2024/07/11, 2024 보안 기초 세미나

암호학과 네트워크 보안

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

이 정 민(jeongmin@pel.sejong.ac.kr)

세종대학교 프로토콜공학연구실

목 차

- 현대 대칭-키 암호
- 현대 대칭-키 암호를 이용한 암호화 기법

목 차

- 현대 대칭-키 암호
- 현대 대칭-키 암호를 이용한 암호화 기법

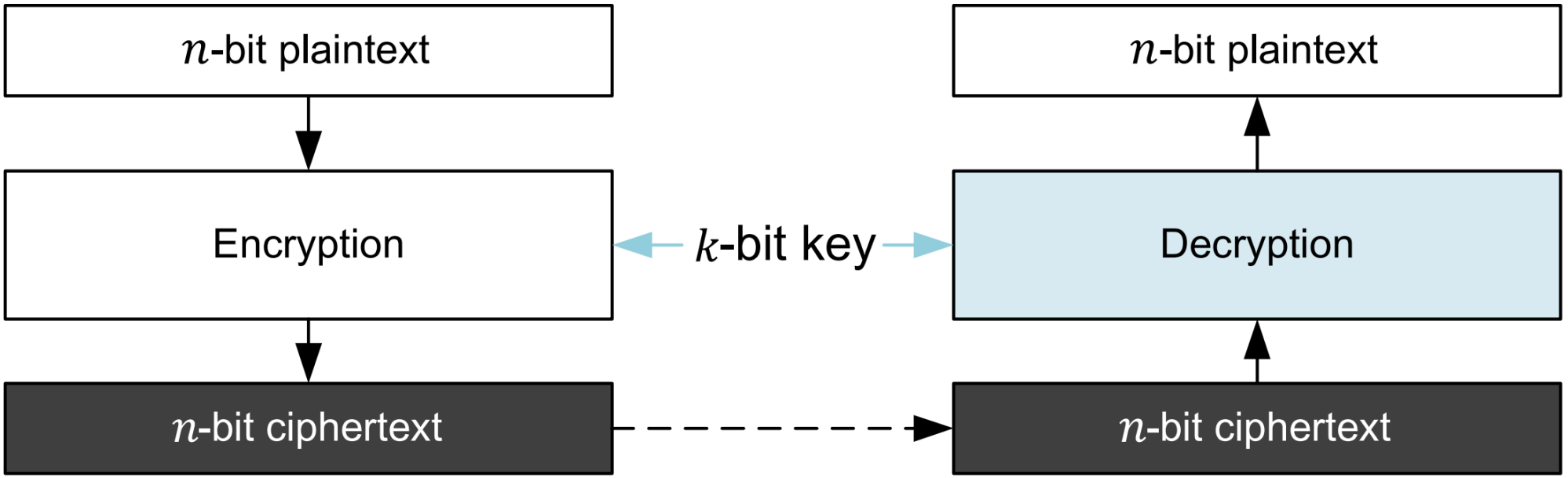
현대 대칭-키 암호

- 개요
  - 고전 대칭 키 암호는 문자 기반(character-oriented) 암호
  - 현대 대칭 키 암호는 비트 기반(bit-oriented) 암호
    - 심볼 길이가 증가(8배 또는 16배)함
    - 많은 심볼을 섞어 안전성이 증가함
    - e.g., 문자 'A' 를 01000001_2로 처리
      - 심볼 'A'가 심볼 0과 1로 변경됨

인코딩 이름	문자 당 바이트 수	비고
UTF-8	가변 (1~6 bytes)	비윈도우 계열 char
UTF-16 or UCS2	2 bytes	윈도우 wchar_t
UTF-32 or UCS4	4 bytes	비윈도우 계열 wchar_t
UTF16 BE	2 bytes	빅 엔디안 시스템
UTF32 BE	4 bytes	빅 엔디안 시스템

현대 대칭-키 암호

- 현대 블록 암호
  - n-비트 평문을 암호화하거나 n-비트 암호문을 복호화 함
    - 메시지 길이가 n-비트보다 작으면 패딩(padding)을 추가함
  - 암호화, 복호화 알고리즘에 k-비트 키를 사용함
  - 복호화는 암호화의 역함수임



현대 대칭-키 암호

- 현대 블록 암호
  - 예제 1

    prob) 8bit ASCII 코드에 대해 64bit 블록 암호를 사용할 때, 100개 문자로 구성된 메시지는 몇 비트의 패딩이 추가되는가?

    sol) 메시지는 총 800bit이며 이는 64bit로 나누어 떨어져야 함.
    메시지의 크기 |M|과 패딩 크기 |pad| 에 대해,

$$
\begin{aligned}
&|M| + |\text{pad}| = 0 \bmod 64 \\
\Rightarrow\ &|\text{pad}| = -800 \bmod 64 \\
\Rightarrow\ &|\text{pad}| = 32 \bmod 64
\end{aligned}
$$

    $\therefore$ 패딩 32bit가 필요 $(832 = 64 \times 13)$

현대 대칭-키 암호

- 현대 블록 암호
  - 현대 블록 암호는 대치 혹은 전치로 동작하도록 설계됨

  - 대치(substitution)
    - 평문 비트의 값을 임의로 대체함
      - e.g., 0을 1로, 1을 0으로 바꿈
    - n 비트 블록에서 2^n의 경우의 수를 가짐

  - 전치(transposition)
    - 평문 비트를 재배열함
      - e.g., 2번째 비트와 5번째 비트의 위치를 바꿈
    - n 비트 블록에서 $n!$의 경우의 수를 가짐

현대 대칭-키 암호

- 현대 블록 암호
  - 예제 2

    prob) 64bit 블록 암호인 암호문에서 1의 개수가 10개라면 평문을 도출하기 위해 필요한 시행착오(trial-and-error test)는 몇 번인가? 초당 10억번 연산 시, 얼마의 시간이 걸리는가?

    (a) 대치 암호

    (b) 전치 암호

    sol)

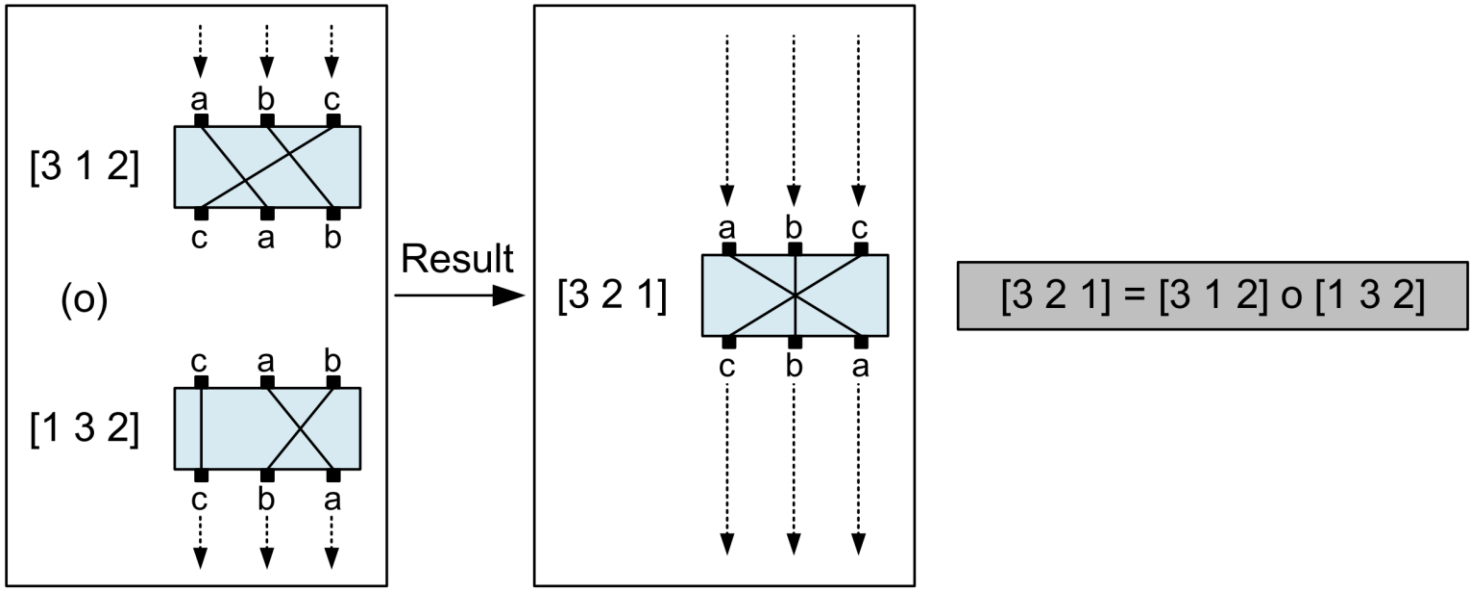
    (a) 대치를 수행하면 각 비트에 대해 2번씩 총 2^{64}번 연산을 수행함. 약 100년의 시간이 걸림

    (b) 위치만 바뀌므로 1의 개수는 유지됨. 64개 비트 중 10개의 1을 조합하는 경우의 수 ${}_{64}C_{10}$임. 약 3분의 시간이 걸림

    따라서, 대치 암호가 전수조사 공격에 더 안전하다고 할 수 있음

현대 대칭-키 암호

- 현대 블록 암호
  - 치환군으로서의 블록 암호
    - 치환군
      - 치환을 모아놓은 집합을 갖는 군
      - 치환에 대한 합성을 연산으로 가짐
      - 합성 연산이 암호의 안전성을 강화하는 지 알 수 있음



현대 대칭-키 암호

- 현대 블록 암호
  - 치환군으로서의 블록 암호
    - 전체 길이 키 암호(full-size key cipher)
      - 입력 값을 출력 값으로 대응시키는 키 길이가 충분하다고 가정
        - 실제 환경에서는 전체 길이 키 암호보다 더 짧은 길이의 키를 사용함
      - 종류
        - 전체 길이 키 전치 블록 암호
        - 전체 길이 키 대치 블록 암호

현대 대칭-키 암호

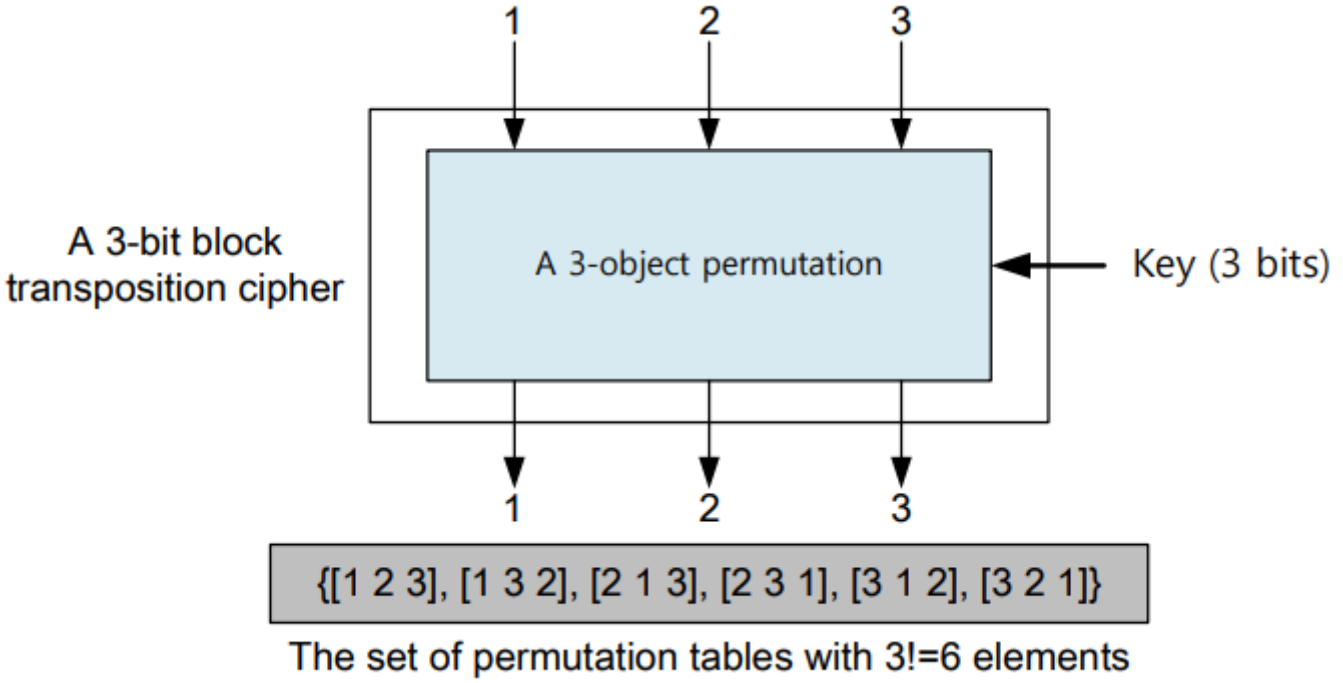
- 현대 블록 암호
  - 치환군으로서의 블록 암호
    - 전체 길이 키 암호: 전체 길이 키 전치 블록 암호(full-size key transposition block ciphers)
      - 비트 위치 재배열을 수행함
      - n 비트 길이에 대해 $n!$개의 치환표로 모델화가 가능함
      - 따라서, $n!$의 키 공간을 가짐
      - 효율적인 키 길이는 $\lceil \log_2 n! \rceil$ 비트임
        - 입력이 가질 수 있는 치환에서 중복 없이 모든 매핑을 수행하는 키 길이

현대 대칭-키 암호

- 현대 블록 암호
  - 전체 길이 키 암호: 전체 길이 키 전치 블록 암호
    - 예제 3

      prob) 블록 길이가 3비트인 블록 전치 암호의 모델과 치환표 집합을 보이시오

sol)



현대 대칭-키 암호

- 현대 블록 암호
  - 치환군으로서의 블록 암호
    - 전체 길이 키 암호: 전체 길이 키 대치 블록 암호(full-size key substitution block ciphers)(1/3)
      - 각 비트의 값을 대체시킴
      - 대치 암호 자체로는 치환표로 모델화가 불가능
      - 디코딩(decoding)과 인코딩(encoding)을 통해 치환표로 모델화 가능
        - 입력 값을 디코딩하고 출력 값을 인코딩함
        - 다음과 같은 과정을 통해 모델화를 수행함
          1. 입력 값을 전치를 적용할 수 있도록 디코딩
          2. 디코딩 된 값에 전치 적용
          3. 전치 적용된 값을 인코딩하여 원래 대치와 동등한 효과를 가지도록 함

현대 대칭-키 암호

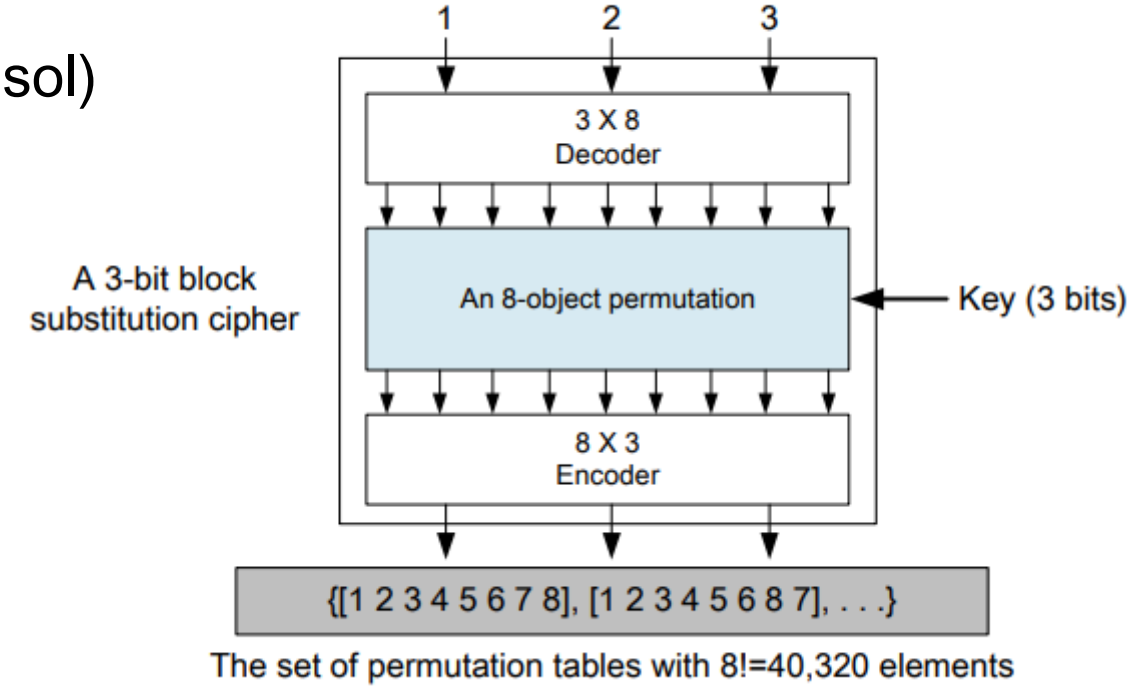
- 현대 블록 암호
  - 치환군으로서의 블록 암호
    - 전체 길이 키 암호: 전체 길이 키 대치 블록 암호(2/3)
      - 디코딩
        - 변환된 데이터를 원래 형식으로 복원
        - e.g., n 비트 정수를 한 비트만 1을 갖고 나머지 $2^n - 1$비트를 0을 갖도록 전환
      - 인코딩
        - 데이터를 특정 형식으로 변환
        - e.g., 1 값을 갖는 비트 위치를 통해 정수 값으로 전환
      - 인코딩, 디코딩에서 위치가 가지는 값의 범위는 0부터 $2^n - 1$임
      - 따라서, $2^n!$ 개의 치환표로 모델화 할 수 있음
      - 효율적인 키 길이는 $\lceil \log_2(2^n!) \rceil$비트임
        - 입력이 가질 수 있는 치환에서 중복 없이 모든 매핑을 수행하는 키 길이

현대 대칭-키 암호

- 현대 블록 암호
  - 전체 길이 키 암호: 전체 길이 키 대치 블록 암호(3/3)
    - 예제 4

prob) 블록 길이가 3비트인 블록 대치 암호의 모델과 치환표 집합을 보이시오

sol)



e.g.,

1. decoding
   $000 \rightarrow 00000001$
   $001 \rightarrow 00000010$

2. permutation
   $00000001 \rightarrow 00010000$
   $00000010 \rightarrow 00100000$

3. encoding
   $00010000 \rightarrow 100$
   $00100000 \rightarrow 101$

현대 대칭-키 암호

- 현대 블록 암호
  - 치환군으로서의 블록 암호
    - 전체 길이 키 암호의 합성
      - 전체 길이 키 전치 암호, 전체 길이 키 치환 암호는 치환 군임
      - 암호가 한 개 이상의 전치 혹은 대치로 구성되면 치환 군의 합성 연산과 동일함
      - 두 개 이상의 암호를 합성하여 사용하는 것은 암호학적으로 한 번 치환한 것과 동일하므로 불필요

현대 대칭-키 암호

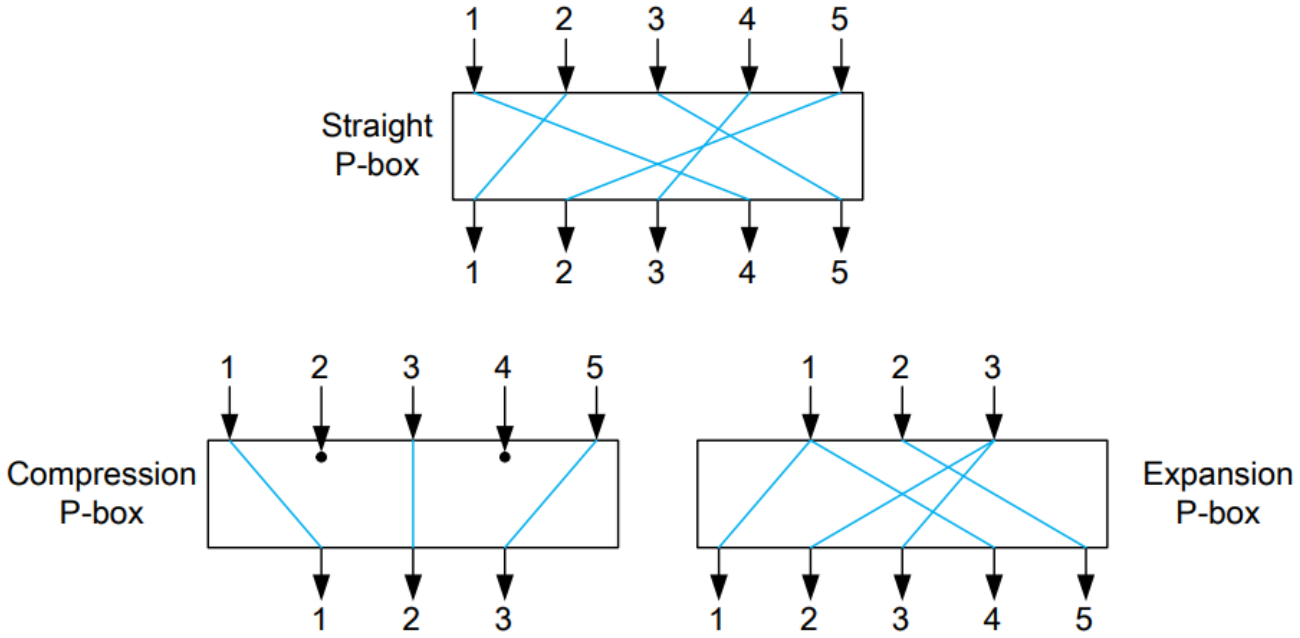
- 현대 블록 암호
  - 치환 군으로서의 블록 암호
    - 부분 길이 키 암호
      - 전체 길이 키의 실용성 보완
        - 부분 길이 키 암호는 키를 부분적으로 사용하여 전체 길이 키 암호의 실용성을 보완함
        - e.g., DES(Data Encryption Standard)의 실사용
          - 64비트 블록 암호 DES에서 최대 길이 키를 사용하면 키 길이는 $\log_2(2^{64}!) \approx 2^{70}$임
          - 하지만, DES의 실제 키 길이는 56비트임에 따라, 실제 대응 가능한 $2^{2^{70}}$가지 중 2^{56} 개의 대응만을 사용함
      - 부분 길이 키 암호의 합성
        - 부분 길이 키 암호가 치환 군이 아니라면, 합성 연산이 더 높은 보안성을 제공함
        - e.g., 다중 DES
          - 부분 길이 키 암호가 전체 길이 키 암호의 부분군이라면 부분 길이 키 암호는 합성으로 정의된 군임
          - $2^{64}!$개 대응을 갖는 군으로 2^{56}개 대응을 갖는 부분군을 구성할 수 없으므로, 56비트 키를 갖는 DES는 군이 아님

현대 대칭-키 암호

- 현대 블록 암호
  - 구성 요소
    - 종류
      - P-박스(P-box, Permutation box)
      - S-박스(S-box, Substitution box)
      - 배타적 논리합(XOR, eXclusive OR)
      - 순환 이동 연산(circular shift operation)
      - 스왑(swap operation)
      - 분할과 결합(split & combine)
    - 특징
      - P-박스는 키가 없는 전치 암호, S-박스는 키가 없는 대치 암호에 속함
        - 키가 없는 암호 알고리즘은 키가 없거나 고정되었다고 간주

현대 대칭-키 암호

- 현대 블록 암호
  - 구성 요소
    - P-박스
      - 단순(Straight) P-박스
      - 축소(Compression) P-박스
      - 확장(Expansion) P-박스



현대 대칭-키 암호

- 현대 블록 암호
  - 구성 요소
    - P-박스의 종류(1/3)
      - 단순 P-박스
        - n 비트를 입력 받고 n 비트를 출력하는 치환 함수
        - $n!$ 개의 대응이 존재
        - e.g., 64×64 치환표에서, 입력의 58번째 비트가 첫 번째 비트로 출력됨

58	50	42	34	26	18	10	02	60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06	64	56	48	40	32	24	16	08
61	53	45	37	29	21	13	05	63	55	47	39	31	23	15	07

