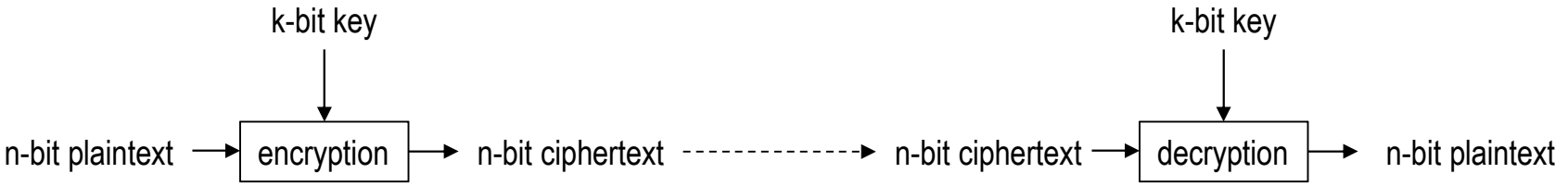
블록암호기술

대칭키 암호 기술

- 블록 암호 기술
  - 개요
  - 대치, 전치 방식 비교
  - Full-size vs. partial-size key cipher
  - 현대 블록 암호 모듈의 주요 요소
    - P-box, S-box 등
  - Product cipher
  - 확산, 혼돈
  - 라운드
  - Product cipher의 두 부류
    - Feistel cipher vs. Non-Feistel cipher
- 스트림 암호 기술

블록 암호 기술:
Introduction

블록 암호 기술(symmetric-key block cipher)



- 암호화 모듈은 평문의 n-bit 블록을 암호화하여 n-bit 암호문을 생성
- 복호화 모듈은 n-bit 암호문 블록을 복호화하여 n-bit 평문 블록을 생성
- 암호화 및 복호화에서 **동일한 k-bit 비밀키(secret key)**가 사용됨
- 평문 메시지가 n 비트보다 적을 경우, n 비트 블록을 만들기 위해 패딩(padding) 필요
- 평문이 n 비트보다 클 경우, 평문을 n 비트 블록(들)로 분할, 마지막 블록에는 필요 시 패딩 추가

- **(Question)** 8-bit 문자 100개로 된 메시지를 64-bit 블록 암호화할 때, 필요한 패딩 비트 수는?

  $|M| + |Pad| \equiv 0 \bmod 64$
  $800 + |Pad| \equiv 0 \bmod 64$
  $|Pad| \equiv -800 \bmod 64$
  $|Pad| \equiv -(64 \times 13) + 32 \bmod 64$
  $|Pad| \equiv 32 \bmod 64$
  → 패딩 비트 수는 32비트
  → 최초 800비트 평문은 64-bit 블록 13개로 분할되며, 암호화 알고리즘은 총 13회 적용됨

블록 암호 기술:
Substitution vs. transposition cipher

(Question) 64-bit block cipher로부터 생성된 어떤 ciphertext 내 비트 1의 개수가 10개라고 할 때, 이 ciphertext에 대응하는 plaintext를 알아내려면 총 몇 개의 서로 다른 plaintext를 검사해야 하는가?

- 64-bit block cipher가 **substitution cipher**인 경우, 최초 평문 내 비트 1의 개수에 대한 정보가 없으므로, 모든 가능한 64-bit 평문 블록들(2^{64}개)에 대한 시도 필요

2^{64} → 초당 십억개 block들을 처리하더라도 수백년 소요됨

- 64-bit block cipher가 **transposition cipher**인 경우, 최초 평문 내 비트 1의 개수 역시 10개일 것이므로, 시도해야 할 서로 다른 64-bit block들의 개수는

$\frac{64!}{10!\times 54!} \approx 151.5 \times 10^9$ → 초당 십억개 block들을 처리한다면 3분 미만 소요

Summary

- 전치 기반 블록 암호는 전수조사 (exhaustive search) 공격에 취약할 수 있음
- 현대 블록 암호 기술은 대치 기반 암호 방식으로 설계될 필요 있음

블록 암호 기술:
Full-size key cipher & full-size key length

Full-size key cipher

- 평문과 암호문 간 모든 가능한 매핑 중 하나를 선택하기에 충분한 크기의 키를 사용하는 cipher

Full-size key transposition block cipher

- 3-bit block을 전치 방식으로 암호화하는 방법의 총 수는 6(3개 원소들의 순열의 수 3!=6)이므로, full-size key cipher를 위한 key의 크기는 $\lceil \log_2 6 \rceil = 3\ bits$

Full-size key substitution block cipher

- 3-bit block을 대치 방식으로 암호화하는 방법의 총 수는 $2^3! = 8! = 40320$이므로, full-size key cipher를 위한 key의 크기는 $\lceil \log_2 8! \rceil = 16\ bits$

plaintext	000	001	010	011	100	101	110	111
	0	1	2	3	4	5	6	7
ciphertext	010	100	000	111	001	110	011	101
	2	4	0	7	1	6	3	5

