을 수행함
      - 각 단계의 연산은 복호화를 위해 역연산이 필요함
      - 입력 비트 수와 출력 비트 수가 같은 S-박스를 주로 사용함
      - 축소 P-박스와 확장 P-박스는 직접적인 역연산이 어려워 사용하기 어려움
      - 복호화에서는 라운드 키를 역순으로 사용함

현대 대칭-키 암호 소개

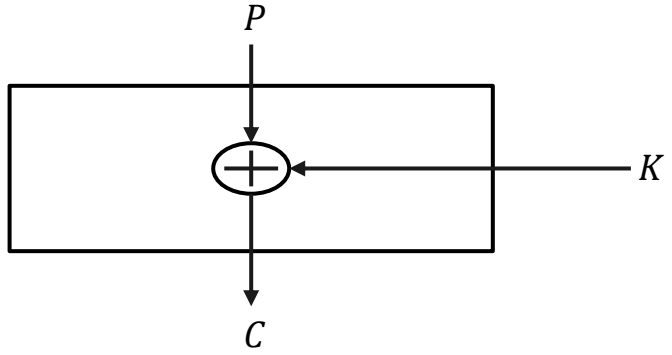
- 블록 암호에 대한 공격
  - 차분 분석(Differential Cryptanalysis)
    - 평문 차분과 암호문 차분의 관계를 이용하여 키를 추정하는 공격
    - 평문 차분을 의도적으로 조작해야 하므로 선택 평문 공격에 사용됨
  - 선형 분석(Linear Cryptanalysis)
    - 평문, 암호문, 키 사이의 선형 근사를 이용하여 키를 추정하는 공격
    - 평문을 선택할 필요 없이 관찰만으로 가능하므로 기지 평문 공격에 사용됨

현대 대칭-키 암호 소개

- 블록 암호에 대한공격
  - 차분 분석(Differential Cryptanalysis) (1/6)
    1. 알고리즘 분석 (1/3)

<단순 암호 구조를 통한 차분 분석 예시>

$C = P \oplus K$



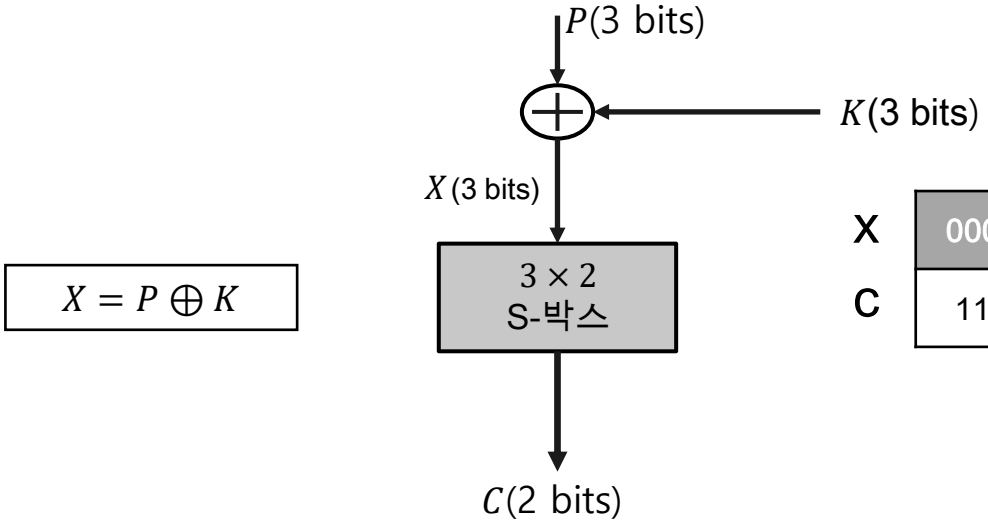
단순 XOR 구조에서는 키 K가 두 번 XOR되어 상쇄됨
따라서 암호문 차분 $C_1 \oplus C_2$는 평문 차분 $P_1 \oplus P_2$와 동일해짐

- $C_1 = P_1 \oplus K, C_2 = P_2 \oplus K \rightarrow C_1 \oplus C_2 = P_1 \oplus K \oplus P_2 \oplus K = P_1 \oplus P_2$
- 단순 XOR 구조에서는 키의 효과가 상쇄되어 평문 차분과 암호문 차분이 동일하다는 관계를 찾을 수 있음

현대 대칭-키 암호 소개

- 블록 암호에 대한 공격
  - 차분 분석(Differential Cryptanalysis) (2/6)
    1. 알고리즘 분석 (2/3)

<S-박스 구조에서의 차분 분석 예시>



$X = P \oplus K$

X:S-박스에 들어가는 입력 값
C:S-박스를 통과해서 나온 출력 값

X	000	001	010	011	100	101	110	111
C	11	00	10	10	01	00	11	00

S-box table

- S-박스가 추가되면 평문 차분과 암호문 차분 사이의 명백한 관계를 찾기 어려움
- $X_1 \oplus X_2 = (\mathrm{P}_1 \oplus \mathrm{K}) \oplus (\mathrm{P}_2 \oplus \mathrm{K}) = P_1 \oplus P_2 \oplus K \oplus K = P_1 \oplus P_2$를 만족하므로 확률적 관계를 알아낼 수 있음
- 이 관계를 기반으로 S-박스의 입출력 표와 차분 분포 표를 작성할 수 있음

현대 대칭-키 암호 소개

- 블록 암호에 대한 공격
  - 차분 분석(Differential Cryptanalysis) (3/6)
    1. 알고리즘 분석 (3/3)

<S-박스 구조에서의 차분 분석 예시>

$C_1 \oplus C_2$

$P_1 \oplus P_2$	00	01	10	11
000	8			
001	2	2		4
010	2	2	4	
011		4	2	2
100	2	2	4	
101		4	2	2
110	4		2	2
111			2	6

입력 차분별 출력 차분의 발생 횟수

$C_1 \oplus C_2$

$P_1 \oplus P_2$	00	01	10	11
000	1	0	0	0
001	0.25	0.50	0	0.50
010	0.25	0.25	0.50	0
011	0	0.50	0.25	0.25
100	0.25	0.25	0.50	0
101	0	0.50	0.25	0.25
110	0.50	0	0.25	0.25
111	0	0	0.25	0.75

발생 횟수를 확률로 변환한 표

- 확률이 높은 입력 차분은 출력 차분을 예측하기 쉬워 공격에 유리함
- $P_1 \oplus P_2 = 111$일 때 $C_1 \oplus C_2 = 11$이 확률 0.75로 발생함 따라서 공격에 유리한 평문 차분은 111임

현대 대칭-키 암호 소개

- 블록 암호에 대한 공격
  - 차분 분석(Differential Cryptanalysis) (4/6)
    2. 선택 평문 공격의 시작
       - 이전의 분석한 내용은 암호의 구조가 변하지 않는 한 지속적으로 사용할 수 있음
       - 이를 기반으로 공격을 위해 필요한 평문을 선택함
         - 차분 확률 분포표에서 가장 높은 확률을 갖는 평문의 차분을 선택해야 함
    3. 키 값의 추측
       - 키 값의 추측을 위해 특정 평문과 암호문 쌍을 찾음
       - 암호문 C로부터 평문 P가 생성되도록 함

현대 대칭-키 암호 소개

- 블록 암호에 대한 공격
  - 차분 분석(Differential Cryptanalysis) (5/6)
    - 일반적인 절차
      1. 각 라운드가 동일하므로 각 S-박스에 대한 차분 분포표를 구성함
      2. 각 라운드가 독립이라고 가정하고, 대응 확률을 결합하여 전체 암호에 대한 분포를 구성함
      3. 구성한 분포를 바탕으로 공격에 유리한 평문 차분과 평문 쌍을 선택함
      4. 선택한 평문 쌍에 대응하는 암호문을 획득하고, 이를 분석하여 키의 특정 비트 정보를 추출함
      5. 더 많은 키 비트 정보를 얻기 위해 위 과정을 반복함
      6. 충분한 키 비트 정보를 얻은 후, 남은 키는 전수조사로 확인함

현대 대칭-키 암호 소개

- 블록 암호에 대한 공격
  - 차분 분석(Differential Cryptanalysis) (6/6)
    - 차분 분석 한계
      - S-box는 비선형 구조이므로 차분 관계가 항상 일정하지 않음
      - 유의미한 분석을 위해 특정 차분을 가진 평문 쌍을 많이 확보해야 함
      - 차분 확률이 낮으면 공격 효과가 작아짐
      - 라운드 수가 증가할수록 차분 경로를 추적하기 어려워짐

현대 대칭-키 암호 소개

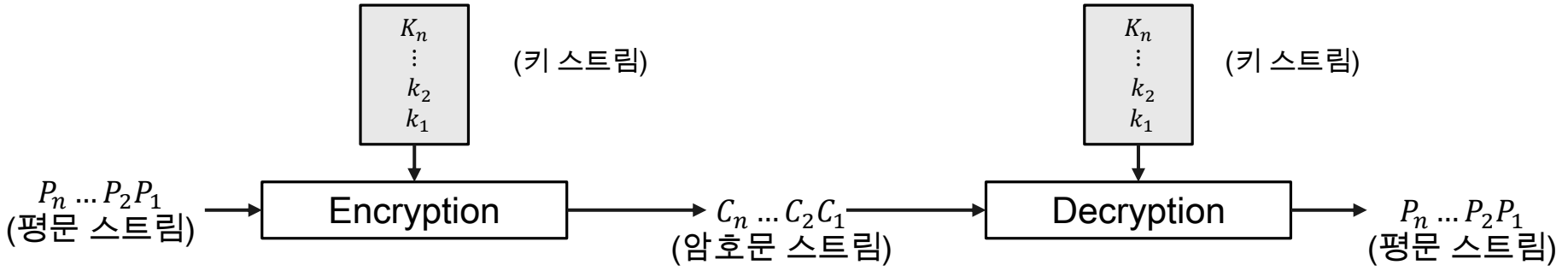
- 블록 암호에 대한 공격
  - 선형분석(Linear Cryptanalysis) (1/2)
    - 선형(Linear)
      - 입력 비트, 출력 비트, 키 비트 간의 관계를 XOR 형태의 선형식으로 표현할 수 있는 성질
    - 선형 근사식(Linear Approximation)
      - 비선형 알고리즘을 구성하는 연산들에 대해 입력, 출력, 키 비트 사이의 선형적 관계가 특정 확률로 근사적으로 성립하는 식
      - S-box는 특정 선형 함수에 의해 확률적으로 근사적으로 될 수 있음
      - 일반적으로 n-비트 평문 및 암호문, m-비트 키가 주어지면 다음과 같은 식을 찾음
        - $(k_0 \oplus k_1 \oplus \cdots \oplus k_x) = (p_0 \oplus p_1 \oplus \cdots \oplus p_y) \oplus (c_0 \oplus c_1 \oplus \cdots \oplus c_z)$
      - 각 식은 확률 $\frac{1}{2} + \varepsilon$로 성립함($\varepsilon$ 는 편차)
        - ε가 클수록 더 효과적임

현대 대칭-키 암호 소개

- 블록 암호에 대한 공격
  - 선형분석(Linear Cryptanalysis) (2/2)
    - 선형 분석 한계
      - S-box는 본래 비선형 구조이므로 근사식이 항상 정확하지 않음
      - 유의미한 통계 분석을 위해 충분한 양의 평문–암호문 쌍이 필요함
      - 확률 편차 ϵ가 작으면 공격 효과가 작아짐
      - 편차가 클수록 더 효과적인 선형 분석이 가능함