현대 대칭-키 암호

- 현대 블록 암호
  - 구성 요소
    - P-박스의 종류(2/3)
      - 축소 P-박스
        - n 비트를 입력 받고 m 비트를 출력하는 치환 함수 $(n > m)$
        - 1부터 n까지 성분 값 중 $n - m$개 비트가 소실됨
        - e.g., 32×24 치환표에서 7, 8, 9, 15, 16, 23, 24, 25번째 입력 비트가 소실됨

01	02	03	21	22	26	27	28	29	13	14	17
18	19	20	04	05	06	10	11	12	30	31	32

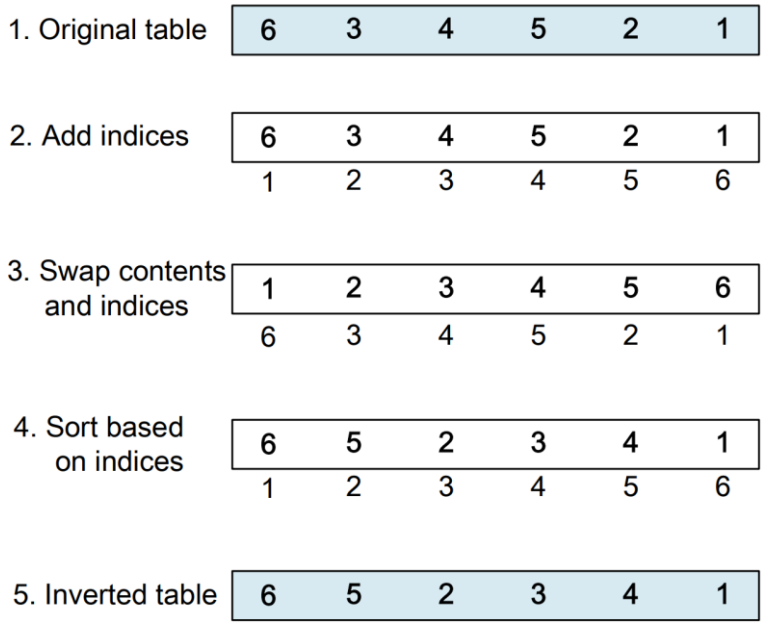
현대 대칭-키 암호

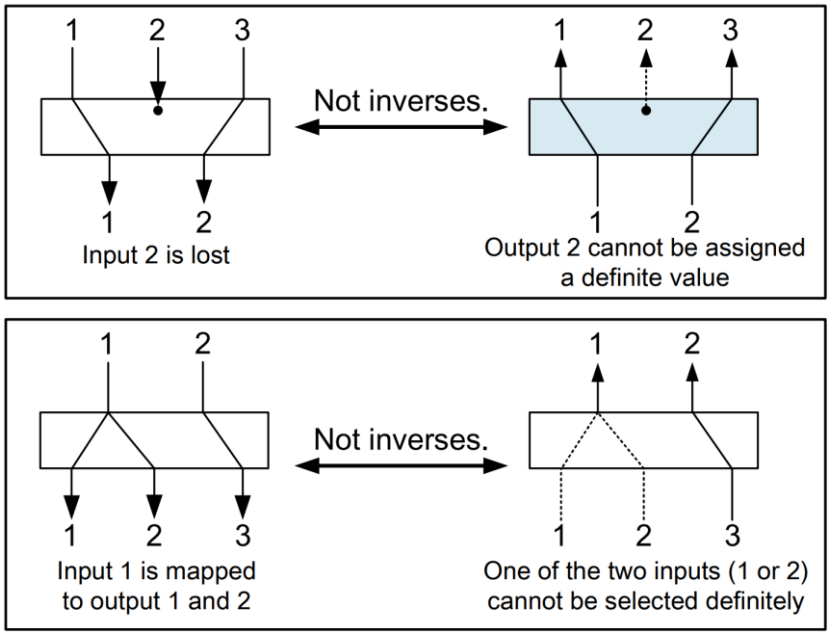
- 현대 블록 암호
  - 구성 요소
    - P-박스의 종류(3/3)
      - 확장 P-박스
        - n 비트를 입력 받고 m 비트를 출력하는 치환 함수 $(n < m)$
        - 1부터 n까지의 성분 값 중 $m - n$ 개의 성분이 중복됨
        - e.g., 12×16 치환표에서 1, 3, 9, 12 번째 입력 비트가 중복되어 출력됨

01 09 10 11 12 01 02 03 04 05 06 07 08 09 12

현대 대칭-키 암호

- 현대 블록 암호
  - 구성 요소
    - P-박스의 특징
      - 역함수의 존재성
        - 단순 P-박스는 역함수가 존재함
        - 축소 P-박스와 확장 P-박스는 역함수가 존재하지 않음





현대 대칭-키 암호

- 현대 블록 암호
  - 구성 요소
    - S-박스
      - 입력과 출력 값의 관계는 다음 수식으로 표현됨

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

      - 수식에 따라 선형 S-박스와 비선형 S-박스로 구분할 수 있음

어떤 함수의 선형성은 하위 두 가지 조건을 만족하는 것으로 정의됨

1. Additivity
$$f(x + y) = f(x) + f(y)$$
2. Homogeneity
$$f(\alpha x) = \alpha f(x)$$
for scalar α

현대 대칭-키 암호

- 현대 블록 암호
  - 구성 요소
    - S-박스의 종류
      - 선형 S-박스
        - 선형 S-박스는 관계식을 다음과 같이 표기함

$$y_1 = a_{1.1}x_1 \oplus a_{1.2}x_2 \oplus \cdots a_{1.n}x_n$$
$$y_2 = a_{2.1}x_1 \oplus a_{2.2}x_2 \oplus \cdots a_{2.n}x_n$$
$$\cdots$$
$$y_m = a_{m.1}x_1 \oplus a_{m.2}x_2 \oplus \cdots a_{m.n}x_n$$

      - 비선형 S-박스
        - 모든 출력 값에 대해 위 관계식을 만족하지는 않음

현대 대칭-키 암호

- 현대 블록 암호
  - 구성 요소
    - 예제 5

prob) 3비트를 입력 받아 2 비트를 출력하는 S-박스가 다음을 만족할때,

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

선형성을 판단하여라.

sol) $a_{1.1} = a_{1.2} = a_{1.3} = a_{2.1} = 1$ 이고 $a_{2.2} = a_{2.3} = 0$ 이므로 위 S-박스는 다음과 같은 행렬로 표현됨에 따라 선형임

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

어떤 체 $\mathbb{F}$ 위에서 정의된 행렬 A는 항상 선형성을 가짐

현대 대칭-키 암호

- 현대 블록 암호
  - 구성 요소
    - 예제 6

      prob) 3비트를 입력 받아 2비트를 출력하는 S-박스가 다음을 만족할때,

      $y_1 = (x_1)^3 + x_2, \qquad y_2 = (x_1)^2 + x_1x_2 + x_3$

      선형성을 판단하여라.

      sol) 선형 관계식이 존재하지 않으므로 비선형임

현대 대칭-키 암호

- 현대 블록 암호
  - 구성 요소
    - S-박스의 특징
      - 대치 암호임에도 입력과 출력의 개수가 달라도 됨
        - 입력 n 비트 워드, 출력 m 비트 워드에 대해 n과 m이 달라도 됨
      - 역함수의 존재성
        - 역함수가 존재할 수도 있고 존재하지 않을 수도 있음
        - 단, 역함수를 가지는 S-박스는 입, 출력 비트 수가 동일함

현대 대칭-키 암호

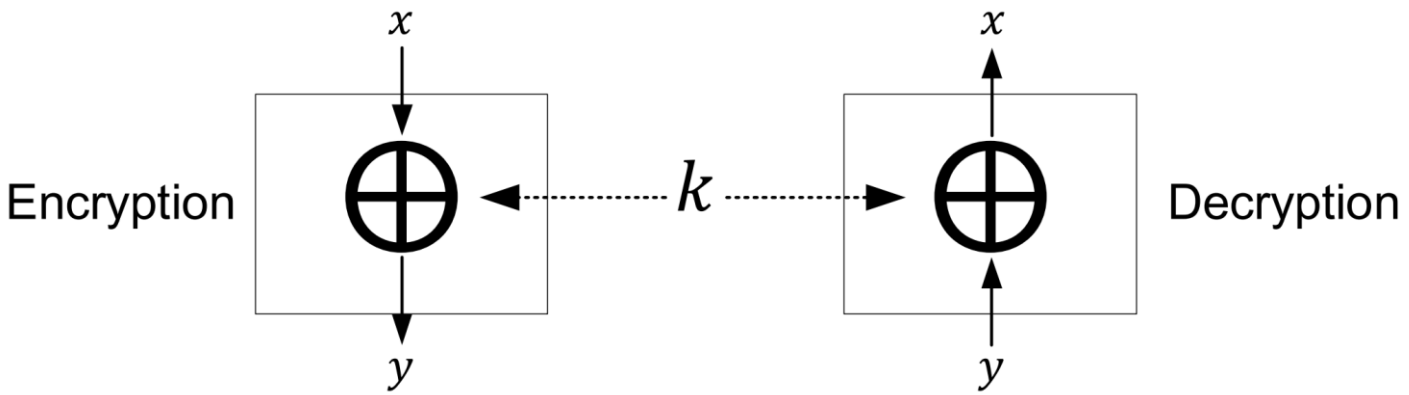
- 현대 블록 암호
  - 구성 요소
    - 배타적 논리합(1/3)
      - 성질
        - 닫힘
          - n 비트 워드 두 개의 XOR 결과 또한 n비트 워드
        - 결합 법칙
          - $x \oplus (y \oplus z) \leftrightarrow (x \oplus y) \oplus z$가 성립함
        - 교환 법칙
          - $x \oplus y \leftrightarrow y \oplus x$가 성립함
        - 항등원의 존재성
          - $x \oplus (00 \ldots 0) = x$
        - 역원의 존재성
          - $x \oplus x = (00 \ldots 0)$
          - 자기 자신을 덧셈 역원으로 가짐

현대 대칭-키 암호

- 현대 블록 암호
  - 구성 요소
    - 배타적 논리합(2/3)
      - 보수(complement)
        - XOR을 통해 1의 보수를 구할 수 있음
        - e.g., x와 x의 보수 x'에 대해 다음이 성립함
          $$x \oplus x' = (11 \ldots 1)$$
          $$x \oplus (11 \ldots 1) = x'$$
        - e.g., 8비트 값 00111011에 대해 다음과 같이 보수를 구할 수 있음
          $$00111011 \oplus 11111111 = 11000100$$

현대 대칭-키 암호

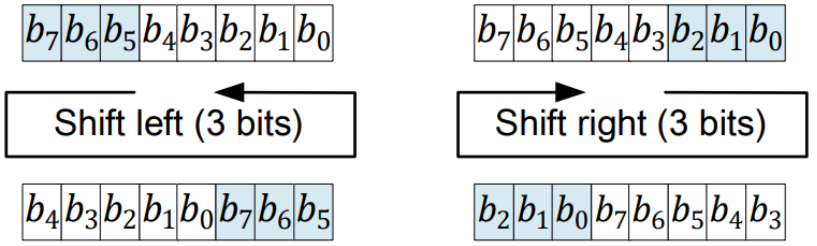
- 현대 블록 암호
  - 구성 요소
    - 배타적 논리합(3/3)
      - 역(inverse)
        - 알고리즘 구성 요소가 단일 연산으로 표현될 수 있으면 각 구성요소에 대한 역을 설명할 수 있음
          - 하나의 입력과 하나의 출력을 갖기 때문
        - 배타적 논리합은 이진 연산이므로 두 입력 중 하나의 입력이 고정되어야 역의 의미가 있음
          - e.g. $x \oplus k = y \leftrightarrow y \oplus k = x$($k$는 고정값)



현대 대칭-키 암호

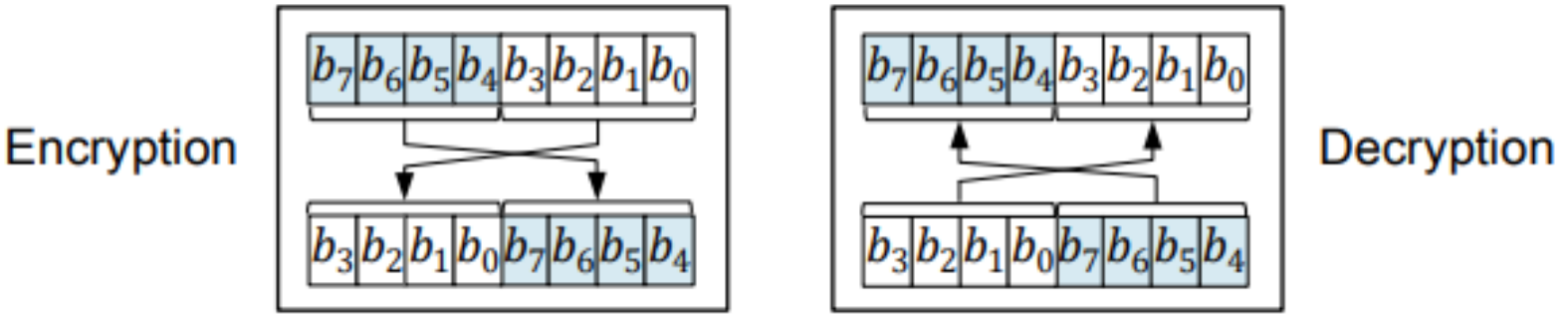
- 현대 블록 암호
  - 구성 요소
    - 순환 이동 연산
      - 왼쪽이나 오른쪽으로 비트 이동을 수행함
        - e.g., $n = 8, k = 3$ 비트만큼 왼쪽 이동
          $b_7b_6b_5b_4b_3b_2b_1b_0 \Rightarrow b_4b_3b_2b_1b_0b_7b_6b_5$
      - 워드의 비트를 섞고 본래 워드 패턴을 숨기는 효과
      - k값은 보통 사전에 정의되어 고정됨
        - $k = 0$ 이거나 $k = n$이면 자기 자신(원래의 비트)과 같음
        - $k \geq n$이면 $k \bmod n$만큼 이동함
      - k가 같을 때, 왼쪽 순환 이동은 오른쪽 순환 이동의 역임

n: 피연산자의 비트 길이
k: 이동 비트 수



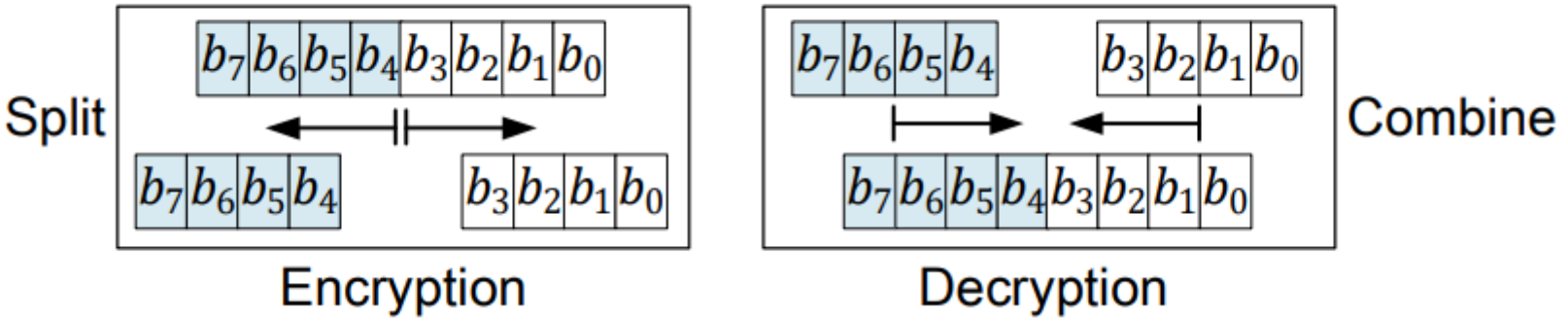
현대 대칭-키 암호

- 현대 블록 암호
  - 구성 요소
    - 스왑
      - 순환 이동 연산에서 $k = \frac{n}{2}$ 인 경우
        - n이 짝수일 때만 연산이 옳게 수행됨($\because k \in \mathbb{Z}$)
      - 자기 자신을 역으로 가짐
        - 암호화에서의 스왑 연산은 복호화에서 스왑 연산으로 상쇄됨



현대 대칭-키 암호

- 현대 블록 암호
  - 구성 요소
    - 분할과 결합
      - 분할
        - 두 개의 동일 길이 워드를 생성하기 위해 n비트 워드의 중앙을 분리(n은 짝수)
      - 결합
        - 두 개의 동일 길이 워드를 n 비트로 연접(concatenate)
      - 분할, 결합 연산은 서로 역관계에 있음



현대 대칭-키 암호

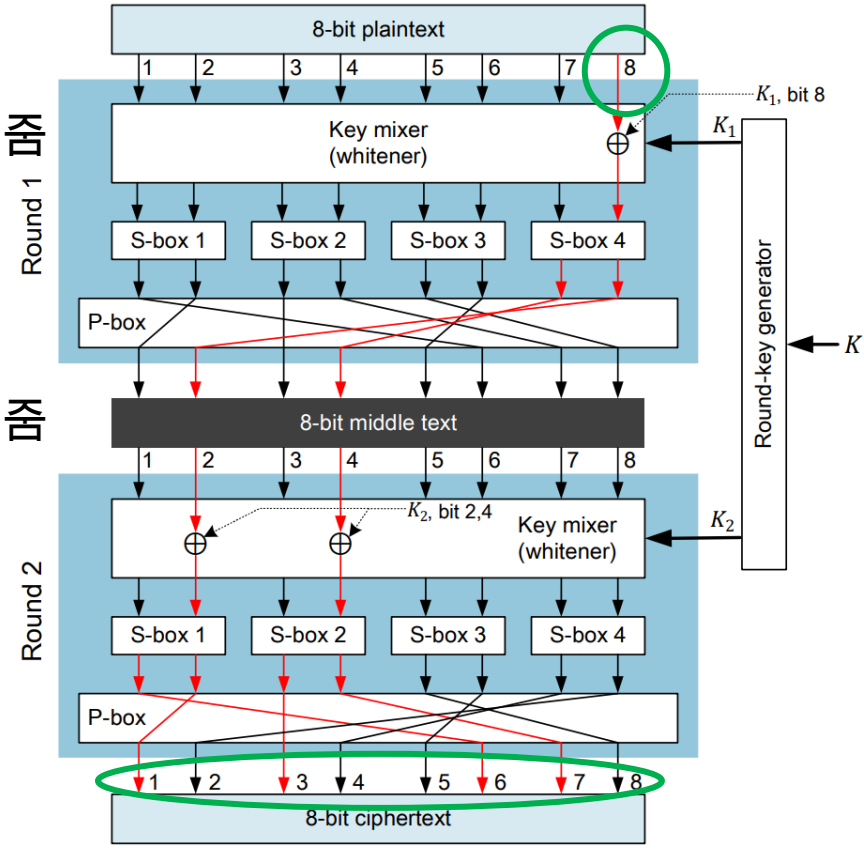
- 현대 블록 암호
  - 합성 암호(product cipher)(1/4)
    - 정의
      - 대치, 치환, 기타 구성 요소를 결합한 복합적인 암호
    - 확산과 혼돈
      - 확산(confusion)
        - 암호문과 평문 사이의 관계를 숨김
        - 평문, 암호문 사이 통계에 기반한 공격을 방어
        - e.g., 평문 단일 비트만 변해도, 암호문의 거의 모든 비트들이 변함
      - 혼돈(diffusion)
        - 암호문과 키 사이의 관계를 숨김
        - 암호문을 이용해 키를 찾고자 하는 공격을 방어
        - e.g., 키의 단일 비트만 변해도, 암호문의 거의 모든 비트들이 변함
      - 확산과 혼돈은 반복적인 합성 암호 사용으로 얻어짐

현대 대칭-키 암호

- 현대 블록 암호
  - 합성 암호(2/4)
    - 라운드(round)
      - 합성 암호가 반복적으로 사용되는 각 단계를 지칭
        - e.g., N 라운드 암호에서 평문은 N번 암호화 되고 암호문은 N번 복호화
      - 중간 상태 값(middle text)
        - 각 라운드를 수행하는 중간 변화되는 상태의 값
      - 각 라운드에 쓰일 서로 다른 키 생성을 위해 키 스케줄(key schedule) 알고리즘을 사용함

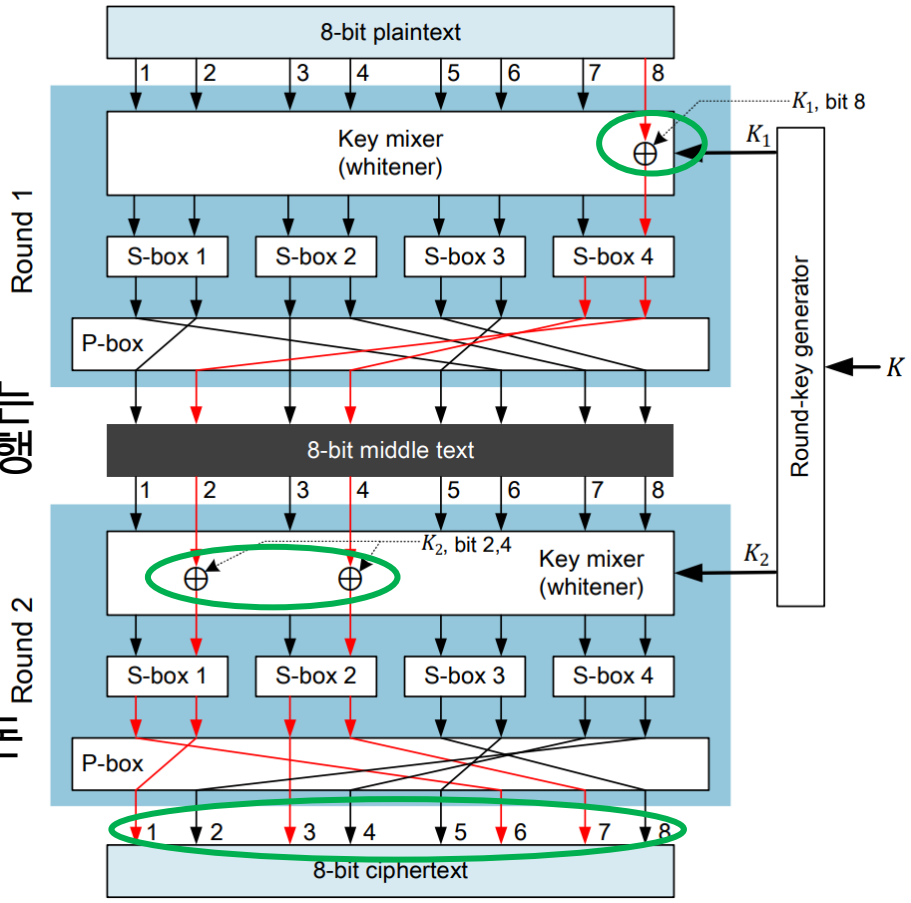
현대 대칭-키 암호

- 현대 블록 암호
  - 합성 암호(3/4)
    - 2 라운드 합성 암호 예시(1/2)
      - 확산
        - 8번째 비트는 키 K_1의 8번째 비트와 XOR 연산을 수행 (key mixing)
        - S-box 4를 통과하여 두 비트에 영향을 줌
        - 두 비트는 P-box를 통과하여 2, 4번 비트가 됨
        - 2 라운드에서 같은 과정을 통해 8번 비트는 1, 3, 6, 7번 비트에 영향을 줌
        - 라운드를 거치며 평문 8번 비트가 암호문에 주는 영향이 커짐 → 확산



