

고전 암호 - 비밀정보/비밀키

근대 암호 - 기계식 암호/1, 2차 대전

현대 암호 - 알고리즘 외 '키'라는 요소 추가 도입

BC 60 - Caesar (시저 암호)

AD 1942 - 애니그마/제이더 (1, 2차 세계대전)

AD 1976 - 공개키

AD 1977 - DES, RSA

AD 2001 - AES

역사 속

✓ 고 주몽 암호

- 일곱 고개 일곱 골짜기 돌 위의 소나무
- 일곱 모가 난 돌 위의 소나무
- 칼 → 신원 인증 ⇒ 부모 자식 인증

✓ 금 갑의 화

- 거문고 갑을 쏘아라 → “금 갑을 쏘면 2명이 죽고, 쏘지 않으면 1명이 죽는다”
- 돼지 두 마리가 싸움
- 노인이 연못 속에서 나와
- “뜯어보면 두 사람이 죽고, 뜯어보지 않으면 한 사람이 죽는다”

✓ 조선시대 연애편지

- 파자 식 한자 암호

籙 → 竹 = 十 八 昔 ⇒ 「28인 제백 대박술」에서 만났시다.

서적	대박	죽	연	여덟	제백
적	죽	이	십	팔	석

木子 → 李 이씨 米 → 一 + 八 → 八 + 八 (88)

左絲右絲 中言下心 糸言糸 사말 권.

走肖 → 趙 趙씨 一三口 牛 豆頭 不出 言午 허락한 허

午 → 午

✓ 조선왕조실록 암호

✓ 명도전

✓ 침수도 = 칼돈문자 - 고조선 문자 (고조선 화폐)

대칭 암호 방식 구성요소

- ✓ 평문
- ✓ 암호 알고리즘
- ✓ 비밀키
 - 키 값 == \rightarrow 대칭 키
 - 키 값 \neq \rightarrow 비대칭키
- ✓ 암호문
- ✓ 복호화 알고리즘

암호문 = 암호화 알고리즘(암호 키, 평문)

평문 = 복호화 알고리즘(암호 키, 암호문)

공격자 \rightarrow 암호문 확인 \rightarrow 암호 키, 평문 예측

평문 \rightarrow 암호문 반환에 사용되는 동작

- ✓ 치환(substitute) S-Box
- ✓ 전치(transposition) P-Box \Rightarrow 모든 암호 기술의 기초

사용한 키의 수

- ✓ single key - 대칭 키/비밀키 암호, 관용 암호
- ✓ two key - 비대칭 키/공개키 암호

평문이 처리되는 방법

- ✓ 블록 암호
- ✓ 스트림 암호

암호 해독

- ✓ 암호 알고리즘 본질에 대한 공격
- ✓ 평문-암호문 쌍에 대한 지식 등을 활용한 평문, 키 값 추론 공격

전사적 공격

- ✓ 모든 키에 대해 시도 함

DES - 56 bit → 1142 years

AES - 128 bit → 5.4×10^{24} years

절대 안전성

- ✓ 암호문을 아무리 많이 사용하더라도 해당 암호문의 평문을 알아낼 수 있는
- ✓ 충분한 정보를 포함하지 않는 경우
- ✓ One Time Pad - 한 번만 사용

계산 안정성

- ✓ 해독 비용이 암호화된 정보의 가치 초과
- ✓ 해독 시간이 유효기간 초과

S-Box - 치환 기법

- ✓ 평문의 문자(비트 열)를 다른 문자(비트 열), 숫자, 심볼 로 바꿈
- ✓ Caesar 가 대표적
- ✓ Shift Cipher
 - 암호화 알고리즘(암호 키, 평문) = $\text{평문} \pm \text{암호 키} \bmod \text{최대 이동 수}$
 - 복호화 알고리즘(암호 키, 암호문) = $\text{암호문} \mp \text{암호 키} \bmod \text{최대 이동 수}$
 - Caesar
 - ♦ 최대 이동 수 = 26
 - ♦ 가능한 키는 25개뿐 → 전사적 키 해독 기법에 취약
 - * 많은 키 사용 시, 전사적 공격은 비실용적
 - * 평문의 언어나 유형을 알지 못할 경우 공격은 더 어려움 → zip 등

✓ 단일 문자 치환 암호 기법

- 각 평문 문자를 임의의 문자로 치환하는 경우
- 총 가능한 키 수 = $26!$

A		B		C	...	Z
26	*	25	*	24	*	1

- 각 문자의 상대 빈도 계산
- 각 문자의 상대 빈도를 영문자의 상대 문자 빈도와 비교

+ 한글에 응용

ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅅ ㅇ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ
ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ

- ♦ 총 23개 키

✓ Playfair 암호

- 다중 문자 치환 암호 기법
 - ♦ 안전도 높임
 - ♦ 특정 키워드 기반 5*5 매트릭스 생성
 - ♦ 해당 키워드를 채우고
 - ♦ 나머지로 남은 공간 채움
 - ♦ 같은 문자 반복 → 사이에 'x' 삽입
 - ♦ 1, 2차 세계대전에서 사용

①	-	-	-	②	.	'1 4'	→	'2 3'
③	-	-	-	④	.	'3 2'	→	'4 1'

✓ Hill 암호

- 선형 방정식
- 행렬의 곱 → 키 알아낼 수 있음

✓ Vernam

- 평문과 동일한 키 길이
 - ♦ 암호문 = 평문 XOR 암호 키
 - ♦ 평문 = 암호문 XOR 암호 키
- 평문과 통계적 연관성이 없어야 함

✓ One Time Pad

- random 한 키 값, 평문과 같은 길이의 키 → perfect secrecy
(vernam 은 키 반복 사용 → 안정성↓)
- 문제점
 - ♦ 대용량 키를 다루기 어려움
 - ♦ 키 분배, 보호의 문제 발생

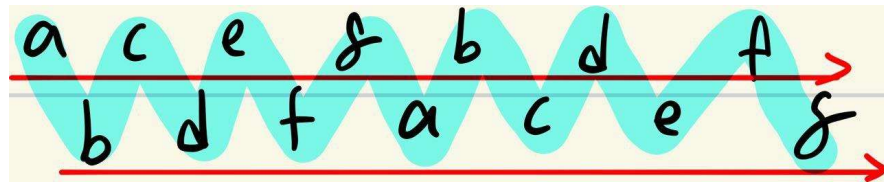
P-Box - 전치 기법

✓ Rail fence 기법

- 평문을 대각선으로 쓰고 열로 읽음
- abcdefgabcdefg

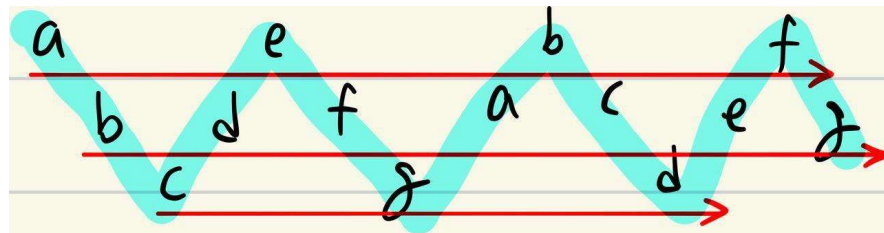
- ♦ 2 fence -

→ acegbdfdbfaceg



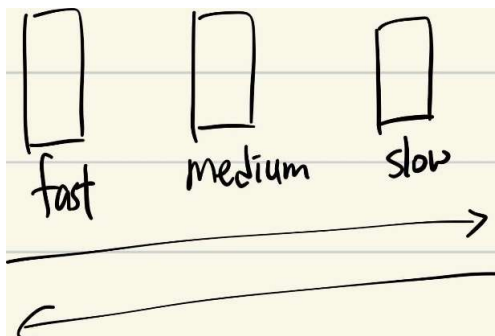
- ♦ 3 fence -

→ aebfdbfacegcbd



✓ Rotor Machine

- 2차 세계대전
- 가변적 치환 기법 사용
- 다수의 실린더로 구성되면 각 실린더는 한 문자에 대한 치환을 수행함
- 3개의 실린더 → 263 개의 다른 알파벳 치환 가능
n 개의 실린더 → n 단 애니그마 머신



보안 기술 범주

- ✓ 보안(security) - 정보를 보호하는 방법
- ✓ 첩보(intelligence) - 정보를 탐지하는 방법

- ✓ 평문 메시지 은닉 방법
 - 스테가노그래피
 - ♦ 메시지의 존재 자체를 은폐
 - ♦ 실제 메시지를 나타냄
 - 암호토크래피
 - ♦ 의미를 알지 못하도록 메시지 변형

암호 발전 과정

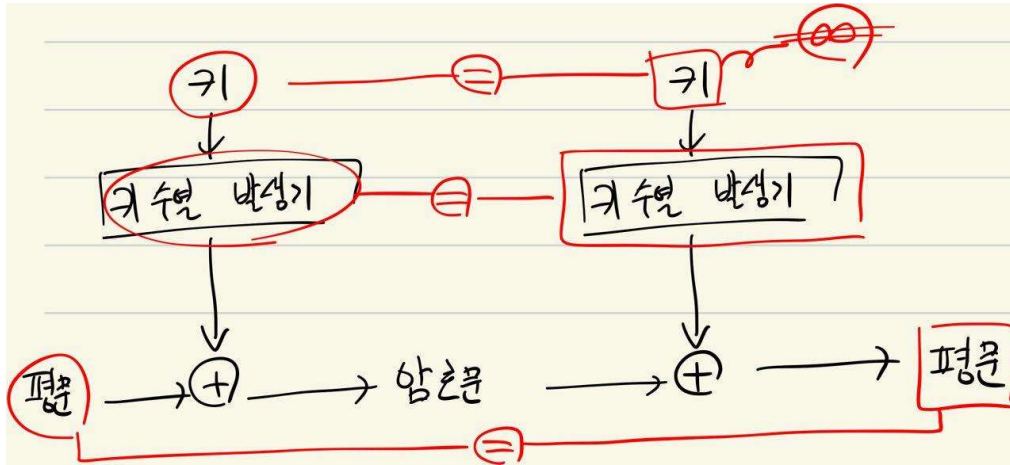
- ✓ BC 60 - 시저 암호
 - 영문
 - 키 길이: 1
- ✓ 16C - 비제네르 암호
 - 암호 구호
 - 키 길이: 가변적
- ✓ 20C - 버남 암호
 - One Time Pad
 - 키 길이: 무한
- ✓ 20C 후반 ~ 21C - 현대 암호
 - 스트림 암호
 - 키 길이: 128 ~ 256 → 짧고 일정

One Time Pad

- ✓ Perfect Cipher (완전 암호) - 유일함
- ✓ 암호학적으로 안전
- ✓ 구현이 어렵다
- ✓ OTP-Real Random (실난수열) 비밀 생성, 보관, 전환, 폐기 ... - 길이 무한 ⇒ 비용 무한
- ✓ 스트림 암호로 발전

스트림 암호

- ✓ 이진화 된 평문과 이진 키 수열의 배타적 논리합(XOR) 연산을 실행하여 암호문을 생성하는 알고리즘
- ✓ 이 때, 출력 키 수열에 대한 특성과 발생 방법이 안전도에 직접적인 영향을 미친다



✓ 특징

- 에러 확산 없음 → 무선 이동 통신에 적합 (OFB)
- 비도 수준에 대한 수학적 정량화가 가능
- 하드웨어 구현이 용이
- 통신 지연 없음
- 고성 통신 가능

✓ 설계 고려사항

- 키 수열 발생기의 암호학적 안전성
- 통신 채널 환경에 적합한 키 수열 동기 방식 성능 (통신 신뢰성)
- 암호화 속도 등에 대한 성능 분석
- 키가 충분히 길어야 함 → LFSR - 128-bit

1. 주기

- ♦ 출력 키 수열은 주기에 대한 최소 값이 보장되어야 한다
- ♦ 반복 주기가 길어야 함
- ♦ $P \gg 2^{128}$ 권장

2. 랜덤 특성

- ♦ 출력 키 수열은 좋은 랜덤 특성을 가져야 한다
- ♦ 키 스트림이 랜덤하게 나타나야 함
- ♦ Menezes / NIST Test

3. 선형 복잡도

- 출력 키 수열은 큰 선형 복잡도를 갖는다
- $LC \gg 2^{128}$ 권장

4. 상관 면역도

- 출력 키 수열은 높은 상관 면역도를 갖는다
- $CI \geq 1 \rightarrow$ 높을수록 좋다

5. 키 수열 사이클 수

- 출력 키 수열은 2개 이상의 키 수열 사이클에서 발생해야 한다
다른 키 \rightarrow 다른 키 수열
- $KC \geq 2$

✓ 장. 단점

▪ 장점

- 동일한 키를 사용하는 블록 암호만큼 안전함
- 블록암호에 비해 간단하고 빠른 처리 속도

▪ 단점

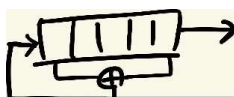
- 동일 키를 가지고 2개 이상 암호화하면, 해독이 간단해질 수도 있다
- *** 하드웨어가 고정, 제한적인 상태 \rightarrow 스트림 암호가 유리함 (무선통신에 적합)