Summary

- Full-size key n-bit 전치 혹은 대치 cipher는 permutation으로 모델링 가능
- Transposition cipher의 경우 full-size 키의 크기는 $\lceil \log_2 n! \rceil$
- Substitution cipher의 경우 full-size 키의 크기는 $\lceil \log_2 (2^n)! \rceil$

블록 암호 기술:
Full-size vs. partial-size key cipher

Full-size vs partial-size key cipher

- DES는 64-bit block cipher임
- DES가 full-size key를 사용했다면 키의 길이는 $\lceil \log_2(2^{64})! \rceil \approx 2^{70} bits$였을 것임
- 그러나 실제 DES의 키는 단지 56 비트임
- 즉 DES는 $2^{(2^{70})}$개의 가능한 매핑 중 2^{56}개 매핑만 사용하는 것임

Keyless cipher

- Keyless cipher는 fixed-key cipher임
- Keyless cipher는 keyed cipher의 구성 요소로 활용됨
- 전치 cipher의 경우, 하나의 미리 정해진 위치이동규칙이 사용됨 → P-box
- 대치 cipher의 경우, 하나의 미리 정해진 매핑이 사용됨 → S-box

블록 암호 기술:
Components

Components of a modern block cipher

- 현시대 블록 cipher들은 일반적으로 partial-key 기반 substitution 방식으로 동작
- 확산과 혼돈 특성 제공 위해, transposition unit(P-box)들, substitution unit(S-box)들, 그리고 다른 unit들의 결합으로 구성됨

P-box

- Straight P-box
- Compression P-box
- Expansion P-box

S-box

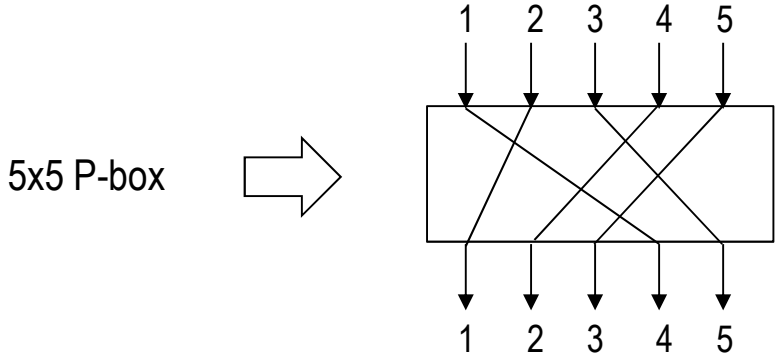
Other units

- XOR
- Circular shift
- Swap
- Split & combine

블록 암호 기술:
Straight P-box

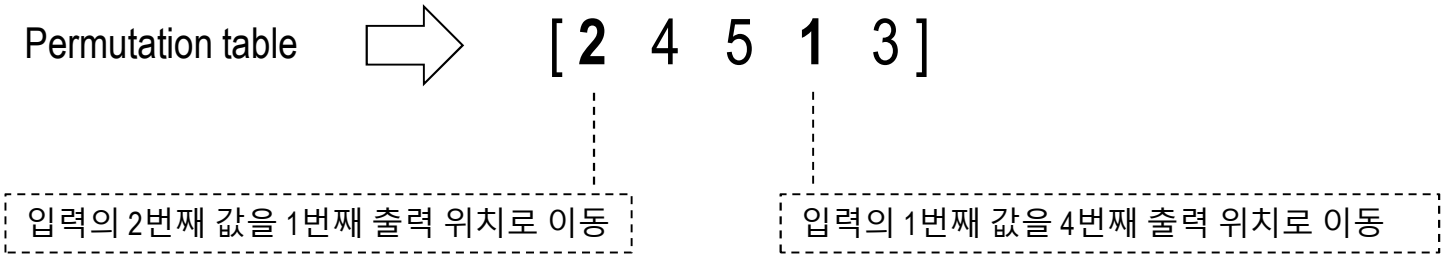
Straight P-box

- n개 각 입력을 n개 각 출력으로 위치 이동
- n!개 가능한 매핑 중 하나를 결정하기 위해 key를 사용할 수 있으나, 대개 keyless로 동작 (즉, 미리 정해진 하나의 매핑이 사용됨)