현대 대칭-키 암호

- 현대 블록 암호
  - 합성 암호(4/4)
    - 2 라운드 합성 암호 예시(2/2)
      - 혼돈
        - 8번째 비트는 키 K_1 의 8번째 비트와 XOR 연산을 수행
        - S-박스와 P-박스를 통과하여 2, 4번 비트에 영향을 줌
        - 2 라운드에서 중간 상태 2, 4번 비트는 K_2의 2, 4 번 비트와 XOR 연산을 수행
        - S-박스와 P-박스를 통과하여 1, 3, 6, 7번 비트에 영향을 줌
        - 라운드를 거치며 각 라운드의 키가 암호문 여러 비트에 영향을 줌 → 혼돈



현대 대칭-키 암호

- 현대 블록 암호
  - 두 가지 종류의 합성 암호
    - Feistel 암호
      - 역함수가 존재하는 구성 요소와 존재하지 않는 구성 요소 모두 사용
      - e.g., DES, SEED
    - non-Feistel 암호(= SPN, Substitution Permutation Network 구조)
      - 역함수가 존재하는 구성 요소만을 사용
      - e.g., AES(Advanced Encryption Standard), ARIA(Academy, Research Institute, Agency)

현대 대칭-키 암호

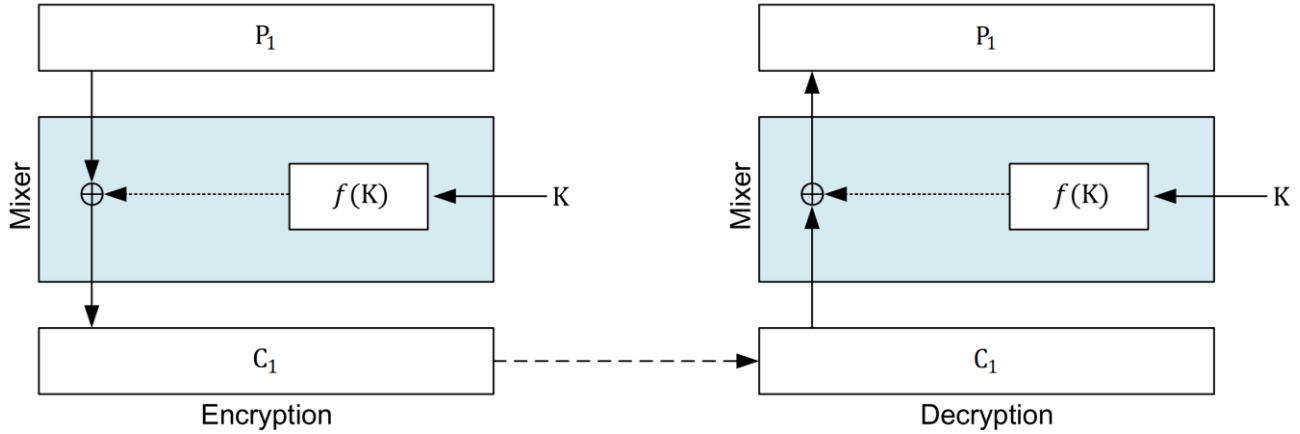
- 현대 블록 암호
  - 두 가지 종류의 합성 암호
    - Feistel 암호 생성(1/4)
      - 구성 요소
        - 자기 자신을 역으로 갖는 것
        - 역함수를 갖는 것
        - 역함수를 가지지 않는 것
      - 첫 번째 사상(mapping)
        - 키 K 와 역함수가 존재하지 않는 함수 $f(\mathrm{K})$에 대해 $f(\mathrm{K})$함수와 평문간 XOR 연산을 수행
        - XOR 수행 결과를 암호문으로 사용
        - $f(\mathrm{K})$함수와 XOR 연산의 결합을 혼합기(mixer)라 지칭

현대 대칭-키 암호

- 현대 블록 암호
  - 두 가지 종류의 합성 암호
    - Feistel 암호 생성(2/4)
      - 혼합기
        - 키 K에 대해 역함수가 존재하지 않는 함수 $f(\mathrm{K})$와 XOR 연산의 결합으로 생성되는 가역적 요소
        - 암호 알고리즘과 복호 알고리즘이 역 관계임

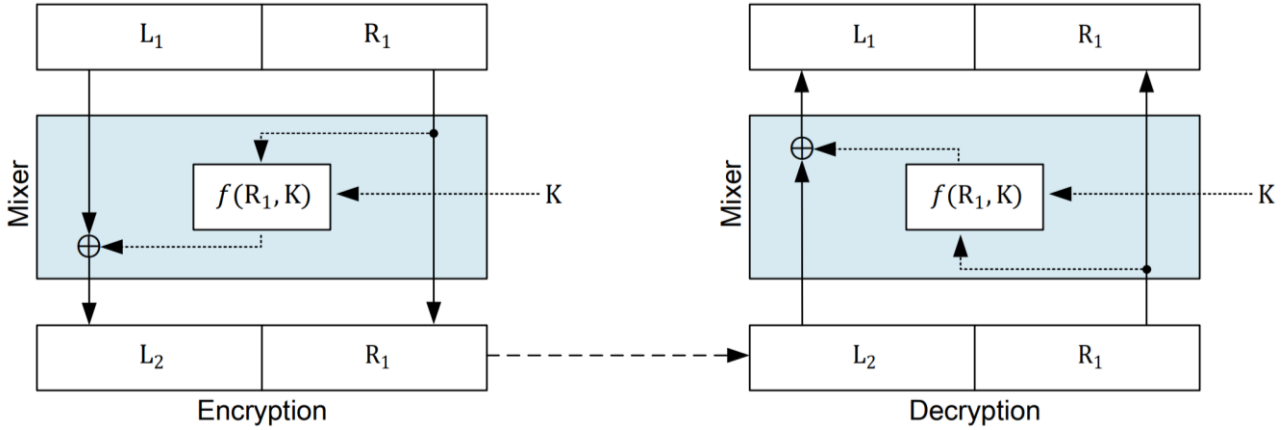
P: 평문
C: 암호문
K: 키

        - 암호화: $\mathrm{P}_1 \oplus f(\mathrm{K}) = \mathrm{C}_1$
        - 복호화: $\mathrm{C}_1 \oplus f(\mathrm{K}) = \mathrm{P}_1 \oplus f(\mathrm{K}) \oplus f(\mathrm{K}) = \mathrm{P}_1$



현대 대칭-키 암호

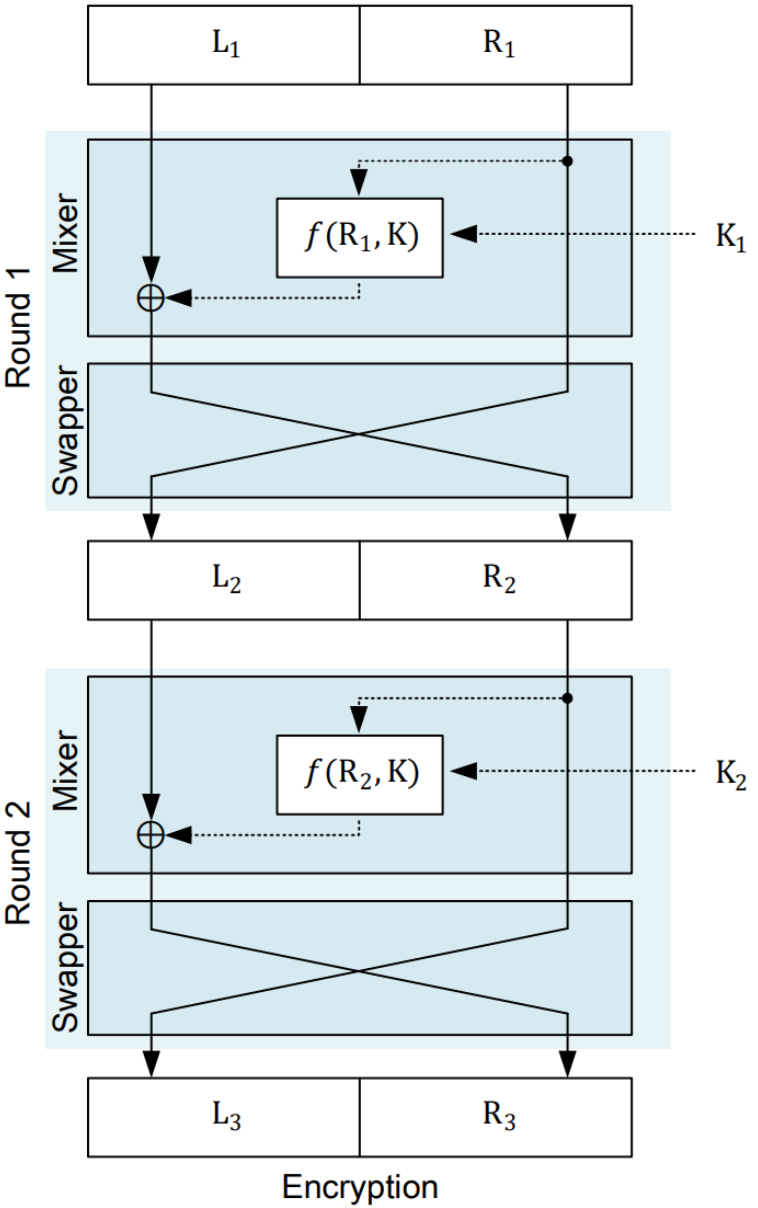
- 현대 블록 암호
  - 두 가지 종류의 합성 암호
    - Feistel 암호 생성(3/4)
      - 평문 활용 혼합기
        - 입력 값으로 키만을 사용하기보다 평문의 일부분도 함께 사용함
        - 키가 사용되는 요소와 사용되지 않는 요소가 복합적으로 구성됨
        - 과정
          1. 평문과 암호문을 동일한 길이의 왼쪽 블록(L)과 오른쪽 블록(R)으로 분할함
          2. R은 함수에 입력되고 L은 함수의 출력 값과 XOR 연산을 수행함
        - 평문의 오른쪽 반쪽(R)은 변하지 않는다는 결점을 가짐



현대 대칭-키 암호

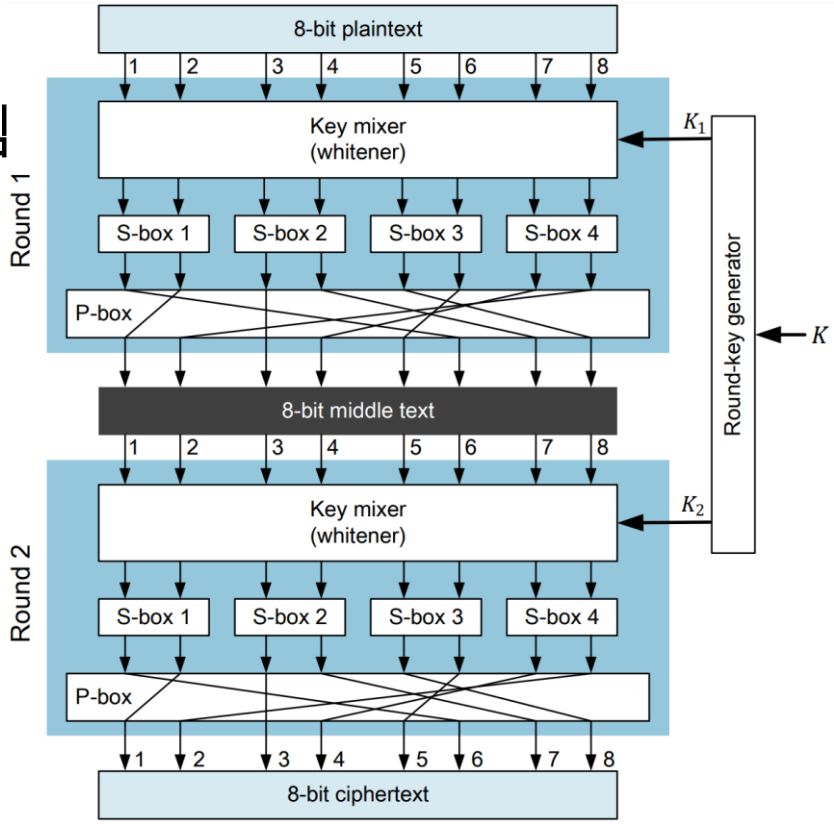
- 현대 블록 암호
  - 두 가지 종류의 합성 암호
    - Feistel 암호 생성(4/4)
      - 최종 구조
        - 스와퍼(swapper)를 추가하여 오른쪽 블록도 암호화
        - 라운드 추가

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



현대 대칭-키 암호

- 현대 블록 암호
  - 두 가지 종류의 합성 암호
    - non-Feistel 암호
      - 역함수가 존재하는 요소만을 사용
        - e.g., 동일 입출력 개수를 가지는 S-박스
        - e.g., 단순 P-박스
      - 암호화 요소는 복호화 요소에 대응됨
      - 평문이 반으로 분할될 필요는 없음
      - 각 요소가 역이 존재하므로 각 라운드가 역이 존재함



현대 대칭-키 암호

- 현대 블록 암호
  - 블록 암호에 대한 공격
    - 차분 분석(DC, Differential Cryptanalysis)
      - Eli Biham과 Adi Shamir이 차분 분석 개념을 소개
        - Biham, E., Shamir, A. “Differential cryptanalysis of DES-like cryptosystems.” *In proc of: Journal of Cryptology*, pp. 3–72, 1991.
      - 선택 평문 공격(Chosen Plaintext Attack)을 수행
      - 암호 키를 찾는 것이 목적임
    - 선형 분석(LC, Linear Cryptanalysis)
      - Mitsuru Matsui이 선형 분석 개념을 소개
        - Matsui, M. “Linear Cryptanalysis Method for DES Cipher.” *In proc of: Workshop on the Theory and Application of Cryptographic Techniques*, pp. 386-397, 1993.
      - 알려진 평문 공격(Known Plaintext Attack)을 수행
      - 암호 키를 찾는 것이 목적임

현대 대칭-키 암호

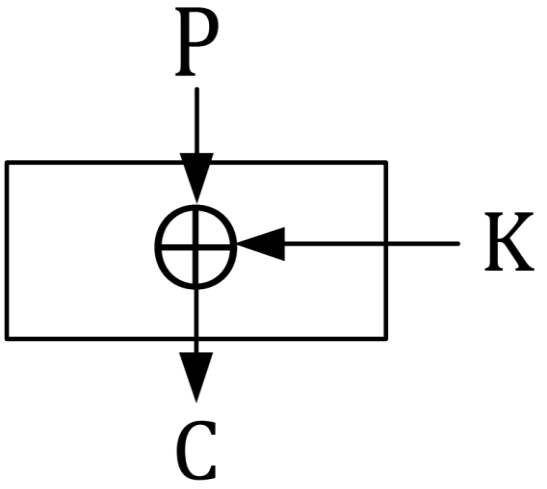
- 현대 블록 암호
  - 블록 암호에 대한 공격
    - 차분 분석(1/6)
      - 예제 7

prob) 암호가 XOR로 구성된다고 가정할 때, 평문의 차분 $P_1 \oplus P_2$와 암호문의 차분 $C_1 \oplus C_2$의 관계는?

sol) $C_1 \oplus C_2 = P_1 \oplus P_2$를 증명함

$C_1 = P_1 \oplus K,\ C_2 = P_2 \oplus K$

$\Rightarrow C_1 \oplus C_2 = P_1 \oplus K \oplus P_2 \oplus K = P_1 \oplus P_2$

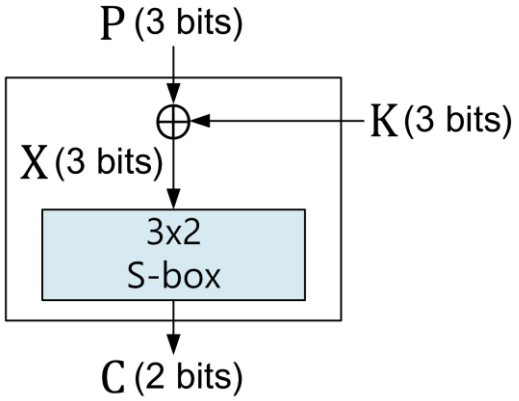


현대 대칭-키 암호

- 현대 블록 암호
  - 블록 암호에 대한 공격
    - 차분 분석(2/6)
      - 예제 8

prob) 암호가 다음과 같은 구조로 구성된다고 가정할 때, 평문의 차분 $P_1 \oplus P_2$와 암호문의 차분 $C_1 \oplus C_2$의 관계는?

sol) S-박스를 추가하여 암호문과 차분 사이의 관계를 찾기 어렵지만, $X = P \oplus K$는 여전히 차분공격이 수행됨($X_1 \oplus X_2 = P_1 \oplus P_2$)
따라서, 입력 차분에 대해 출력 차분이 균등하게 분포하지 않는다는 점을 통해 확률적 관계를 알아내어 차분 공격을 수행할 수 있음



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

S-box table

현대 대칭-키 암호

- 현대 블록 암호
  - 블록 암호에 대한 공격
    - 차분 분석(3/6)
      - 입·출력 차분 표
        - 3비트를 암호화하므로 차분($P_1 \oplus P_2$)은 2^3가지 경우가 존재

```
ddt = [[0 for col in range(4)] for row in range(8)]
s_box = [0b11, 0b00, 0b10, 0b10, 0b01, 0b00, 0b11, 0b00]

## computing ddt
for i in range(8):
    for j in range(8):
        ddt[i ^ j][s_box[i] ^ s_box[j]] += 1

## output formatting
print('___| 00 01 10 11')
for i in range(8): # i = P_1 ^ P_2
    print(format(i, '03b')+': ', end='')
    for j in range(4):
        print(f'{ddt[i][j]:2} ', end='')
    print()
```

$X_1 \oplus X_2$ \ $C_1 \oplus C_2$

___	00	01	10	11
000:	8	0	0	0
001:	2	2	0	4
010:	2	2	4	0
011:	0	4	2	2
100:	2	2	4	0
101:	0	4	2	2
110:	4	0	2	2
111:	0	0	2	6

현대 대칭-키 암호

- 현대 블록 암호
  - 블록 암호에 대한 공격
    - 차분 분석(4/6)
      - 차분 분포표(DDT, Differential Distribution Table)
        - 입·출력 차분 표에 대해 확률을 구한 것
      - 절차

1. 선택 평문 공격
   - 차분 분포표에서 가장 높은 확률을 갖는 평문 차분 쌍을 찾음
2. 키 값 추측
   - 찾아낸 쌍으로부터 선택 암호문 공격을 수행

$P_1 \oplus P_2$ \ $C_1 \oplus C_2$	00	01	10	11
000	1	0	0	0
001	0.25	0.25	0	0.50
010	0.25	0.25	0.50	0
011	0	0.50	0.25	0.25
100	0.25	0.25	0.5	0
101	0	0.5	0.25	0.25
110	0.5	0	0.25	0.25
111	0	0	0.25	0.75

현대 대칭-키 암호

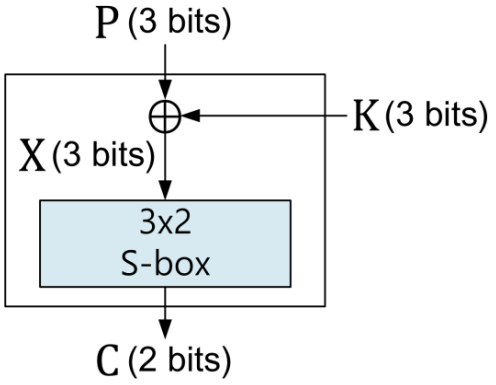
- 현대 블록 암호
  - 블록 암호에 대한 공격
    - 차분 분석(5/6)
      - 예제 9

prob) 50% 확률로 $P_1 \oplus P_2 = 001 \rightarrow C_1 \oplus C_2 = 11$를 가지는 차분 분포표에서 사전에 선택 평문 공격을 통해 구한 쌍으로부터 $C_1 = 00 \rightarrow P_1 = 010$, $C_2 = 11 \rightarrow P_2 = 011$을 만족하는 평문을 얻었다고 할 때, 키에 대한 정보를 도출하시오

sol)

- $K = X \oplus P$임
- $C_1 = 00 \rightarrow X_1 = 001 \text{ or } X_1 = 101 \text{ or } X_1 = 111$
  $\text{If } X_1 = 001 \Rightarrow K = 011$
  $\text{If } X_1 = 101 \Rightarrow K = 111$
  $\text{If } X_1 = 111 \Rightarrow K = 101$
- $C_2 = 11 \rightarrow X_2 = 000 \text{ or } X_2 = 110$
  $\text{If } X_2 = 000 \Rightarrow K = 011$
  $\text{If } X_2 = 110 \Rightarrow K = 101$

$\therefore$ K의 오른쪽 첫 비트는 1임



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

S-box table

현대 대칭-키 암호

- 현대 블록 암호
  - 블록 암호에 대한 공격
    - 차분 분석(6/6)
      - 전체적인 절차
        1. 각 라운드가 동일하여 도청자 Eve는 S-박스에 대한 차분 분포표를 구성할 수 있음
        2. 각 라운드가 독립이라 가정, 라운드에 대응하는 확률을 곱해서 전체에 대한 분포표를 얻음
        3. 전체 분포표에 기반하여 공격에 필요한 평문 목록을 구성함
        4. 암호문을 선택하고 대응하는 평문을 얻음, 결과 분석을 수행
        5. 4 과정을 반복함
        6. 충분한 키 비트 정보를 얻으면 전수 조사를 시도함

현대 대칭-키 암호

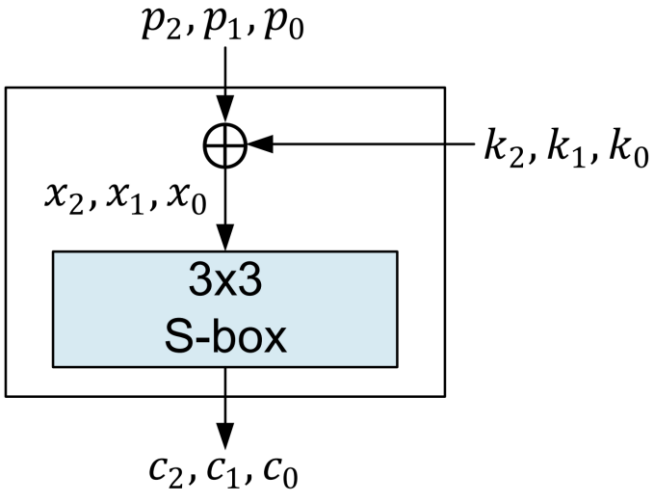
- 현대 블록 암호
  - 블록 암호에 대한 공격
    - 선형 분석(1/2)
      - 암호 알고리즘이 1 라운드로 구성되었다고 가정
      - 암호문, 평문, 키의 관계

$$c_0 = p_0 \oplus k_0 \oplus p_1 \oplus k_1$$
$$c_1 = p_0 \oplus k_0 \oplus p_1 \oplus k_1 \oplus p_2 \oplus k_2$$
$$c_2 = p_1 \oplus k_1 \oplus p_2 \oplus k_2$$

$$\therefore k_2 = (p_2) \oplus (c_0 \oplus c_1)$$
$$k_0 = (p_0) \oplus (c_1 \oplus c_2)$$
$$k_1 = (p_1) \oplus (c_0 \oplus c_1 \oplus c_2)$$

$$c_0 = x_0 \oplus x_1$$
$$c_1 = x_0 \oplus x_1 \oplus x_2$$
$$c_2 = x_1 \oplus x_2$$