✓ 유형

- 선형 귀환 이동 레지스터 방식 (LFSR, Linear Feedback Shift Register)
- 비선형 조합 함수 방식 (Nonlinear combining function)
- 비선형 필터 발생기 (Nonlinear filtered generator)
- 클럭 조절 수열 발생기 (Clock-controlled sequence generator)
- 병렬형 스트림 암호 (Parallel stream cipher)
- 워드기반 스트림 암호 (WBSC, Word Based Stream Cipher)

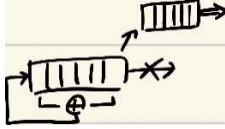
✓ 요소

- LFSR, 원시 다항식 (primitive polynomial)



- 랜덤 이진 수열을 발생시키는 가장 기본적인 암호 요소이나
레지스터 길이의 2배가 되는 출력을 알면 해독 가능

- 클럭 조절형 LFSR (Clock-controlled LFSR)
 - ♦ LFSR의 클럭 수를 조절하여 안전성을 높이는 DKAG 요소
- 비선형 조합 함수 (Nonlinear combining function)
 - ♦ 출력 수열의 비선형도를 높이기 위해 여러 개의 선형 LFSR을 조합하는 메모리형 조합 함수
- 비선형 필터 함수 (Nonlinear filtered function)



- ♦ 선형 LFSR의 각 state memory를 비선형적으로 조합하여 출력을 발생시키는 비선형 함수
- NFSR (Nonlinear Feedback Shift Register)
 - ♦ LFSR에 대응되는 비선형 귀환 이동 레지스터
- 키 수열 동기 방식
 - ♦ 연속 동기 방식, 초기 동기 방식, 절대 동기 방식 등
송. 수신 키 수열에 대하여 각각 bit-by-bit 일치시키는 동기 방식
- Zero-suppression algorithm
 - ♦ 출력 수열의 연속 '0'을 일정 비트 이하로 억제

✓ 동기식 스트림 암호

- 장점
 - ♦ 고속의 암호화 처리
 - ♦ 고난이도
 - ♦ 적은 오류 전파 율
 - ♦ 능동적 wiretapping으로부터 보호
- 단점
 - ♦ 낮은 확산 효과
 - ♦ 자기동기 능력의 결여
 - ♦ 키 수열 동기
 - ♦ Zero-suppression

✓ 자체 동기식 스트림 암호

- 장점
 - ♦ Self-synchronizing (키 수열 동기 = 난수 동기)
 - ♦ 수학적인 동작에 대한 예측이 어려움 (랜덤 성)

- 단점
 - ♦ 에러확산 없음 ... 생김
 - ♦ # 블록 암호의 경우, 1-bit 에러가 128-bit 블록 전체로 확산됨
 - ♦ 입력 값의 노출
 - 암호문이 LFSR의 입력 값으로 사용되므로
 - 공격자들에게 LFSR에 대한 입력 값들이 노출됨
 - ♦ 비트열에 대한 수학적 성질의 결정이 곤란

✓ LFSR - 순서 이진비트 생성을 위한 기계

- 장점
 - ♦ 하드웨어를 이용한 구현이 용이하다
 - ♦ 주기가 길고 또한 여러 유형의 우수한 통계적 특성을 지니고 있기 때문에
 - ♦ 키 수열 생성기에 폭 넓게 이용됨

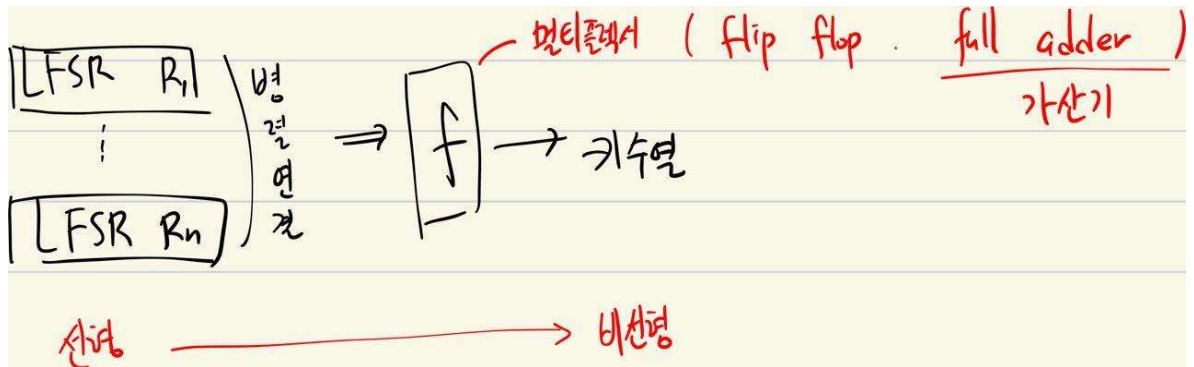
- 단점
 - ♦ 키 수열은 그 예측이 매우 용이하다 → 비선형으로 만들 필요가 있다
- n-bit → n단 LFSR → all zero를 제외한 모든 경우의 수를 만들어 냄
 - ♦ *Feedback tab(귀환 탭) 구성이 중요하다

- 응용
 - ♦ 19단 + 22단 + 23단 → 264 → A5 암호
 - ♦ 암호 프로토콜 - nonce라는 랜덤 수
 - ♦ 보안장비 - 최소 256/512 단 이상 사용
 - ♦ CATV 스크램블러 - LFSR-7 또는 LFSR-32
 - ♦ USB 소프트웨어 락 - 소프트웨어 복제 방지 인증용 USB (LFSR-32 수준)
 - ♦ C언어 rand() 함수 → DES(random key)
 - ♦ 2자리 로또 발생기
 - ♦ 4자리 랜덤 수열 생성기/6자리 랜덤 수열 생성기 (OTP)
 - ♦ 가위바위보 게임
 - ♦ 윷놀이 발생기
 - ♦ 서양화(포커) - 0 ~ 53
 - ♦ 동양 화투 게임(초기값 ~ 다음 카드 발생) - 0 ~ 47
 - ♦ Gold 수열 발생기 - 랜덤 수열 (5단 LFSR + 5단 LFSR → 10단 - 210 스크램블)

- 랜덤성 시험

- Frequency Test - 0 1 $\rightarrow n/2$ --- 0
- Serial Test - 00 01 10 11 $\rightarrow n/4$ --- 0
- Poker Test(block) - 00 01 10 11 $\rightarrow \text{block}/4$ --- X
- Generalized Serial Test - 000 001 010 011 100 101 110 111 $\rightarrow n/8$ --- 0

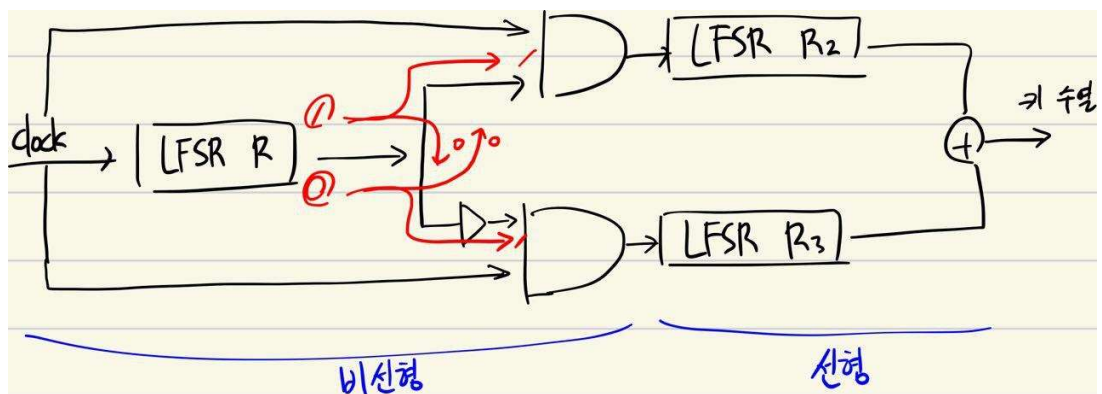
- 비선형 결합 생성기



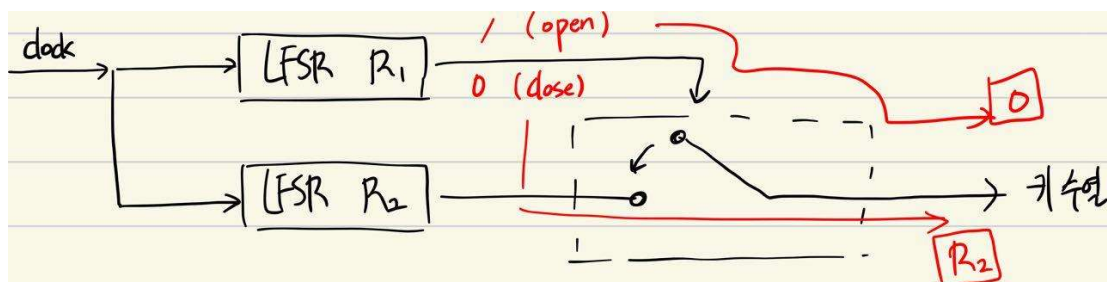
- Clock-Controlled 생성기

- LFSR에 불규칙적인 클럭 신호를 가하는 것

- Alternating Step Generator



- Shrinking Generator



- Linear Congruential Generator
 - ♦ 빠름
 - ♦ 예측 가능
 - ♦ 좋은 성능

의사 난수

✓ 난수

- 특정한 배열 순서나 규칙을 가지지 않는 연속적인 임의의 수
- 재생 불가

✓ 의사 난수

- 컴퓨터에 의해 만들어진 난수
- 아주 긴 주기를 가진 숫자 열
- 재생 가능

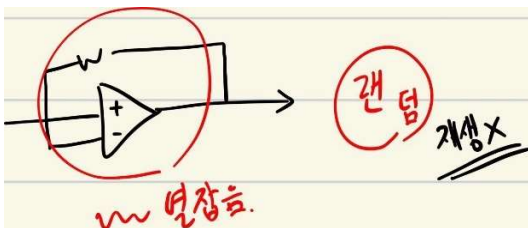
✓ 임의성

- 균일 분포 - 수열의 비트 분포가 동일, 0과 1의 출현 빈도가 거의 동일
- 독립성 - 수열의 어느 부분도 다른 부분에 의해 예측될 수 없음

✓ 비예측성

- 수열의 잇따른 다음 수의 순서에 대해 예측이 불가능해야 함

✓ TRNG



- 실제로 랜덤 한 소스를 입력으로 사용
- 키보드 입력 타이밍 패턴 및 마우스 움직임
- 디스크의 전기적 활동, 시스템 클럭의 순간 값

✓ PRNG

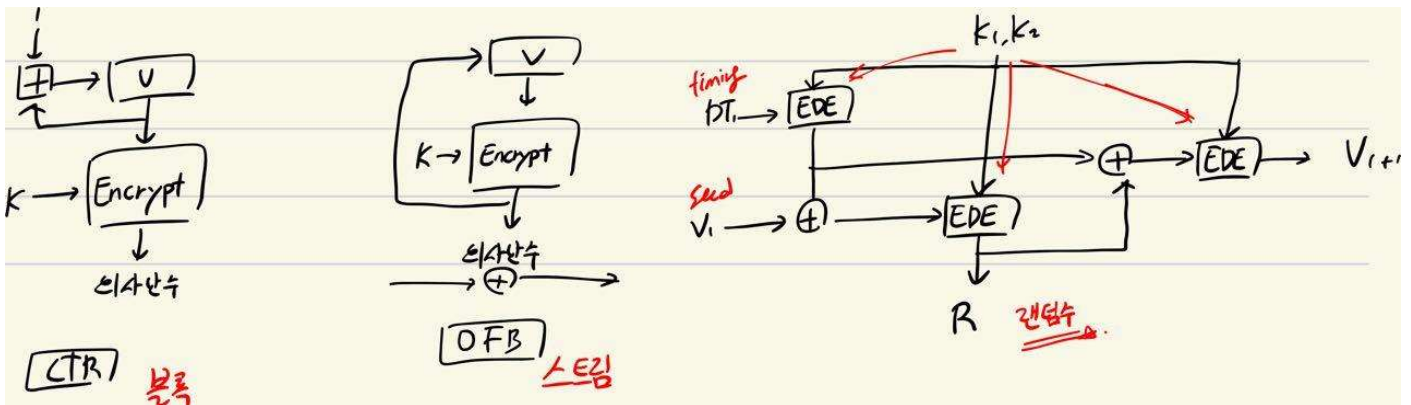
- 고정 값 seed를 입력 받아 결정적 알고리즘을 사용하여 출력 비트 열 생성
- 제한이 없는 비트 열 생성하는데 사용
 - ♦ 알고리즘과 seed를 알고 있는 공격자는 비트 열 재생성 가능