Permutation table for P-box

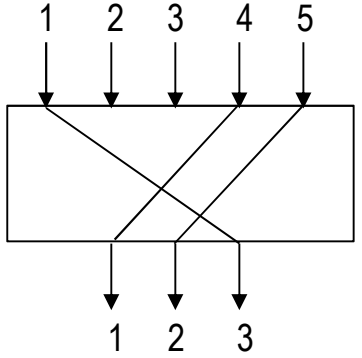
- P-box의 입출력 위치 이동 규칙 표현



블록 암호 기술:
Compression P-box, Expansion P-box

Compression P-box

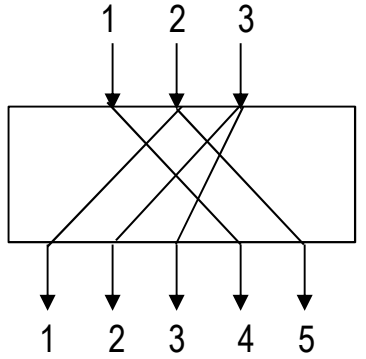
- n개 각 입력을 m개 각 출력으로 위치 이동 (n>m)
- 입력 비트 일부는 출력으로 나가지 않음
- 보통 keyless로 동작
- 비트 위치 변경 및 비트 수 축소 시 사용



5x3 P-box [4 5 1]

Expansion P-box

- n개 각 입력을 m개 각 출력으로 위치 이동 (n<m)
- m-n개 항목들이 반복 출력됨
- 보통 keyless로 동작
- 비트 위치 변경 및 비트 수 확장 시 사용



3x5 P-box [2 3 3 1 2]

블록 암호 기술:
Invertibility of P-box

Invertibility of P-box

- Straight P-box는 가역적, 축소 및 확장 P-box는 비가역적
- Straight P-box는 암호화 모듈에, 그것의 inverse는 복호화 모듈에 사용할 수 있음

Permutation table의 inverse 구하기

최초 permutation table	Contents	**6**	**3**	**4**	**5**	**2**	**1**

Index 추가	Contents	6	3	4	5	2	1
	Index	1	2	3	4	5	6

Contents와 Index 교환	Contents	1	2	3	4	5	6
	Index	6	3	4	5	2	1

Index 기준으로 contents 정렬	Contents	6	5	2	3	4	1
	Index	1	2	3	4	5	6

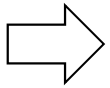
Inverted table	Contents	**6**	**5**	**2**	**3**	**4**	**1**

블록 암호 기술:
S-box

S-box

- 소형 대치 기반 cipher로 볼 수 있음
- 입력과 출력의 크기가 다를 수 있음
- Keyed 혹은 keyless일 수 있으나, 보통 keyless임 → 즉 입출력 매핑이 미리 정해져 있음
- 가역적 S-box가 되기 위해서는 입출력 비트수가 같아야 함

3x2 S-box



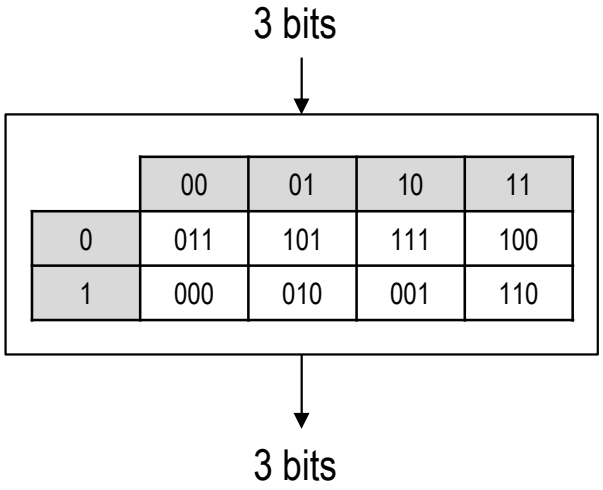
leftmost bits

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

rightmost bits

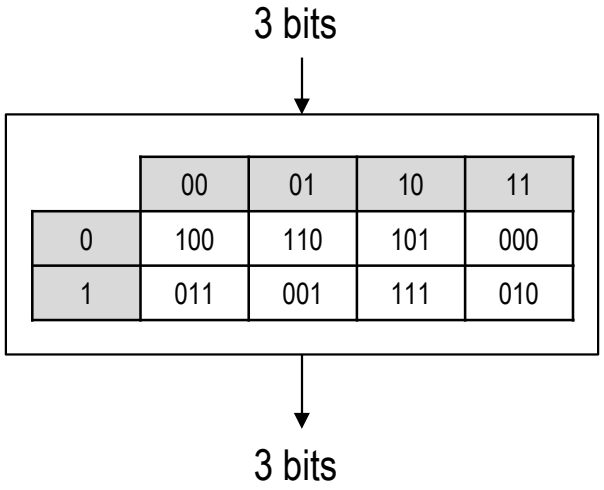
예) 입력 비트열 010 → 출력 비트열 01
예) 입력 비트열 101 → 출력 비트열 00

블록 암호 기술:
Invertible S-box



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

암호화 모듈에 사용



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

복호화 모듈에 사용

Product Cipher

Reference: https://www.britannica.com/topic/product-cipher

- Product cipher
  - Claude Shannon에 의해 소개됨
  - 평문에 어떤 변형(예:대치)을 가한 결과에 또다른 변형(예:전치)을 가하는 방식으로, 평문에 2회 이상 변형을 적용하여 암호화하는 방법으로 개별 변형만 적용한 경우보다 더 안전한 암호문 생성할 수 있음
  - 대치, 전치 및 다른 요소들을 결합한 cipher