S-box



현대 대칭-키 암호

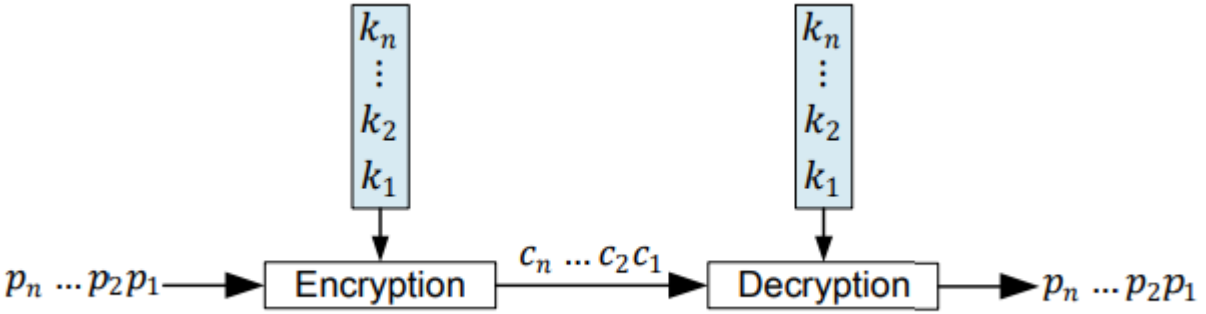
- 현대 블록 암호
  - 블록 암호에 대한 공격
    - 선형 분석(2/2)
      - 한계점
        - 앞선 수식은 3번의 알려진 평문 공격으로 k_0, k_1, k_2를 구할 수 있으나, 실제로는 예시처럼 간단하지 않고 S-박스는 선형이 아님
        - 선형 근사식(Linear Approximation)을 사용하여 해결
      - 선형 근사식
        - S-박스가 선형이 아닌 경우 특정 선형함수에 의해 확률적으로 근사할 수 있음
        - n비트 평문, m비트 키에 대해 다음 형태를 갖는 수식을 찾을 수 있음

          $(k_0 \oplus k_1 \oplus \cdots \oplus k_x) = (p_0 \oplus p_1 \oplus \cdots \oplus p_y) \oplus (c_0 \oplus c_1 \oplus \cdots \oplus c_z)$

          $1 \le x \le m,\ \ 1 \le y \le n,\ \ 1 \le z \le n$을 만족
        - 근사 확률(LAP, Linear Approximation Probability)은 편차(bias) ε에 대해 $\frac{1}{2} + \varepsilon$을 만족함
        - ε이 큰 값을 갖는 수식은 ε이 작은 값을 갖는 수식보다 효과적

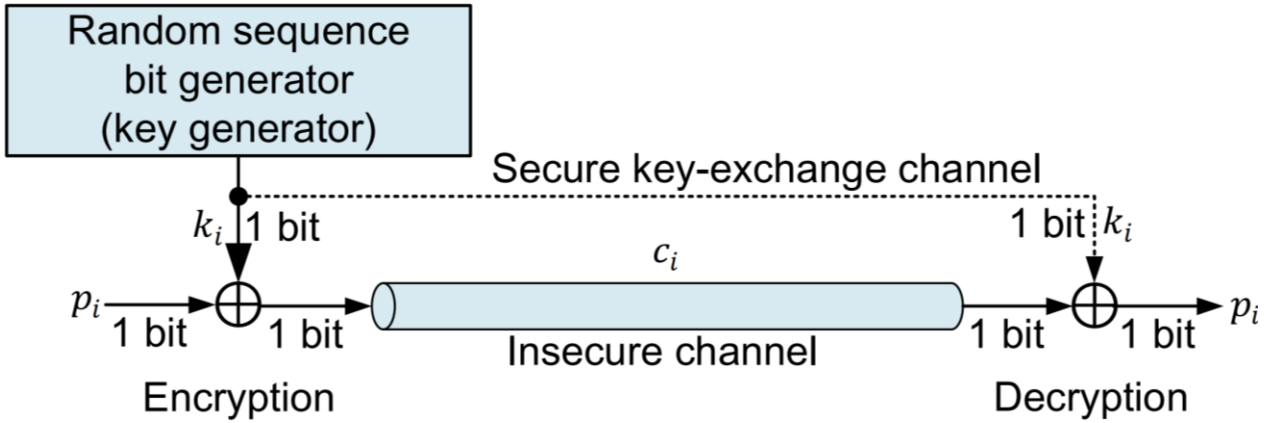
현대 대칭-키 암호

- 현대 스트림 암호
  - 평문 스트림의 r비트 워드를 키 스트림 r비트 워드를 사용하여 r비트 암호문 워드로 암호화
  - 종류
    - 동기식(synchronous) 스트림 암호
      - 키는 평문 혹은 암호문과 독립적임
    - 비동기식(nonsynchronous) 스트림 암호
      - 키는 평문 혹은 암호문에 종속적임



현대 대칭-키 암호

- 현대 스트림 암호
  - 동기식 스트림 암호(1/10)
    - 일회용 패드(one-time pad)
      - 정의
        - 메시지와 동일한 길이의 무작위 키를 사용하여 각 문자를 암호화(XOR)하는 기법
      - 조건
        - 암호문과 키의 길이가 같아야 함
        - 키는 임의의 심볼로 구성되어야 함
        - 한 번 사용한 키는 어떤 메시지에서도 재사용되어서는 안 됨
        - 키 스트림을 전달할 수 있는 안전한 채널이 존재해야 함
      - 특징
        - 전수조사로만 공격이 가능함
        - 키 공유 문제로 실사용이 어려움

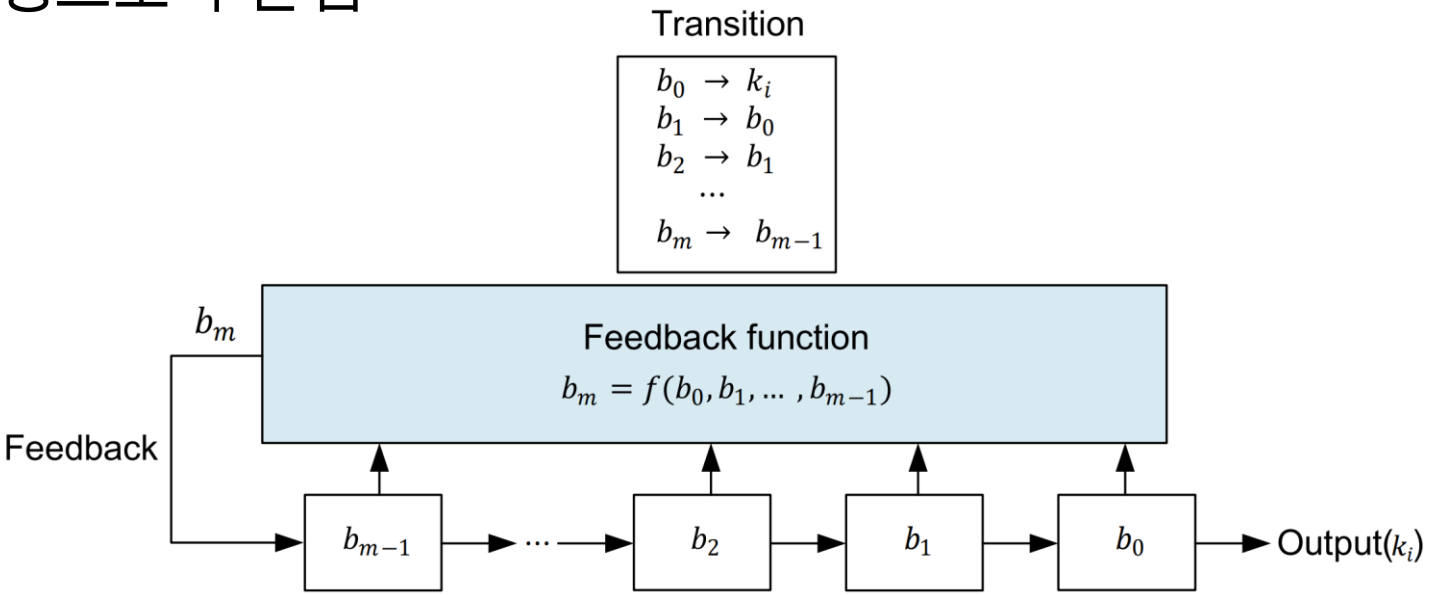


현대 대칭-키 암호

- 현대 스트림 암호
  - 동기식 스트림 암호(2/10)
    - 귀환 이동 레지스터(FSR, Feedback Shift Register)(1/2)
      - 정의
        - 현재 레지스터 값과 입력 비트를 이용해 다음 값을 생성하는 키 스트림 생성기
      - 구성
        - 이동(shift)과 귀환(feedback) 함수로 구성
        - 쉬프트 레지스터는 b_0부터 b_{m-1}까지 m개 단일 비트 셀로 나열
        - 셀은 m 비트 워드로 초기화되며 초기화 값을 시드(seed)라고 함

현대 대칭-키 암호

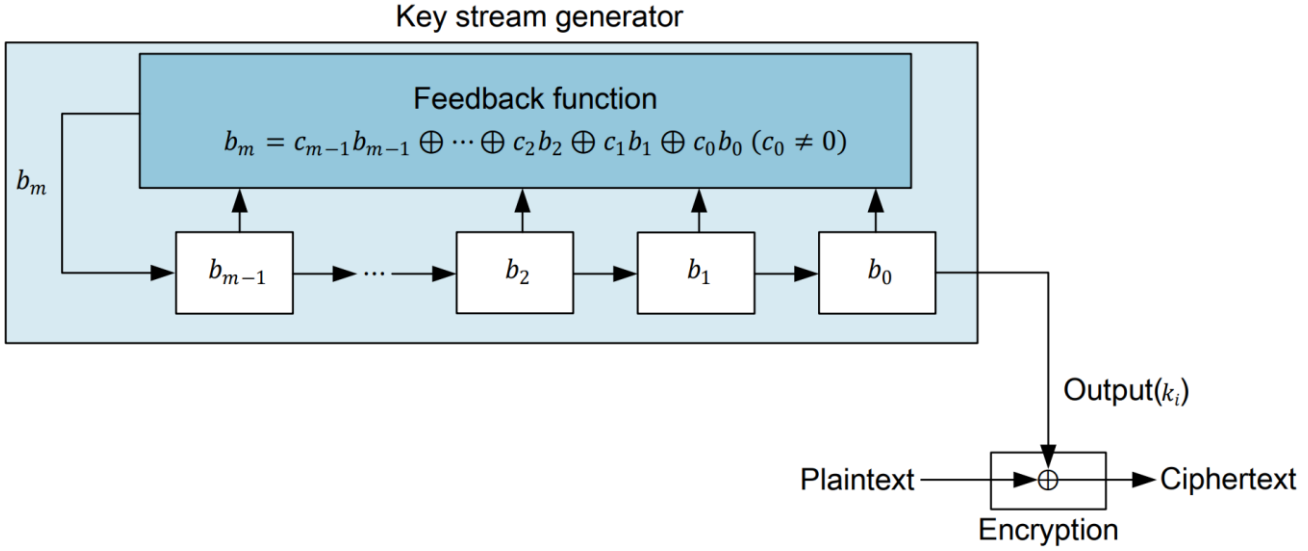
- 현대 스트림 암호
  - 동기식 스트림 암호(3/10)
    - 귀환 이동 레지스터(2/2)
      - 동작
        - b_0는 k_i로 자신의 값을 출력
        - 매 출력마다 오른쪽으로 쉬프트 되며 b_{m-1}은 귀환 함수의 출력을 받음
      - 선형, 비선형으로 구분됨



현대 대칭-키 암호

- 현대 스트림 암호
  - 동기식 스트림 암호(4/10)
    - 선형 귀환 이동 레지스터(LFSR, Linear FSR) (1/6)
      - 정의
        - 귀환 함수로 선형 함수를 사용하는 귀환 이동 레지스터
      - 귀환 함수
        - $b_m = c_{m-1}b_{m-1} \oplus \cdots \oplus c_2 b_2 \oplus c_1 b_1 \oplus c_0 b_0 \ (c_0 \neq 0)$
        - c_0가 0이면 최초 k만 b_0값을 가지고 그 다음 키부터는 $b_1, b_2, \cdots, b_{m-1}$의 셀을 가지는 LFSR 처럼 동작하여 의미가 없음

b_i: i번째 셀
c_i: 계수, 0 또는 1의 값을 가짐
m: 셀의 개수



현대 대칭-키 암호

- 현대 스트림 암호
  - 동기식 스트림 암호(5/10)
    - 선형 귀환 이동 레지스터(2/6)
      - 규칙
        - 시드가 모두 0이면, 생성된 키 스트림은 사용하지 않음
        - 평문 스트림이 모두 0이면 암호화에서 제외함
      - 주기성
        - 주기는 LFSR 구조와 시드에 기반함
        - 주기는 m비트 시드에 대해, $2^m - 1$을 넘지 못함
      - 최대 주기 조건
        - 셀의 개수가 짝수이어야 함
        - 귀환 함수를 특성 다항식(m차 다항식)으로 변환하였을 때, 원시 다항식(primitive polynomial)을 만족해야 함
          - $b_m = c_{m-1}b_{m-1} + \cdots + c_1 b_1 + c_0 b_0 \rightarrow x^m = c_{m-1}x^{m-1} + \cdots + c_1 x^1 + c_0 x^0$
        - 원시 다항식은 $e = 2^k - 1\ (k \geq 2)$를 만족하는 최소 정수 e에 대해 $x^e + 1$을 나누는 기약 다항식
        - 일반적으로, 다항식을 랜덤하게 선택하여 원시 다항식인지를 체크함 (이미 알려진 원시 다항식 리스트가 존재)

현대 대칭-키 암호

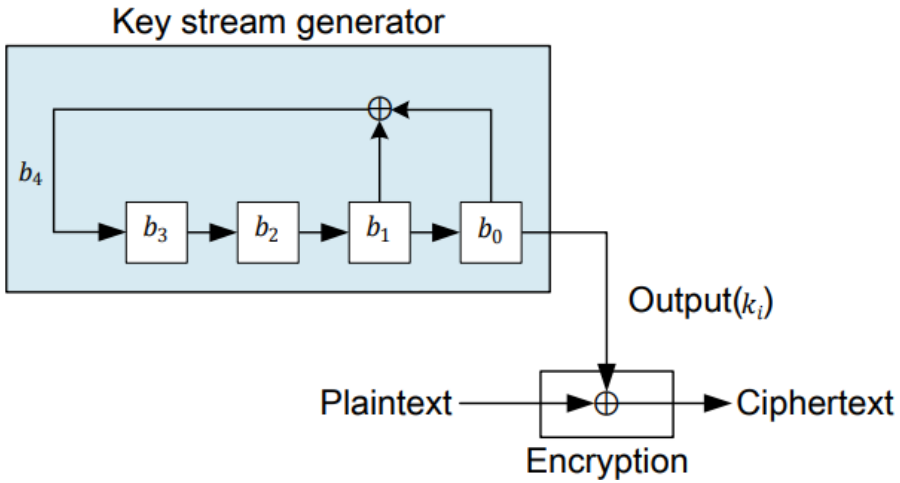
- 현대 스트림 암호
  - 동기식 스트림 암호(6/10)
    - 선형 귀환 이동 레지스터(3/6)
      - 최대 주기 조건
        - 셀의 개수가 짝수이어야 함
        - 귀환 함수를 특성 다항식(m차 다항식)으로 변환하였을 때, 원시 다항식 (primitive polynomial)을 만족해야 함
          - e.g., $b_m = c_{m-1}b_{m-1} + \cdots + c_1 b_1 + c_0 b_0 \rightarrow x^m = c_{m-1}x^{m-1} + \cdots + c_1 x^1 + c_0 x^0$
      - 원시 다항식
        - $e = 2^k - 1\ (k \geq 2)$를 만족하는 최소 정수 e에 대해 $x^e + 1$을 나누는 기약 다항식
        - 일반적으로, 다항식을 랜덤하게 선택하여 원시 다항식인지를 체크함 (이미 알려진 원시 다항식 리스트가 존재)

현대 대칭-키 암호

- 현대 스트림 암호
  - 동기식 스트림 암호(7/10)
    - 선형 귀환 이동 레지스터(4/6)
      - 예제 10
        - 다음 그림의 특성 다항식은 기약 다항식인 $x^4 + x + 1$이고, 이는 $e = 2^4 - 1 = 15$인 $x^e + 1$을 나눔

          $$x^{15} + 1 = (x^4 + x + 1)(x^{11} + x^8 + x^7 + x^5 + x^3 + x^2 + x + 1)$$

        - 또한, 셀의 개수가 4개로 짝수의 셀을 가지므로 최대 주기$(2^4 - 1)$를 가짐



현대 대칭-키 암호

- 현대 스트림 암호
  - 동기식 스트림 암호(8/10)
    - 선형 귀환 이동 레지스터(5/6)
      - 코드를 통한 이해

```python
def feedback(*args):
    res = 0
    for feedback_arg in args:
        res ^= feedback_arg

    return res

seed = [1, 0, 0, 0] # for index readability
b = seed

for _ in range(60):
    bm = feedback(b[0], b[1])
    k = b[0] # generate key
    for i in range(1, len(b)): ## shift
        b[i-1] = b[i]
    b[len(b) - 1] = bm ## feedback ==> compute b4

    if _ %15 == 0 and _ != 0:
        print(' ', end='')
    print(k, end='')
```

```
$ python -u "g:\내 드라이브\연구실\세미나\암호학과_네트워크_보안
9_암호학과 네트워크 보안_5장, 8장_현대 대칭-키 암호 소개, 현대 대
암호를 이용한 암호화 기법_이정민_v1\LFSR.py"
100010011010111 100010011010111 100010011010111 100010011010111
```

15 bits

현대 대칭-키 암호

- 현대 스트림 암호
  - 동기식 스트림 암호(9/10)
    - 선형 귀환 이동 레지스터(6/6)
      - 원시 다항식 리스트 예제
        - 괄호 ()로 감싸진 부분은 원시 다항식이 아닌 기약 다항식
        - e.g., 차수 4의 첫 번째 원시 다항식은 $0\text{x}13$으로 10011_2임
          $\therefore x^4 + x + 1$인 원시 다항식을 구할 수 있음

n	Polynomials (in hexadecimal format)
1	3 2
2	7
3	B D
4	13 19 $(1F)$
5	25 29 $2F$ 37 $3B$ $3D$
6	43 (45) 49 57 58 61 $6D$ 73
7	83 87 91 $9D$ $A7$ AB $B9$ BF $C1$ CB $D3$ $D4$ $E5$ EF $F1$ $F7$ FD
8	$(11B)$ $11D$ $12B$ $12D$ (139) $(13F)$ $14D$ $15F$ 163 165 169 171 (177) $(17B)$ 187 $(18B)$ $(19F)$ $(1A3)$ $1A9$ $(1B1)$ $(1BD)$ $1CF$ $(1D7)$ $(1D8)$ $1E7$ $(1F3)$ $1F5$ $(1F9)$

현대 대칭-키 암호

- 현대 스트림 암호
  - 동기식 스트림 암호(10/10)
    - 비선형 귀환 이동 레지스터(NLFSR, Nonlinear FSR)
      - b_m이 $b_0, b_1, \ldots, b_{m-1}$의 비선형 함수로 정의된 귀환 이동 레지스터
        - e.g., $b_4 = (b_3 \text{ AND } b_2) \text{ OR } (b_1 \text{ AND } \overline{b_0})$
      - LFSR의 선형성으로 인한 취약성을 보완함
        - Berlekamp-Massey 알고리즘과 같은 방법들을 사용하여 LFSR 내부 상태를 추측하고 키 스트림을 예측할 수 있음
      - 최대 주기를 갖는 NLFSR을 설계하기 위한 수학적 기초지식이 널리 알려지지 않아 NLFSR이 널리 사용되지는 않음
    - 결합(combination)
      - LFSR 비트를 비선형으로 조합하여 사용함
      - 몇몇 LFSR은 최대 주기를 갖도록 설계된 다음, 비선형 함수와 결합하여 사용됨

현대 대칭-키 암호

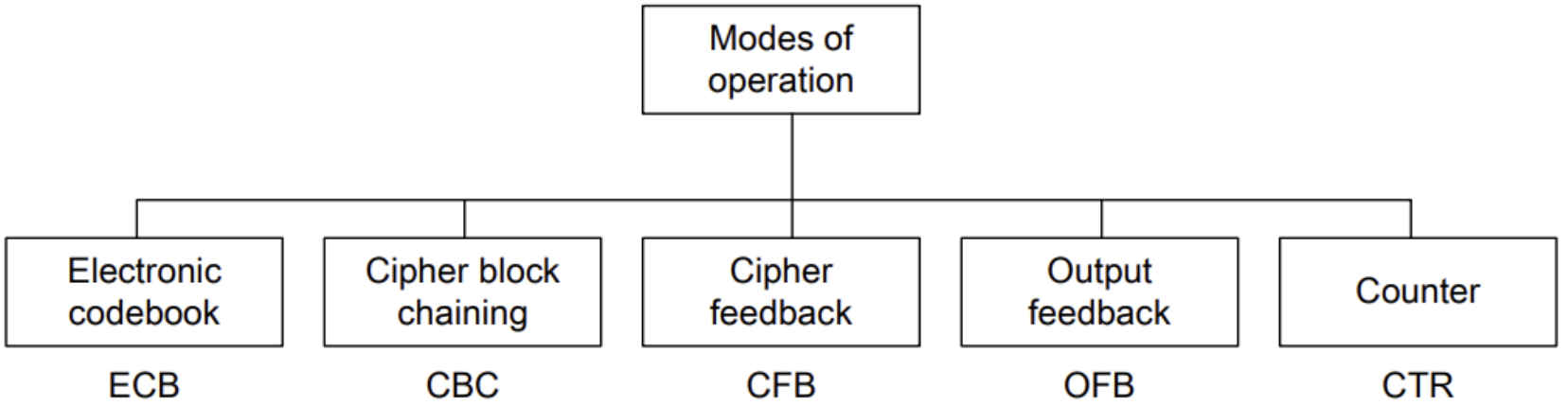
- 현대 스트림 암호
  - 비동기식 스트림 암호
    - 특징
      - 자기 동기성
        - 동기식 스트림 암호는 암호 전송 과정에서 암호문의 비트가 누락된 경우, 오류 시점부터 복호화가 실패하여 동기화가 필요함
        - 비동기식 스트림 암호는 암호 비트가 손상된 경우 일부분만 복호화에 실패하며, 이후에는 다시 정상적인 복호화 값을 얻을 수 있음
    - e.g., CFB(Cipher FeedBack) 모드 블록 암호

목 차

- 현대 대칭-키 암호
- 현대 대칭-키 암호를 이용한 암호화 기법

현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - 정의
    - 하나의 키를 사용하여 블록 암호를 반복적으로 이용하는 절차
  - 등장 배경
    - 일반적으로 평문은 64비트나 128비트 블록 크기보다 길
    - 이러한 문제를 해결하기 위해 임의 길이의 텍스트를 암호화하는 운영 모드를 설계함



현대 대칭-키 암호를 이용한 암호화 기법

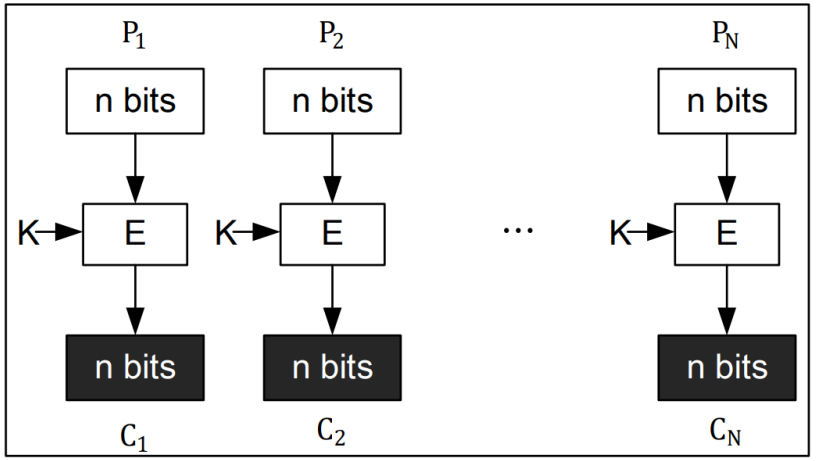
- 운영 모드
  - ECB(Electronic CodeBook) 모드(1/7)
    - 동작
      - 평문을 일정 크기의 블록으로 나누고 각 블록을 동일한 키로 암호화함
    - 특징
      - 운영 모드 중에서 가장 간단한 모드
      - 평문 크기가 블록 크기의 배수가 아니라면, 패딩 필요
      - 블록 간 독립성 존재