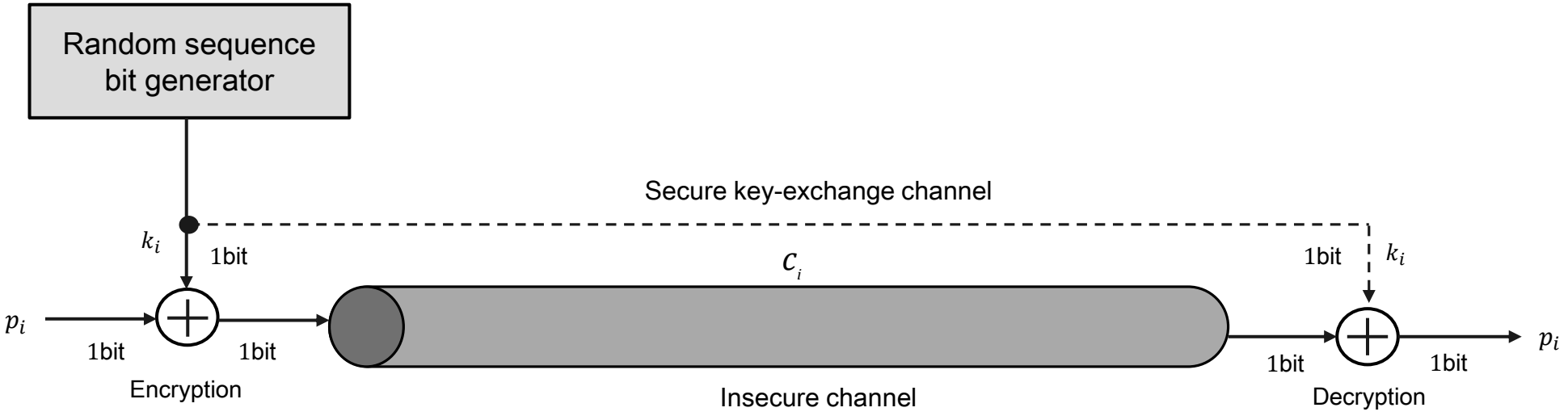
현대 대칭-키 암호 소개

- 현대 스트림 암호(Modern Stream Cipher)
  - 정의
    - 평문 스트림을 키 스트림과 결합하여 대응되는 암호문 스트림을 생성하는 암호 방식임
  - 종류
    - 동기식 스트림 암호
      - 키 스트림이 평문 혹은 암호문과 독립적으로 생성됨
        - e.g ., 일회용 패드(One-Time Pad, OTP)
    - 비동기식 스트림 암호
      - 키 스트림이 이전의 평문 또는 암호문에 종속되어 생성됨
        - e.g ., CFB(Cipher Feedback) 모드



현대 대칭-키 암호 소개

- 현대 스트림 암호(Modern Stream Cipher)
  - 동기식(Symchronous) 스트림 암호 (1/12)
    - 일회용 패드(One-Time Pad, OTP) (1/2)
      - 랜덤하게 선택된 키 스트림을 사용하는 동기식 스트림 암호
      - 암호화와 복호화는 평문을 키 스트림과 XOR 연산으로 수행됨
      - 공격자가 키나 평문을 추측 할 수 없음
      - 평문의 통계적 특성이 암호문에 드러나지 않음
      - 안전한 키 공유가 필요하여 실제 사용은 어려움



현대 대칭-키 암호 소개

- 현대 스트림 암호(Modern Stream Cipher)
  - 동기식(Symchronous) 스트림 암호 (2/12)
    - 일회용 패드(One-Time Pad, OTP) (2/2)
      - 예제 5.17

다음 경우 각각에 대하여 일회용패드의 암호문의 패턴은 무엇인가?

a. 평문이 n-비트의 0으로 구성되는 경우
b. 평문이 n-비트의 1로 구성되는 경우
c. 평문이 0과 1로 교차되어 구성되는 경우
d. 평문이 랜덤 비트의 스트림인 경우

$$C_i = P_i \oplus k_i$$

a. $0 \oplus k_i = k_i$ 이므로 암호문 스트림은 키 스트림과 같음
b. $1 \oplus k_i = \overline{k_i}$ 이므로 암호문 스트림은 키 스트림의 보수가 됨
c. 암호문 각 비트는 대응되는 키 스트림 비트와 같거나 보수가 됨
d. 두 개의 랜덤 비트에 대해 XOR을 수행한 결과이므로 암호문은 완전히 랜덤함

현대 대칭-키 암호 소개

- 현대 스트림 암호(Modern Stream Cipher)
  - 동기식(Symchronous) 스트림 암호 (3/12)
    - 귀환 이동 레지스터(FSR, Feedback Shift Register) (1/10)
      - 정의
        - 비트를 한 칸씩 이동시키며 출력하고, 이동으로 생긴 빈 자리를 귀환 함수의 결과로 채워 키 스트림을 생성하는 장치
      - 구성요소
        - 귀환(Feedback) 함수
        - 이동(Shift) 레지스터

현대 대칭-키 암호 소개

- 현대 스트림 암호(Modern Stream Cipher)
  - 동기식(Symchronous) 스트림 암호 (4/12)
    - 귀환 이동 레지스터(FSR, Feedback Shift Register) (2/10)
      - 귀환 함수
        - 레지스터의 셀 값들을 받아 레지스터에 채울 새 비트를 생성하는 함수
        - 이동 후 맨 왼쪽에 들어갈 비트를 결정함
        - 키 스트림의 패턴과 주기를 결정하는 핵심 요소
        - $b_m = f(b_0, b_1, \ldots, b_{m-1)})$

현대 대칭-키 암호 소개

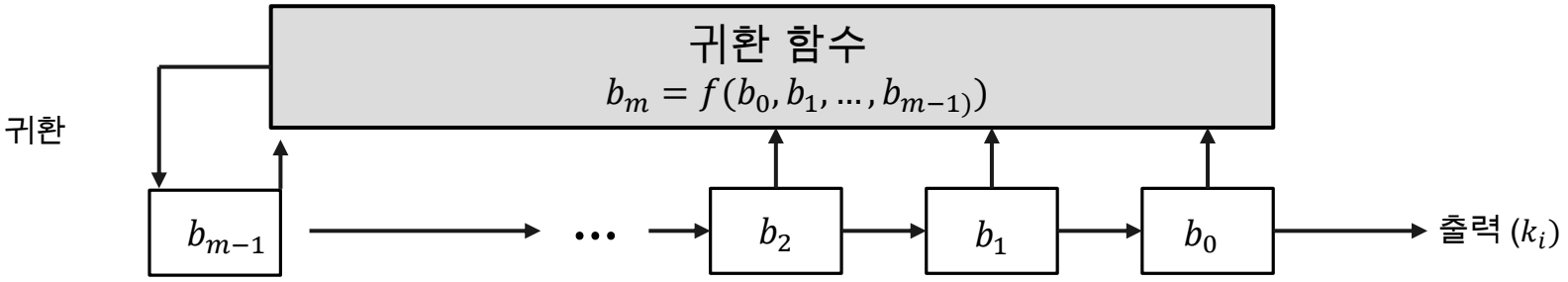
- 현대 스트림 암호(Modern Stream Cipher)
  - 동기식(Symchronous) 스트림 암호 (5/12)
    - 귀환 이동 레지스터(FSR, Feedback Shift Register) (3/10)
      - 이동 레지스터
        - 구성
          - b_0 에서 b_{m-1}까지의 m개의 셀로 이루어짐
          - 각 셀은 1비트 값을 저장함
          - 전체 레지스터는 m비트 길이를 가짐
        - 초기 상태
          - 시작 시 레지스터는 하나의 m비트 초기값으로 설정됨
          - 이 초기 값을 시드(seed) 라고 함

현대 대칭-키 암호 소개

- 현대 스트림 암호(Modern Stream Cipher)
  - 동기식(Symchronous) 스트림 암호 (6/12)
    - 귀환 이동 레지스터(FSR, Feedback Shift Register) (4/10)
      - 이동 레지스터
        - 동작 방식
          - 출력 비트가 필요할 때마다 각 셀의 값이 오른쪽으로 한 칸씩 이동함
          - 오른쪽 최하위 셀 b_0는 출력 값 k_i로 사용됨
          - 비워진 가장 왼쪽 자리(b_{m-1})에는 귀환 함수가 만든 새 비트가 채워짐

비트 이동 규칙

$b_0 \rightarrow k_i$
$b_1 \rightarrow b_0$
$b_2 \rightarrow b_1$
...
$b_m \rightarrow b_{m-1}$



현대 대칭-키 암호 소개

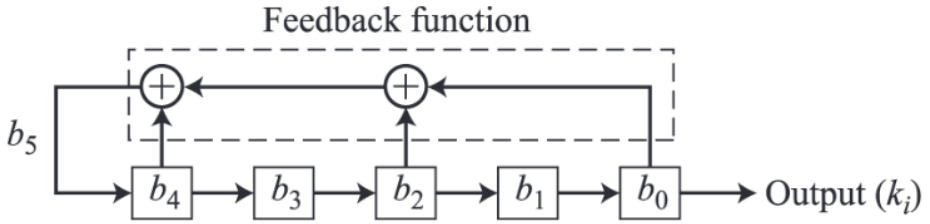
- 현대 스트림 암호(Modern Stream Cipher)
  - 동기식(Symchronous) 스트림 암호 (7/12)
    - 귀환 이동 레지스터(FSR, Feedback Shift Register) (5/10)
      - 선형 귀환이동 레지스터(LFSR, Linear Feedback Shift Register)
        - 귀환 함수가 선형인 FSR의 한 형태
        - b_m 은 $b_0, b_1, \cdots, b_{m-1}$의 선형 함수임
          - $b_m = c_{m-1}b_{m-1} \oplus \cdots \oplus c_2 b_2 \oplus c_1 b_1 \oplus c_0 b_0 \ (c_0 \neq 0)$
        - 체 $GF(2)$상에서 덧셈은 XOR로 표현됨
        - c_i 는 0 또는 1의 값을 가지며, 1인 경우에만 계산에 참여함
        - c_0 는 출력 값이 귀환 되어야 하므로 항상 1임

c_i:LFSR의 귀환 함수에서 어떤 비트를 사용할지 결정하는 계수

현대 대칭-키 암호 소개

- 현대 스트림 암호(Morden Stream Cipher)
  - 동기식(Symchronous) 스트림 암호 (8/12)
    - 귀환 이동 레지스터(FSR, Feedback Shift Register) (6/10)
      - 선형 귀환이동 레지스터(LFSR, Linear Feedback Shift Register)
        - 예제 5.18

$b_5 = b_4 \oplus b_2 \oplus b_0$을 만족하고 5개의 셀을 갖는 선형 귀환 이동레지스터 설계



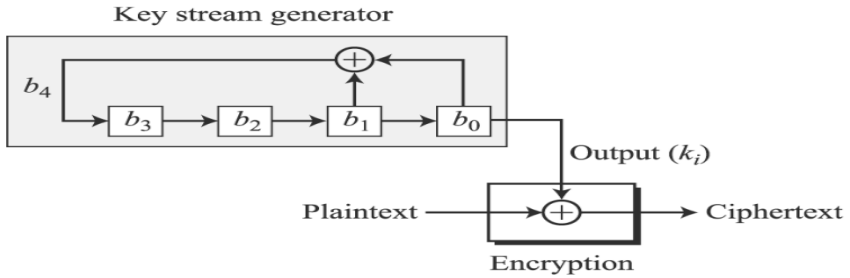
현대 대칭-키 암호 소개

- 현대 스트림 암호(Modern Stream Cipher)
  - 동기식(Symchronous) 스트림 암호 (9/12)
    - 귀환 이동 레지스터(FSR, Feedback Shift Register) (7/10)
      - 선형 귀환이동 레지스터(LFSR, Linear Feedback Shift Register)
        - 의사난수열
          - 난수처럼 보이지만 규칙적인 알고리즘에 의해 생성되는 수열
          - LFSR은 시드와 귀환 함수에 따라 의사난수 키 스트림을 생성함
          - 생성된 키 스트림은 일정 길이 이후 주기적으로 반복됨
          - 최대 주기는 $2^m - 1$이며, m은 레지스터 길이임
          - 모든 비트가 0인 시드는 제외함

현대 대칭-키 암호 소개

- 현대 스트림 암호(Modern Stream Cipher)
  - 동기식(Symchronous) 스트림 암호 (10/12)
    - 귀환 이동 레지스터(FSR, Feedback Shift Register) (8/10)
      - 선형 귀환이동 레지스터(LFSR, Linear Feedback Shift Register)
        - 의사난수열
          - 예제 5.19

$b_4 = b_1 \oplus b_0$ 을 만족하고 4개의 셀을 갖는 선형 귀환 이동 레지스터 설계 단, 시드가 $(0001)_2$일 때 20번의 변환에 대한 출력 값