- 요구사항

- seed를 알지 못하는 공격자는 의사 난수를 결정할 수 없어야 함
- 임의성 - 생성된 비트 스트림이 결정적일지라도 랜덤하게 보여야 함
- 균일성 - 비트 열 생성에 0과 1이 거의 동일하게 존재
- 확장성 - 무작위로 추출된 어떤 비트열도 랜덤해야 함
- 일관성 - 생성기동작은 초기값 전반에 대해 일관되어야 함

- 구축 방법

- 대칭 블록 암호를 PRNG 메커니즘의 핵심으로 사용
- 어떤 블록 평문에 대해서도 대칭 블록 암호는 확실히 랜덤 한 출력 블록 생산
- 암호문에는 평문을 추론하도록 하는 정보를 제공하는 패턴이나 규칙이 없음



- ✓ PRF

- 고정된 길이의 의사 난수 비트 열 생성에 사용

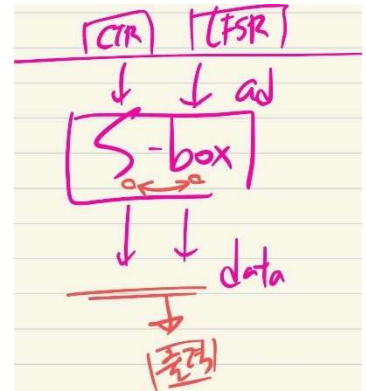
RC4

- ✓ RSA Security에서 Ron Rivest가 1987년에 고안한 스트림 암호 방식

- 바이트 단위로 작동되고 가변적 키 길이를 가지는 스트림 암호방식
- 랜덤 순열의 사용을 기반으로 함
- 웹 브라우저와 서버 사이의 통신 표준으로 규정된 SSL/TLS에서 사용
- 무선 LAN 표준인 WEP와 WPA 프로토콜에서 사용

✓ S의 초기화

- S 항목들을 오름차순으로 0부터 255까지의 값이 되도록 설정
 - ♦ $S[1] = 0, S[2] = 1 \dots S[255] = 256$
- 임시 벡터 T 생성
 - ♦ 키 값의 길이가 256바이트일 때
 - K를 그대로 T로 전달
 - ♦ 키 값의 길이가 256바이트 미만일 때
 - 키의 길이 Keylen만큼 T에 전달하고, T가 채워질 때까지 K를 반복하여 채움



✓ S의 초기 순열 교환 - swap

- $S[0] \sim S[255]$ 까지 수행
- 각 $S[i]$ 에 대하여 $T[i]$ 구조를 적용하여 $S[j]$ 를 생성
- $S[i]$ 와 $S[j]$ 를 교환

✓ S에서 일어나는 동작은 오직 교환이며, 그 결과는 순열

✓ 스트림 생성

- S벡터 초기화 후 입력키는 사용되지 않음
- 스트림 생성은 $S[0]$ 부터 $S[255]$ 까지 포함하여 수행
 - ♦ 각 $S[i]$ 는 S의 현재상태에 있는 내용에 따라 S의 다른 바이트와 교환
 - ♦ $S[255]$ 에 도달한 뒤에는 계속하여 $S[0]$ 부터 다시 시작
- 암호화 - k값과 평문의 다음 바이트를 XOR 연산하여 암호화 수행
- 복호화 - k값과 암호문의 다음 바이트를 XOR 연산하여 복호화 수행

진 난수 생성기

✓ 엔트로피 소스

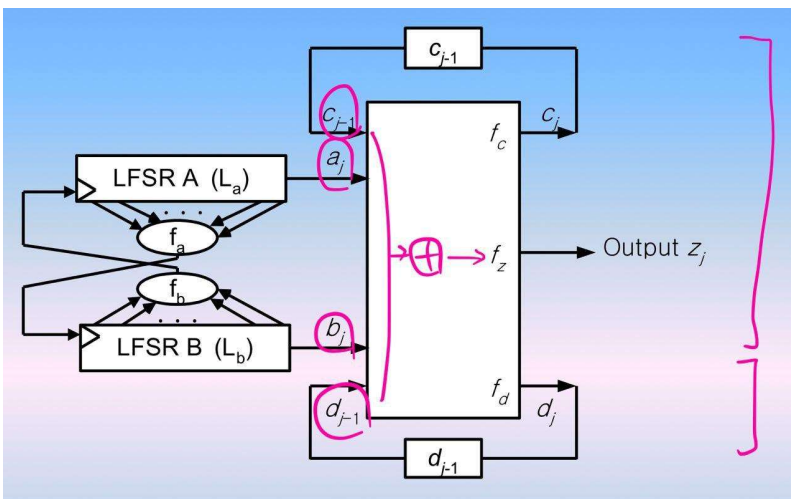
- 진 난수 생성기는 임의성을 만들기 위해 비 결정적 소스 사용
 - ♦ 예측 불가능한 자연계의 처리 과정들을 측정하는 방법 사용
 - 전리 방사선의 펄스 탐지기, 가스 방전관, 누전 축전기
 - ♦ 인텔[JUN99]
 - 사용되지 않는 저항기들 사이에서 측정된 전압을 증폭시켜 열잡음을 샘플링 하는 상업화된 칩 개발

- 임의성의 소스 후보[RFC 4086]
 - ♦ 음향/영상입력
 - 실세계의 아날로그 소스를 디지털화 하는 입력들을 가지고 작동
 - ♦ 디스크 드라이브
 - 환경에 따라 회전 속도가 랜덤하게 변동되는 것을 측정

✓ 편향

- Deskewing 알고리즘
 - ♦ 0 또는 1 중 한 쪽으로 편향된 출력물이 생성될 가능성 존재
 - ♦ 이를 감소시키거나 없애기 위한 기법
 - 해시함수를 이용 (MD5, SHA-1 등)
 - ♦ 운영체제는 난수를 생성하는 내장형 메커니즘 제공
 - ♦ 리눅스는 4개의 엔트로피 소스 (키보드, 마우스, 디스크 I/O 연산, 특정 인터럽트)를 사용하여 버퍼 풀어 저장, 이 중 일정 개수를 읽어서 SHA-1 해시 함수 통과

핑퐁 (Ping-Pong Family)



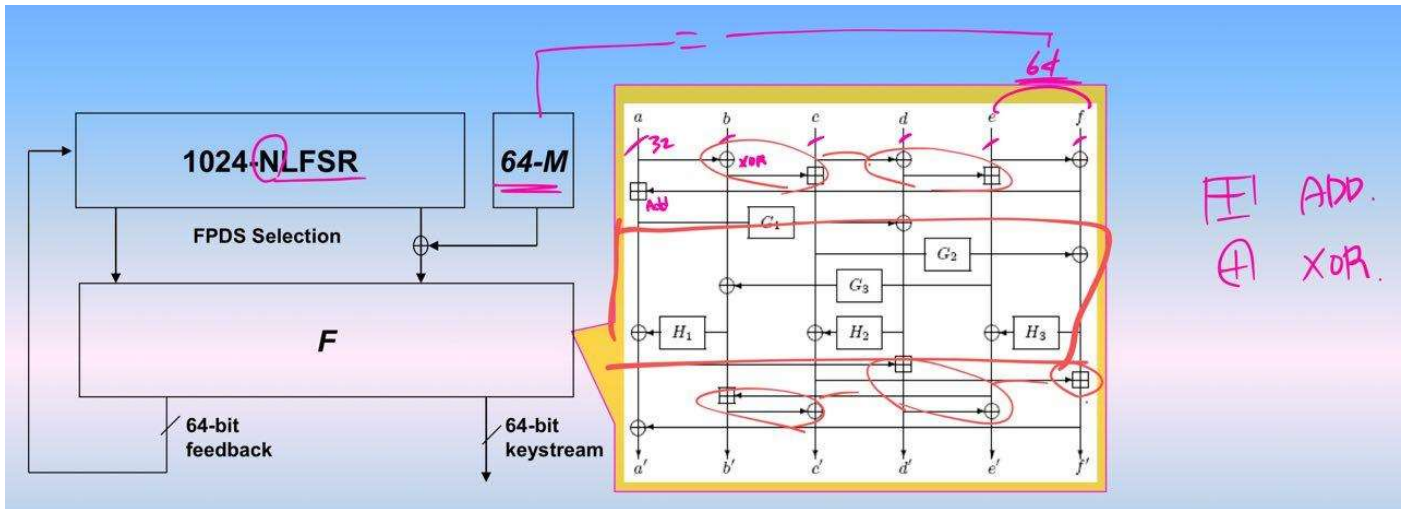
No. of clocks f_a

L_{a2}	L_{a5}	f_a
0	0	1
0	1	2
1	0	3
1	1	4

Handwritten notes: $(L_b)^{2/2}$ and f_a 리 반복

Dragon

- ✓ 고속 워드 기반 스트림 암호 개발
 - Word 기반 스트림 암호화 방식 - ICISC'2004 발표



DES

- ✓ 핵심
 - 블록 암호는 평문 블록 전체를 가지고 동일한 길이의 암호문 블록을 생성하는 암호화, 복호화 방식
 - 많은 블록 암호가 Feistel 구조를 띄고 있음
 - ♦ 동일한 라운드 수로 구성되어 동작
 - ♦ 각 라운드는 데이터의 절반이 치환으로 수행 이후 데이터의 두 개의 반을 교환하는 순열 수행
 - ♦ 원본의 키는 확장되어 각 라운드마다 다르게 사용
 - DES는 최근까지 가장 널리 사용되는 암호 알고리즘
 - ♦ 수학적(수식)으로 해독 불가
 - ♦ 컴퓨터로는 해독됨
 - 차등 암호분석과 선형 암호분석은 암호 분석의 중요한 두 가지 방법
 - ♦ 차등 암호분석 → 2^{49}
 - ♦ 선형 암호분석 → 2^{43}
 - ♦ DES는 두 가지 공격에 매우 강함을 보임

✓ 블록 암호 vs 스트림 암호

■ 블록 암호

- ◆ 암호화 적용
 - 텍스트/음성/영상/동영상을 블록 크기로 나누어 암호화
- ◆ 암호화 속도
 - 소프트웨어 적용 시에는 간편하고, 저속
 - 하드웨어 적용 시에는 고속
- ◆ 암호화 안전성
 - 상대적으로 안전성이 높다
key 길이 > 80-bit 이상
→ 통상적으로 128-bit
- ◆ 채널 에러
 - 채널 에러가 블록 전체로 확산되어 불리하며
OFB와 같은 스트림 방식으로 적용
- ◆ 비트 스트리밍 서비스
 - 블록을 비트로 전환시킨 후에 적용해야 함
OFB mode

■ 스트림 암호

- ◆ 암호화 적용
 - 텍스트/음성/영상/동영상을 비트 스트림으로 나누어 암호화
- ◆ 암호화 속도
 - 소프트웨어 또는 하드웨어에서 간편하고, 고속
- ◆ 암호화 안전성
 - 상대적으로 안전성이 낮다
key 길이 > 80-bit 이상, 추가로 init, Value 필요
→ 통상적으로 128-bit
- ◆ 채널 에러
 - 채널 에러에 대한 확산이 없어 채널 오류가 많은 회선에 유리함
무선통신 구간 등
- ◆ 비트 스트리밍 서비스
 - 간편하고 쉽게 적용된다
기본적으로 비트 처리됨

✓ 원리

- 스트림 암호
 - ♦ 한 번에 1-bit 혹은 1-byte의 디지털 데이터 스트림을 암호화
 - ♦ 키 스트림은 평문 비트 스트림만큼의 길이를 가짐
- 블록 암호
 - ♦ 평문 블록 전체를 가지고 같은 크기의 암호문 블록 생성
 - ♦ 전형적으로 64-bit 또는 128-bit 사용
- Feistel 암호 구조의 동기
 - ♦ 블록 암호는 n -bit 암호문 블록을 생성하기 위해 n -bit 평문 블록을 이용하여 연산
 - ♦ $2n$ 가지의 서로 다른 평문 블록 존재
- 두 개 이상의 기본 암호 연속적 수행(치환, 순열 번갈아 수행) - S-P 생성 원리
 - ♦ 치환 (S-Box)
 - 평문의 각 원소 또는 원소의 그룹을 다른 원소에 사상 → 바꿔 치기
 - ♦ 순열 (P-Box)
 - 평문 원소의 순서는 순열의 순서대로 재배치
 - ♦ 확산과 혼돈
 - Claude Shannon 소개 (SHAN49)
 - Shannon - 매우 이상적인 암호는 암호문에 대한 모든 통계적 정보가 사용된 키와 독립적이어야 한다
 - 통계적 분석에 기초한 암호 해독 방지
 - 확산
 - 평문의 통계적 구조가 암호문에 광범위하게 분산
 - 평문과 암호문 관계 복잡
 - 각 평문 숫자가 다수의 암호문 숫자 값에 영향
 - 혼돈
 - 암호문의 통계적 구조와 암호 키 값 사이의 관계 복잡
 - 키를 이용한 암호문 생성 방법 복잡
 - 키 추론 어려움