현대 대칭-키 암호를 이용한 암호화 기법

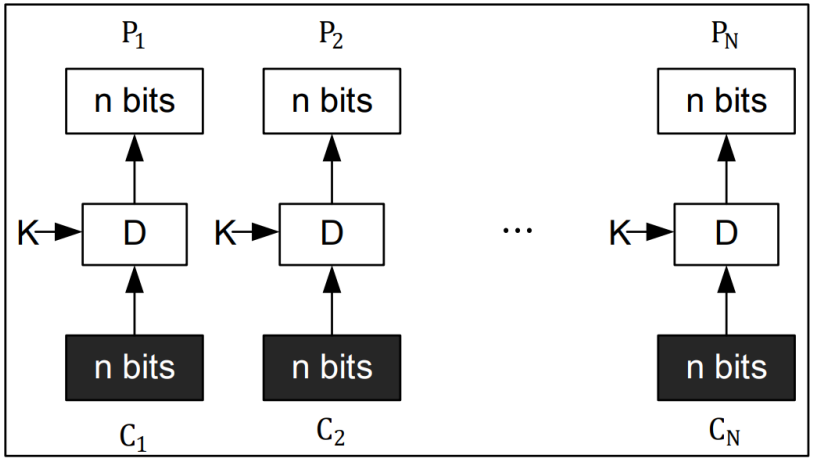
- 운영 모드
  - ECB 모드(2/7)
    - 암호화
      - $C_i = E_K(P_i)$
    - 복호화
      - $P_i = D_K(C_i)$

E : Encryption
P_i : Plaintext block i
K : Secret key

D : Decryption
C_i : Ciphertext block i



Encryption



Decryption

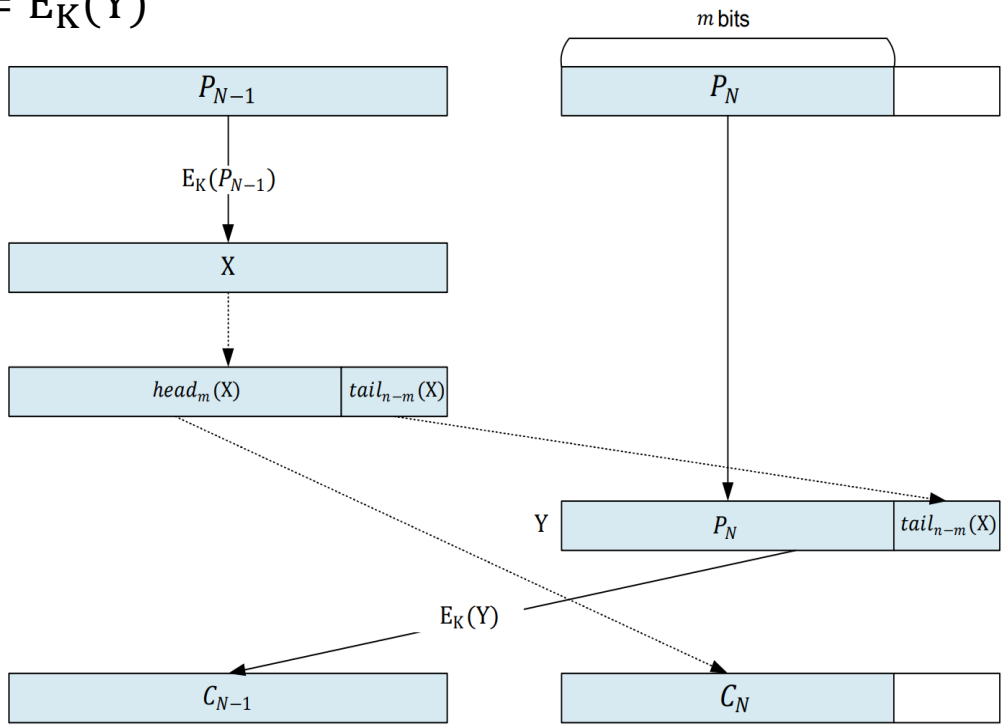
현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - ECB 모드(3/7)
    - CTS(CipherText Stealing) (1/2)
      - 정의
        - 운영 모드를 사용할 때, 평문과 같은 크기로 암호문을 생성하도록 수행하는 방법
      - 동작
        - CTS에서 두 평문 블록 $\mathrm{P_{N-1}}, \mathrm{P_N}$은 기존과 다르게 암호화 됨($\mathrm{P_{N-1}}$은 n비트 블록, $\mathrm{P_N}$은 m비트 블록 $(m < n)$)
        - $\mathrm{X} = \mathrm{E_K}(\mathrm{P_{N-1}}) \rightarrow \mathrm{C_N} = head_m(\mathrm{X})$
        - $\mathrm{Y} = \mathrm{P_N} | \, tail_{n-m}(\mathrm{X}) \rightarrow \mathrm{C_{N-1}} = \mathrm{E_K}(\mathrm{Y})$

$head_m$: 왼쪽 최상위 m비트 선택하는 함수
$tail_{n-m}$: 오른쪽 최상위 $n-m$ 비트를 선택하는 함수

현대 대칭-키 암호를 이용한 암호화 기법

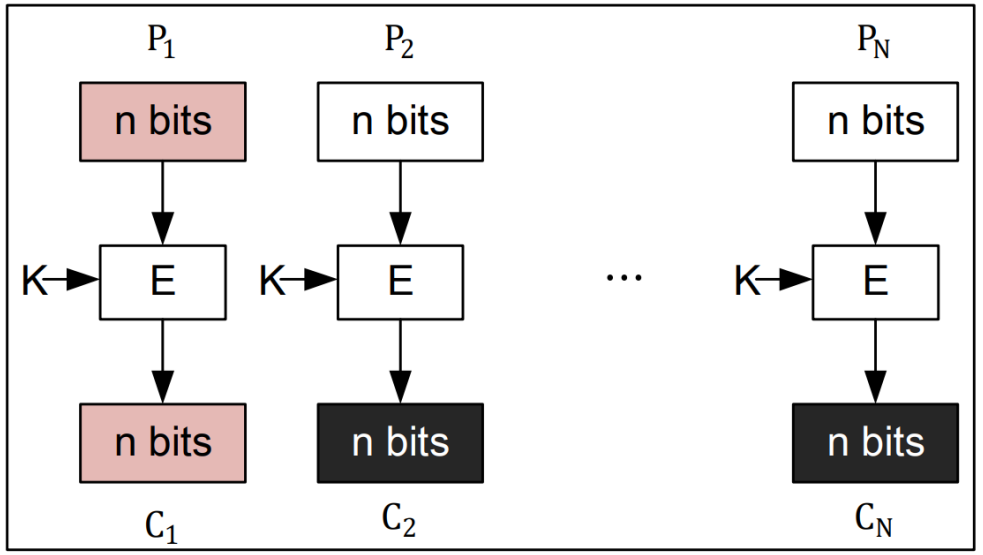
- 운영 모드
  - ECB 모드(4/7)
    - CTS(2/2)
      - 동작
        - $\mathrm{X} = \mathrm{E_K}(\mathrm{P_{N-1}}) \rightarrow \mathrm{C_N} = head_m(\mathrm{X})$
        - $\mathrm{Y} = \mathrm{P_N} | \, tail_{n-m}(\mathrm{X}) \rightarrow \mathrm{C_{N-1}} = \mathrm{E_K}(\mathrm{Y})$

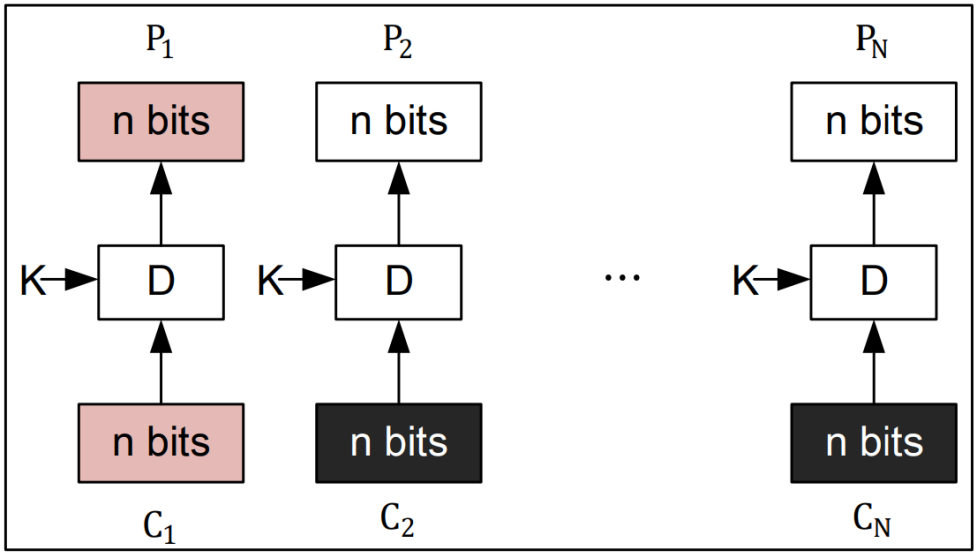


현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - ECB 모드(5/7)
    - 오류 파급
      - 암호화, 복호화 모두 오류가 발생한 블록에 대응되는 블록만 오류를 발생시키고 다른 블록에 영향을 주지 않음

오류가 발생하는 부분은 붉은 색(■)으로 표시





현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - ECB 모드(6/7)
    - 장점
      - 암호문의 단일 비트 오류가 다른 평문 블록에는 영향을 주지 않음
        - CTS 사용 시, N번째와 $N-1$번째 평문, 암호문 간에 오류 전파가 발생함
      - 암·복호화 모두 병렬 처리 가능
        - CTS 사용 시, N번째와 $N-1$번째 평문, 암호문은 병렬 처리가 불가능함
    - 단점
      - 블록 단위의 패턴이 유지
        - 같은 평문 블록은 대응되는 암호문 블록도 같음
      - 블록 간 독립성으로 인해, 키를 몰라도 특정 암호문을 변조 가능
        - e.g., 회사가 은행에 보내는 암호문의 7번째 블록이 급여 정보일 때, 도청자 Eve는 Alice의 7번째 암호문을 가로채서 자신의 암호문에 붙여 넣으면, Eve는 Alice의 급여를 받을 수 있음

현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - ECB 모드(7/7)
    - 응용
      - 병렬 처리, 블록의 독립성이 필요한 분야에서 사용될 수 있음
      - e.g., 데이터베이스 레코드
        - 병렬 처리: 다수의 데이터베이스 레코드를 병렬적으로 암호화 가능
        - 블록의 독립성: 레코드가 블록 단위라면, 다른 레코드에 영향을 주지 않고 수정한 레코드에 임의로 암·복호화 가능

현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CBC(Cipher Block Chaining) 모드(1/6)
    - 동작
      - 각 평문 블록은 이전 암호문 블록과 XOR을 수행하고 암호화
      - 첫 번째 평문 블록에서는 IV(Initialization Vector)를 사용하여 XOR
    - 특징
      - IV 사용
      - 패딩 필요
      - 블록간 연관성 존재
      - 자기 복구
        - e.g., 암호문 블록 C_j의 오류는 평문 블록 P_{j+2}부터 영향을 주지 않음

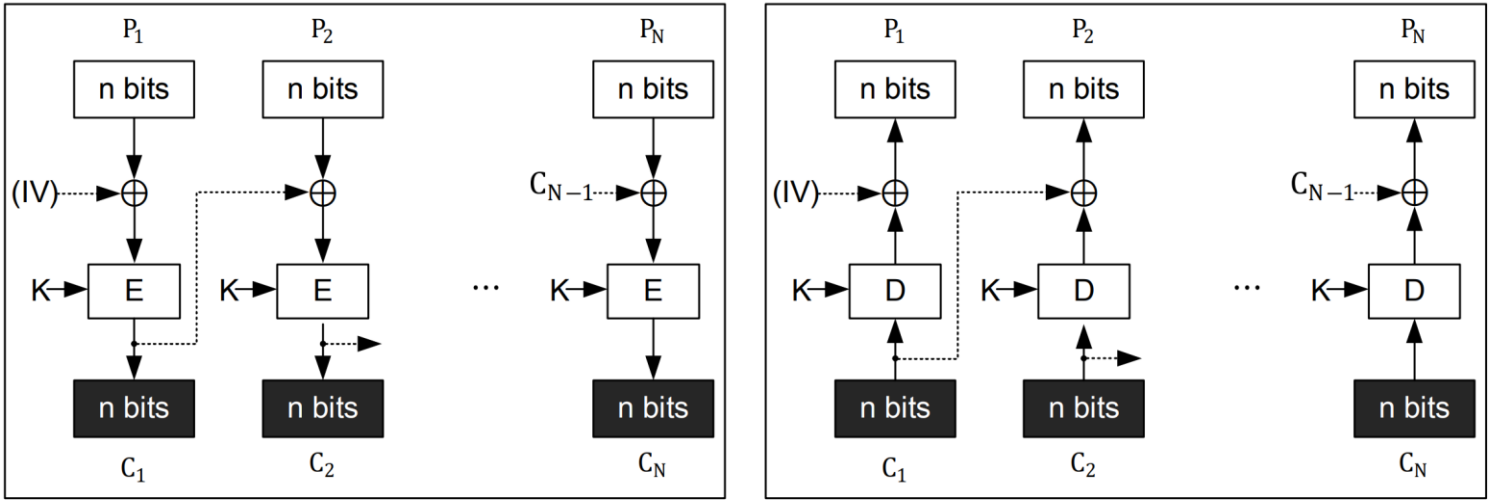
현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CBC 모드(2/6)
    - IV
      - 정의
        - 암호의 초기 상태를 제공하는 임의의 값
      - 특징
        - 송신자와 수신자 간에 공유되어야 함
          - 반드시 비밀일 필요는 없음
        - 변조되어서는 안됨(무결성 보장)
          - 변조되면 첫 번째 블록의 결과가 바뀜(복호화)
          - 이를 위해, 안전한 채널로 IV를 전송하는 것이 권장됨

현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CBC 모드(3/6)
    - 암호화
      - $C_0 = IV,\ \ C_i = E_k(P_i \oplus C_{i-1})$
    - 복호화
      - $C_0 = IV,\ \ P_i = D_k(C_i) \oplus C_{i-1}$

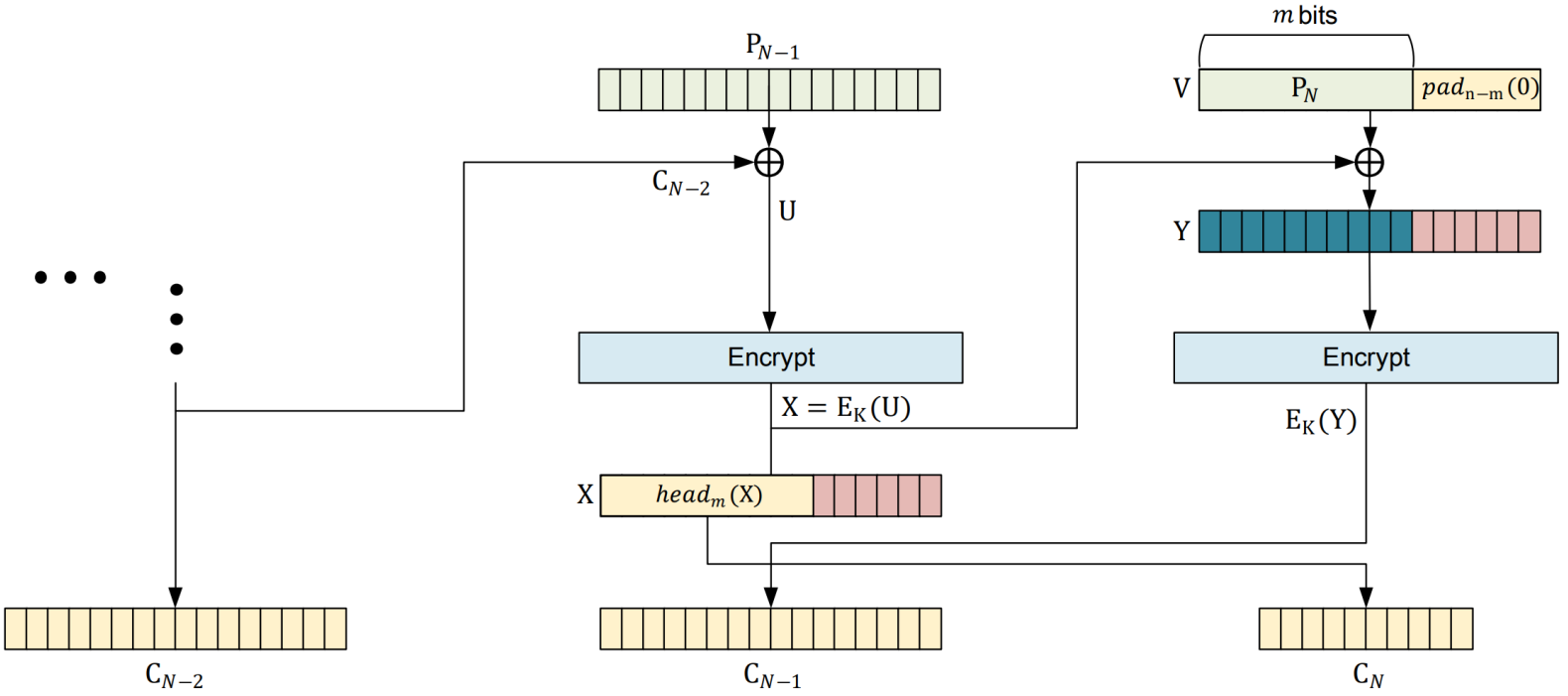
E : Encryption
P_i : Plaintext block i
K : Secret key
D : Decryption
C_i : Ciphertext block i
IV : Initialization vector (C_0)



현대 대칭-키 암호를 이용한 암호화 기법

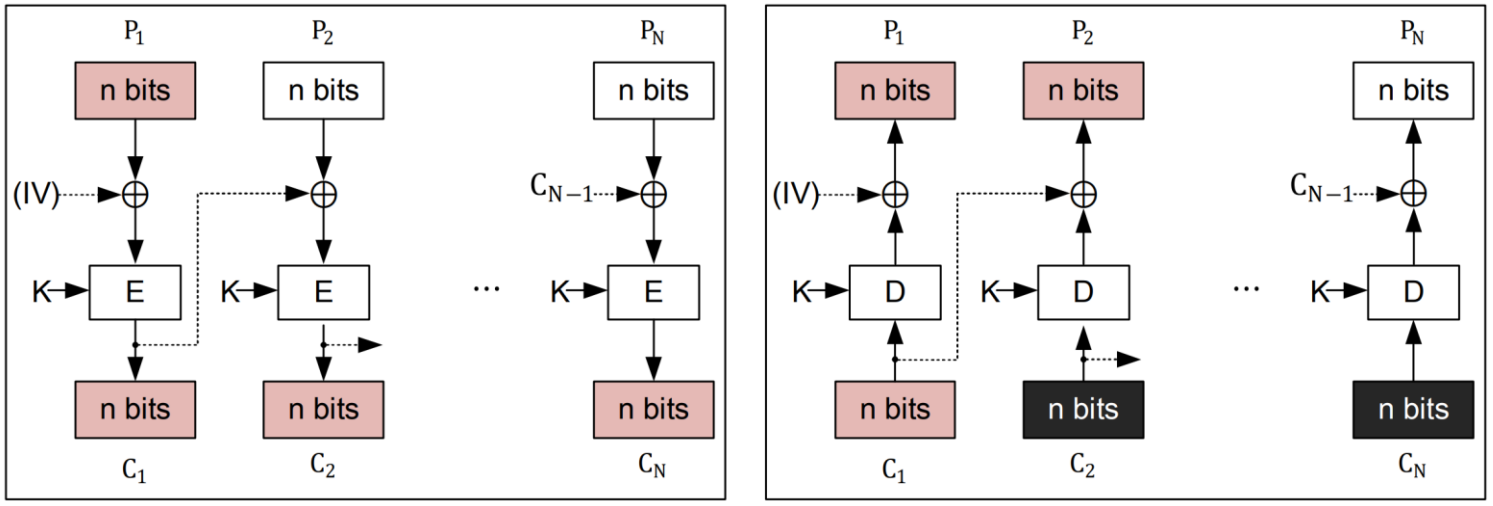
- 운영 모드
  - CBC 모드(4/6)
    - CTS
      - 동작
        - $\mathrm{U} = \mathrm{P}_{N-1} \oplus \mathrm{C}_{N-2} \rightarrow \mathrm{X} = \mathrm{E}_{\mathrm{K}}(\mathrm{U}) \rightarrow \mathrm{C}_N = head_m(\mathrm{X})$
        - $\mathrm{V} = \mathrm{P}_N | pad_{n-m}(0) \rightarrow \mathrm{Y} = \mathrm{X} \oplus \mathrm{V} \rightarrow \mathrm{C}_{N-1} = \mathrm{E}_{\mathrm{K}}(\mathrm{Y})$

$pad_{n-m}(0)$: $n-m$개의 0을 덧붙이는 함수



현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CBC 모드(5/6)
    - 오류 파급
      - 암호화
        - 한 평문 블록에 오류가 발생하면, 현재 암호문 블록과 이후 모든 암호문 블록에 오류가 발생함
      - 복호화
        - 한 암호문 블록에 오류가 발생하면, 현재 평문 블록과 다음 평문 블록에만 오류가 발생함



현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CBC 모드(6/6)
    - 장점
      - 블록 단위의 패턴이 유지되지 않음
        - 동일 평문은 서로 다른 암호화 블록으로 암호화 됨
      - 복호화 병렬 처리 가능
    - 단점
      - 암호문 에러 시, 다음 평문 블록에도 영향을 미침
      - 암호화 병렬 처리 불가

현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CFB(Cipher FeedBack) 모드(1/7)
    - 동작
      - 이전 암호문을 암호 알고리즘에 입력하여 키 스트림을 얻고, 얻어낸 키를 평문과 XOR연산을 수행
      - 암호화와 복호화 모두 블록 암호의 암호화 함수를 사용
      - 초기 쉬프트 레지스터는 IV로 설정
    - 특징
      - IV 사용
      - 패딩 필요 없음
      - 블록 간 연관성 존재
      - 블록 암호를 스트림 암호로 바꿀 수 있음

현대 대칭-키 암호를 이용한 암호화 기법

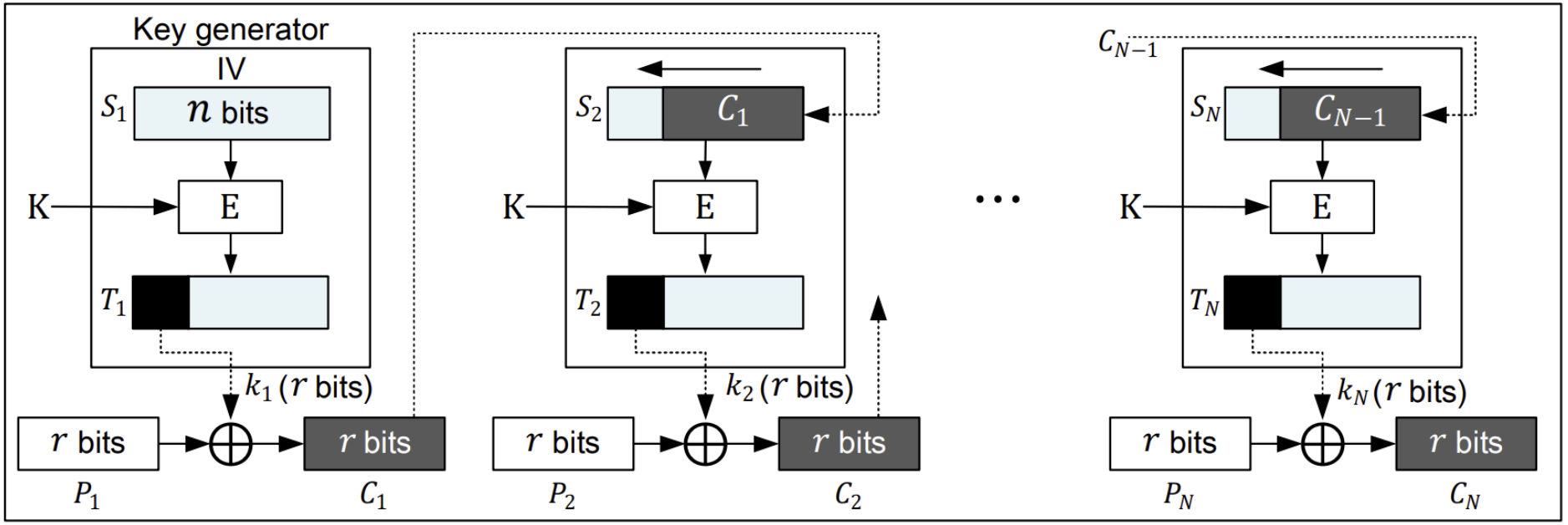
- 운영 모드
  - CFB 모드(2/7)
    - 암호화
      - $C_i = P_i \oplus \text{SelectLeft}_r\{E_K[\text{ShiftLeft}_r(S_{i-1})|C_{i-1}]\}$