Diffusion, Confusion

- Diffusion vs confusion
  - 안전한 암호 알고리즘이 가져야 할 두 가지 특성
    - Claude Shannon에 의해 소개됨
  - Diffusion (확산)
    - 암호문과 평문 간 관련성을 숨김으로써 공격자가 암호문 통계분석을 통해 평문을 알아 내지 못하도록 함
    - 암호문 내 각 심볼(문자 혹은 비트)은 평문 일부 혹은 전체에 의존
    - 즉 평문 내 단일 심볼 변경은 암호문 일부 혹은 전체 심볼 변경 야기
  - Confusion (혼돈)
    - 암호문과 키 간 관련성을 숨김
      - 암호문으로부터 키를 알아내지 못하도록 함
    - 키 내 단일 비트 변경은 암호문 대부분 변경되도록

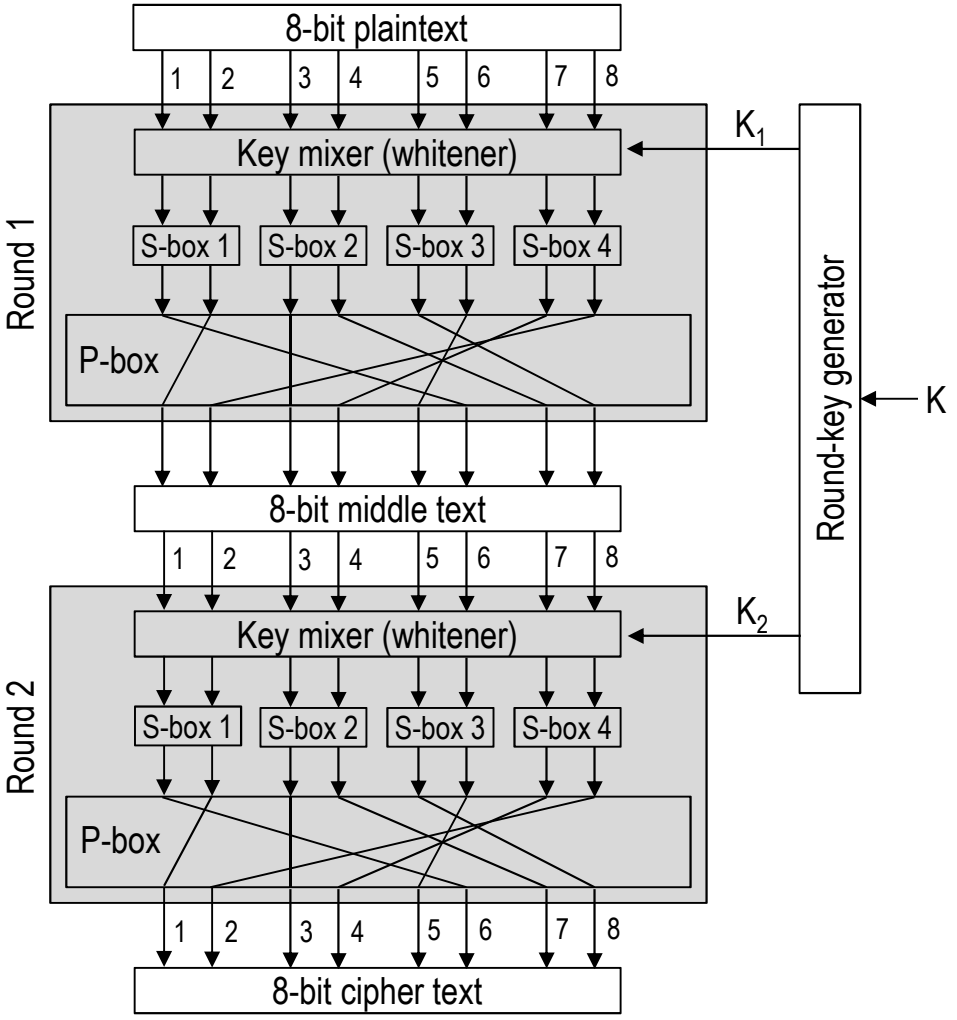
Product Cipher: Rounds

Rounds (라운드)