▪ P-Box

- ♦ n-bit - n-bit 블록 순열
 - 입력 n2개가 순서를 바꾸어서 다른 번호로 출력됨
 - 입력 - 출력의 순서를 바꾸어서 diffusion 시키는 원리
- ♦ P-Box diffusion
 - 선형화 하면서 데이터를 확산시켜 암호해독이 어렵게 한다
 - 랜덤하게 섞으면 예측이 어렵다
 - 랜덤 → 확산이 잘된 것
 - 입 - 출력은 선형관계이지만 S-Box와 함께 복잡성을 증대시킨다
- ♦ n*n P-Box 생성 방법
 - n*n P-Box를 설계하기 위하여 먼저 1 ~ n (or 0 ~ n-1)까지의 숫자를
 - 랜덤하게 발생시킨다
 - 입 - 출력의 동일 여부 확인
 - 랜덤하게 발생시킨 숫자대로 순열을 분포한다

▪ S-Box

- ♦ n-bit - n-bit 블록 치환
 - n-bit 입력으로 n2개 값 중 하나 선택하고
 - 내부 치환에 의해 n2개 출력 값 중 하나 대응하여 n-bit 출력
 - 입력이 내부 비선형 변환 함수를 거쳐서 confusion 되는 원리
- ♦ S-Box confusion
 - 비선형화 시키면 해독이 어렵다
 - 랜덤하게 섞으면 예측이 어렵다
 - 입 - 출력의 어떤 관계를 예측할 수 없다
- ♦ n*n S-Box 생성 방법
 - n*n S-Box를 설계하기 위해 먼저 1~2n (or 0 ~ 2n-1)까지의 숫자를
 - 랜덤하게 발생시킨다
 - 입 - 출력의 동일 여부 확인
 - 랜덤하게 발생시킨 숫자대로 순열을 분포한다
 - 외부 입력 - 내부 십진수 변환, 내부 십진수 - 외부 출력 변환 실시하면
 - 전체적으로 외부 n*n S-Box 변환으로 보인다
 - S-Box 검증 기준 확인

♦ 사례

- 미국 DES - 1977년 FIPS Pub. 46
 - 6×4 S1~S8-Box 8개 = $(4 \times 4$ S-Box 4개씩 묶음) * 8개
- 미국 AES - 2001년 FIPS Pub. 197
 - 8×8 S-Box 1개 (암호화)/ 8×8 S-1-Box 1개 (복호화)
- 한국 SEED - 1999년 TTAS.K0 12.004/R1
 - 8×8 S-Box 4개
- 한국 ARIA - 2005년 KS X 1213
 - 8×8 S1-Box, S2-Box, S1-1-Box, S2-1-Box 4개

▪ Feistel 암호 구조

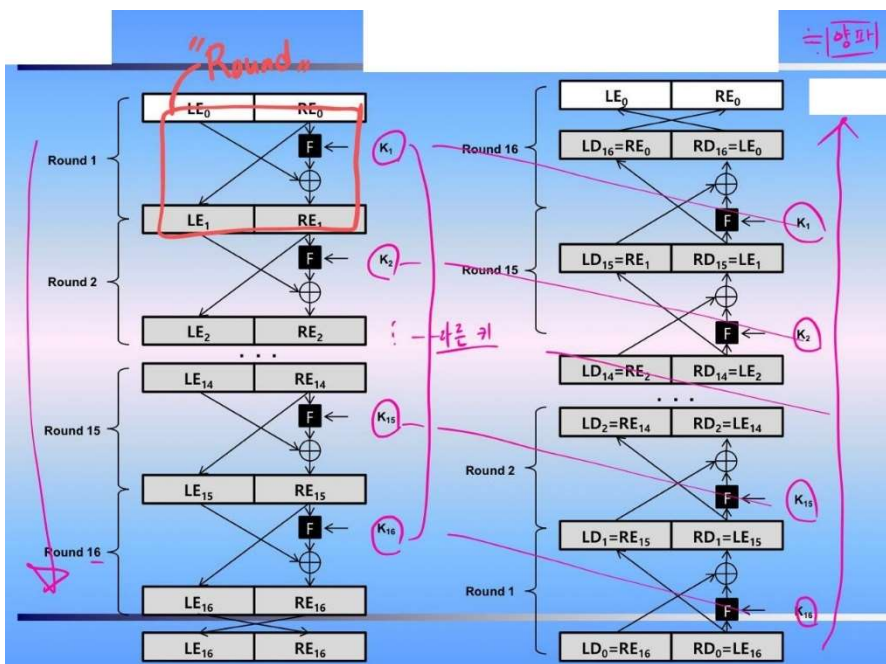
♦ 처리구조

- 길이 $2w$ 비트인 평문 블록(LR) 분할 처리
- K(main key)로부터 유도된 n 개의 키(K_i) 사용
- n 회의 동일한 반복 구조 실행

♦ 고전 Feistel 구조

♦ 하나의 반복 구조 → 16회 반복

- 오른쪽 반 R_0 에 반복 함수 F 적용
 - 반복 서브키 K_1 적용 ($K \neq K_i$)
- 왼쪽 반 L_0 와 XOR (치환 작용)
- 좌우 양쪽 결과를 교환 (순열 작용)



- ◆ 설계 고려사항
 - 빠른 소프트웨어 암호화, 복호화
 - 어플리케이션 또는 유틸리티 함수에 내재
 - 알고리즘의 실행속도가 중요
 - 분석의 용이성
 - 암호 해독의 취약성에 대한 알고리즘의 분석이 쉬움
 - 고도의 신뢰성과 보안 강도를 위한 개발 용이

- ◆ 설계 특성
 - 블록 크기(64-bit) → 128-bit
 - 큰 블록은 보안 강화하지만 암호/복호화 속도는 저하
 - 암호/복호화 속도를 고려 64-bit 일반적
 - 키 크기 (128-bit) → 128-bit
 - 큰 키는 보안 강화하지만 암호/복호화 속도는 저하
 - 암호/복호화 속도를 고려하여 128-bit 일반적
64-bit 이하는 해독 용이
 - 반복 수
 - 다중 반복과정은 보안성 강화
 - 16회 반복이 일반적
 - 서브키 생성 알고리즘
 - 서브키 생성 방법이 복잡할수록 강력
 - 반복 함수 (F)
 - 적용되는 반복함수가 복잡할수록 강력

- ◆ 복호화 알고리즘
 - 암호화와 복호화에 같은 키 사용을 위하여 서브키 k_i 를 역순으로 사용

✓ 특징

- 1977년 미국표준국(ANSI)에서 미 연방 정보처리 표준 46(FIPS 46)으로 채택
- 64-bit 블록 암호 알고리즘
- 56-bit 키를 사용
 - ♦ 64-bit 중 8-bit는 parity check로 사용
8, 16, 24, 32, 40, 48, 56, 64 총 8개
- 기본 구조
 - ♦ round 수 - 16 round
 - ♦ 복호화는 암호화의 역순
- 최근에는 DES암호화를 세 개의 키로 세 번 반복함으로써 암호의 강도를 높인 Triple-DES 사용

✓ 치환과 전치 혼합방법, 블록 암호, 관용 암호 방식

✓ 역사

- 1960년 후반 - Feistel에 의해 컴퓨터 암호 과제 선정
- 1971년 - LUCIFER 개발 (128-bit 키 사용)
- 1973년 - Tuchman-Meyer 과제로 DES 개발
 - ♦ 수학적으로 안전/컴퓨터로는 해독됨
- 1977년 - 데이터 암호 표준으로 채택
- 1994년 - NIST에서 재사용 인가
- 1999년 - 3중 DES의 표준 발표 (FIPS PUB 46-3) → 2001년 - AES 표준화

✓ 초기 순열

- 64-bit 평문 메시지 순열 $X = IP(M)$



M1	M2	M3	M4	M5	M6	M7	M8	1	M58	M50	M42	M34	M26	M18	M10	M2
M9	M10	M11	M12	M13	M14	M15	M16	2	M60	M52	M44	M36	M28	M20	M12	M4
M17	M18	M19	M20	M21	M22	M23	M24	3	M62	M54	M46	M38	M30	M22	M14	M6
M25	M26	M27	M28	M29	M30	M31	M32	4	M64	M56	M48	M40	M32	M24	M16	M8
M33	M34	M35	M36	M37	M38	M39	M40	5	M57	M49	M41	M33	M25	M17	M9	M1
M41	M42	M43	M44	M45	M46	M47	M48	6	M59	M51	M43	M35	M27	M19	M11	M3
M49	M50	M51	M52	M53	M54	M55	M56	7	M61	M53	M45	M37	M29	M21	M13	M5
M57	M58	M59	M60	M61	M62	M63	M64	8	M63	M55	M47	M39	M31	M23	M15	M7

✓ 순열 표

▪ 초기 순열 IP (64 → 64)

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

▪ 역 초기 순열 IP-1 (64 → 64)

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

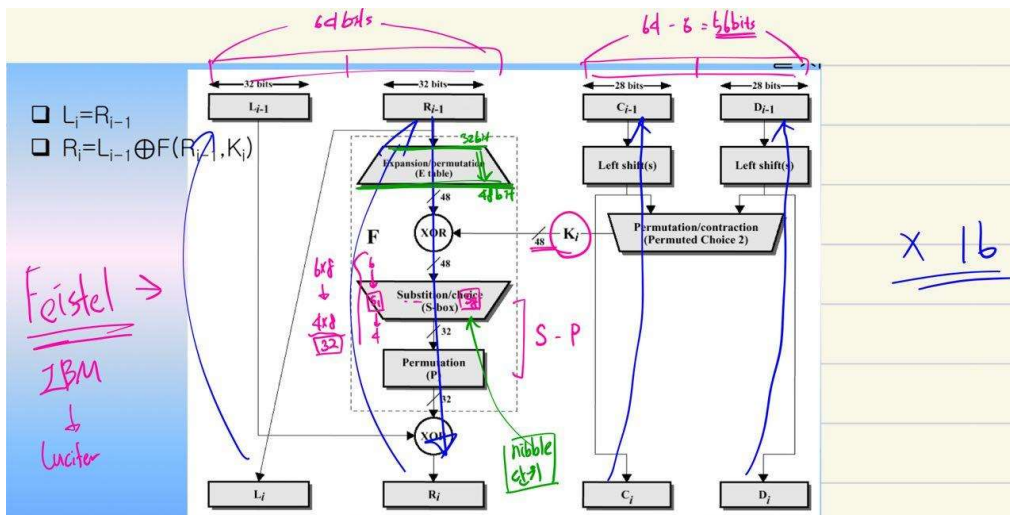
▪ 확장 순열 (32 → 48)

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

▪ 순열 함수 (32 → 32)

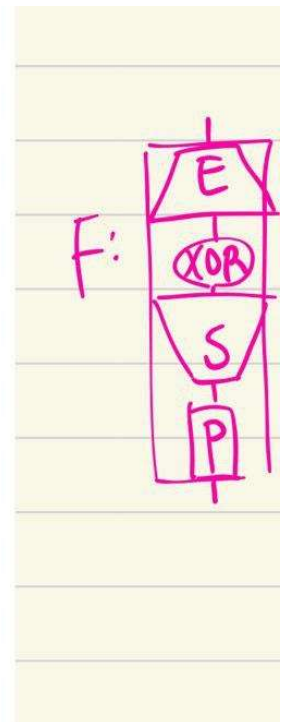
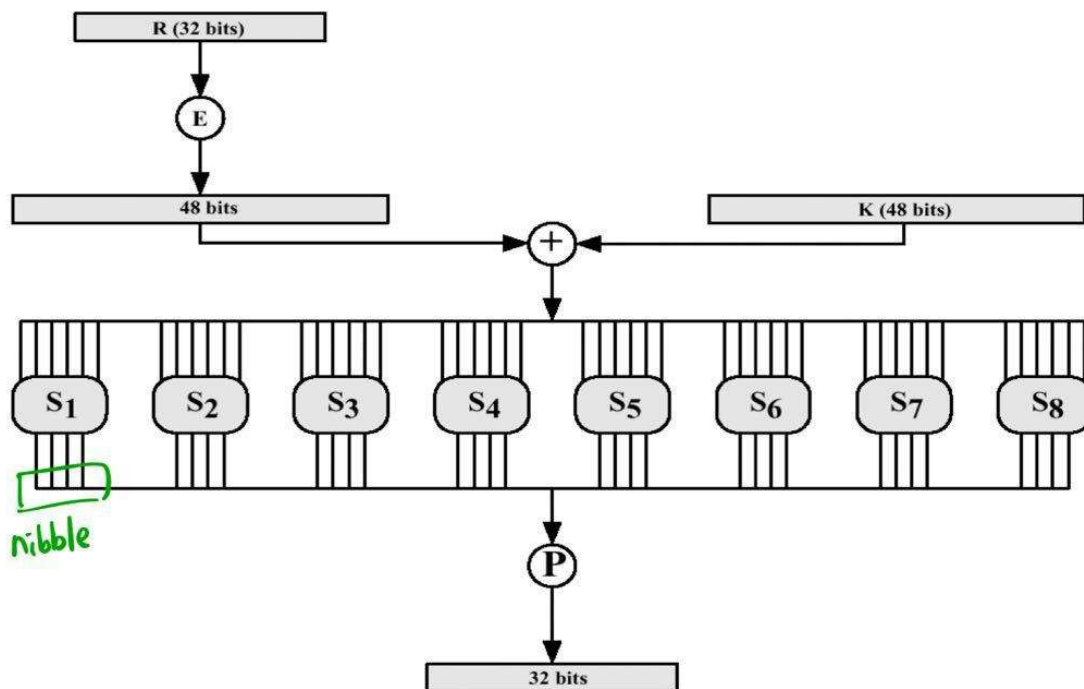
16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	16	32	27	3	9
19	13	30	6	22	11	4	25