States	b_4	b_3	b_2	b_1	b_0	k_i
Initial	1	0	0	0	1	
1	0	1	0	0	0	1
2	0	0	1	0	0	0
3	1	0	0	1	0	0
4	1	1	0	0	1	0
5	0	1	1	0	0	1
6	1	0	1	1	0	0
7	0	1	0	1	1	0
8	1	0	1	0	1	1
9	1	1	0	1	0	1
10	1	1	1	0	1	0
11	1	1	1	1	0	1
12	0	1	1	1	1	0
13	0	0	1	1	1	1
14	0	0	0	1	1	1
15	1	0	0	0	1	1
16	0	1	0	0	0	1
17	0	0	1	0	0	0
18	1	0	0	1	0	0
19	1	1	0	0	1	0
20	1	1	1	0	0	1

- 시드와 귀환 함수로부터 결정론적으로 생성됨
- 통계적으로 랜덤하게 보이지만 주기가 존재함
- 시드 $(0001)_2$로 주기 15의 의사난수열 생성($2^4 - 1 = 15$ 와 일치)
- 출력: 100010011010111(15비트, 이후 반복)

현대 대칭-키 암호 소개

- 현대 스트림 암호(Modern Stream Cipher)
  - 동기식(Symchronous) 스트림 암호 (11/12)
    - 귀환이동 레지스터(FSR, Feedback Shift Register) (9/10)
      - 선형 귀환 이동 레지스터(LFSR, Linear Feedback Shift Register)
        - 특성 다항식(Characteristic Polynomial)
          - LFSR의 귀환 함수를 GF(2) 상의 다항식으로 나타낸 것
          - LFSR의 구조와 키 스트림의 주기 특성을 분석하는 데 사용함
          - 최대 주기를 가지기 위해서는 특성 다항식이 원시 다항식이어야 함
          - LFSR의 특성다항식 정의
            - m-비트 LFSR에서 비트 상태가 $b_0, b_1, \cdots, b_{m-1}$이고, 귀환 함수가 $b_m = c_{m-1}b_{m-1} \oplus \cdots \oplus c_1 b_1 \oplus c_0 b_0$임
            - 여기서 $c_i \in \{0, 1\}$ 이며, $GF(2)$상에서 정의됨
            - 특성 다항식은 $f(x) = x^m + c_{m-1}x^{m-1} + \cdots + c_1 x + c_0$

원시다항식(Primitive Polynomial)
기약 다항식이면서 최소 주기가 $2^m - 1$이 되도록 하는 다항식
c_i:LFSR의 귀환 함수에서 어떤 비트를 사용할지 결정하는 계수

현대 대칭-키 암호 소개

- 현대 스트림 암호(Modern Stream Cipher)
  - 동기식(Symchronous) 스트림 암호 (12/12)
    - 귀환이동 레지스터(FSR, Feedback Shift Register) (10/10)
      - 비선형 귀환 이동 레지스터(NLFSR, Nonlinear Feedback Shift Register)
        - 귀환 함수가 선형이 아닌 FSR의 한 형태임
        - b_m이 비선형 함수로 정의되며, 이외에는 LFSR과 동일한 구조를 가짐
          - $b_m = f(b_{m-1}, b_{m-2}, \ldots, b_1, b_0)$
        - LFSR의 선형성으로 인해 키 스트림이 예측될 수 있다는 취약점을 보완할 수 있음
      - 결합(Combination)
        - 스트림 암호는 선형과 비선형 구조를 결합하여 사용함
        - 여러 LFSR을 최대 주기를 갖도록 설계한 뒤 비선형 함수와 결합함
        - 선형 구조는 단순하고 빠르지만 예측 가능성이 있음
        - 비선형 구조는 키 스트림의 예측 가능성을 낮춰 보안성을 높임
        - 결합을 통해 속도와 보안성을 만족하는 키 스트림 생성기를 구현할 수 있음

현대 대칭-키 암호 소개

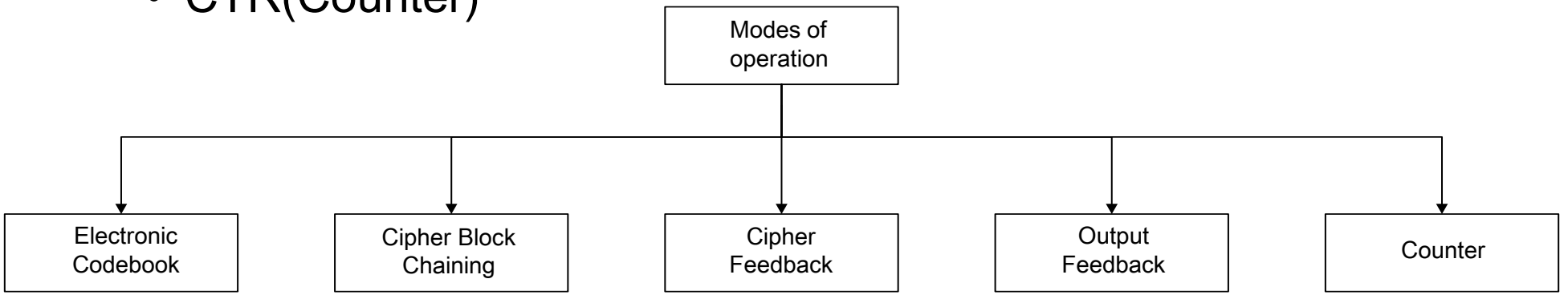
- 현대 스트림 암호(Modern Stream Cipher)
  - 비동기식(Nonsymchronous) 스트림암호
    - 정의
      - 키 스트림이 키뿐만 아니라 이전 평문 및 암호문에도 의존하여 생성되는 스트림 암호
    - 특징
      - 동기식 스트림 암호와 달리, 키 스트림이 평문이나 암호문과 독립적이지 않음
      - 이전의 평문이나 암호문의 값이 다음 키 스트림 생성에 영향을 줌
      - 키 스트림은 이전 입력 또는 출력에 따라 변화함

목 차

- 보충
- 현대 대칭-키 암호 소개
- 현대 대칭-키 암호를 이용한 암호화 기법

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - 임의의 길이의 텍스트를 암호화 하는 방법
    - 종류
      - ECB(Electronic Codebook)
      - CBC(Cipher Block Chaining)
      - CFB(Cipher Feedback)
      - OFB(Output Feedback)
      - CTR(Counter)



현대 대칭-키 암호를 이용한 암호화 기법

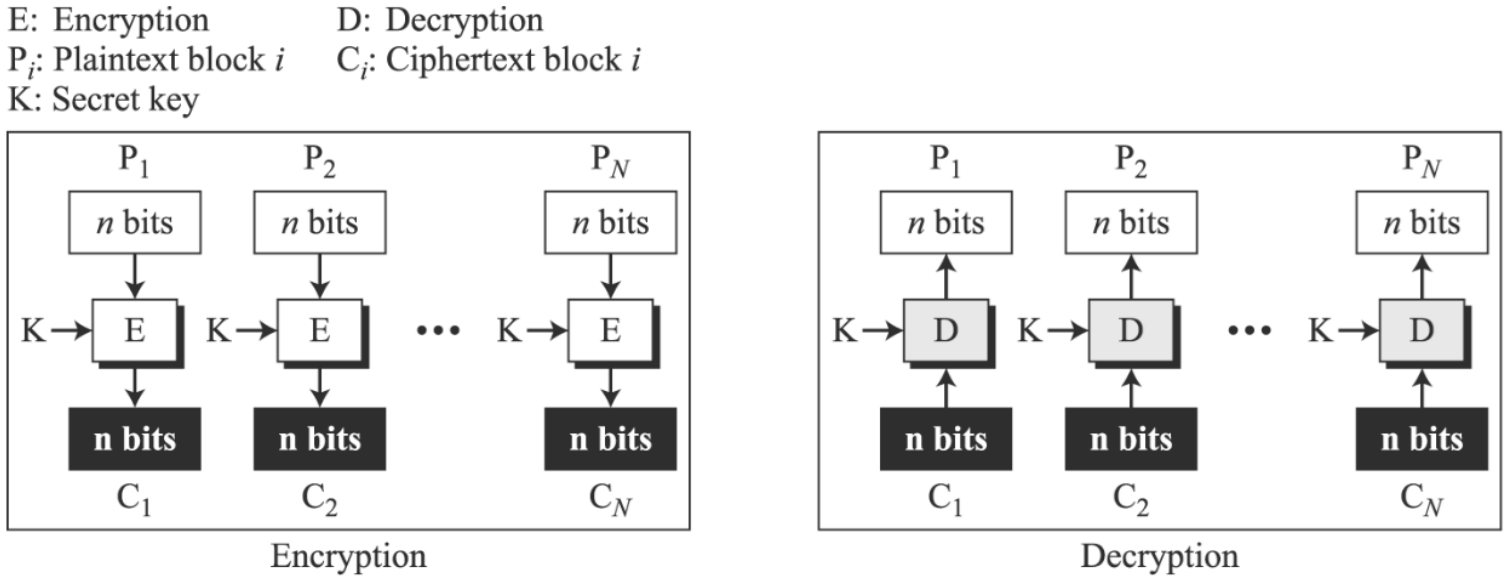
- 현대 블록 암호의 사용
  - 운영 모드
    - 초기 벡터(IV, Initialization Vector)
      - 정의
        - 첫 번째 블록 암호화에 사용하는 초기 블록
      - 특징
        - 첫 번째 블록에 대해 이전 암호문 블록 대신 사용됨
        - 송신자와 수신자가 공유해야 함
        - 비밀일 필요는 없지만 무결성이 보장되어야 함
        - 암호화 시 IV가 달라지면 첫 번째 암호문 블록이 달라짐
        - 복호화 시 IV가 변조되면 첫 번째 평문 블록이 영향을 받음

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - ECB(Electronic Codebook) 모드 (1/7)
      - 정의
        - 평문을 블록 단위로 나누어 각 블록을 서로 독립적으로 암호화하는 가장 단순한 운영 모드
      - 특징
        - 각 블록을 서로 연결하지 않고 독립적으로 암호화하므로 가장 간단한 운영 모드임
        - 평문을 N개의 n비트 블록으로 나누어 처리함
        - 각 블록은 서로 독립적으로 암호화 및 복호화됨
        - 모든 블록에 대해 같은 키를 사용함
        - 평문 길이가 블록 크기의 배수가 아니면 패딩(padding) 이 필요함

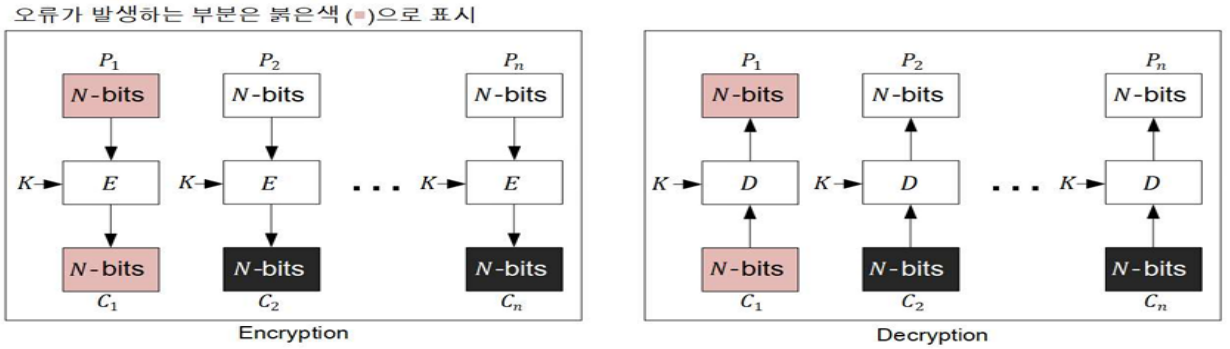
현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - ECB(Electronic Codebook) 모드 (2/7)
      - 동작 과정
        - 암호화: $C_i = E_k(P_i)$
        - 복호화: $P_i = D_k(C_i)$



현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - ECB(Electronic Codebook) 모드 (3/7)
      - 오류 파급
        - 각 블록을 독립적으로 처리하므로, 암호화 또는 복호화 과정에서 특정 블록에 오류가 발생해도 해당 블록에만 영향을 주고 다른 블록으로 전파되지 않음