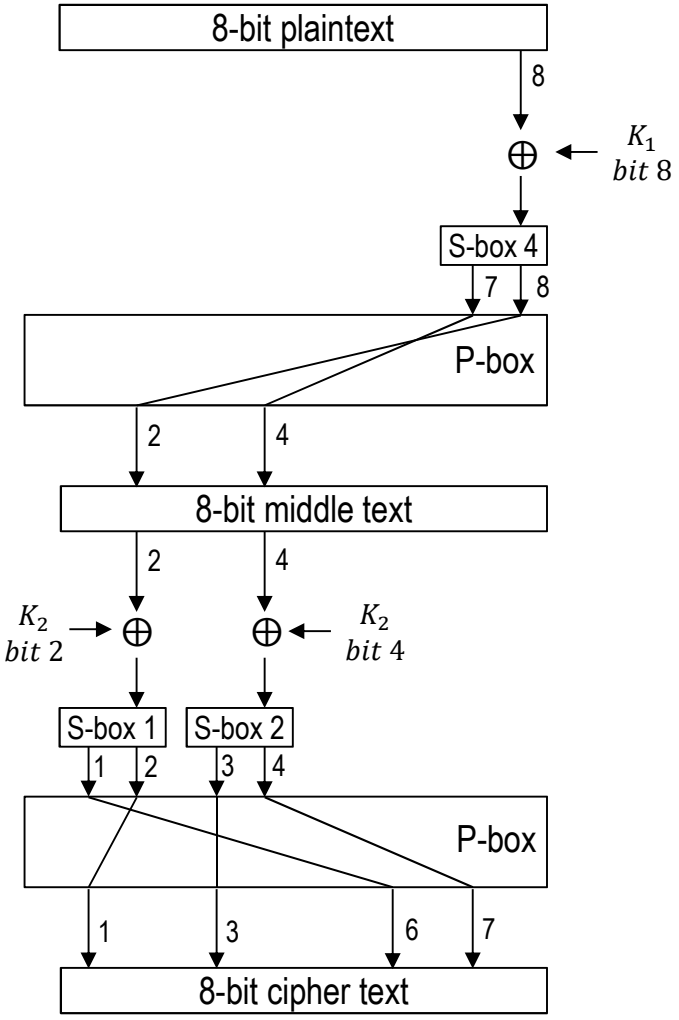
- Diffusion, confusion은 iterated product cipher를 통해 성취 가능
- 각 iteration은 하나의 round라고 불리며 S-box, P-box 및 다른 요소들의 결합으로 구성됨
- Block cipher는 key schedule 혹은 key generator를 사용하며 이는 cipher key로부터 각 라운드에서 사용될 서로 다른 키들을 생성
- N-round cipher의 경우 평문은 N번 암호화됨. 또한 N번 복호화됨
- Round 들 간에 만들어지는 text를 middle text라고 부름

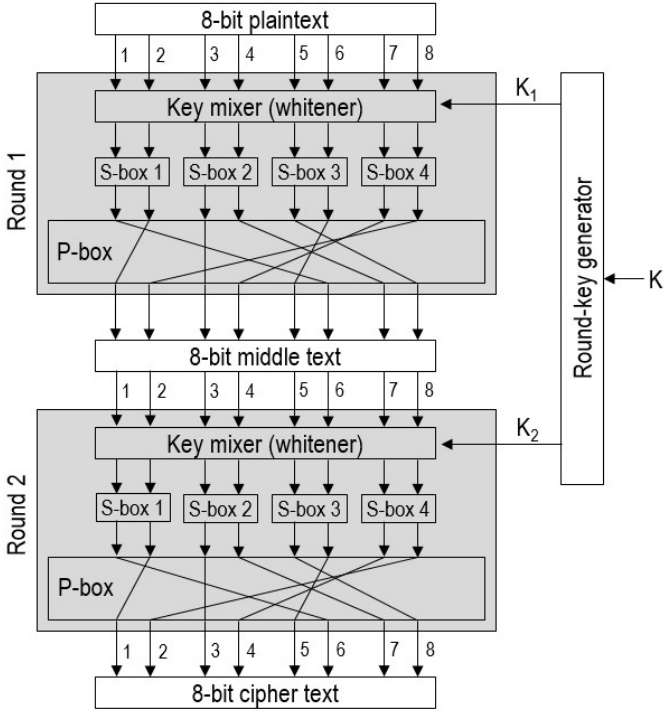
Practical ciphers

- 실용적 cipher들은 더 큰 데이터 블록, 더 많은 S-box, 더 많은 라운드를 사용하여 diffusion과 confusion을 향상시킴
- 더 많은 S-box를 사용하는 더 많은 라운드를 사용함으로써 암호문과 평문의 관계를 숨겨 diffusion 정도를 높임
- 더 많은 라운드는 더 많은 라운드 키 사용을 의미하므로 암호문과 키의 관계를 숨겨 confusion 정도를 높임