✓ 알고리즘 단일 반복과정



✓ 함수 F

- 8개의 S-Box로 구성
- 각 S-Box는 6-bit로 입력, 4-bit로 출력 생성



✓ S-Box Table

- 4행*16열의 표
- 입력되는 6-bit 중
처음과 끝 비트를 이용하여 행을 찾아가고
중간 4개의 비트를 이용하여 열을 찾아가는다

□ S-Box Table

중간 양 끝

양 끝	중간	0000	...	1111													
00		14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
01		0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
10		4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
11		15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

❖ 예) 0 1 1 0 1 1 ----- (6비트 입력)

행 (1행) 열 (13열) → '011011'을 4비트로 출력: "0 1 0 1"

10행 13열 → 5 → 0101₍₂₎

00행 ~ 15행

00행
10행
20행

✓ DES 쇄도 효과 → 에러 확산 → 무선 X

▪ 평문 변화

- 평문이나 키의 작은 변화가 암호문에 대하여 중요한 변화를 일으키게 하는 효과
- 평문이나 키를 1-bit 바꿀 때, 암호문의 여러 비트가 변함

▪ 키 변화

- 각 라운드가 경과할 때마다 바뀔 정도가 달라지는데
 $0 \rightarrow 1$ 또는 $1 \rightarrow 0$ 으로 완전히 바뀌는 것은 50% 뿐이므로
변화율이 50% 이상이 되면 완전히 바뀐 것으로 봐야한다

✓ 강점

▪ 56-bit 키 사용

- 미 정부의 채택 이후 DES의 보안 수준에 대한 우려가 지속
- 마이크로 세컨드 당 하나의 암호를 수행하는 장치 1백만개로 구성된 병렬기계 구성
 - 평균 탐색시간 10시간으로 축소
- 1998년 7월 EFF(Electronic Frontier Foundation)가
\$250,000 이하의 비용으로 제작된 DES해독기를 통해 DES 알고리즘 해독

▪ DES 알고리즘 자체 성질을 이용한 해독 가능성

▪ 8개의 치환 표 또는 S-Box

▪ S-Box의 경우

- 전체 알고리즘에 대한 설계기준이 공개된 적이 없다
- S-Box의 약점에 대해 알고 있는 공격자가 해독할 수 있게 설계되었을 가능성에 대한 의혹 제기
- 수 년간 S-Box의 수많은 정규성과 예측 못할 동작 발견
- 그럼에도 S-Box에 대하여 치명적인 약점을 발견한 사람은 아직까지 없다
적어도 그런 발견이 공개된 적이 없다

▪ Timing 공격 - P.kocher → D.P.A.

- 암호문을 복호화 하는데 걸리는 시간을 관측하여 키 또는 평문의 정보를 획득
- 입력에 따라 처리시간이 다르다는 것을 이용
- 아직까지는 성공 사례가 없으나 흥미로운 연구

▪ 차분 암호 해독

- DES에 적용해서 널리 알려짐
- 2^{55} 미만의 복잡도로 DES를 해독
- 2^{47} 의 선택 평문을 가지고 2^{47} 차수로 DES를 해독

- 선형 암호 해독
 - ♦ 2^{43} 기지 평문으로 해독 가능
 - ♦ 기지 평문이 선택 평문보다 구하기 쉽지만 DES 공격으로서의 선형 암호 해독은 가능성이 없음
 - ♦ 선형 암호 해독의 타당성 주장이 약함

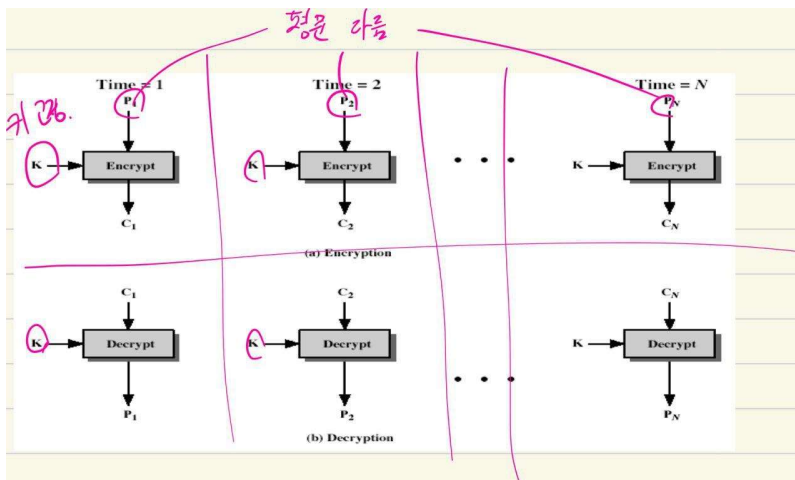
✓ 설계기준

- S-Box 설계기준
 - ♦ S-Box의 각 행은 (0~15)까지 정수의 랜덤 한 배치 순열
 - ♦ 어떤 S-Box도 출력에 대하여 선형(linear)도 아핀(affine)도 아니다
 - ♦ S-Box의 6-bit 입력 중 1-bit 가 변경되면
출력은 최소 2-bit 이상 달라야 함
(2/4, 즉 50% 이상 변경됨)
 - ♦ S-Box의 6-bit 입력 중, 가운데 2-bit만 변경되면
출력은 최소 2-bit 이상 달라야 함
(2/4, 즉 50% 이상 변경됨)
 - ♦ S-Box의 입력 중 첫 2-bit가 다르고, 마지막 2-bit가 같으면 두 출력은 달라야 함
 - ♦ 입력들 사이에 0이 아닌 6-bit 차이는 그러한 차이를 나타내는 입력은
32쌍 중 오직 8쌍은 같은 출력을 야기
- 순열 P 설계기준
 - ♦ i번째 반복에서 각 S-Box로부터의 4개의 출력 비트가 분산되어 2-bit는
(i+1)번째 반복의 중간비트에 영향을 미치고 다른 2-bit는 양 끝 비트에 영향을 줌
 - ♦ 각 S-Box의 출력 4-bit는 다음 반복에서 여섯 개의 다른 에 영향을 주며,
같은 S-Box에 영향을 주지 않음
 - ♦ 두 개의 S-Box j, k에서 j=k에 대하여 S_j 의 한 출력비트는 S_k 의 중간비트에 영향을 주어서는 안
됨
- 알고리즘의 확산을 증가시키기 위한 기준
- 라운드 수
 - ♦ 라운드 수가 많으면 F함수가 약하더라도 암호 해독은 어려움
 - ♦ 전사적 키 탐색 공격 이상의 노력이 요구되도록 라운드 수 선정

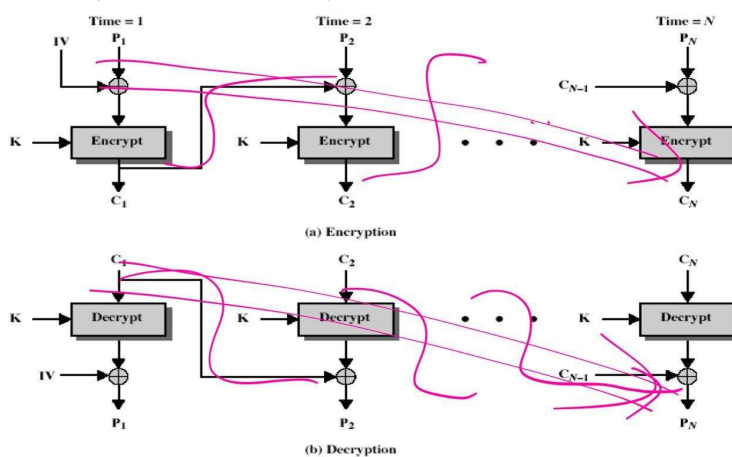
- F함수 설계

- ♦ 치환 결과를 복구하기 난해해야 함
- ♦ 엄격한 쇄도 기준
 - 모든 i 와 j 에 대하여 어떤 입력비트 i 가 바뀌면 S-Box의 출력비트 j 는 1/2의 확률로 바뀌어야 함
- ♦ 비트 독립 기준
 - 입력비트 i 가 바뀌면 출력비트 j, k 가 바뀌어야 함

- ✓ ECB – Electronic Codebook Book

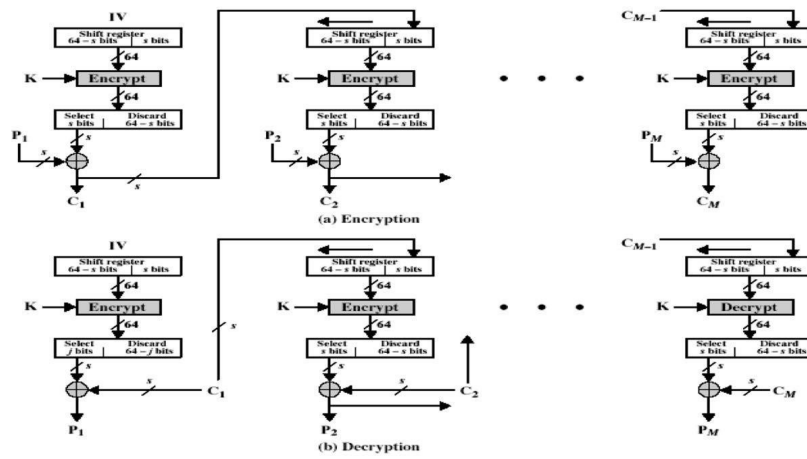


- ✓ CBC – Cipher Block Chaining



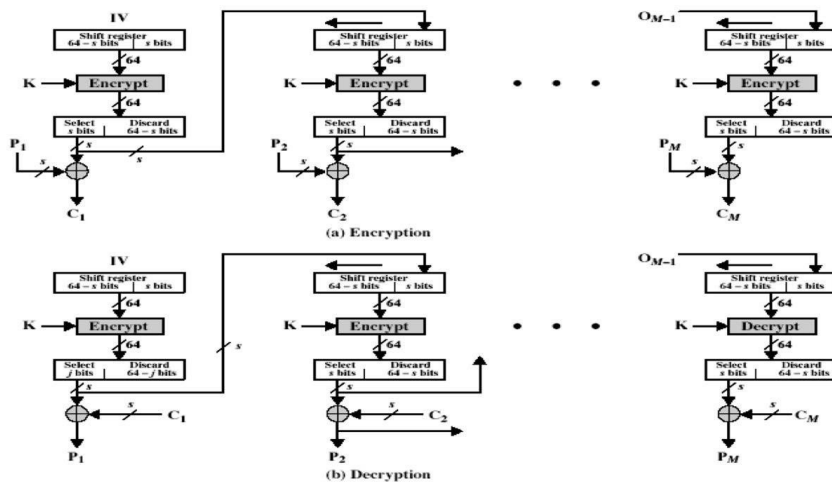
- 압축, 무결성, 암호화

✓ CFB – Cipher FeedBack



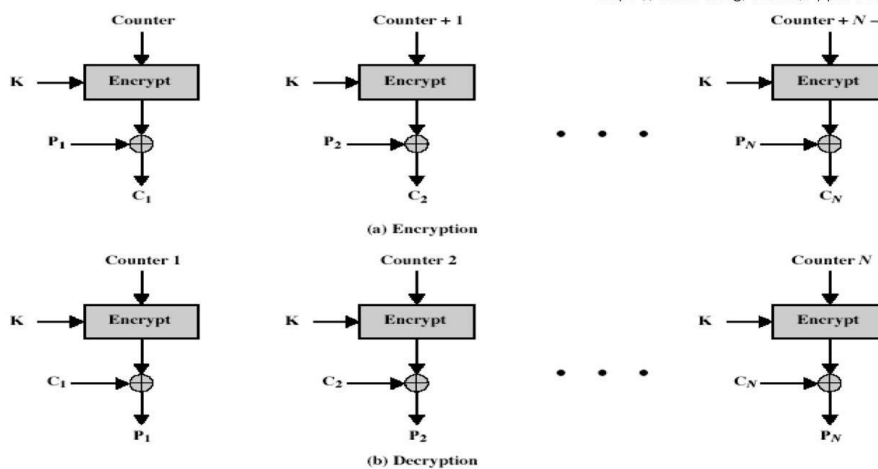
- 동기화

✓ OFB – Output FeedBack



- 스트림 암호

✓ CTR – CounteR



- CFB와 유사
- 무선 사용

AES

✓ 핵심

- AES는 DES를 대체하는 블록 암호로 상업적 용도로 개발
 - ♦ 128-bit 블록 사이즈를 사용
 - ♦ 128, 192, 256-bit 를 key size로 사용
민간 군용
- Feistel 구조를 사용하지 않지만, 각 라운드는 4가지 암호화 과정으로 이루어짐
 - ♦ 바이트 치환 변환 (Byte Substitute)
 - ♦ 행 이동 (Shift Row)
 - ♦ 열 혼합 (Mixing Column)
 - ♦ 라운드 키 더하기 (Add Round Key)