현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - ECB(Electronic Codebook) 모드 (4/7)
      - 보안성 문제
        1. 블록 단위의 패턴이 유지됨
           - 평문에서 같은 값을 가지는 블록은 대응되는 암호문 블록도 같은 값을 가짐
           - 암호문 블록의 반복을 통해 평문의 반복 패턴을 추측할 수 있음
           - 공격자는 같은 암호문 블록들 중 하나만 전수조사해도 관련 평문 정보를 알아낼 수 있음
        2. 블록 간 독립성이 공격 기회를 제공함
           - ECB 모드에서는 각 블록이 서로 독립적이기 때문에, 공격자는 특정 암호문 블록을 이전에 가로챈 다른 암호문 블록으로 교체할 수 있음
           - 공격자가 암호문 블록을 교체하거나 재배열하여 복호화된 평문 블록을 조작할 수 있음

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - ECB(Electronic Codebook) 모드 (5/7)
      - 알고리즘
        - 평문을 n-비트 단위의 N개 블록으로 분할
        - 각 블록을 같은 키로 독립적으로 암호화
        - 복호화도 같은 키를 사용하여 각 암호문 블록을 독립적으로 복호화

알고리즘 8.1 ECB 모드의 암호화

```
ECB_Encryption (K, Plaintext blocks)
{
    for (i = 1 to N)
    {
        C_i ← E_K (P_i)
    }
    return Ciphertext blocks
}
```

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - ECB(Electronic Codebook) 모드 (6/7)
      - 암호문 훔치기(CTS, Ciphertext Stealing)
        - 정의
          - 패딩 없이 평문과 같은 길이의 암호문을 생성하는 방법
        - 사용 요소
          - P_{N-1}: n비트 평문 블록
          - P_N: m비트 평문 블록 $(m < n)$
          - $head_m$: 왼쪽 최상위 m비트를 선택하는 함수
          - $tail_{n-m}$: 오른쪽 최상위 $n-m$비트를 선택하는 함수
        - 동작 절차
          - $X = E_k(P_{N-1}) \quad \rightarrow C_N = head_m(X)$
            $Y = P_N | tail_{n-m}(X) \rightarrow C_{N-1} = E_K(Y)$
            - P_{N-1}을 암호화하여 임시 블록 X를 생성함
            - X의 앞 m비트는 마지막 암호문 C_N으로 사용함
            - X의 남은 $n-m$비트를 짧은 마지막 평문 P_N 뒤에 붙여 Y를 생성함
            - Y를 암호화하여 C_{N-1}을 생성함

현대 대칭-키 암호를 이용한 암호화 기법

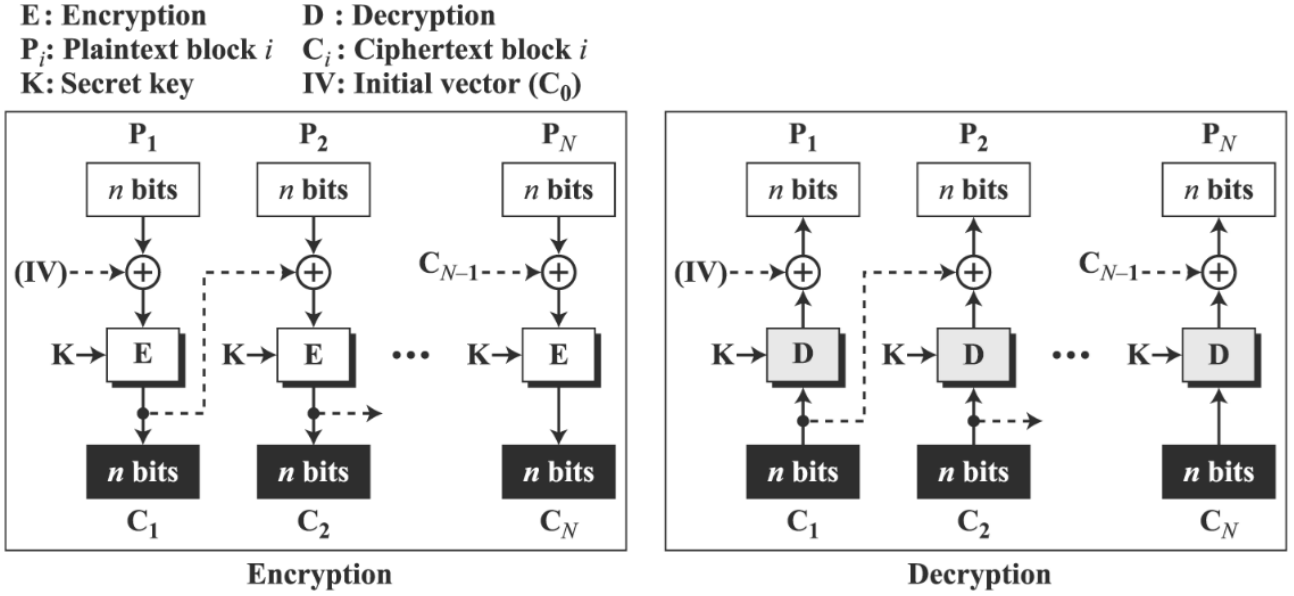
- 현대 블록 암호의 사용
  - 운영 모드
    - ECB(Electronic Codebook) 모드 (7/7)
      - 장점
        - 구조가 단순하고 구현이 쉬움
        - 각 블록을 서로 독립적으로 처리할 수 있음
        - 이전 블록의 결과를 기다리지 않고 각 블록을 따로 암호화할 수 있음
        - 특정 블록에 오류가 발생해도 다른 블록으로 전파되지 않음
      - 단점
        - 같은 평문 블록은 항상 같은 암호문 블록으로 변환됨
        - 평문의 반복 패턴이 암호문에도 남을 수 있음
        - 암호문 블록을 교체하거나 재배열하는 공격에 취약함
        - 평문 길이가 블록 크기의 배수가 아니면 패딩이 필요함

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CBC(Cipher Block Chaining) 모드 (1/7)
      - 정의
        - 각 평문 블록을 이전 암호문 블록과 XOR한 후 암호화하는 운영 모드
      - 특징
        - 첫 번째 블록 암호화에는 초기 벡터(IV, initialization vector) 가 사용됨
        - 같은 평문 블록이라도 이전 암호문 블록에 따라 서로 다른 암호문으로 생성되므로, ECB와 달리 평문의 반복 패턴이 그대로 드러나지 않음
        - 암호화는 이전 암호문 블록에 의존하므로 순차적으로 수행됨
        - 복호화는 각 암호문 블록을 독립적으로 복호화한 뒤 이전 암호문과 XOR함
        - 오류는 현재 블록과 다음 블록까지만 영향을 줌

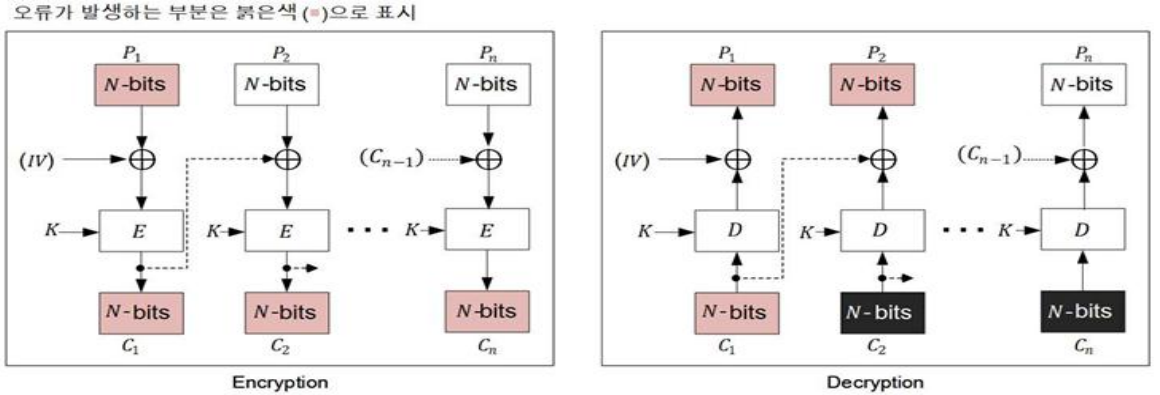
현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CBC(Cipher Block Chaining) 모드 (2/7)
      - 동작 과정
        - 암호화: $C_0 = IV, C_i = E_k(P_i \oplus C_{i-1})$
        - 복호화: $C_0 = IV, P_i = D_k(C_i) \oplus C_{i-1}$



현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CBC(Cipher Block Chaining) 모드 (3/7)
      - 오류 파급
        1. 암호화 과정
           - 평문 블록 P_j에서 오류가 발생하면, 해당 오류는 C_j에 반영됨
           - CBC는 이전 암호문 블록을 다음 블록 암호화에 사용하므로, C_j의 오류가 다음 블록들에도 연쇄적으로 영향을 줄 수 있음
        2. 복호화 과정
           - 암호문 블록 C_j에서 오류가 발생하면,평문 블록 P_j와 P_{j+1}에서는 대부분의 비트에 오류가 발생함



현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CBC(Cipher Block Chaining) 모드 (4/7)
      - 보안성 문제
        1. IV 재사용 문제
           - 같은 키와 같은 IV를 재사용하면 같은 첫 번째 평문 블록은 같은 첫 번째 암호문 블록으로 생성될 수 있음
           - 메시지 사이의 일부 패턴이 드러날 수 있음
        2. 암호문 조작 가능성
           - 암호문이 변조되었는지 확인하는 기능이 없음
           - 공격자가 암호문 블록을 추가하거나 변경할 수 있음

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CBC(Cipher Block Chaining) 모드 (5/7)
      - 알고리즘
        - $C_0 = IV$ 로 초기화
        - 각 평문 블록 P_i에 대해 P_i와 이전 암호문 블록 C_{i-1}를 XOR
        - 그 결과를 키 K로 암호화하여 C_i생성
        - 복호화 시에는 C_i를 복호화한 뒤 C_{i-1}와 XOR하여 P_i복원

알고리즘 8.2 CBC mode의 암호 알고리즘

```
CBC_Encryption (IV, K, Plaintext blocks)
{
    C0 ← IV
    for (i = 1 to N)
    {
        Temp ← Pi ⊕ Ci−1
        Ci ← EK (Temp)
    }
    return Ciphertext blocks
}
```

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CBC(Cipher Block Chaining) 모드 (6/7)
      - 암호문 훔치기(CTS, Ciphertext Stealing)
        - 사용 요소
          - P_{N-1}: n-비트 블록
          - P_N :m-비트 블록 $(m < n)$
          - C_{N-2}: 이전 암호문 블록
          - $pad_{n-m}(0)$: 0 덧붙이기 함수
          - $head_m$: 상위 m비트 선택 함수
        - 동작 절차
          - $U = P_{N-1} \oplus C_{N-2} \quad \rightarrow X = E_K(U) \rightarrow C_N = head_m(X)$
            $V = P_N | pad_{n-m}(0) \rightarrow Y = X \oplus V \rightarrow C_{N-1} = E_K(Y)$
            - P_{N-1}과 C_{N-2}를 이용해 임시 블록 X를 생성함
            - X의 일부를 마지막 암호문 C_N으로 사용함
            - 짧은 마지막 평문 P_N을 보완한 뒤 X와 결합하여 C_{N-1}을 생성함
            - 이를 통해 패딩 없이 평문과 같은 길이의 암호문을 생성함

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CBC(Cipher Block Chaining) 모드 (7/7)
      - 장점
        - 같은 평문 블록이라도 이전 암호문 블록이 다르면 서로 다른 암호문 블록이 생성됨
        - 패턴 노출이 줄어들어 통계적 분석과 패턴 기반 공격에 더 강함
        - 복호화는 이전 암호문 블록만 있으면 각 블록을 독립적으로 계산할 수 있음
        - 병렬 처리가 가능하여 여러 블록을 동시에 처리할 수 있음
      - 단점
        - 암호문 오류가 현재 평문 블록과 다음 평문 블록에 영향을 줌
        - 암호화는 이전 암호문 블록이 먼저 생성되어야 하므로 순차적으로 처리해야 함
        - 암호화는 병렬 처리가 어려워 대량 데이터 처리 시 속도 측면에서 불리함