Product Cipher: Rounds





Diffusion 측면

- 평문 bit 8은 암호문 내 1,3,6,7 비트들에 영향을 미침
- 역으로 암호문 내 각 비트는 평문 내 여러 비트들의 영향 받음

Confusion 측면

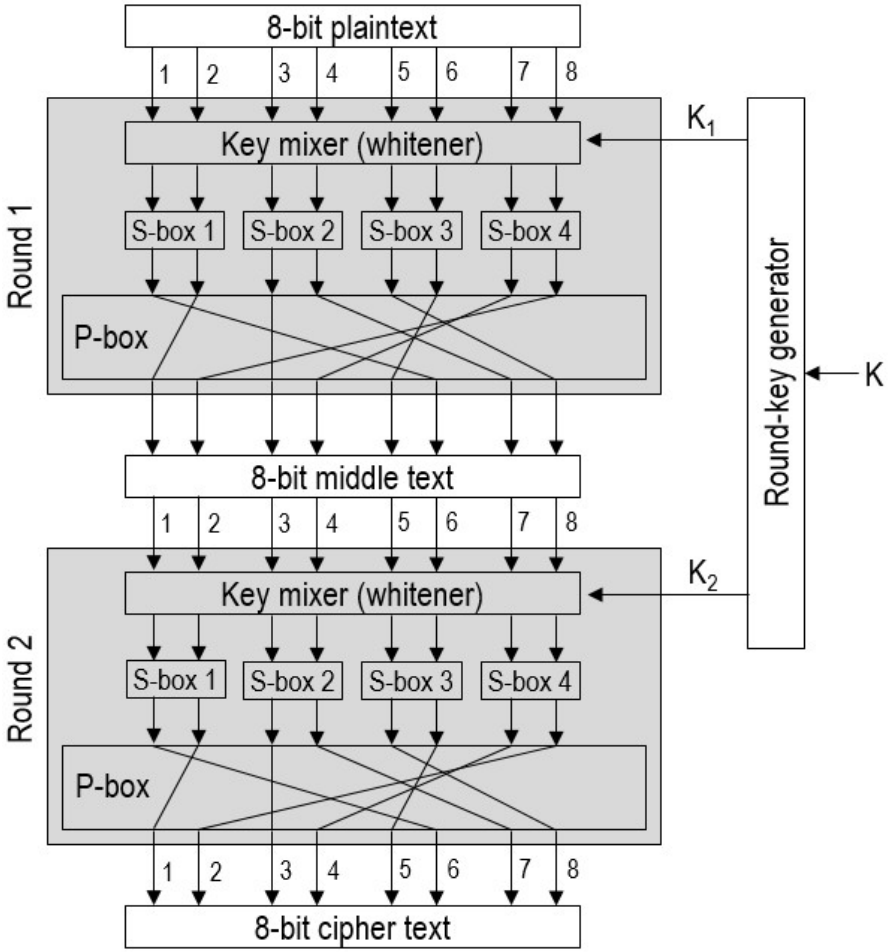
- 암호문 내 1,3,6,7 비트들이 키 내 3개 비트들의 영향 받음
- 역으로 각 라운드 키의 각 비트가 암호문 내 여러 비트에 영향 미침

Two Classes of Product Ciphers

- Two Classes of Product Ciphers
  - 현대 블록 cipher들은 product cipher들이며 Feistel cipher와 non-Feistel cipher의 두 부류로 나뉨
  - Feistel cipher
    - 가역적(invertible) 요소와 비가역적(non-invertible) 요소 모두 사용
    - 예) block cipher DES
  - Non-Feistel cipher
    - 가역적 요소만 사용
    - 예) block cipher AES

Non-Feistel Cipher

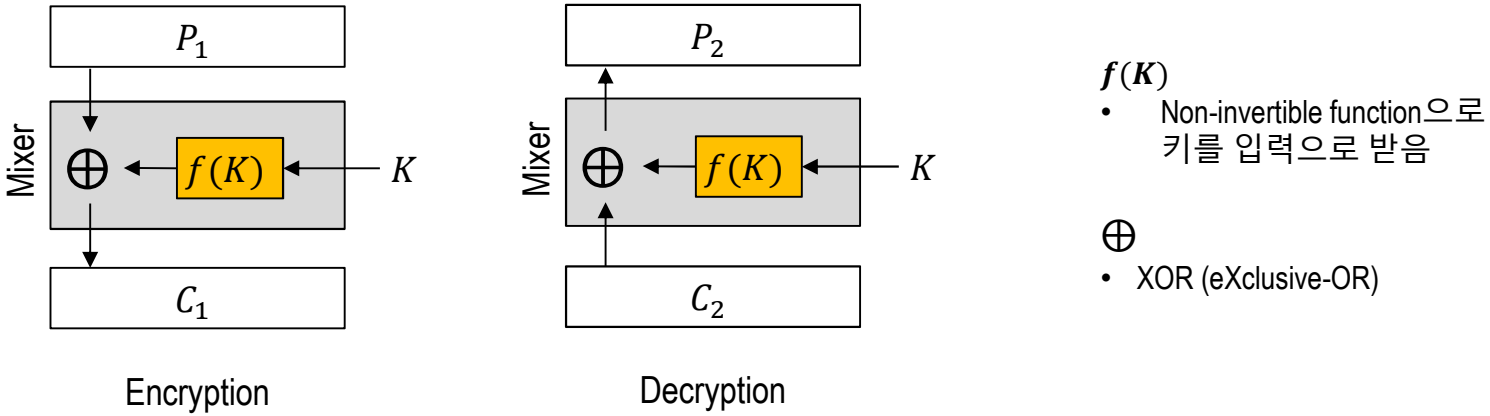
- Non-Feistel cipher
  - 가역적 요소만 사용
  - 비가역적 요소 사용 불가
    - 축소 P-box
    - 확장 P-box