✓ 유한체 연산

- AES와 유한체 집합
 - ♦ AES는 8-bit를 기본 단위로 동작하며 덧셈, 곱셈, 나눗셈 연산을 유한체를 바탕으로 연산
 - 2K word → 1개의 S-Box로 더 빠르게 연산 가능
 - ♦ Rijndael 알고리즘 개발자들은 더 이상 약분할 수 없는 8차 다항식 30개를 선택하여 사용
- 유한체의 집합 S의 수학적 연산 정의

✓ 기준

- 초기 기준 (1997)
 - ♦ Security – effort to practically cryptanalyse
보안성 향상
 - ♦ Cost – computational
비용 절감
 - ♦ Algorithm & implementation characteristics
- 최종 기준 (2000)
 - ♦ General security
일반적 보안성
 - ♦ Software & hardware implementation ease
소프트웨어 및 하드웨어에 구현 용이
 - ♦ Implementation attacks
부 채널 공격 (T.A./D.P.A.)
 - ♦ Flexibility (in en/decrypt, keying, other factors)
유연함(암호화/복호화, keying 등)

✓ 후보자 - 1차(안전성), 2차(구현성)

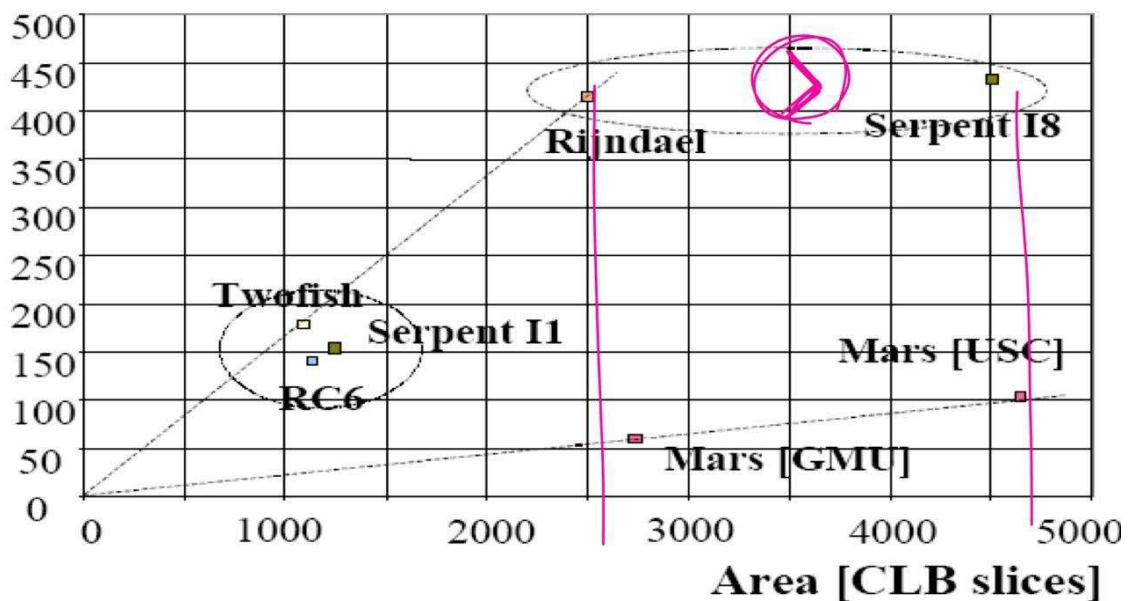
▪ 99년 8월 최종 후보자

- ♦ MARS (IBM) - 복잡함 --- X
- ♦ RC6 (USA) - 보안성 저조 --- X
- ♦ Rijndael (Belguim) - clean, fast, good security margin (깔끔함, 빠름, 보안성 우수)
- ♦ Serpent (Euro) - 느림 --- X
- ♦ Twofish (USA) - 복잡함 --- X

▪ 2000년 최종

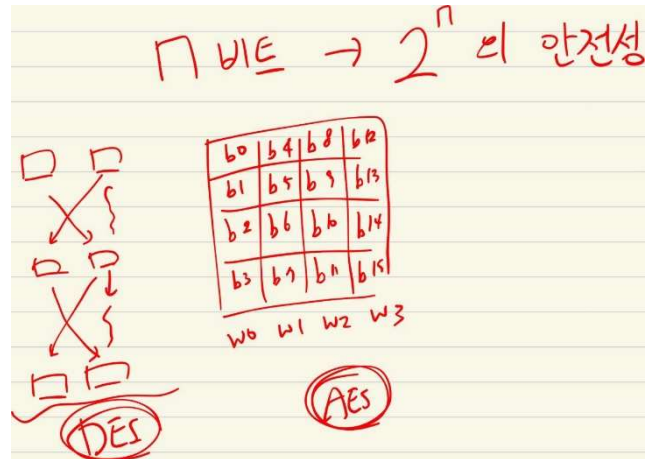
- ♦ 신 기술보다 구 기술이 오히려 능률이 높았다...
- ♦ Rijndael 알고리즘이 대부분의 테스트에서 우수하였다
Serpent18 알고리즘이 약간 더 우수하였지만
- ♦ ASIC 0.5(마이크로)m의 테스트에서는
Rijndael 알고리즘은 우수한 점수를 얻었고
Serpent18 알고리즘은 테스트를 실행하지 않았다
- ♦ 최종적으로 Rijndael 알고리즘은 Serpent18 알고리즘보다
칩의 면적과 속도 면에서 우수함을 보이게 되어 최종적으로 AES 암호 알고리즘으로 채택되었다.

Throughput [Mbit/s]



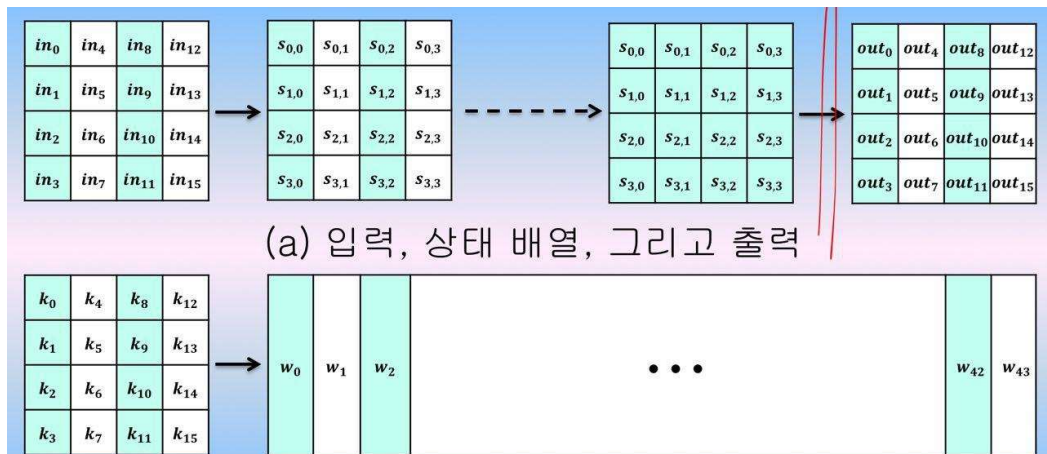
✓ Rijndael

- Designed by Rijmen-Deamen in Belgium
- 128, 192, 256-bit keys
10 12 14 word
- 128bit data
- Feistel 암호보다 반복적이다
- resistant against known attacks
이미 알고 있는 공격방식에 대응
- speed and code compactness on many CPUs
많은 CPU에서 속도 향상 및 코드 압축
- design simplicity
심플한 디자인



✓ 구조

- 일반 구조
 - 암호화 복호화 과정에서 128-bit 블록이 쓰이며, 4*4 바이트 행렬로 나타냄
 - 블록은 상태 배열로 복사되며 암호화와 복호화 각 단계에서 수정되어 쓰이고 마지막 단계에서 출력 행렬로 복사됨
 - 키는 비트 행렬로 나타내어지며, 키 스케줄러 연산에 따라 확장
 - 128-bit 키의 확장은 행렬 안에서 1워드로 4-bit가 할당되며,
총 스케줄러는 128-bit 키에서 44 word 생성
총 스케줄러는 192-bit 키에서 52 word 생성
총 스케줄러는 256-bit 키에서 60 word 생성
- AES 데이터 구조

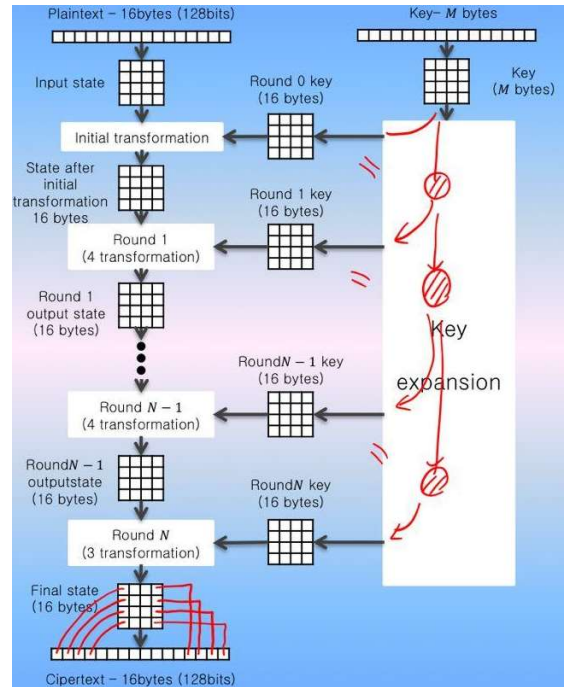


- AES 암호화 과정

- 키 길이는 16, 24, 32-byte이며
키 길이에 따라
AES-128, AES-192 AES-256으로 나뉨
- 평문의 블록 사이즈는 128-bit(16-byte)

라운드 수	키 길이(bytes)
10	16 4.4
12	24 4.6
14	32 4.8

word



- AES 파라미터

- N 라운드로 이루어진 암호화에서 라운드 수는 키 길이에 의존

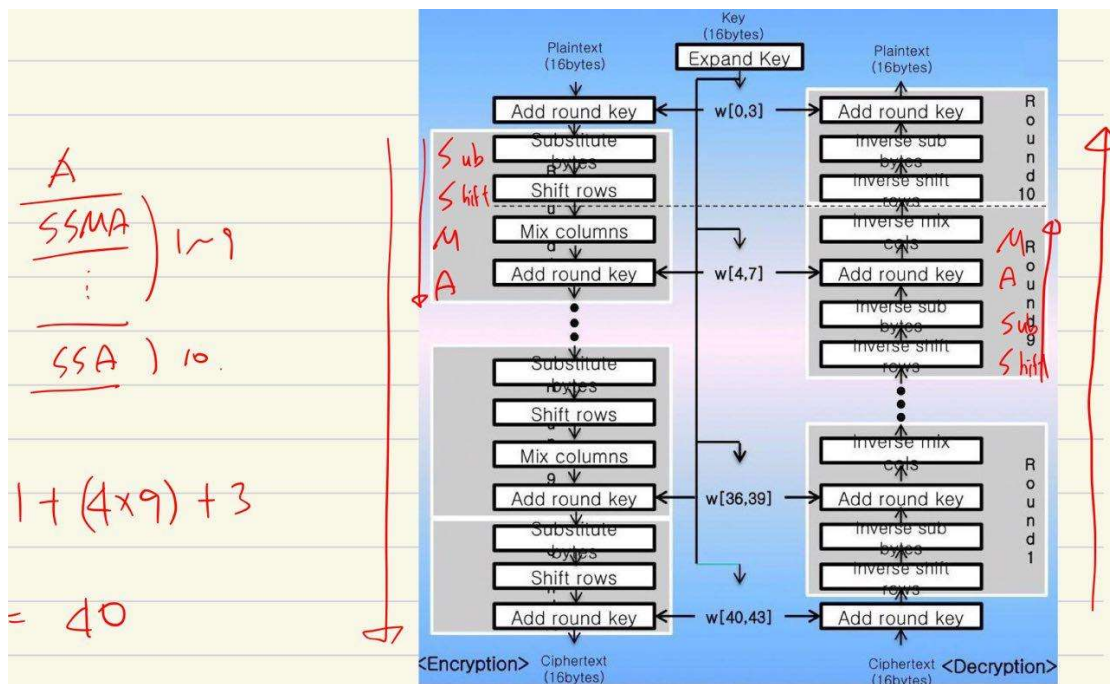
키 길이 (워드/바이트/비트)	4/16/128	6/24/192	8/32/256
평균 블록사이즈 (워드/바이트/비트)	4/16/128	4/16/128	4/16/128
라운드 수	10	12	14
라운드 키 길이 (워드/바이트/비트)	4/16/128	4/16/128	4/16/128
확장 키 길이 (워드/바이트/비트)	44/176	52/208	60/240