현대 대칭-키 암호를 이용한 암호화 기법

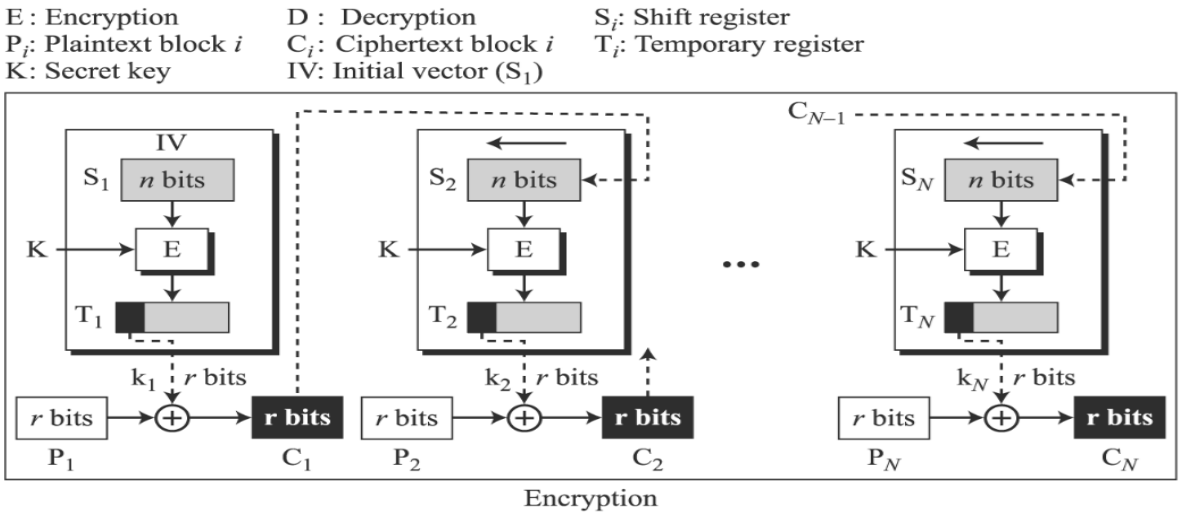
- 현대 블록 암호의 사용
  - 운영 모드
    - CFB(Cipher Feedback) 모드 (1/7)
      - 정의
        - 이전 암호문에 의해 갱신된 값을 암호화하여 만든 키 스트림을 평문과 XOR하는 운영모드
      - 특징
        - 블록 암호로 키 스트림을 만들고, 그 키 스트림을 평문과 XOR해서 암호화함
        - 키 스트림이 이전 암호문 블록에 의해 결정되므로 비동기식 스트림 암호처럼 동작함
          - 이전 암호문 블록을 블록 암호의 입력으로 사용하여 다음 키 스트림을 생성함 따라서 키 스트림이 암호문 흐름에 따라 갱신되므로 비동기식 스트림 암호처럼 동작함
        - 블록 암호 출력 중 일부 r비트만 사용하여 비트 단위나 문자 단위 처리가 가능함
        - 평문 길이만큼 키 스트림을 생성해 XOR하므로 블록 크기에 맞추는 패딩이 필요 없음
        - 초기 이동 레지스터 값으로 IV를 사용함
        - 암호화와 복호화 모두 이전 암호문 블록을 이용하므로 순차적 처리해야 함

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CFB(Cipher Feedback) 모드 (2/7)

$ShiftLeft_r$: r만큼 왼쪽 쉬프트
$SelectLeft_r$: 상위 r비트 선택

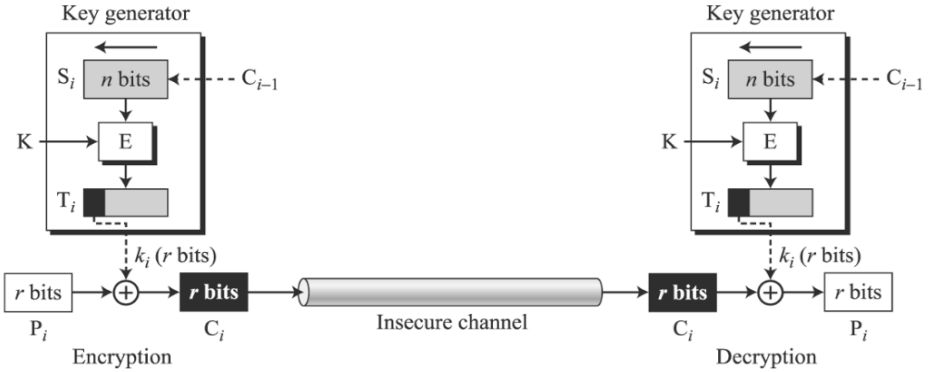
      - 동작 과정
        - 암호화:$C_i = P_i \oplus SelectLeft_r\{E_k[ShiftLeft_r(S_{i-1}) \mid C_{i-1}]\}$
        - 복호화:$P_i = C_i \oplus SelectLeft_r\{E_k[ShiftLeft_r(S_{i-1}) \mid C_{i-1}]\}$



- 이전 암호문 블록을 이용해 키 스트림 K_i를 생성함
- 암호화와 복호화는 평문 블록과 암호문 블록의 역할만 바뀔 뿐 구조는 동일함
- 복호화 시에도 블록 암호의 암호화 함수를 사용함

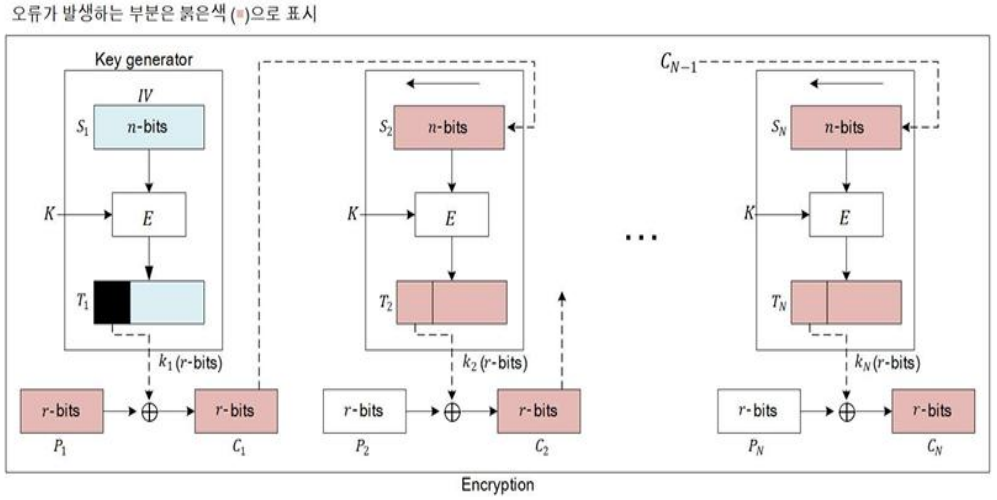
현대 대칭-키 암호를 이용한 암호화 기법

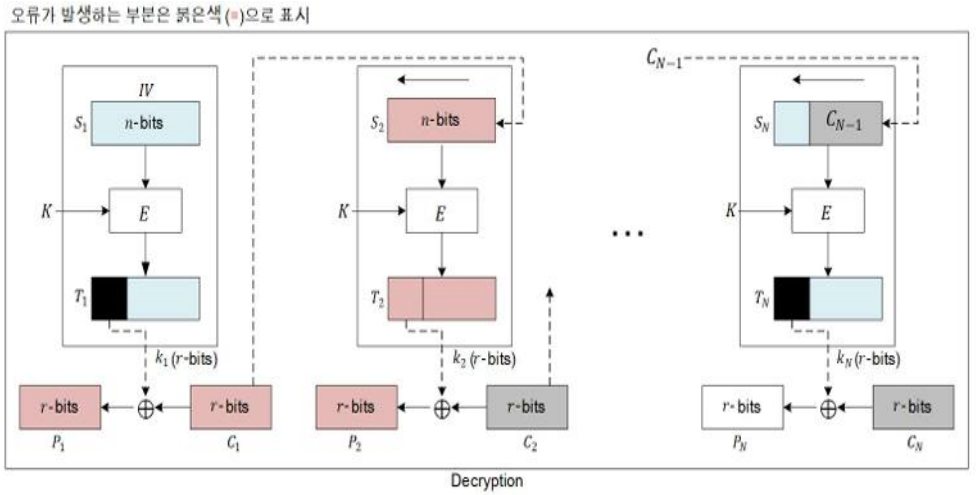
- 현대 블록 암호의 사용
  - 운영 모드
    - CFB(Cipher Feedback) 모드 (3/7)
      - 스트림 암호로서 CFB 모드
        - CFB 모드는 DES나 AES와 같은 블록 암호를 이용하는 운영 모드이지만, 결과적으로는 스트림 암호처럼 동작함
        - 생성되는 키스트림은 이전 암호문에 의존하므로 비동기식 스트림 암호에 해당함
        - 암호화와 복호화 과정에서 블록 암호와 이전 암호문 블록 Ci–1이 키스트림 생성에 사용됨
        - 생성된 키스트림의 일부를 평문 또는 암호문과 XOR하여 암호화와 복호화를 수행함



현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CFB(Cipher Feedback) 모드 (4/7)
      - 오류 파급
        - 암호화에서는 한 평문 블록에 오류가 발생하면, 현재 암호문 블록과 이후 모든 암호문 블록에 오류가 발생함
        - 복호화에서는 한 암호문 블록에 오류가 발생하면, 현재 평문 블록과 이후 이동 레지스터에 오류가 존재하는 동안의 평문 블록에 오류가 발생함





현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CFB(Cipher Feedback) 모드 (5/7)
      - 보안성 문제
        1. *IV* 재사용 문제
           - 같은 키와 같은 IV를 재사용하면 첫 번째 키스트림이 동일하게 생성됨
           - 같은 첫 번째 평문 블록은 같은 첫 번째 암호문 블록이 될 수 있음
           - 메시지 사이의 일부 패턴이나 시작 부분의 유사성이 드러날 수 있음
        2. 암호문 조작 가능성
           - 암호문이 변조되었는지 확인하는 기능이 없음
           - 공격자가 암호문 일부를 바꾸면, 복호화된 평문도 일부 바뀔 수 있음

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CFB(Cipher Feedback) 모드 (6/7)
      - 알고리즘

T:임시 값/ 임시 레지스터

- $S_1 = IV$ 로 초기화
- S_i를 블록 암호로 암호화 하여 T 생성함
- T의 상위 r비트를 선택하여 키스트림 생성함
- P_i와 k_i를 XOR하여 C_i생성함
- C_i 를 이동 레지스터에 피드백하여 S_{i+1}를 구성함
- S_{i+1}는 다음 키 스트림 k_{i+1} 생성에 사용됨

알고리즘 8.3 CFB 모드의 암호 알고리즘

```
CFB_Encryption (IV, K, r)
{
  i ← 1
  while (more blocks to encrypt)
  {
  input (P_i)
  if (i = 1)
    S ← IV
  else
    {
    Temp ← shiftLeft_r (S)
    S ← concatenate (Temp, C_i−1)
    }
  T ← E_K(S)
  k_i ← selectLeft_r (T)
  C_i ← P_i ⊕ k_i
  output (C_i)
  i ← i + 1
  }
}
```

현대 대칭-키 암호를 이용한 암호화 기법

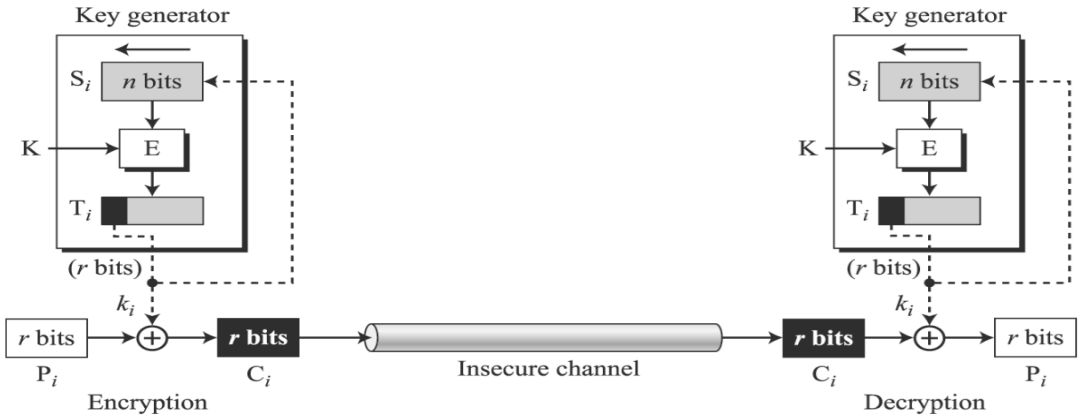
- 현대 블록 암호의 사용
  - 운영 모드
    - CFB(Cipher Feedback) 모드 (7/7)
      - 장점
        - 비트 단위, 문자 단위 등 작은 단위 처리가 가능함
        - 패딩이 필요 없음
        - 큰 블록 전체를 기다리지 않아도 되어 실시간 처리가 가능함
        - 같은 평문이라도 동일한 암호문으로 고정되지 않아 패턴 노출이 줄어듦
      - 단점
        - 이전 암호문에 의존하므로 병렬 처리에 불리함
        - 같은 키를 사용할 경우 매 메시지마다 다른 IV가 필요함
        - 암호문 오류가 발생하면 이후 몇 블록까지 오류가 전파될 수 있음
        - 작은 단위마다 블록 암호를 반복 호출하므로 처리 효율이 낮을 수 있음