ShiftLeft_r: r만큼 왼쪽 쉬프트
SelectLeft_r: 상위 r비트 지정

E : Encryption
P_i : Plaintext block i
K : Secret key

D : Decryption
C_i : Ciphertext block i
IV : Initialization vector (S_1)

S_i : Shift register
T_i : Temporary register



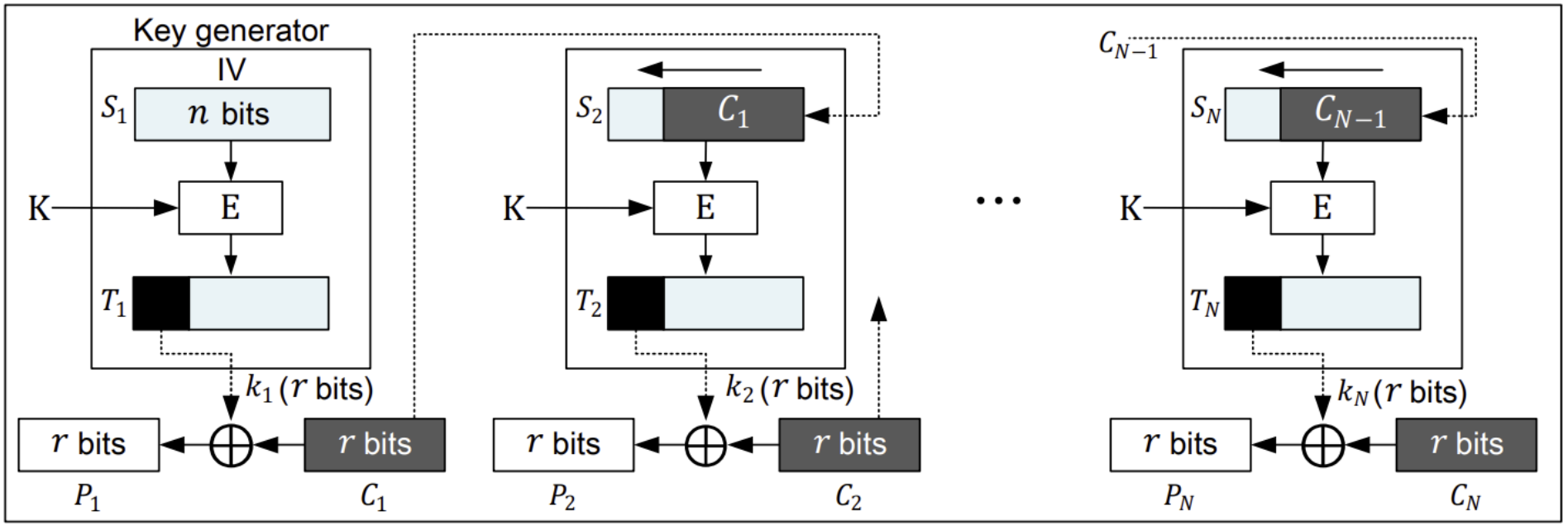
Encryption

현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CFB 모드(3/7)
    - 복호화
      - $\mathrm{P}_i = \mathrm{C}_i \oplus \mathrm{SelectLeft}_r\{\mathrm{E_K}[\mathrm{ShiftLeft}_r(\mathrm{S}_{i-1})|\mathrm{C}_{i-1}]\}$

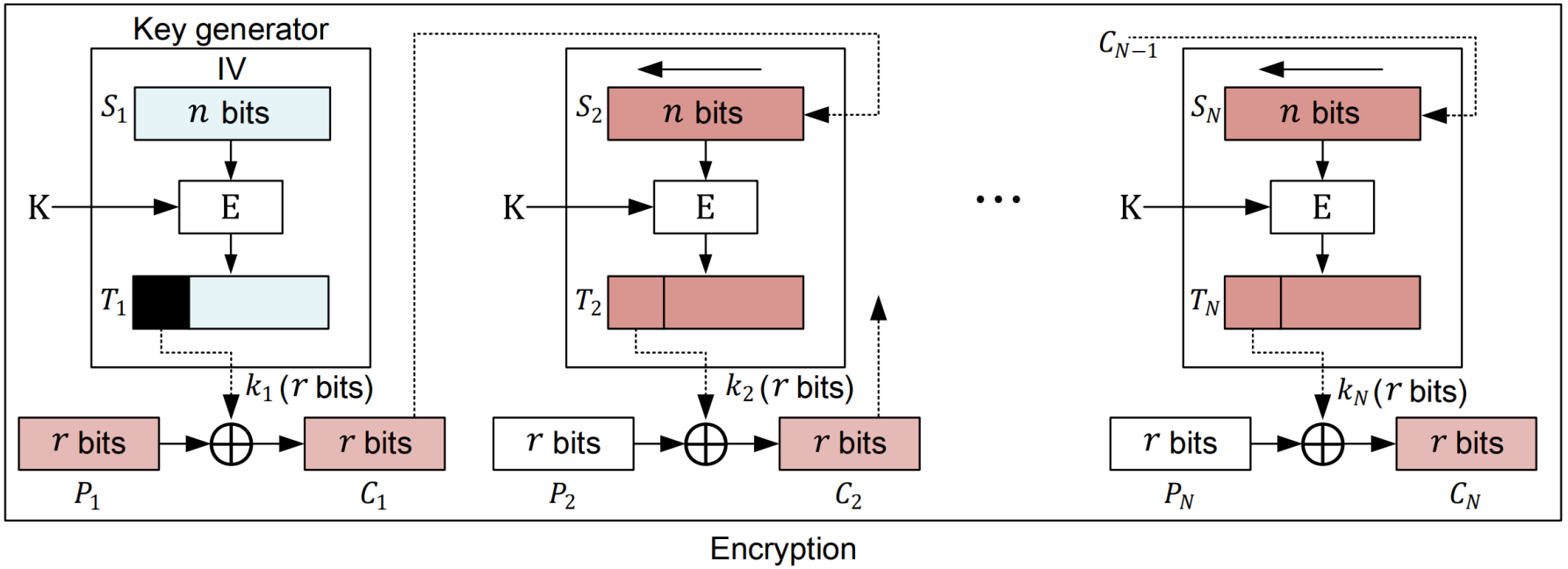
$\mathrm{ShiftLeft}_r$: r만큼 왼쪽 쉬프트
$\mathrm{SelectLeft}_r$: 상위 r비트 지정

E : Encryption
P_i : Plaintext block i
K : Secret key
D : Decryption
C_i : Ciphertext block i
IV : Initialization vector (S_1)
S_i : Shift register
T_i : Temporary register



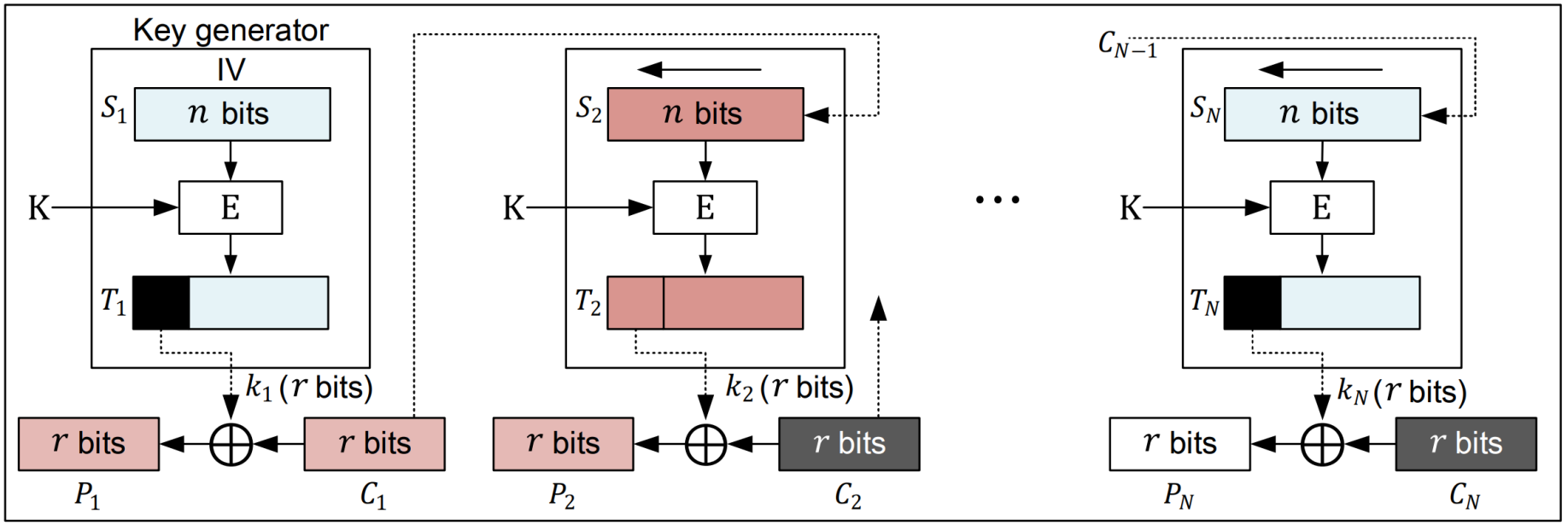
현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CFB 모드(4/7)
    - 오류 파급
      - 암호화
        - 평문 블록에 오류가 발생하면, 현재 암호문 블록과 이후 모든 암호문에 오류가 발생함



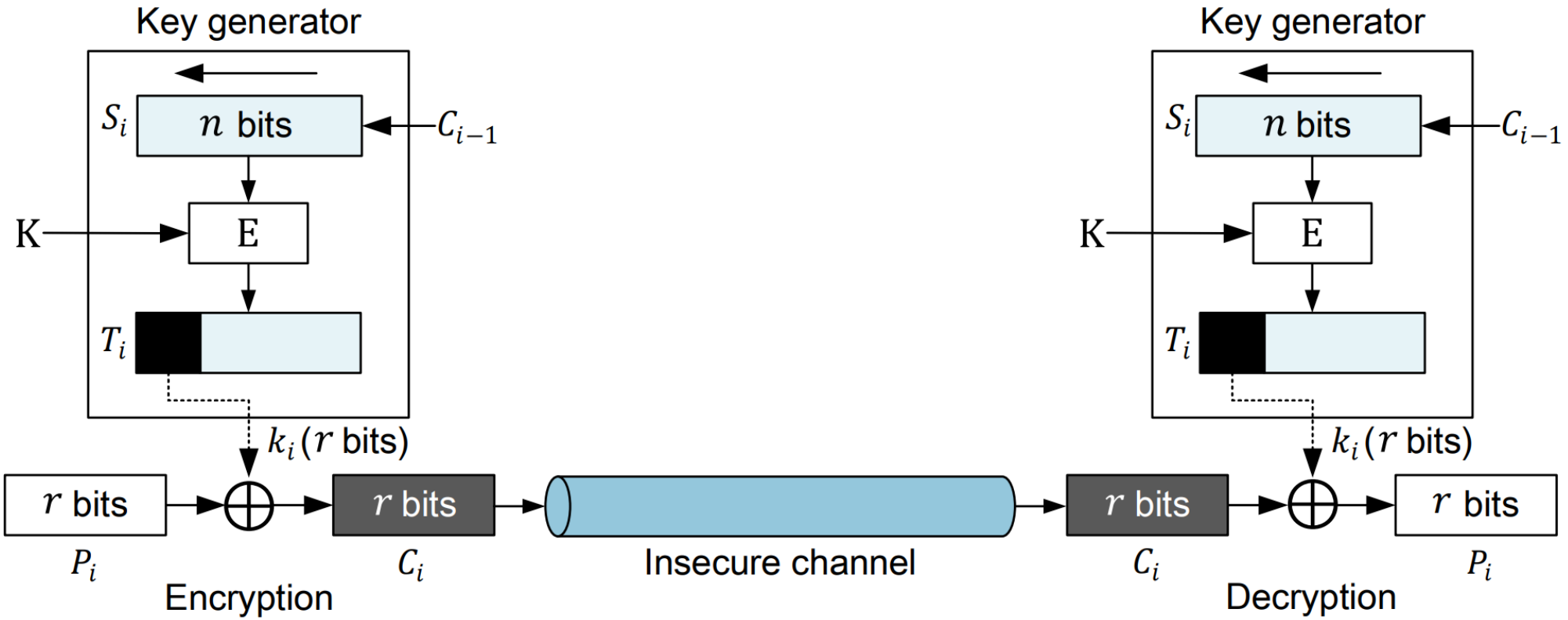
현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CFB 모드(5/7)
    - 오류 파급
      - 복호화
        - 암호문 블록에 오류가 발생하면, 현재 평문 블록과 이후 Shift register에 오류가 존재하는 동안의 평문 블록에 오류가 발생함



현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CFB 모드(6/7)
    - 스트림 암호로서의 CFB 모드
      - 키 스트림이 암호문에 의존하는 비동기식 스트림 암호임



현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CFB 모드(7/7)
    - 장점
      - 블록 단위의 패턴이 유지되지 않음
      - 복호화 병렬 처리 가능
    - 단점
      - 암호문 에러 시, 다음 평문 블록에도 영향을 미침
      - 암호화 병렬 처리 불가

현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - OFB(Output FeedBack) 모드(1/7)
    - 동작
      - 이전 암호문을 암호 알고리즘에 입력하지 않고, 이전 암호 알고리즘의 출력을 입력하여 얻어낸 키 스트림으로 평문과 XOR을 수행
      - 암호화와 복호화 모두 블록 암호의 암호화 함수를 사용
      - 초기 쉬프트 레지스터는 IV로 설정
    - 특징
      - CFB에서 오류 파급을 개선함
      - IV 사용
      - 패딩 필요 없음
      - 블록 간 독립성 존재
      - 블록 암호를 스트림 암호로 바꿀 수 있음

현대 대칭-키 암호를 이용한 암호화 기법

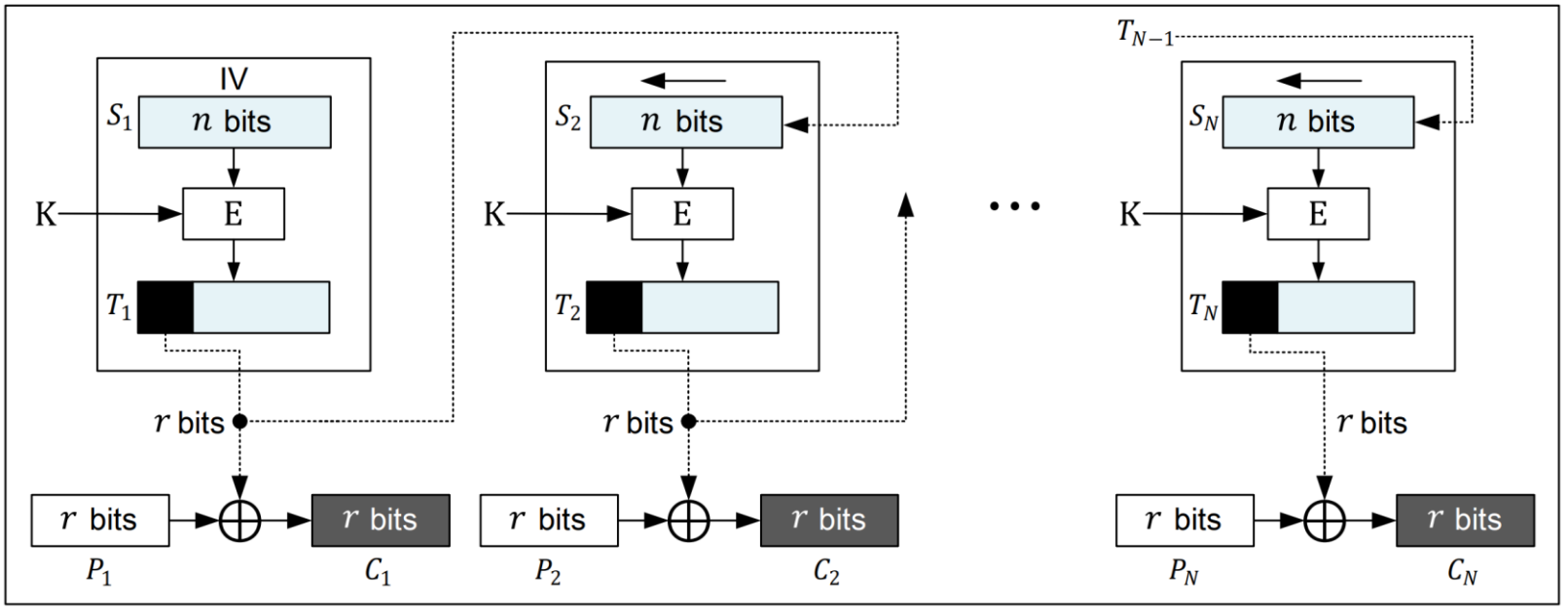
- 운영 모드
  - OFB 모드(2/7)
    - 암호화
      - $C_i = P_i \oplus \text{SelectLeft}_r\{E_K[\text{ShiftLeft}_r(S_{i-1}) | \text{SelectLeft}_r(T_{i-1})]\}$

ShiftLeft_r: r만큼 왼쪽 쉬프트
SelectLeft_r: 상위 r비트 지정

E : Encryption
P_i : Plaintext block i
K : Secret key

D : Decryption
C_i : Ciphertext block i
IV : Initialization vector (S_1)

S_i : Shift register
T_i : Temporary register

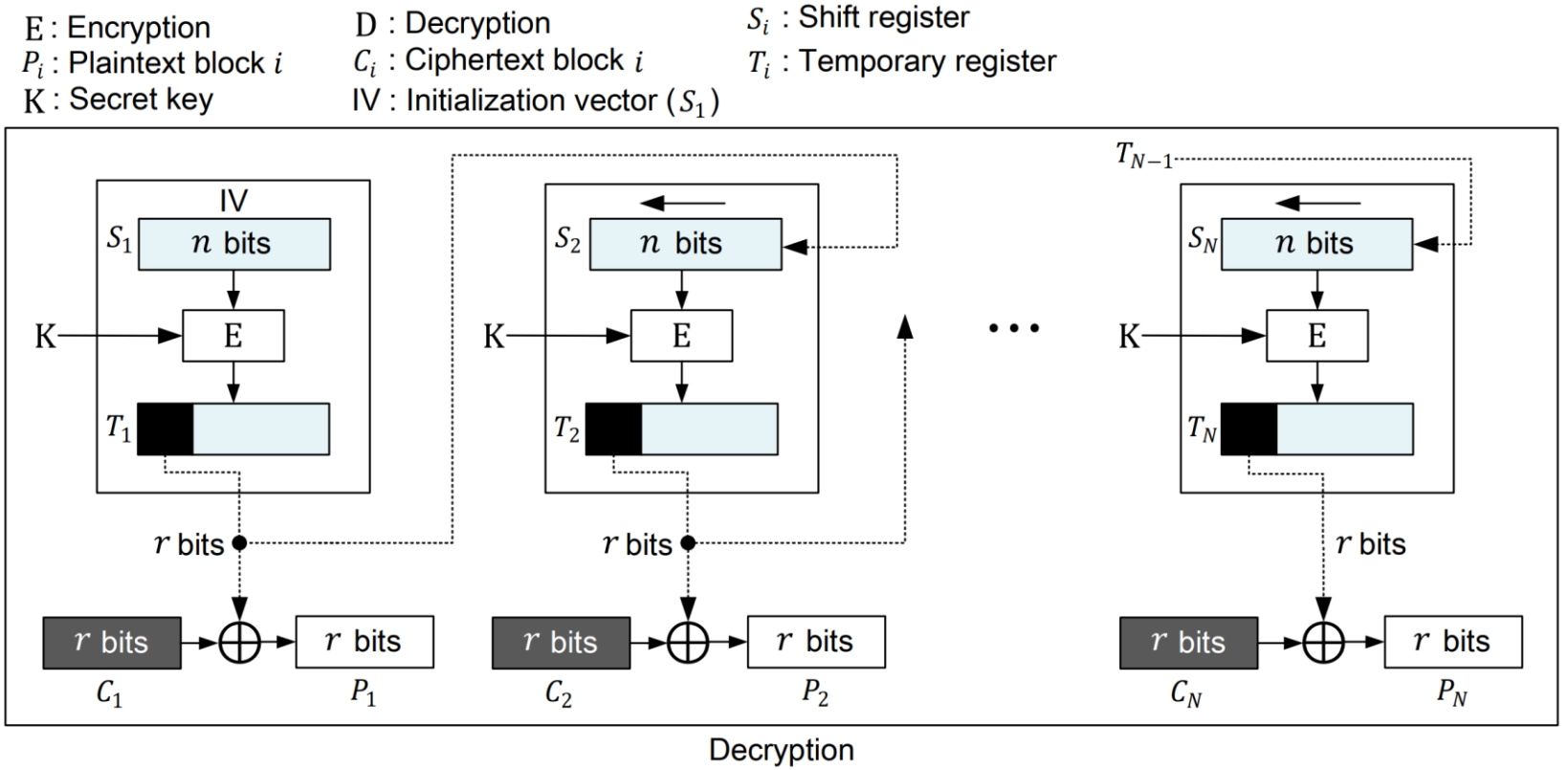


Encryption

현대 대칭-키 암호를 이용한 암호화 기법

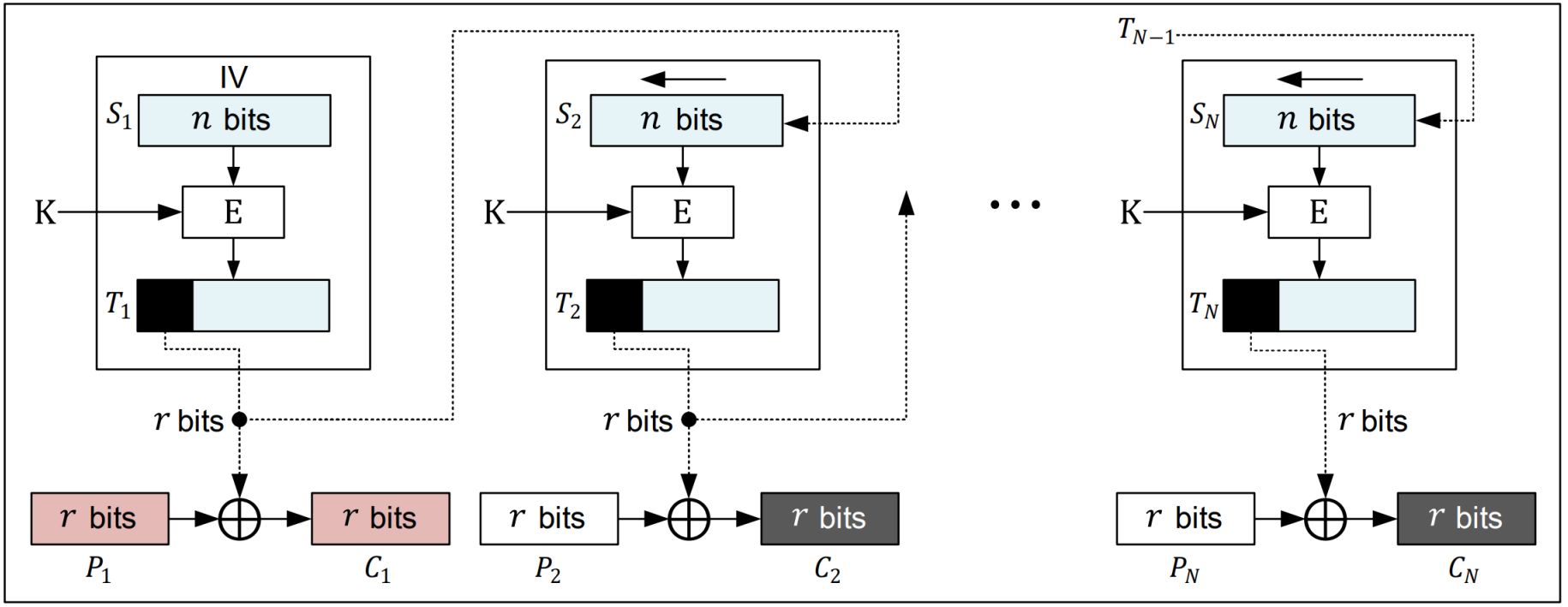
- 운영 모드
  - OFB 모드(3/7)
    - 복호화
      - $\mathrm{P}_i = \mathrm{C}_i \oplus \mathrm{SelectLeft}_r\{\mathrm{E_K}[\mathrm{ShiftLeft}_r(\mathrm{S}_{i-1})|\mathrm{SelectLeft}_r(\mathrm{T}_{i-1})]\}$