4×4 $+8$ $+8$
 2라운드 증가

- 세부 구조

- Rijndael 알고리즘을 포함하여, AES의 최종 평가에 오른 두 후보 알고리즘은 Feistel 구조를 사용하지 않으며, 각 라운드에서 순열과 변환이 행하여 지는 동안 전체 데이터 블록이 병렬 처리됨
- 입력으로 주어지는 키는 44개의 32-bit word, $w[i]$ 로 확장되며 4개의 서로 다른 word(128-bit)는 각 라운드에서 라운드 키로서 사용

- 1번의 순열(permutation)과정과 3번의 치환(substitution)과정으로 이루어진 4단계로 구성
 - 바이트 치환 변환 (Byte Substitute)
 - 블록의 바이트 대 바이트 치환 변환을 수행하기 위해 하나의 S-Box를 사용
 - 행 이동 (Shift Row)
 - 행렬의 행 이동을 통한 순열 과정
 - 열 혼합 (Mixing Column)
 - $GF(2^8)$ 산술 식을 사용한 치환 과정
 - 라운드 키 더하기 (Add Round Key)
 - 현재 블록과 확장된 키의 일부로 단순 비트 단위의 XOR 과정
- 암호화와 복호화 모두에서 라운드 키 추가 단계를 시작으로
4단계 과정으로 이루어진 9회의 라운드가 진행되고, → SSMA
3단계 과정으로 이루어진 마지막 10번째 라운드가 실행됨 → SSA
- 라운드 키 추가 단계에서만 유일하게 키 값을 사용하며,
이 때문에 암호는 라운드 키 추가 단계로 시작되며 또는 끝남.
처음과 마지막에 적용되는 각 단계는 키에 대한 내용을 모르더라도
역으로 갈 수 있으며 더 이상의 보안성이 추가되지 않음
- 각 단계는 역이 가능하며 라운드 키 추가단계에 대한 역함수는
같은 라운드 키를 블록에 XOR하여 얻음
- 복호화 알고리즘은 확장키의 역순을 사용함으로써 이루어지나,
복호화 알고리즘은 암호화 알고리즘과 동일하지는 않음
- 암호화와 복호화를 위한 상태 배열은 동일함
- AES가 가지는 특수한 구조로 암호화의 역이 가능하기 위해
암호화와 복호화 모두 마지막 라운드는 3단계만으로 구성



✓ 변환기능

■ 바이트 치환 변환 (Byte Substitute)

- AES는 S-Box라고 불리는 바이트 값들의 16*16 행렬을 정의

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

< S-box >

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

< Inverse S-box >

- 256가지의 모든 가능한 8-bit 값의 순열을 포함
- 상태 배열의 각 개별 바이트는 다음과 같은 방법으로 대응
- 바이트의 가장 왼쪽 4-bit는 행 값(row value)
가장 오른쪽 4-bit는 열 값(column value) 으로 사용됨
- 행 값과 열 값은 S-Box에서 8-bit의 출력 값을 선택하기 위한 인덱스로 사용됨
- 총 8-bit이며 4자리씩 끊어 16진수 2자리로 나타낸다

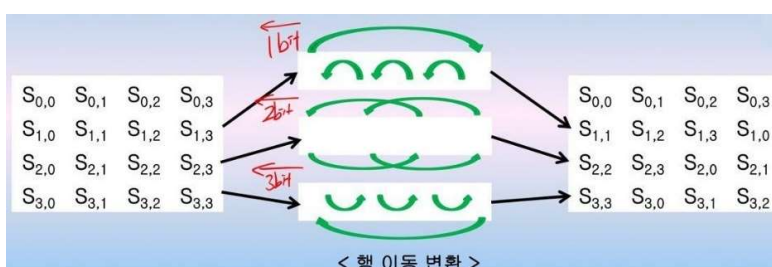
EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

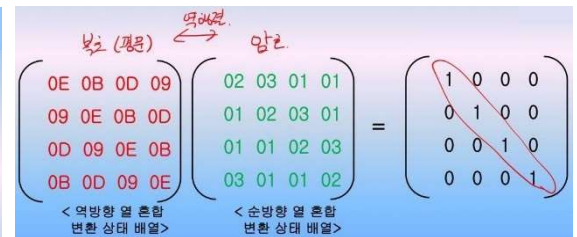
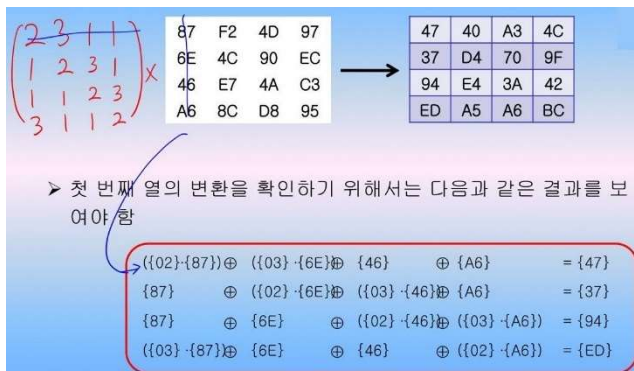
■ 행 이동 (Shift Row)

- 행 이동 변환에서 상태 배열의 첫 번째 행은 바뀌지 않음
- 두 번째 행은 1-byte,
세 번째 행은 2-byte,
네 번째 행은 3-byte씩 왼쪽으로 순환 이동
- 역 행 이동 변환은
첫 번째 행은 바뀌지 않고,
두 번째 행은 1-byte,
세 번째 행은 2-byte,
네 번째 행은 3-byte씩 각각 오른쪽으로 순환 이동



열 혼합 (Mixing Column)

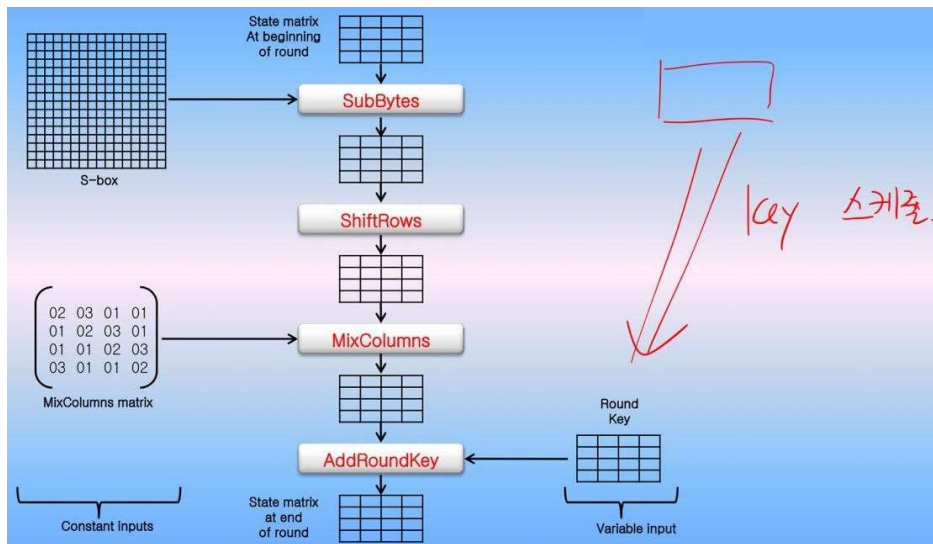
- 열 혼합 변환은 열 단위로 계산됨
- 열의 각 바이트는 그 열의 모든 4-byte 함수의 새로운 값으로 mapping 됨
- 아래의 그림과 같은 상태 배열의 행렬 곱으로 계산됨
- 순 방향과 역 방향의 상태 배열
- 순 방향과 역 방향 열 혼합 수행 시, 방향에 맞는 상태 배열을 사용해서 계산해야 함
- 두 배열은 서로 역원 관계



라운드 키 더하기 (Add Round Key)

- 라운드 키 추가 변환에서 상태 배열의 128-bit는 라운드 키의 128-bit와 비트단위로 XOR 연산을 수행
- 상태 배열의 한 열 4-byte와 라운드 키의 1-word 사이의 열 연산





✓ 키 확장

▪ 키 확장 알고리즘

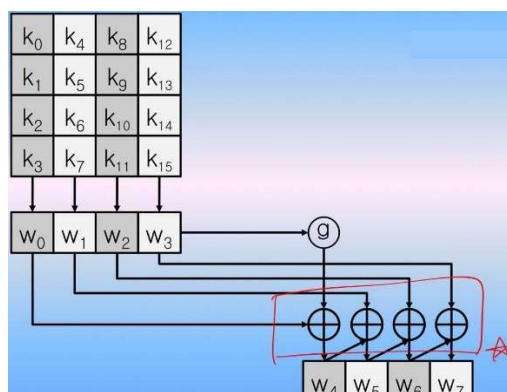
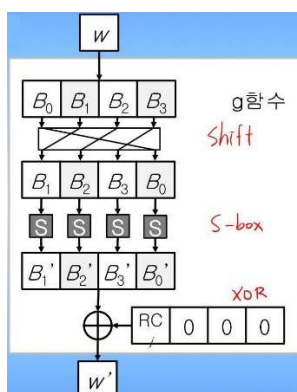
- 4-word(16-byte)의 입력을 받아 44(176)의 선형 배열로 확장
- 키는 확장된 키의 첫 4로 복사
- 추가되는 word(w[i])는 word(w[i-1])와 4개 이전 word(w[i-4])에 의해 결정
 - 이전 4-word 중 3개는 단순 XOR
 - 첫 번째 word에서는 4번째 XOR을 수행하기 전에, 순환이동, S-Box, 이전 라운드 상수와의 XOR 연산 수행

```

KeyExpansion (byte key[16], word w[44])
{
    word temp;
    for (i=0; i<4; i++) {
        w[i] = (key[4*i], key[4*i+1], key[4*i+2], key[4*i+3]);
    }
    for (i=4; i<44; i++) {
        temp = w[i - 1];
        if (i mod 4 == 0)
            temp = SubWord(RotWord(temp)) XOR Rcon[i/4];
        w[i] = w[i - 4] XOR temp;
    }
}

```

Handwritten notes: *순환* (circular shift), *S-Box*, *상수 XOR* (constant XOR), *함수* (function), *P.48*, *⊕* (XOR symbol).



- 원리

- ♦ 알려진 공격에 대해 저항성을 갖도록 설계
- ♦ 설계에 사용된 세부 기준
 - 암호화 키 또는 라운드 키의 일부분을 알아도 다른 라운드 키 비트 계산이 불가능
 - 역 변환 가능
 - 다양한 프로세스에서의 빠른 연산
 - 대칭성을 제거하기 위한 라운드 상수의 사용
 - 암호화 키 확산을 통해 라운드 키 변환
각 키 비트의 변화는 라운드 키 비트에 영향을 미침
 - 암호화 키의 차이점만으로부터 라운드 키 차이점의 전체를 결정하는 것을 방지하기 위한
충분한 비선형성
 - 간결한 표현

✓ 구현

- 복호화

- ♦ AES 복호화는 암호화 과정을 역 순으로 수행하기 때문에 암호화 과정과 동일한 과정이 아님
- ♦ 복호화 과정을 정의할 수는 있음
 - 모든 단계를 역순으로 수행
 - 키 스케줄링을 다르게 하여 수행
- ♦ 결과 값이 변경되지 않는 경우 동작 → S-S-M-A
 - 바이트 치환과 행 이동을 맞바꿈
 - 열 혼합과 라운드 키 추가를 맞바꿈
- ♦ .