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - OFB(Output Feedback) 모드 (1/6)
      - 정의
        - 키스트림을 생성하여 평문과 XOR하는 방식의 운영 모드
      - 특징
        - 키스트림이 평문이나 암호문과 독립적이므로 동기식 스트림 암호에 해당함
        - 생성된 키스트림을 평문 또는 암호문과 XOR하여 암호화 및 복호화를 수행함
        - 초기 키스트림 생성을 위해 IV가 필요함
        - 암호화 및 복호화 모두 키스트림 생성은 순차 처리됨
        - 키스트림을 미리 생성할 수 있어 XOR 단계는 빠르게 처리 가능함
        - 암호문에 오류가 생겨도 대응되는 평문 비트에만 영향을 줌

현대 대칭-키 암호를 이용한 암호화 기법

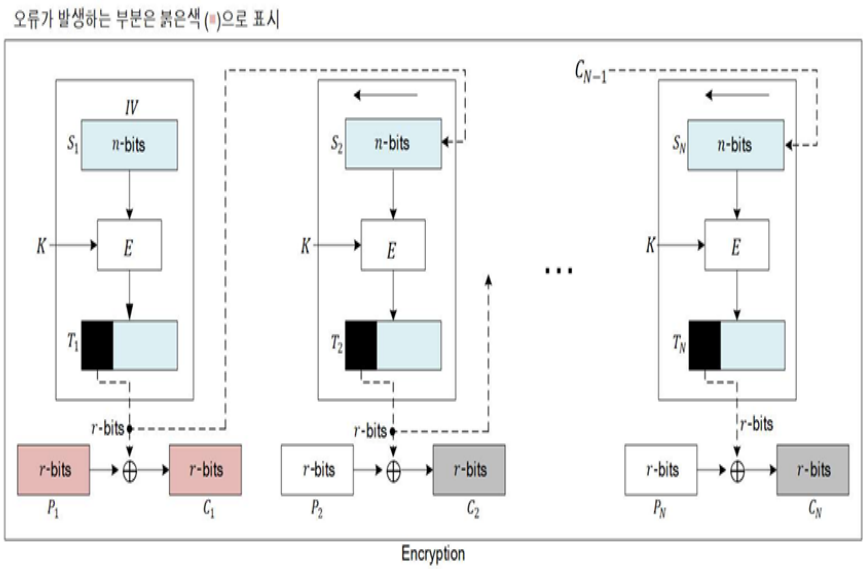
- 현대 블록 암호의 사용
  - 운영 모드
    - OFB(Output Feedback) 모드 (2/6)
      - 동작과정
        - 암호화:$S_0 = IV, T_i = E_k(S_{i-1}), k_i = SelectLeft_r(T_i), C_i = P_i \oplus k_i$
        - 복호화:$S_0 = IV, T_i = E_k(S_{i-1}), k_i = SelectLeft_r(T_i), P_i = C_i \oplus k_i$

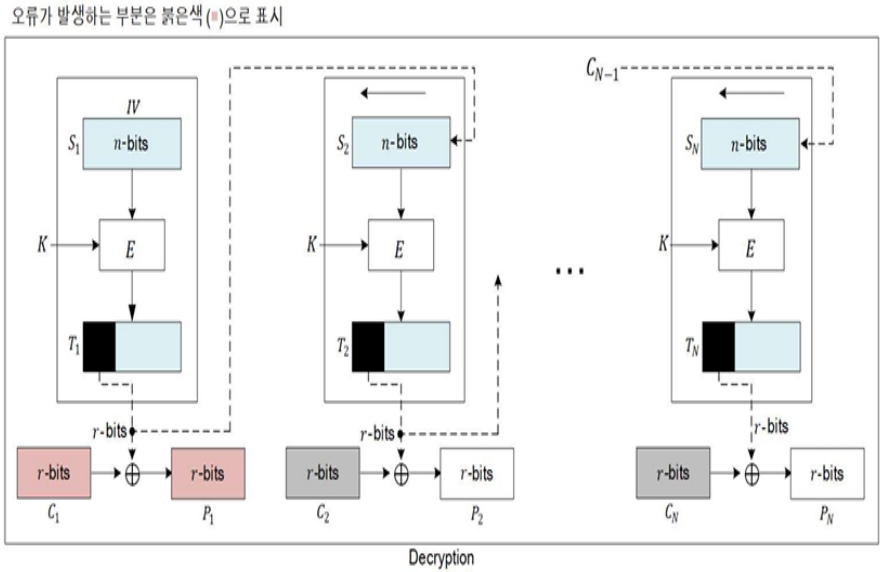


- IV에서 시작해 키스트림 k_i를 순차적으로 생성함
- 생성된 k_i를 평문 또는 암호문과 XOR하여 암호화 및 복호화를 수행함
- 키스트림은 평문 및 암호문과 독립적으로 생성됨

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - OFB(Output Feedback) 모드 (3/6)
      - 오류 파급
        - 암호화: 한 평문 블록에 오류가 발생하면, 현재 암호문 블록에만 오류가 발생함
        - 복호화: 한 암호문 블록에 오류가 발생하면, 현재 평문 블록에만 오류가 발생함





현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - OFB(Output Feedback) 모드 (4/6)
      - 보안상 문제
        1. *IV* 재사용 문제
           - 같은 키와 같은 IV를 재사용하면 같은 키스트림이 생성됨
           - 평문 사이의 관계가 노출될 수 있음
        2. 암호문 조작 가능성
           - OFB 모드 자체에는 암호문이 중간에 변조되었는지 확인하는 기능이 없음
           - 공격자가 암호문 비트를 변경하면 평문의 같은 위치 비트도 변경됨

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - OFB(Output Feedback) 모드 (5/6)
      - 알고리즘
        - $S_1 = IV$ 로 초기화
        - S_i 를 블록 암호의 암호로 암호화 하여 T생성
        - T의 상위 r비트를 선택하여 키스트림 k_i생성
        - P_i 와 XOR하여 C_i생성
        - 이전 암호문이 아니라 이전 출력값을 피드백하여 다음 단계를 진행함

알고리즘 8.4 OFB 모드에 대한 암호화 알고리즘

```
OFB_Encryption (IV, K, r)
{
    i ← 1
    while (more blocks to encrypt)
    {
        input (P_i)
        if (i = 1)   S ← IV
        else
        {
            Temp ← shiftLeft_r (S)
            S ← concatenate (Temp, k_{i-1})
        }
        T ← E_K (S)
        k_i ← selectLeft_r (T)
        C_i ← P_i ⊕ k_i
        output (C_i)
        i ← i + 1
    }
}
```

현대 대칭-키 암호를 이용한 암호화 기법

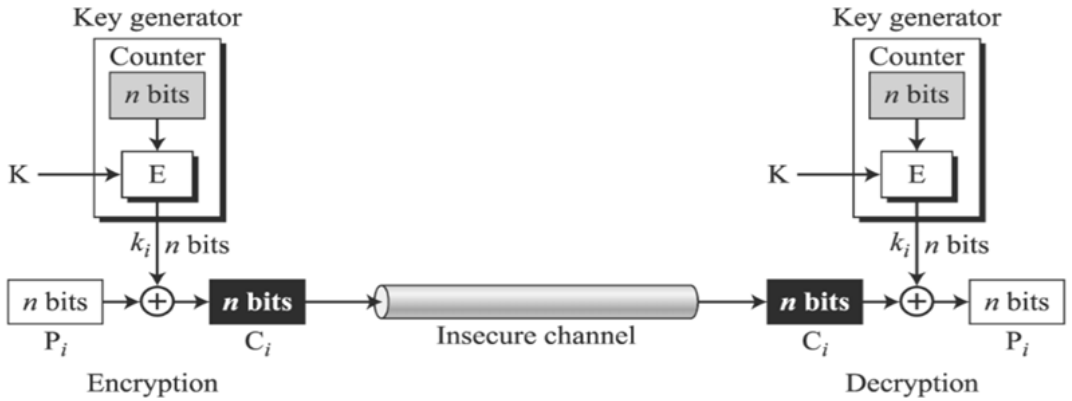
- 현대 블록 암호의 사용
  - 운영 모드
    - OFB(Output Feedback) 모드 (6/6)
      - 장점
        - 키스트림을 사용하므로 평문 패턴 노출이 적음
        - 암호문의 오류가 다른 평문 블록으로 전파되지 않음
        - 키스트림을 미리 생성할 수 있어 실시간 처리에 유리함
      - 단점
        - 키스트림 생성은 이전 출력 블록에 의존하므로 병렬 처리에 불리함
        - 암호문 비트가 조작되면 대응되는 평문 비트도 변경됨
        - 무결성 검증 기능이 없어 암호문 조작을 통한 메시지 위조가 가능함

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CTR(Counter) 모드 (1/8)
      - 정의
        - 카운터 값을 암호화해 만든 키 스트림을 평문과 XOR하여 암호화하는 운영 모드
      - 특징
        - 각 블록마다 고유한 카운터 값을 사용함
        - 블록 간 독립성이 존재함
        - 암호화와 복호화 모두 병렬 처리가 가능함
        - 생성된 키 스트림을 필요한 길이만큼 사용하므로 패딩이 필요하지 않음
        - IV로 첫 카운터 값을 다르게 설정하여 같은 키 스트림이 반복되지 않도록 함
        - 블록 암호를 사용하지만 결과적으로는 스트림 암호처럼 동작함

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CTR(Counter) 모드 (2/8)
      - 스트림 암호로서의 CTR 모드
        - 블록 암호를 이용하지만 스트림 암호처럼 동작함
        - 각 블록은 서로 다른 카운터 값으로부터 키 스트림 생성함
        - 키 스트림이 이전 암호문과 독립적임
        - 생성된 키 스트림을 평문과 암호문에 XOR하여 처리함