$\mathrm{ShiftLeft}_r$: r만큼 왼쪽 쉬프트
$\mathrm{SelectLeft}_r$: 상위 r비트 지정



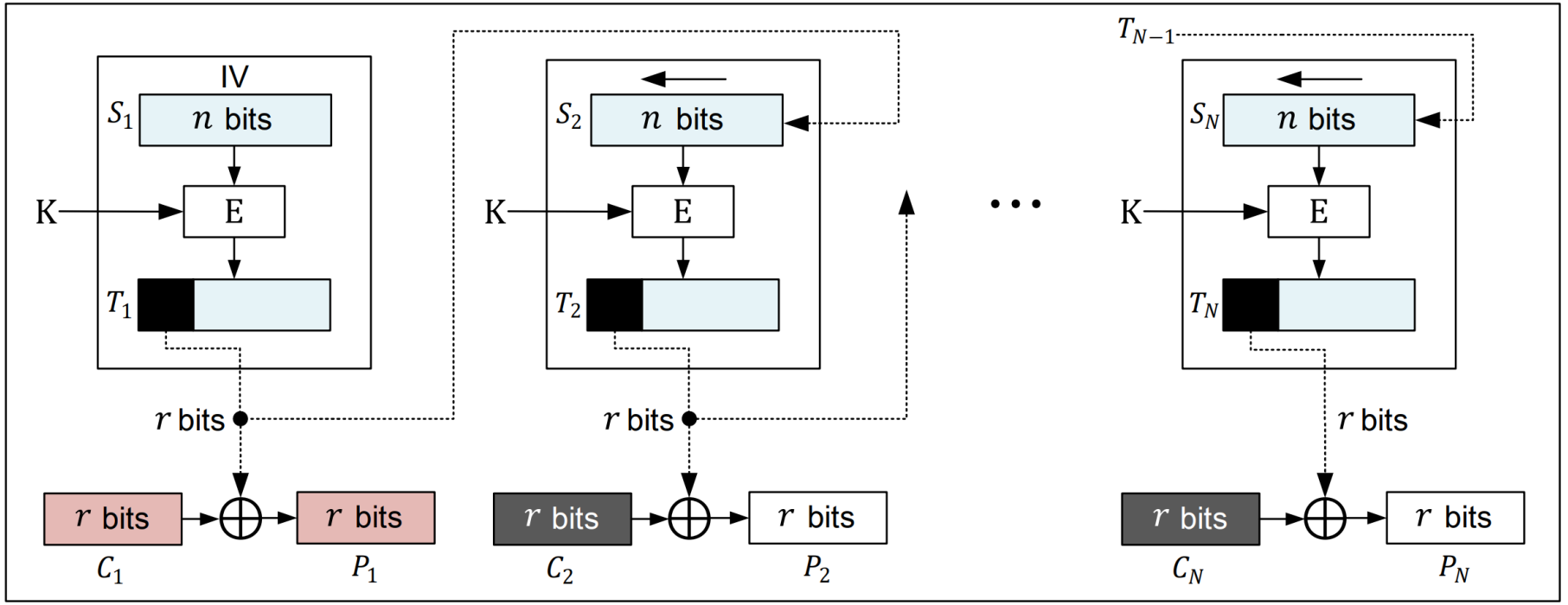
현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - OFB 모드(4/7)
    - 오류 파급
      - 암호화
        - 평문 블록에 오류가 발생하면, 현재 암호문 블록에만 오류가 발생함



현대 대칭-키 암호를 이용한 암호화 기법

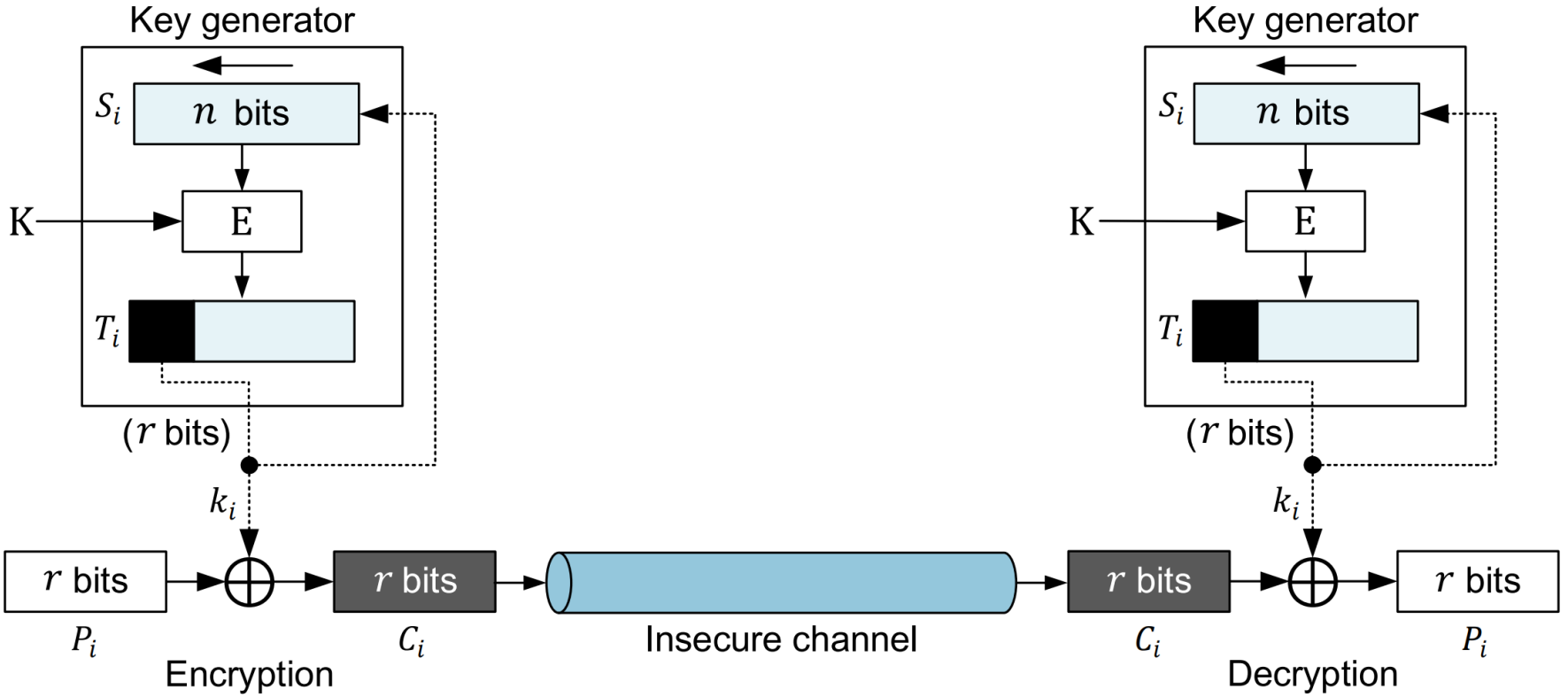
- 운영 모드
  - OFB 모드(5/7)
    - 오류 파급
      - 복호화
        - 암호문 블록에 오류가 발생하면, 현재 평문 블록에만 오류가 발생함



Decryption

현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - OFB 모드(6/7)
    - 스트림 암호로서의 OFB 모드
      - 키 스트림이 평문이나 암호문과 독립이므로 동기식 스트림 암호임



현대 대칭-키 암호를 이용한 암호화 기법

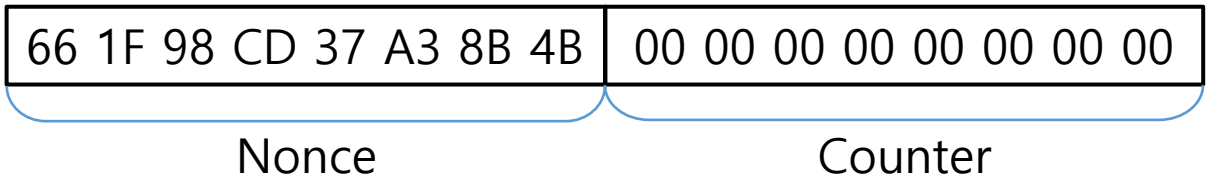
- 운영 모드
  - OFB 모드(7/7)
    - 장점
      - 블록 단위의 패턴이 유지되지 않음
        - 동일 평문은 서로 다른 암호화 블록으로 암호화 됨
      - 암호문 에러 시에도 다른 평문 블록에 영향을 주지 않음
    - 단점
      - 암·복호화 병렬 처리 불가
      - 비트 플립 공격 시, CTR 모드를 제외한 다른 운영모드에 비해 탐지가 어려움

현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CTR(Counter) 모드(1/8)
    - 동작
      - 카운터를 암호화하고 평문 블록과 XOR을 수행
      - 암호화와 복호화 모두 블록 암호의 암호화 함수를 사용
      - 초기 카운터는 IV로 초기화
      - 카운터는 사전에 정의된 규칙에 따라 증가함
    - 특징
      - IV 사용
      - 패딩 필요 없음
      - 블록 간 독립성 존재
      - 블록 암호를 스트림 암호로 바꿀 수 있음
      - 암호화 시 피드백이 없음

현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CTR 모드(2/8)
    - 카운터
      - 카운터는 일반적으로 다음의 구조를 가짐



66 1F 98 CD 37 A3 8B 4B 00 00 00 00 00 00 00 00 (초기값)

66 1F 98 CD 37 A3 8B 4B 00 00 00 00 00 00 00 01 (카운터+1)

66 1F 98 CD 37 A3 8B 4B 00 00 00 00 00 00 00 02 (카운터+2)

•
•
•

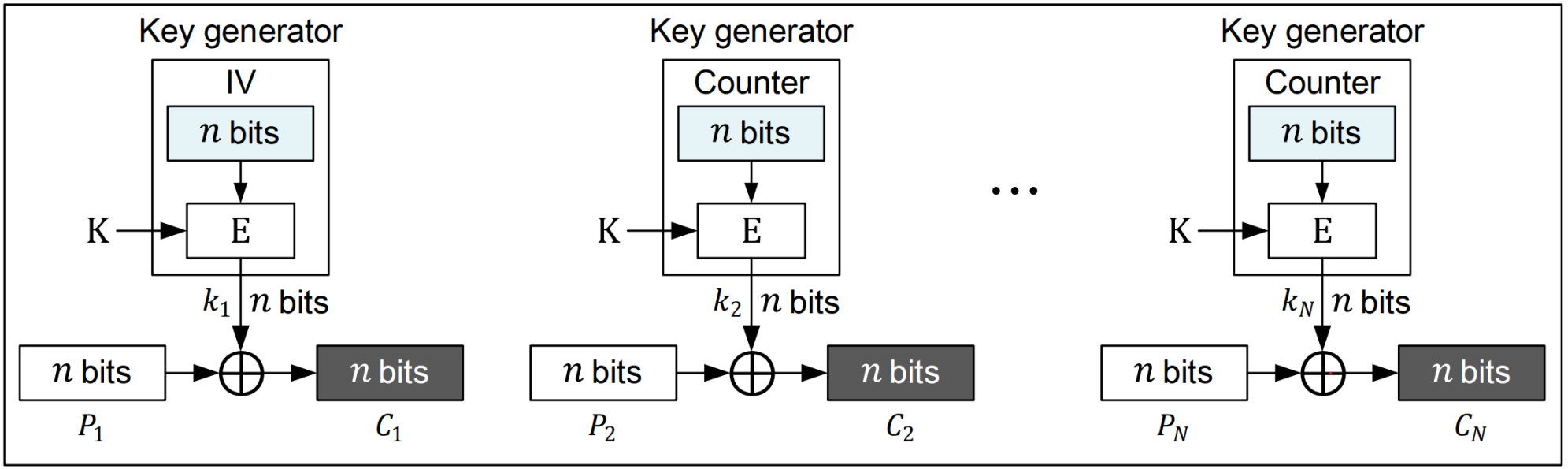
현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CTR 모드(3/8)
    - 암호화
      - $C_i = P_i \oplus E_{ki}(Counter)$

E : Encryption
P_i : Plaintext block i
K : Secret key

IV : Initialization vector
C_i : Ciphertext block i
k_i : Encryption key i

The counter is incremented for each block.



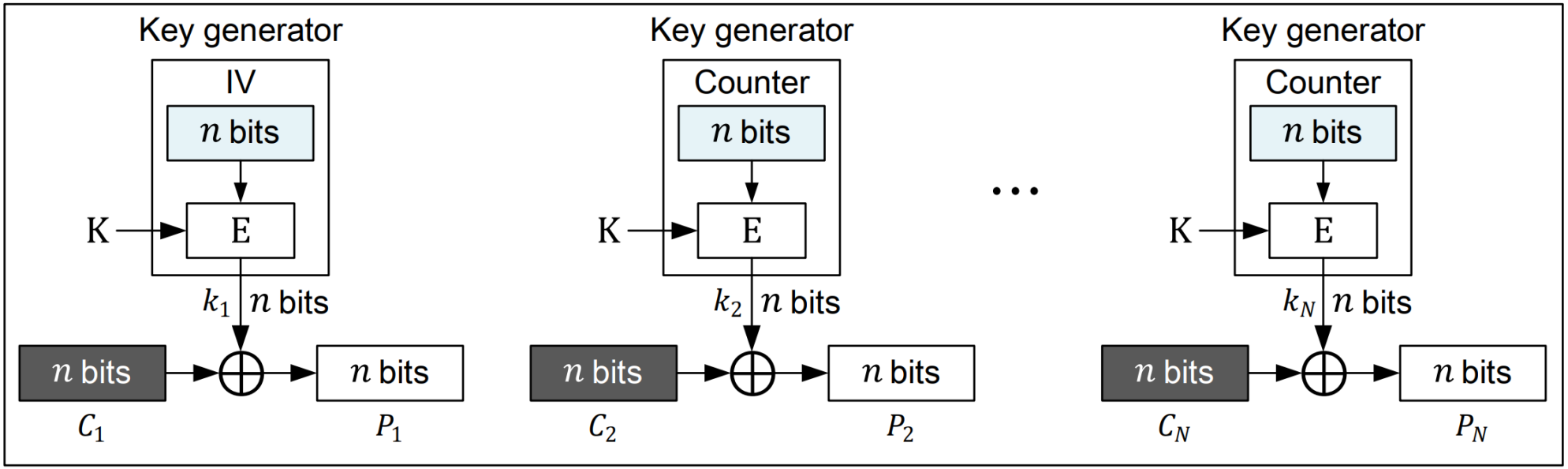
Encryption

현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CTR 모드(4/8)
    - 복호화
      - $\mathrm{P_i} = \mathrm{C_i} \oplus \mathrm{E_{ki}}(\mathrm{Counter})$

E : Encryption
P_i : Plaintext block i
K : Secret key
IV : Initialization vector
C_i : Ciphertext block i
k_i : Encryption key i

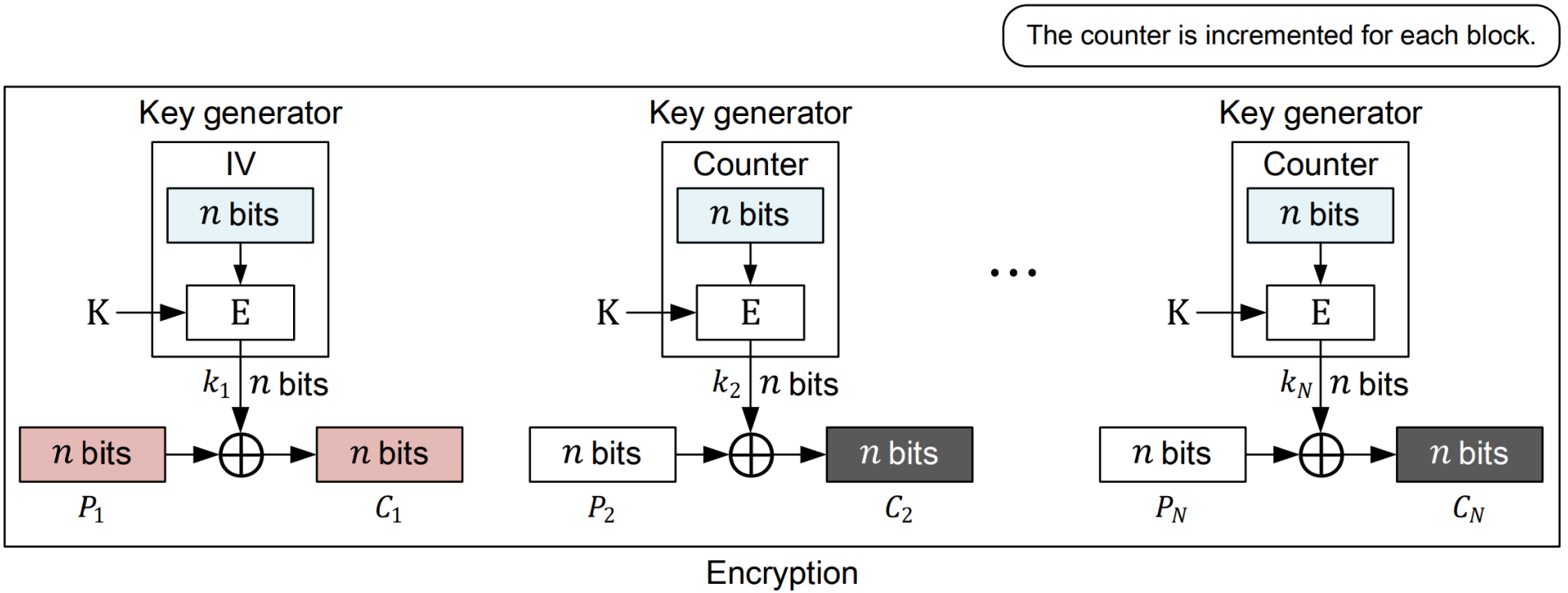
The counter is incremented for each block.



Decryption

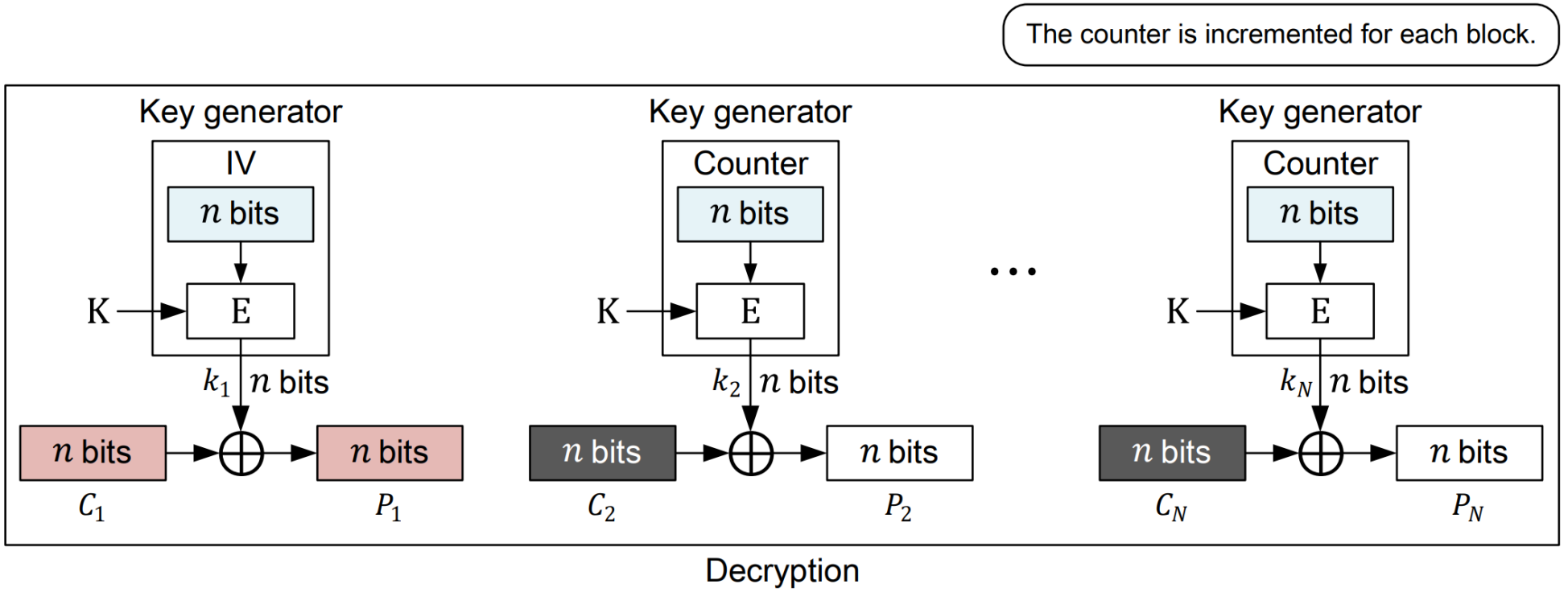
현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CTR 모드(5/8)
    - 오류 파급
      - 암호화
        - 평문 블록에 오류가 발생하면, 현재 암호문 블록에만 오류가 발생함