Non-Feistel cipher의 예

- Key whitener는 두 입력에 대한 XOR (self-invertible)
- 가역적 2x2 S-box
- 가역적 straight P-box
- 즉 각 라운드의 모든 구성 요소들이 가역적임
- 암호화에서 K_1 , K_2를 적용하므로 복호화에서는 K_2 , K_1 순으로 적용 필요

Feistel cipher: First Design



$\boldsymbol{f(K)}$
- Non-invertible function으로 키를 입력으로 받음

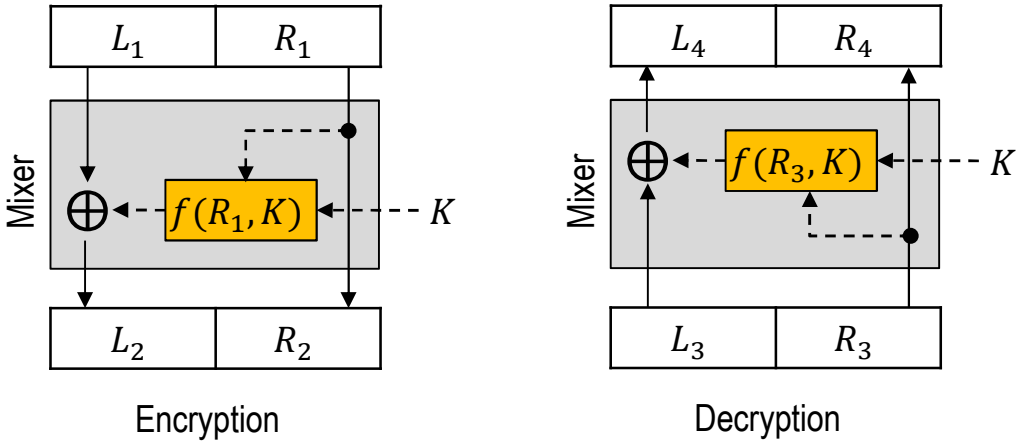
$\oplus$
- XOR (eXclusive-OR)

$\boldsymbol{f(K)}$는 non-invertible이지만 암호화, 복호화 절차가 서로의 역(inverse)이 되는 이유
- $Encryption$: $C_1 = P_1 \oplus f(K)$
- 전송 중 오류 없음 가정: $C_1 = C_2$
- $Decryption$: $P_2 = C_2 \oplus f(K) = C_1 \oplus f(K) = P_1 \oplus f(K) \oplus f(K) = P_1 \oplus (000 \ldots 00) = P_1$

예제

평문, 암호문 4-bit, 키 3-bit 길이이고, $f(K)$는 키를 입력받아 키 비트열의 first bit, third bit의 두 비트값의 십진수를 제곱한 결과를 4비트로 출력한다고 할 때, 평문 P=0111, 키 K=101에 대해 암복호화 과정은 다음과 같다
- 암호화: $11_{(2)} = 3,\ 3^2 = 9 = 1001_{(2)},\ C = P \oplus f(K) = 0111 \oplus 1001 = 1110$
- 복호화: $C \oplus f(K) = 1110 \oplus 1001 = 0111$