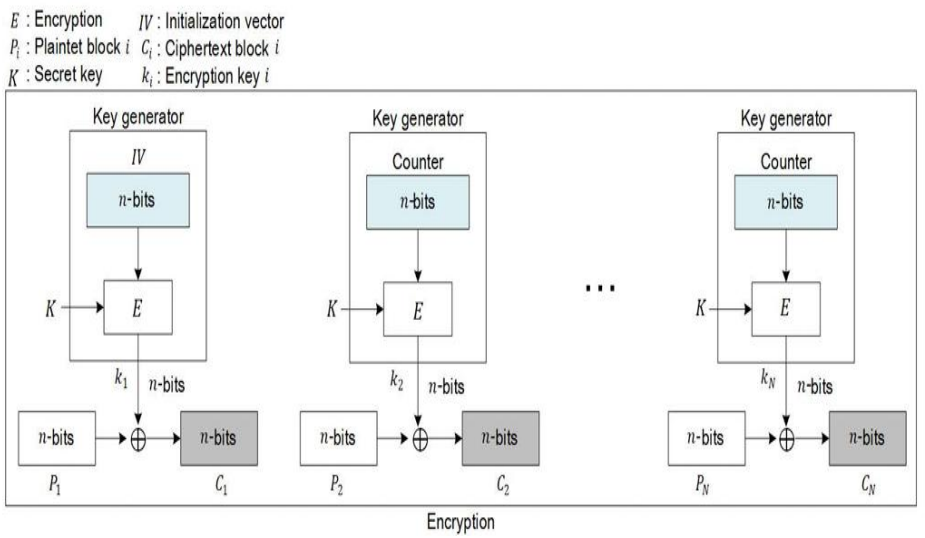


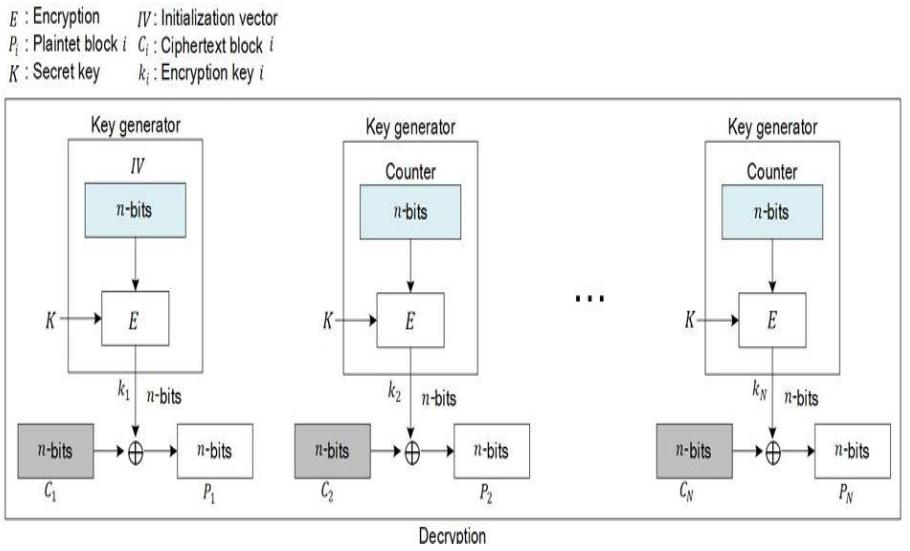
현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CTR(Counter) 모드 (3/8)
      - 카운터(Counter)
        - 블록마다 서로 다른 키스트림을 만들기 위해 사용하는 값
        - 일반적으로 블록이 처리될 때마다 1씩 증가시킴
        - 카운터 값은 블록마다 달라야 하며, 같은 키에서 반복되면 안 됨

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CTR(Counter) 모드 (4/8)
      - 동작과정
        - 암호화: $C_i = P_i \oplus E_{ki}\ (Counter)$
        - 복호화: $P_i = C_i \oplus E_{ki}\ (Counter)$
          - 복호화 시에도 암호화 함수를 사용함



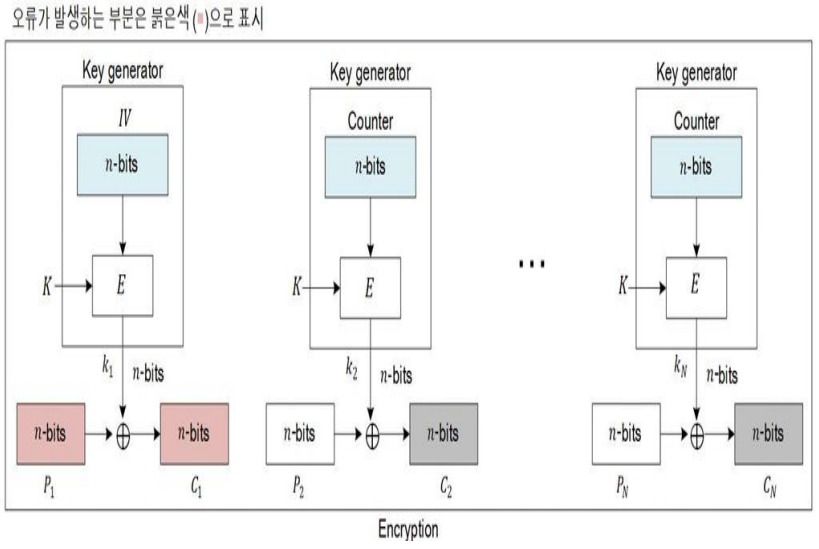


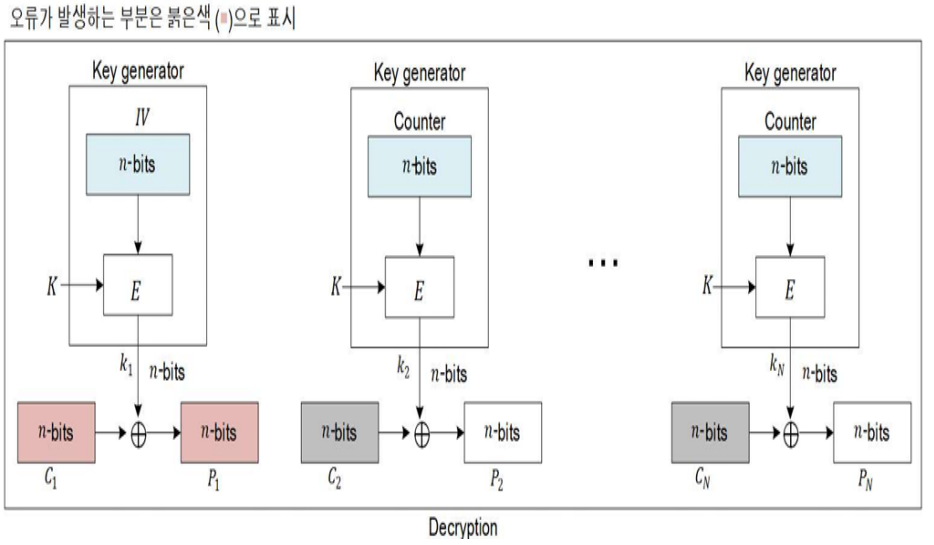
현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CTR(Counter) 모드 (5/8)
      - 보안성 문제
        1. 카운터 값 재사용 문제
           - 같은 키와 같은 카운터 값을 재사용하면 같은 키스트림이 생성됨
           - 같은 키스트림이 반복되면 평문 사이의 관계가 노출될 수 있음
        2. 암호문 조작 가능성
           - 암호문이 변조되었는지 확인하는 기능이 없음
           - 공격자가 암호문 비트를 변경하면 대응되는 평문 비트도 변경될 수 있음

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CTR(Counter) 모드 (6/8)
      - 오류 파급
        - 암호화:한 평문 블록에 오류가 발생하면, 현재 암호문 블록에만 오류가 발생함
        - 복호화:한 암호문 블록에 오류가 발생하면, 현재 평문 블록에만 오류가 발생함





현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CTR(Counter) 모드 (7/8)
      - 알고리즘
        - Counter $= IV$로 초기화
        - 블록마다 Counter 증가
        - Counter를 암호화하여 키 스트림 생성
        - 생성된 키 스트림을 평문과 XOR하여 암호문 생성
        - 복호화도 동일한 방식으로 수행

알고리즘 8.5 CTR 모드의 암호 알고리즘

```
CTR_Encryption (IV, K, Plaintext blocks)
{
    Counter ← IV
    for (i = 1 to N)
    {
        Counter ← (Counter + i − 1) mod 2^N
        k_i ← E_K (Counter)
        C_i ← P_i ⊕ k_i
    }
    return Ciphertext blocks
}
```

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드
    - CTR(Counter) 모드 (8/8)
      - 장점
        - 각 블록이 서로 독립적이므로 암호화와 복호화 모두 병렬 처리할 수 있음
        - 여러 블록을 동시에 처리할 수 있어 속도 면에서 유리함
        - 이전 암호문 블록에 의존하지 않아 랜덤 접근에 유리함
        - 키 스트림을 생성해 XOR하므로 오류 파급이 매우 작음
        - 필요한 길이만큼만 사용 가능하여 패딩이 필요 없음
      - 단점
        - 카운터 값이 재사용되면 같은 키 스트림이 반복되어 보안성이 크게 약화될 수 있음
        - 암호문이 임의로 변경되면 대응되는 평문에도 직접 영향을 줄 수 있음

현대 대칭-키 암호를 이용한 암호화 기법

- 현대 블록 암호의 사용
  - 운영 모드간 비교

운영모드	패딩 필요	*IV* 사용	블록간 관계	병렬 처리	스트림 암호로 변경	동기식/ 비동기식
ECB	O	X	독립적	암호화, 복호화	X	-
CBC	O	O	암호문 의존	복호화	X	-
CFB	X	O	암호문 의존	복호화	O	비동기식 스트림암호
OFB	X	O	이전 키 스트림 의존	불가	O	동기식 스트림 암호
CTR	X	O	독립적	암호화, 복호화	O	동기식 스트림 암호

현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - 스트림 암호(Stream Cipher)
    - 정의
      - 평문 스트림과 키스트림을 XOR하여 암호문 스트림을 생성하는 암호 방식
    - 특징
      - 비트 또는 바이트 단위의 데이터를 암호화하기 위해 사용함
      - 키 스트림을 생성한 뒤 평문과 XOR하여 암호화함
      - 블록 암호보다 빠른 속도를 가지며 실시간 처리에 유리함
    - 종류
      - RC4
      - A5/1

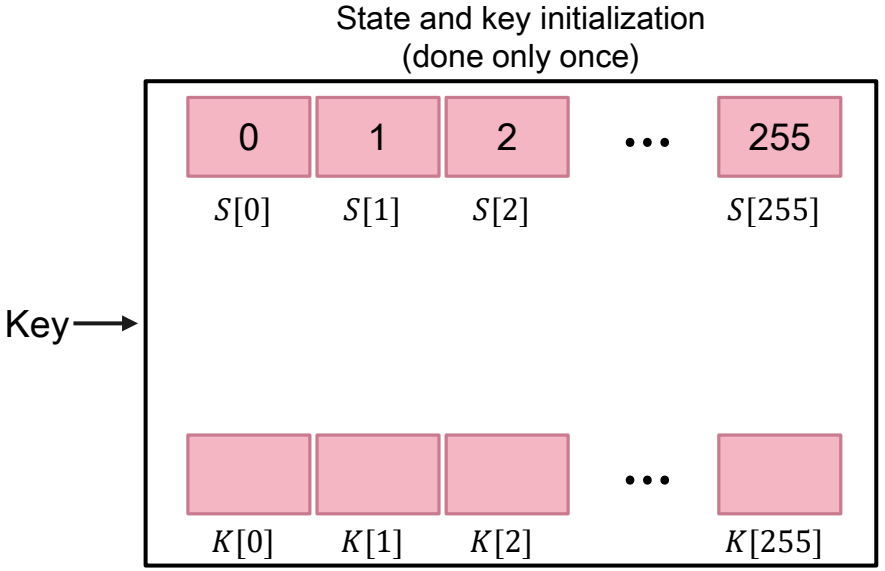
현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - 스트림 암호(Stream Cipher)
    - RC4 (1/6)
      - 정의
        - 256바이트 상태 배열을 기반으로 키스트림을 생성하고 이를 평문과 XOR하여 암호화하는 스트림 암호
      - 상태(state)
        - RC4는 256바이트로 구성된 상태 배열 $S[0], S[1], \ldots, S[255]$을 사용함
        - 각 원소는 0부터 255 사이의 1바이트 값을 가짐
        - 이 상태 배열을 계속 치환하며 키스트림을 생성함

현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - 스트림 암호(Stream Cipher)
    - RC4 (2/6)
      - 동작과정