- *** 암호화 → S - S - M - A
 sub shift mix add

- *** 복호화 → S - S - A - M
 shift sub add mix

- 구현의 관점
 - ♦ 8-bit CPU에서 매우 효율적으로 구현 가능
 - 바이트 치환 연산은 바이트 레벨 연산으로써 256-byte의 테이블만이 요구됨
 - 행 이동은 단순한 바이트 이동임
 - 열 혼합 연산은 모든 연산이 바이트로 실행되는 $GF(2^8)$ 상의 행렬 곱셈을 필요로 하며, 테이블 참조와 바이트 단위 XOR 연산으로 단순화될 수 있음
 - 라운드 키 추가는 바이트 단위 XOR 연산임
 - 32-bit의 word를 사용하기 위해 재정의
 - 256-word로 이루어진 4개의 테이블을 미리 연산 가능
 - 매 라운드의 각 열은 4번 테이블 참조와 4번의 XOR을 사용하여 계산 가능
 - 테이블을 저장하는데 4KB 필요
- ♦ 설계자들은 이러한 구현 용이성이 AES 암호로서 선택된 핵심 이유라고 여기고 있음

부 채널 공격

- ✓ 소형 정보보호장치 대중화
 - 스마트카드, USB 보안장치, USIM 카드 RFID/USM 암호 모듈, PDA, crypto-module 등
- ✓ P.Kocher에 의해 제안된 소형 정보보호장치에 대한 가장 위협적인 공격
 - TA, FA, SPA/DPA/CPA, SEMA/DEMA 등
- ✓ 인체의 부 채널
 - 전통적 방식
 - ♦ 맥박수, 외형관측, 눈동자 관찰, 땀 분비 관찰, 소변/대변 색 관찰
 - 현대식 방식
 - ♦ 청진기, X-RAY, CT, MRI, 신체해부(수술)
- ✓ 캠퍼스의 부 채널
 - 정문, 후문이 아닌 쪽 문이 부채널에 해당한다
- ✓ 구분
 - 정규 채널 → 데이터가 입. 출력되는 경로인 정상적 경로
 - 부 채널 → 정규 채널이 아닌 그 외의 방법으로 접근하는 경로를 말한다

- Traditional Assumptions – no side-channel leaked

이론상 – 부 채널 누출 없음

- Actual Systems – side-channel leaked

실제 시스템 – 부 채널 누출됨

- Power consumption, Electro-magnetic radiation, Timing, Errors, Etc.

전력 소비

전자 방사선

타이밍 오류들

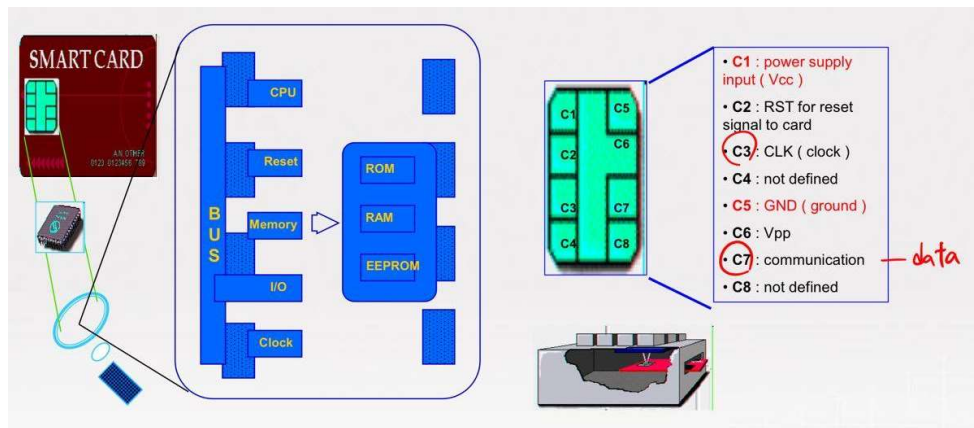
- 입. 출력에 따른 공격 분류

	정규 출력	누설 출력	내부 신호의 직접 관측
정규 입력	부채널공격(Side Channel Attack) TA, PA, EMA 공격 비침투형 공격 (Non-Invasive attack)		
정규 외 입력	오류주입공격 (Fault-insertion attack) 준침투형 공격 (Semi-Invasive attack)		
모듈 내부로의 입력		침투형 공격 (Invasive attack)	

- 역사

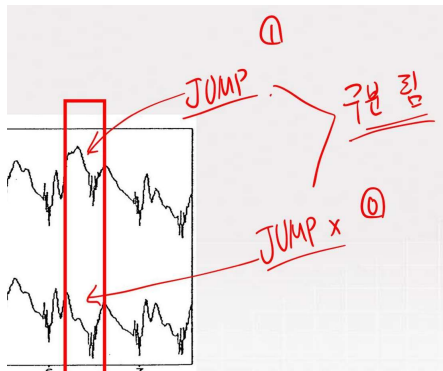
- 1996 – TA (Timing Attack)
- 1997 – FA (Differential Fault Analysis)
- 1999 – DPA (Differential Power Analysis)
- 2001 – EMA (Electro Magnetic Analysis)

- 스마트카드



▪ SPA (Simple Power Analysis)

- ♦ 스마트카드에서 연산 되는 암호 프로세서의 전력소비를 직접 관찰하여 카드에 저장되어 있는 비밀키를 직접 공격하는 방법
- ♦ 스마트카드 외부에서 관측 → 키 또는 순간 동작 중인 연산에 대한 정보를 추론하는 공격 방법
- ♦ 16-round DES 측정 파형 → 0.4ms/1-round interval, @ 3.7MHz clock, 0.27microsecond
- ♦ 2~3 round DES 측정 파형
- ♦ 28-bit 키 레지스터 C와 D
 - 2-round에서 1-bit left shifting
 - 3-round에서 2-bit left shifting
- ♦ Jump 명령이 구분됨



- ♦ To RAS
 - S(제공 연산) → 반드시 수행
 - M(곱 연산) → 결과가 1이 된다 진폭이 커져 결과가 달라진 것 확인 가능
- ♦ To ECC
 - D(2배 연산) → 반드시 수행
 - A(덧셈 연산) → 결과가 1이된다. 진폭이 커지면서 결과가 달라진 것 확인 가능

▪ DPA

- ♦ 평문 입력 → 암호문 출력
 - 순간 전력 샘플 데이터 수집 → 평균 차분
- ♦ 비밀키가 반응하는 지점에서 함수가 내장된 연산 비밀키와 correlation 특성을 이용한 공격
- ♦ 키가 동작하면 진폭이 눈에 띄게 커지게 되고 키가 동작하지 않으면 진폭이 거의 0가 된다

◆ 단계 - 일반화

1. 전체 구하려는 스마트 카드의 n-bit 비밀키 K라 정의하고
최상위 혹은 최 하위 비트의 순서로 순차적으로 키의 일부가 입력되어 연산 된다고 가정
2. 먼저 키의 일부인 K_i 또는 K_i, \dots, K_j 를 미리 가능한 키 영역에서 추측
3. 추측한 키와 전력 신호 데이터를 구할 때 쓴 평문을 입력으로 연산을 수행한 후
분류 함수를 이용하여 전력신호 데이터를 분류한다

$S_0 = \{S_i[j] \mid D(\text{key}, \text{data}) = 0 \text{ or low hamming weight}\}$

$S_1 = \{S_i[j] \mid D(\text{key}, \text{data}) = 1 \text{ or high hamming weight}\}$

4. 양분한 데이터를 각각 평균하여 차분신호를 구한다

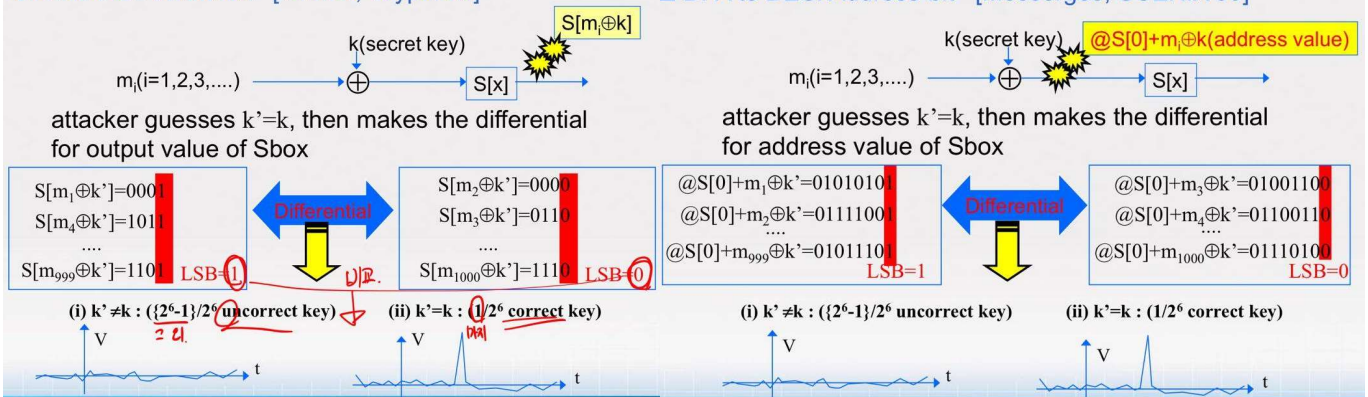
$$\Delta_D[j] = \left[\frac{1}{|S_0|} \sum_{S_i[j] \in S_0} S_i[j] \right] - \left[\frac{1}{|S_1|} \sum_{S_i[j] \in S_1} S_i[j] \right]$$

5. 평문과 전력소비신호의 수가 많을 시 추측한 키가 옳다면 신호가 non-zero이고
맞지 않다면 거의 0에 수렴한다

◆ To DES

DPA to DES : Data-bit [Kocher, Crypto'99]

DPA to DES: Address-bit [Messerges, USENIX'99]



■ CPA

◆ 첨두간 기준주기와 보간법을 적용한 전력신호의 정렬

- 기준 전력신호를 정하고
기준 전력신호로부터 기준 주기(t)를 구하고
전력신호의 첨두간 주기 동안에 기준 주기와 같도록 보간법을 적용하여 정렬

- ◆ AES의 공격(일정 클릭)

- 128-bit의 비트 열 입력

- 128,192, 256-bit의 비트 열 암호화 키로 구성

- 기본적으로 8-bit 비트 열인 바이트 단위로 실행

- SubBytes(), ShiftRow(), MixColumn(), AddRoundKey()의 함수로 구성

- 암호화 키 비트 열 크기 – 128, 192, 256

- 반복 라운드 수 – 10, 12, 14 라운드

- 제로통과점 기준으로 정렬된 전력신호 상관계수 추정 값보다

- 첨두점 기준으로 정렬된 전력신호 상관계수 추정 값이

- 20%정도 향상된 결과가 나왔다

- ◆ 결과

- 상관계수가 최대 위치와 전력신호의 첨두점의 위치가 동일

- 첨두점을 기준으로 전력신호를 정렬하는 방법이 전력분석공격에 유리

- EMA (Electro Magnetic Analysis)

- ◆ 공격 원리

- 암호장치의 내부 게이트에서 전류의 흐름에 의하여 발생하는

- 전자파장에 대한 정보의 노출을 분석하는 공격방법

- ◆ 공격 방법

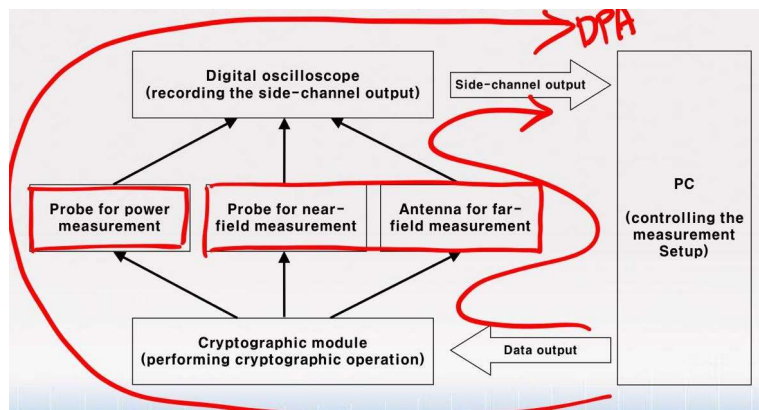
- Simple EMA / Differential EMA

- ◆ 공격 대상

- 스마트카드, USB Key-token, RFID, 암호 모듈/장치

- ◆ 유럽 SCARD(Side Channel Analysis Resistant Design)

- Power attack 및 EM attack을 위한 장치 구성 블록도



- ECC 암호에 대한 SEMA 공격

- 160-bit EC 점 곱셈연산에서 전자파 방사파형 – SEMA

- DA → 1 D → 0