Feistel cipher: Second Design



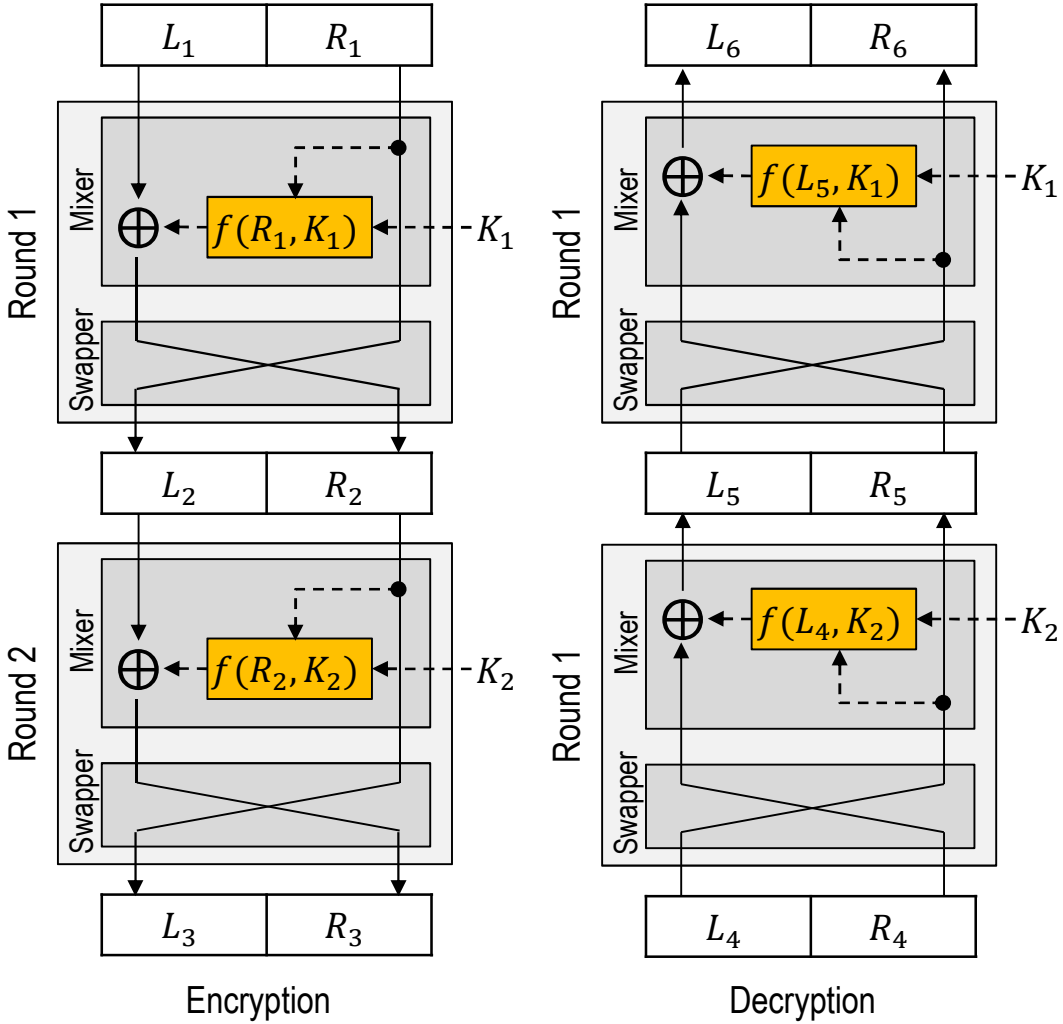
Second Design

- 함수의 복잡도 증대를 위해 키 기반 입력과 함께 키에 무관한 입력을 취하도록 변경
- 비가역 함수 f는 평문의 우측 절반 R_1과 키 K를 입력으로 받도록 변경
- 여전히 암복호화는 서로의 역이 됨
- 단점: 평문의 우측 절반이 변경되지 않아 공격자 접근 가능

암호화, 복호화 절차가 서로의 역(inverse)이 되는 이유

- $Encryption$: $L_2 = L_1 \oplus f(R_1, K)$
- 전송 중 오류 없음 가정: $L_3 = L_2,\ R_3 = R_2$
- $Decryption$: $L_4 = L_3 \oplus f(R_3, K) = L_2 \oplus f(R_2, K) = L_1 \oplus f(R_1, K) \oplus f(R_1, K) = L_1$

Feistel cipher: Final Design (2 rounds case)



Final Design

- Swapper 도입 → 좌측 절반과 우측 절반을 교환
- 암호화의 swapper 효과는 복호화의 swapper에 의해 상쇄됨
- 2개 라운드 키 K_1 , K_2가 암호화와 복호화에서 역순 적용됨

암호화, 복호화 절차가 서로의 역(inverse)이 되는 이유

- 전송 중 오류 없음 가정: $L_4 = L_3$, $R_4 = R_3$
- $L_5 = L_2$, $R_5 = R_2$인지 증명

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

$R_5 = L_4 = L_3 = R_2$

- $L_6 = L_1$, $R_6 = R_1$인지 증명

$L_6 = R_5 \oplus f(L_5, K_1) = L_4 \oplus f(L_2, K_1) = L_3 \oplus f(L_2, K_1) = R_2 \oplus f(L_2, K_1) = L_1 \oplus f(R_1, K_1) \oplus f(R_1, K_1) = L_1$

$R_6 = L_5 = L_2 = R_1$

References

- Behrouz A. Forouzan, Cryptography and Network Security, McGraw-Hill, 2008
- William Stallings, Cryptography and Network Security: Principles and Practice, Sixth Edition, Prentice Hall, 2014
- Christof Paar, Jan Pelzl, Understanding Cryptography: A Textbook for Students and Practitioners, Springer, 2010
- 김명환, 수리암호학개론, 2019
- 정민석, 암호수학, 경문사, 2017
- 최은미, 정수와 암호론, 북스힐, 2019
- 이민섭, 정수론과 암호론, 교우사, 2008