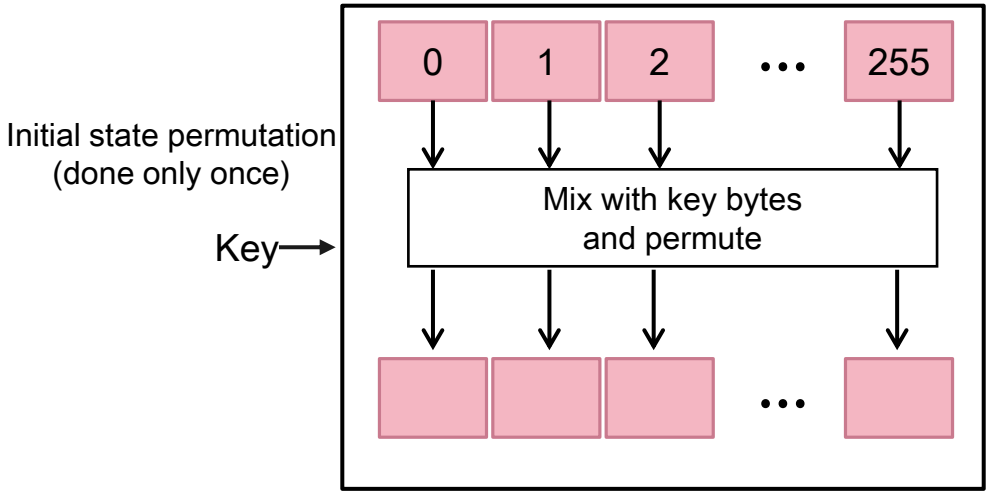
1. 상태 배열 S 초기화
   - $S[i]$ 는 0부터 255까지 순차적으로 초기화됨
   - $K[i]$ 는 256바이트가 될 때까지 비밀 키를 반복 저장함




```
for (i=0 to 255)
{
    S[i] <- i
    K[i] <- Key [i mod Key Length]
}
```

현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - 스트림 암호(Stream Cipher)
    - RC4 (3/6)
      2. KSA(Key Scheduling Algorithm)를 통한 초기 치환
         - $j = 0$ 으로 초기화함
         - $i = 0$ 부터 255까지 반복하며 $j = (j + S[i] + K[i])\ mod 256$ 계산
         - $S[i]$와 $S[j]$를 교환하여 상태 배열을 섞음
         - 이 과정이 끝나면 S는 키에 따라 초기 치환된 상태가 됨




```
j <- 0
for(i=0 to 255)
{
    j<-(j+S[i]+K[i]) mod 256
    swap(S[i], S[j])
}
```

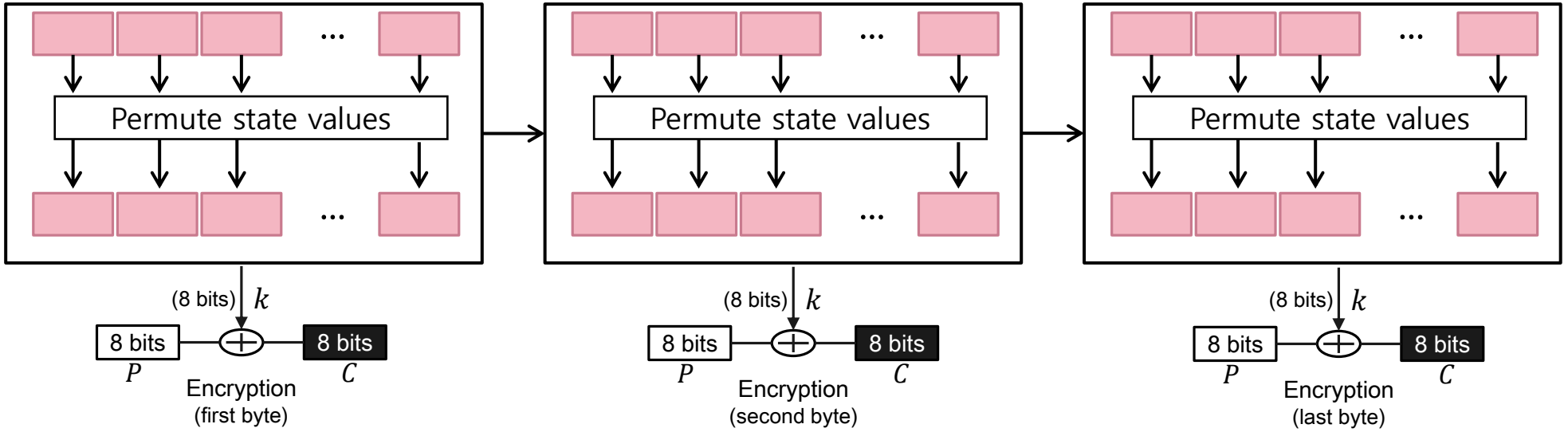
현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - 스트림 암호(Stream Cipher)
    - RC4 (4/6)
      3. 키스트림 생성
         - i 와 j 값을 갱신함
         - $S[i]$ 와 $S[j]$를 교환함
         - $S[(S[i]+S[j])mod256]$ 값을 키 스트림 바이트 k로 출력함
         - 이 과정을 반복하여 키 스트림을 바이트 k를 연속적으로 생성함

```
i <- (i+1) mod 256
j <- (j + S[i] mod 256
swap (S[i], S[j])
k <- S[(S[i]+S[j]) mod 256]
```

현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - 스트림 암호(Stream Cipher)
    - RC4 (5/6)
      4. 암호화 및 복호화
         - 암호화: 키 스트림 k와 평문 P를 XOR하여 암호문 C 생성
           $C = P \oplus k$
         - 복호화: 같은 키 스트림 k와 암호문 C를 XOR하여 평문 P 복원
           $P = C \oplus k$
         - XOR 연산의 성질 때문에 암호화와 복호화는 동일한 구조를 가짐



현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - 스트림 암호(Stream Cipher)
    - RC4 (6/6)
      - 보안성 문제
        - 비밀 키 길이가 128비트(16바이트)보다 작으면 안전하지 않다고 여겨짐
        - RC4 사용 시 충분히 긴 키를 사용하는 것이 권장됨
        - 세션마다 서로 다른 키를 사용하는 것이 권장됨

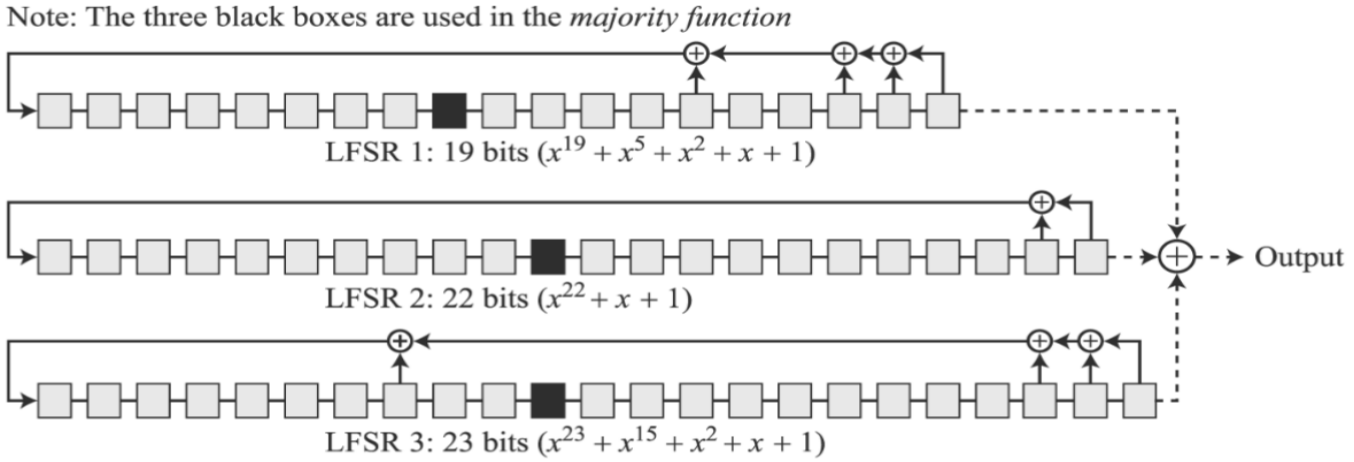
현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - 스트림 암호(Stream Cipher)
    - A5/1 (1/5)
      - 정의
        - GSM 휴대전화 통신에서 음성 데이터를 암호화하기 위해 사용된 스트림 암호
      - 특징
        - GSM 이동통신에서 사용된 스트림 암호
        - 여러 개의 LFSR을 이용해 키 스트림 생성함
        - 생성된 키 스트림을 평문과 XOR하여 암호화함
        - 실시간 통신에 적합함

현대 대칭-키 암호를 이용한 암호화 기법

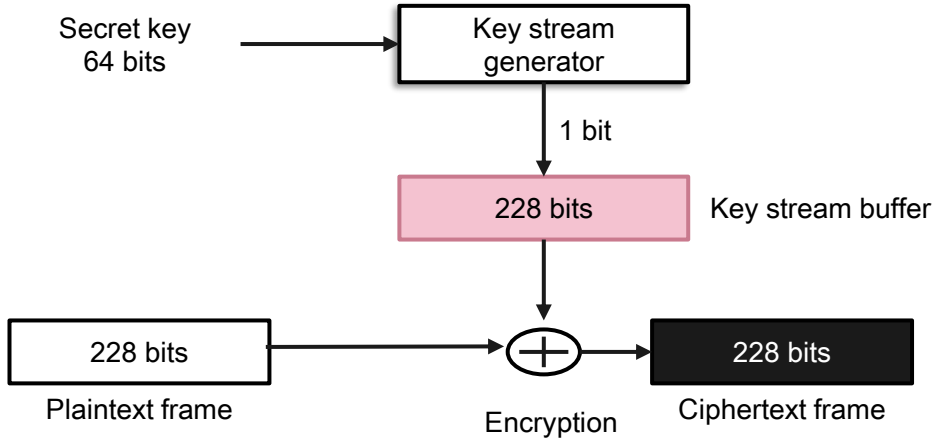
- 스트림 암호의 사용
  - 스트림 암호(Stream Cipher)
    - A5/1 (2/5)
      - 키 생성기
        - 길이가 19, 22, 23인 3개의 LFSR을 사용함
        - 각 LFSR은 고유한 특성 다항식을 가지며, 클로킹 비트를 기준으로 이동 여부가 결정됨
        - 세 LFSR의 출력이 결합되어 최종 키 스트림 비트를 생성함
        - 생성된 키스트림은 228비트 버퍼에 저장되어 평문 프레임과 XOR됨

클로킹
LFSR의 비트들을 한 단계 이동시키고 상태를 갱신하는 동작



현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - 스트림 암호(Stream Cipher)
    - A5/1 (3/5)
      - 동작과정
        1. 초기화
           - 초기화는 64비트 비밀 키와 22비트 프레임 번호를 사용함
           - 먼저 세 LFSR의 모든 셀을 0으로 초기화함
           - 64비트 키를 한 비트씩 각 LFSR에 섞으며 모든 레지스터를 클로킹함
           - 이후 22비트 프레임 번호를 이용해 같은 과정을 반복함
           - 마지막으로 100회 클로킹하여 내부 상태를 충분히 섞음



현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - 스트림 암호(Stream Cipher)
    - A5/1 (4/5)
      2. 키스트림 비트 생성
         - 매 단계마다 먼저 majority 함수를 계산함
           - 클로킹 비트 중 둘 이상이 1이면 1, 아니면 0을 출력함
         - majority 값과 일치하는 LFSR만 클로킹함
         - 클로킹 후 각 LFSR의 출력 비트를 XOR하여 1비트 키 스트림을 생성함
         - 이 과정을 반복하여 전체 키 스트림을 만듦

현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - 스트림 암호(Stream Cipher)
    - A5/1 (5/5)
      3. 암호화
         - 생성된 키 스트림은 228비트 버퍼에 저장됨
         - 평문 프레임과 키 스트림을 XOR하여 암호문 프레임을 생성함
      4. 복호화
         - 복호화도 같은 키 스트림과 XOR하여 수행함
         - 암호화와 복호화는 동일한 XOR 구조를 가짐

      - 보안성 문제
        - A5/1에 대해서는 여러 공격 방법이 존재함
        - 대표적으로 기지 평문 공격이 존재함
        - 현대 기준에서는 보안성이 약한 편임

현대 대칭-키 암호를 이용한 암호화 기법

- 키 관리와 키 생성
  - 키 관리
    - 대칭 키 암호에서는 안전한 통신을 위해 송신자와 수신자가 비밀 키를 공유해야 함
    - 구성원이 n명일 경우, 서로 통신하기 위해 필요한 비밀 키 수는 $n(n-1)/2$
  - 키 생성
    - 대칭 키 암호 환경에서는 비밀 키를 안전하게 생성해야 함
    - 키를 선택할 때는 비밀요소가 누출되는 것을 막기 위해 예측 가능성이 없어야 함
    - 따라서 키는 난수 생성기를 이용해 생성해야 함
    - 세션 키 역시 충분한 무작위성을 가져야 함

Thanks!

김 민 식(ms6127@naver.com)