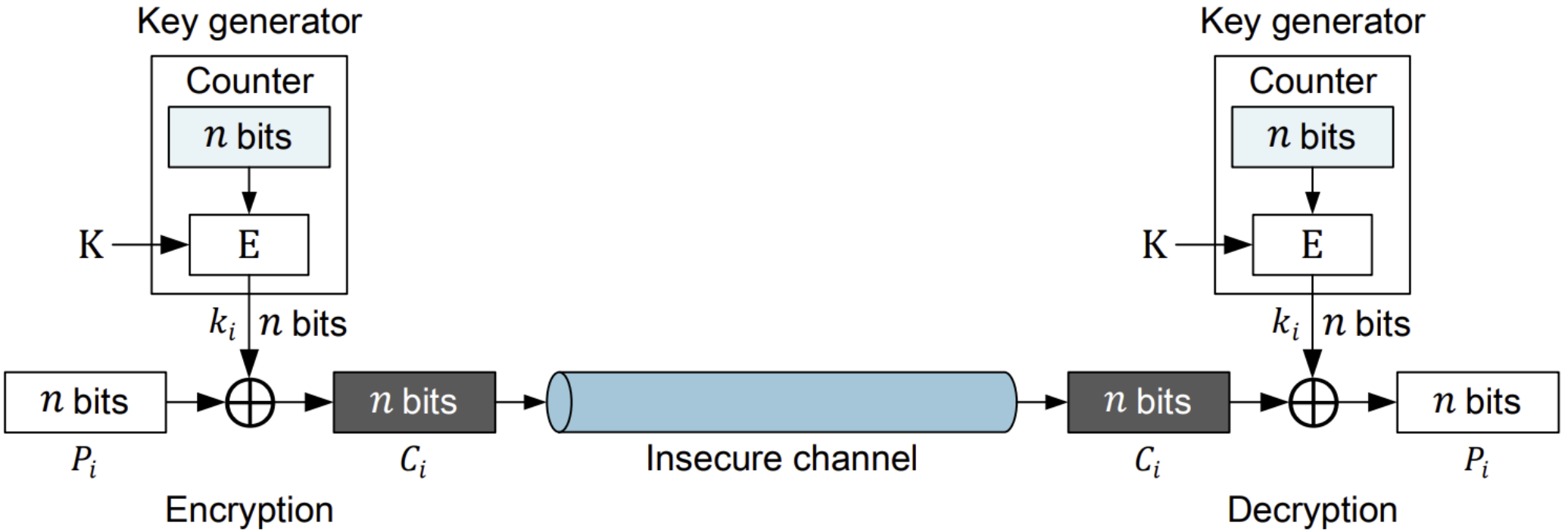
현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CTR 모드(6/8)
    - 오류 파급
      - 복호화
        - 암호문 블록에 오류가 발생하면, 현재 평문 블록에만 오류가 발생함



현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CTR 모드(7/8)
    - 스트림 암호로서의 CTR 모드
      - 키 스트림이 평문이나 암호문과 독립이므로 동기식 스트림 암호임



현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - CTR 모드(8/8)
    - 장점
      - 블록 단위의 패턴이 유지되지 않음
        - 동일 평문은 서로 다른 암호화 블록으로 암호화 됨
      - 암호문 에러 시에도 다른 블록에 영향을 주지 않음
      - 암·복호화 모두 병렬 처리 가능
    - 단점
      - 비트 플립 공격 시, OFB 모드를 제외한 다른 운영모드에 비해 탐지가 어려움

현대 대칭-키 암호를 이용한 암호화 기법

- 운영 모드
  - 운영 모드 비교
    - CBC, CTR 모드가 권장 사용됨
    - ECB, CBC, CTR 모드는 큰 단위의 데이터, CFB, OFB 모드는 작은 단위 데이터 암호화에 유리함
      - 문자나 비트 같은 작은 단위의 데이터 암호화 기법을 위해 스트림 암호가 필요

운영 모드	패딩 필요	IV 사용	블록 간의 관계	병렬 처리	스트림 암호로서의 사용	단위 데이터 크기
ECB	O	X	독립	암·복호화	N/A	n
CBC	O	O	연관	복호화	N/A	n
CFB	X	O	연관	복호화	비동기식	$r \leq n$
OFB	X	O	독립	불가	동기식	$r \leq n$
CTR	X	O	독립	암·복호화	동기식	n

현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - 스트림 암호 (Stream Cipher)
    - 특징
      - 블록 암호보다 빠른 속도를 가짐
      - 블록 암호와 달리 실시간 처리 가능
      - 무선 통신 등의 환경에 주로 사용됨
    - 종류
      - RC4, A5/1 등

현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - RC4
    - 개요
      - 바이트 단위로 작동하도록 설계된 스트림 암호
      - 다양한 크기의 키(1~256 바이트)를 사용함
      - WEP(Wired Equivalent Privacy)와 WPA(Wi-Fi Protected Access)에서 사용함

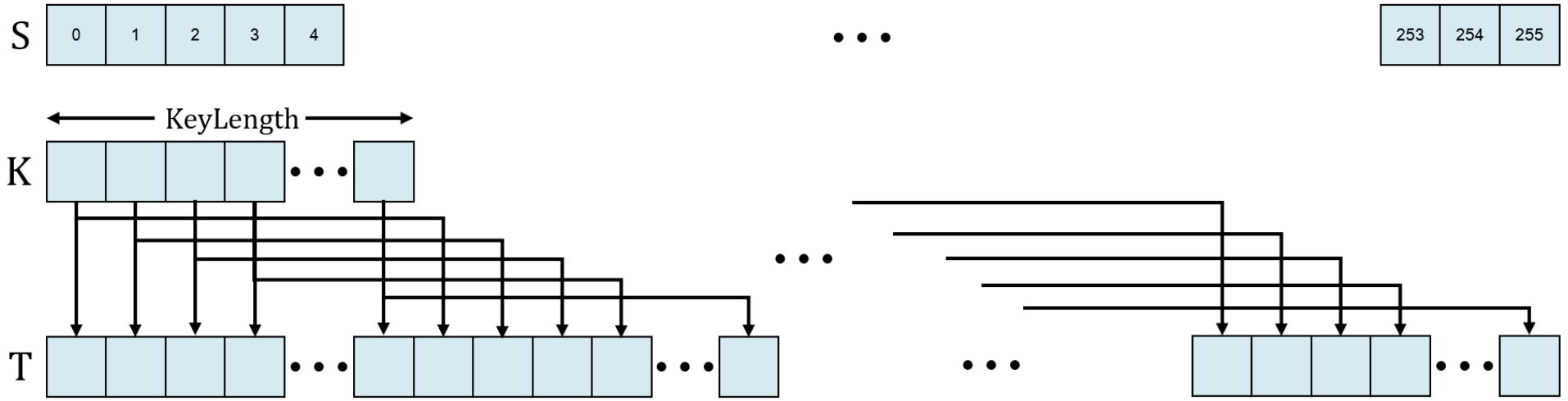
현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - RC4
    - 구성 요소
      - 키(K)
      - 상태(state) 배열(S)
      - 임시(temporary) 배열(T)
    - 과정
      1. S와 T의 초기화
      2. KSA(Key Scheduling Algorithm)를 통한 S의 초기 치환
      3. PRGA(Pseudo-Random Generation Algorithm)를 통한 키 스트림 생성
      4. 암호화 수행

현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - RC4
    - 과정(1/4)
      1. S와 T의 초기화
         - S는 $0, 1, \cdots, 255$로 초기화 됨
         - T는 256바이트가 채워질 때까지 K를 반복적으로 저장함

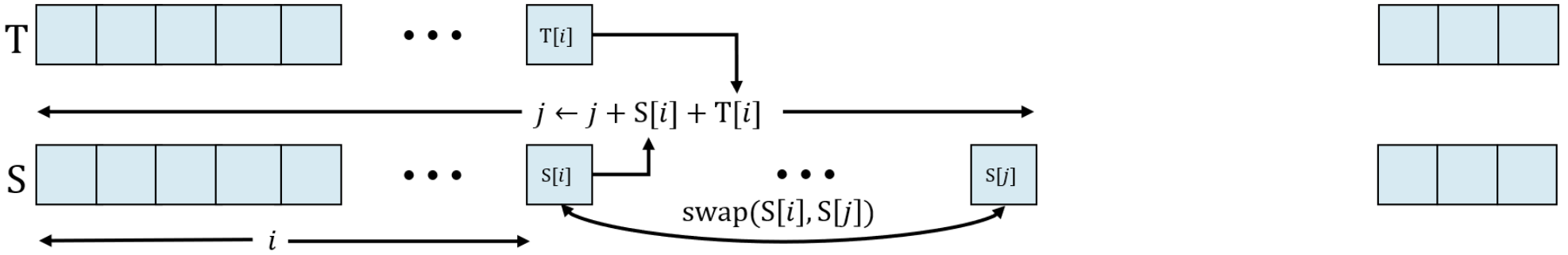
$$
\begin{aligned}
&\text{for}(i = 0 \text{ to } 255)\\
&\{\\
&\quad \mathrm{S}[i] \leftarrow i\\
&\quad \mathrm{T}[i] \leftarrow \mathrm{K}[i \bmod \text{KeyLength}]\\
&\}
\end{aligned}
$$



현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - RC4
    - 과정(2/4)
      2. KSA를 통한 S의 초기 치환
         - T에 따라 $S[i]$를 S의 다른 바이트와 교환
         - S는 완벽히 섞임

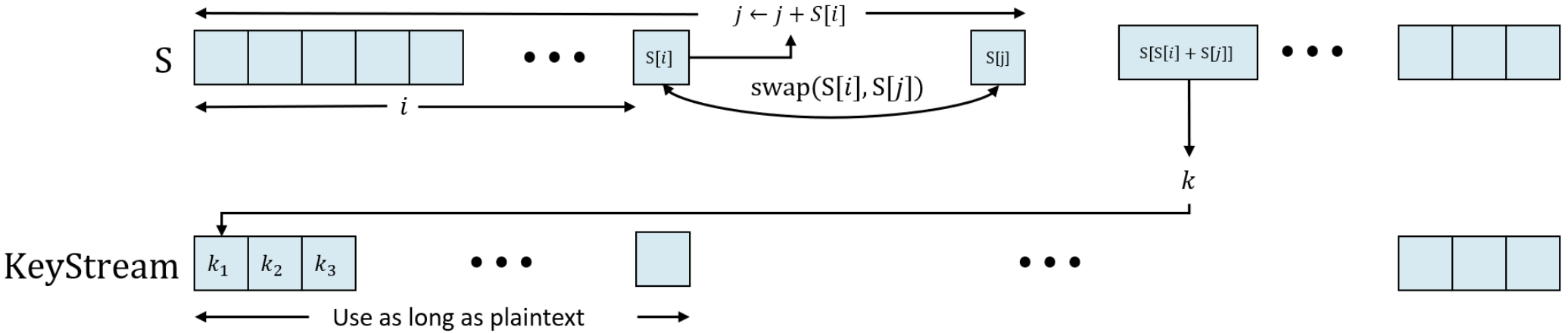
$$
\begin{aligned}
&j \leftarrow 0 \\
&\text{for}(i = 0 \text{ to } 255) \\
&\{ \\
&\quad j \leftarrow (j + \mathrm{S}[i] + \mathrm{T}[i]) \bmod 256 \\
&\quad \text{swap}(\mathrm{S}[i], \mathrm{S}[j]) \\
&\}
\end{aligned}
$$



현대 대칭-키 암호를 이용한 암호화 기법

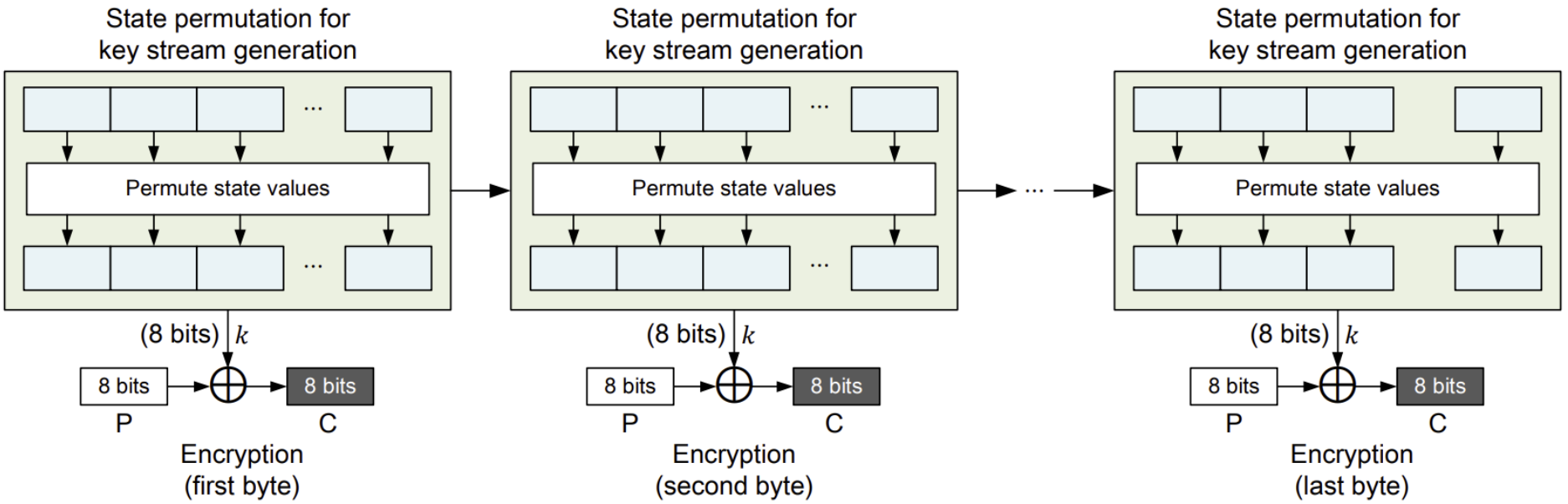
- 스트림 암호의 사용
  - RC4
    - 과정(3/4)
      3. PRGA를 통한 키 스트림 생성
         - i값에 따른 j 값 계산
         - S[i]와 S[j]의 교환
         - S[i]와 S[j]에 따른 키 스트림(k) 생성

```
i, j ← 0
while(true)
{
   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
  - RC4
    - 과정(4/4)
      - 암호화 수행
        - 키 스트림 k와 평문 1바이트를 XOR함
        - 이를 반복하여 암호문을 얻음



현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - RC4
    - 보안 취약성
      - 키 스트림의 일부 값이 편향됨
        - e.g., FMS(Fluhrer, Matin and Shamir) 공격을 통한 WEP 해독이 가능
      - 의사 난수가 완벽한 난수가 아니므로 안전성 문제가 있음
        - 의사 난수 예측으로 인한 키 스트림 분석 공격 가능

현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - A5/1
    - 개요
      - LFSR을 이용해서 키 스트림을 생성하는 스트림 암호
      - 64비트의 키를 사용함
      - 휴대전화 통신을 위한 GSM(Global System for Mobile Communication)에서 사용됨

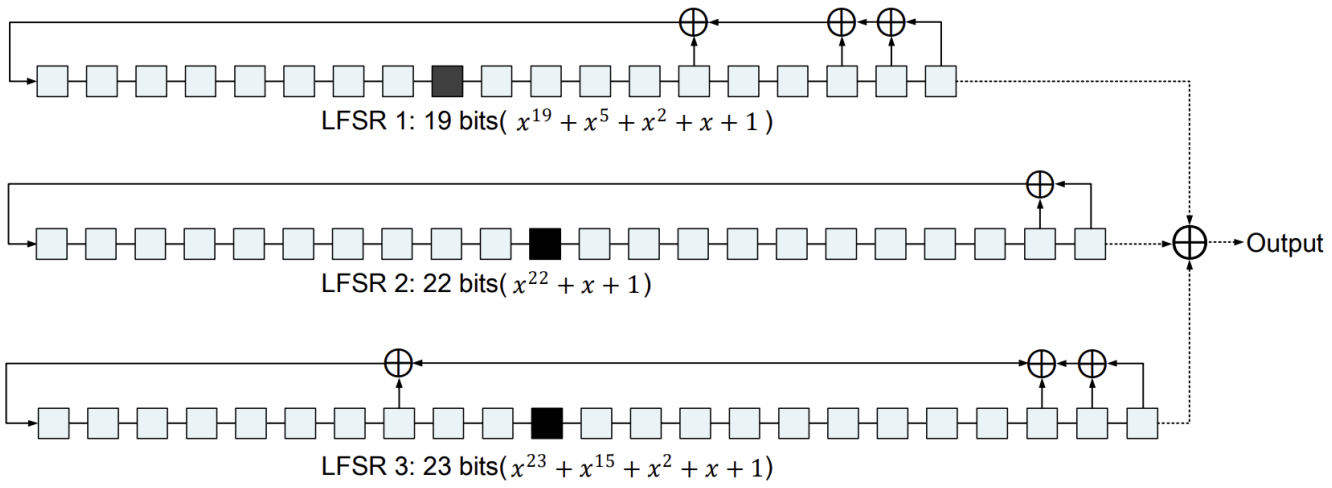
현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - A5/1
    - 구성 요소
      - 키(K)
      - 프레임 번호(FrameNumber)
      - 3개의 LFSR
        - 각각 19, 22, 23 비트 길이를 가짐
      - 클럭킹(clocking) 비트(C)
      - majority 함수
    - 과정
      1. LFSR 초기화
      2. 시프트 레지스터 동기화
      3. 키 스트림 생성
      4. 암호화

클럭킹(clocking)은 LFSR이
한 개의 셀만큼 이동하는 과정을 뜻함

현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - A5/1
    - 과정(1/4)
      1. LFSR 초기화
         - 세 개의 LFSR 셀을 모두 0으로 초기화
         - 클럭킹 비트 설정
         - 키와 LFSR을 XOR
         - 프레임 번호와 LFSR을 XOR



$all\_three\_LFSR \leftarrow \{0,\}$
$C_1 \leftarrow LFSR_1[10]$
$C_2 \leftarrow LFSR_2[11]$
$C_3 \leftarrow LFSR_3[11]$

for $(i = 0$ to $63)$
{
  XOR(K$[i]$, all_three_LFSR)
  Clock(all_three_LFSR)
}

for $(i = 0$ to $21)$
{
  XOR(FrameNumber$[i]$, all_three_LFSR)
  Clock(all_three_LFSR)
}

현대 대칭-키 암호를 이용한 암호화 기법

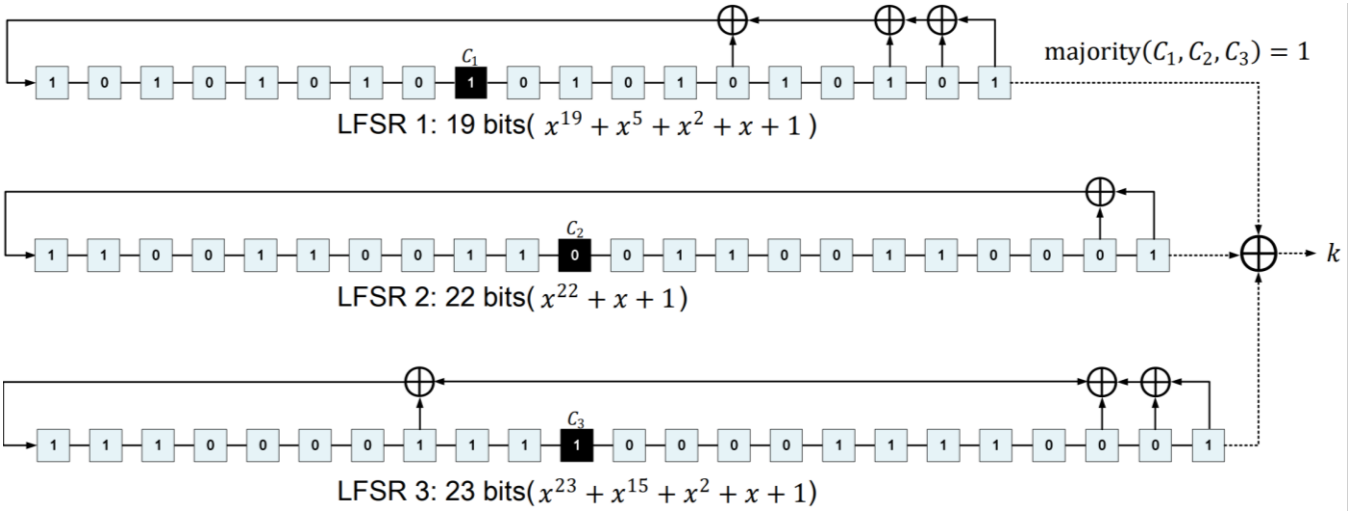
- 스트림 암호의 사용
  - A5/1
    - 과정(2/4)
      2. 시프트 레지스터 동기화
         - 100 사이클 동안 전체 클럭킹 수행
         - 키와 프레임 번호를 확산시킴

```
for (i = 0 to 99)
{
   Clock(all_three_LFSR)
}
```

현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - A5/1
    - 과정(3/4)
      3. 키 스트림 생성
         - 클럭킹 비트를 사용하여 majority 비트를 계산
         - majority 비트와 같은 클럭킹 비트를 가지는 LFSR만 선택적 클럭킹
         - 각 LFSR의 오른쪽 최상위 비트에 따른 키 스트림 생성

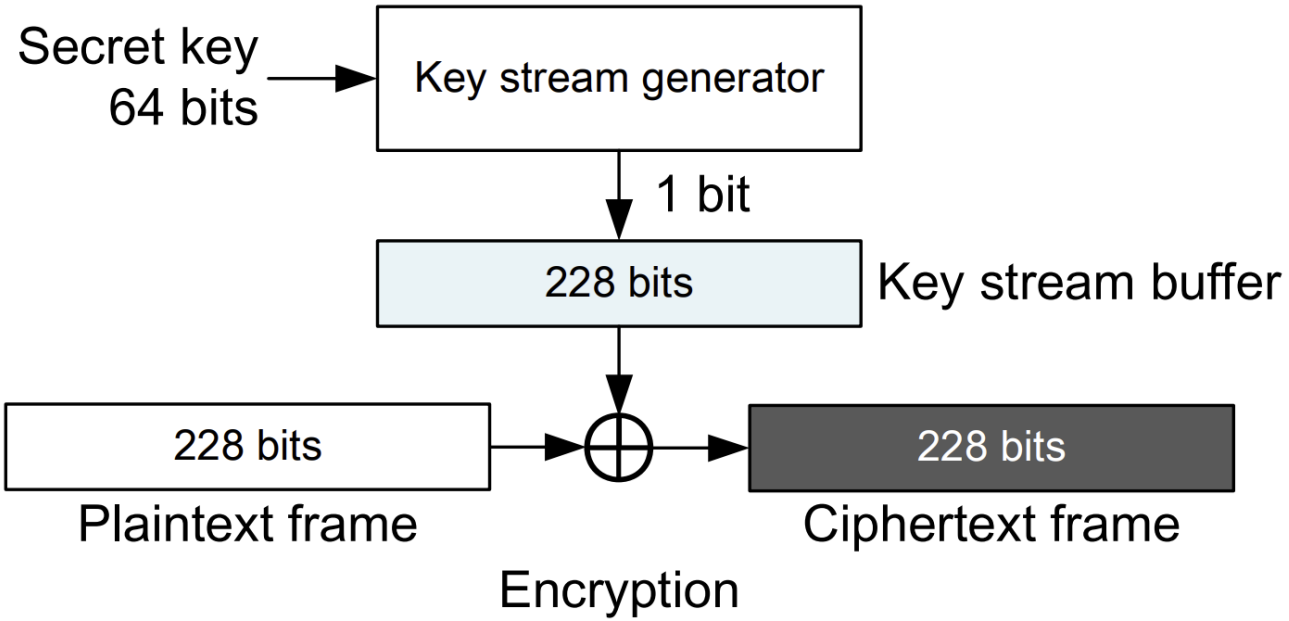
majority(b_1, b_2, b_3)는 b_1, b_2, b_3중 다수 값의 비트를 출력함




```
while (true)
{
  for(i = 0 to 2)
    if C_i == majority(C_1, C_2, C_3)
      Clock(LFSR_i)
  k = XOR_LSB(LFSR_1, LFSR_2, LFSR_3)
}
```

현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - A5/1
    - 과정(4/4)
      4. 암호화
         - 키 스트림을 버퍼에 저장
         - 228 비트 프레임과 XOR을 수행



현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - A5/1
    - 보안 취약성
      - LFSR 구조의 예측 가능성
        - 시간-메모리 트레이드 오프 공격 수행 가능
      - 초기화 절차의 구조적 약점
        - 2~5분의 대화 평문으로 몇 분 안에 해독 가능

현대 대칭-키 암호를 이용한 암호화 기법

- 스트림 암호의 사용
  - 상용 스트림 암호
    - Rabbit
      - 2003년에 처음 제안됨
      - 128비트 키와 64비트 IV를 사용함
      - 높은 부하 환경에서 암호화 스트리밍 미디어에 유용함
    - ChaCha20
      - Salsa20을 개선하여 2008년에 처음으로 제안됨
      - 256비트 키와 128비트 IV를 사용함
      - TLS(Transport Layer Security)에서 주로 사용하는 스트림 암호임

Thanks!

이 정 민(jeongmin@pel.sejong.ac.kr)