2023/02/01, 2023 겨울방학 보안기초 세미나

# Network Security Essentials

## - Chapter_3 공개 키 암호와 메시지 인증(1) -

강 민 채(minchae@pel.sejong.ac.kr)

세종대학교 프로토콜공학연구실

# 목 차

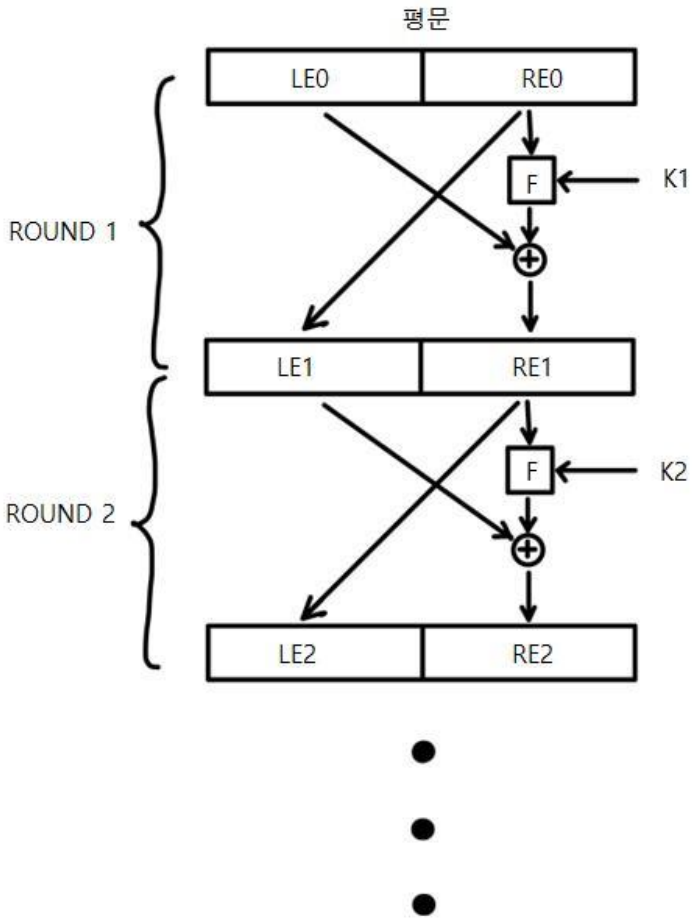
- 보충
  - DES, AES에서의 대체/치환
  - 페이스텔 구조에서의 Sub Key 생성과정
  - 페이스텔 구조에서의 F함수
  - 스트림 암호와 RC4
  - 암호 블록 운용 모드
- 메시지 인증 방법
- 안전 해시 함수
- 메시지 인증 코드

# 목 차

- 보충
  - DES, AES에서의 대체/치환
  - 페이스텔 구조에서의 Sub Key 생성과정
  - 페이스텔 구조에서의 F함수
  - 스트림 암호와 RC4
  - 암호 블록 운용 모드
- 메시지 인증 방법
- 안전 해시 함수
- 메시지 인증 코드

# 보충

- DES에서의 대체/치환
  - 한 라운드의 암호화 과정
    - 평문 블록을 두 조각으로 나누고, 왼쪽 반의 데이터를 오른쪽 반의 데이터에 라운드 함수를 적용한 것과 XOR한것으로 대체한다
    - 두 개의 반쪽짜리 데이터를 치환한다
  - 대체, 치환이 둘 다 적용됨

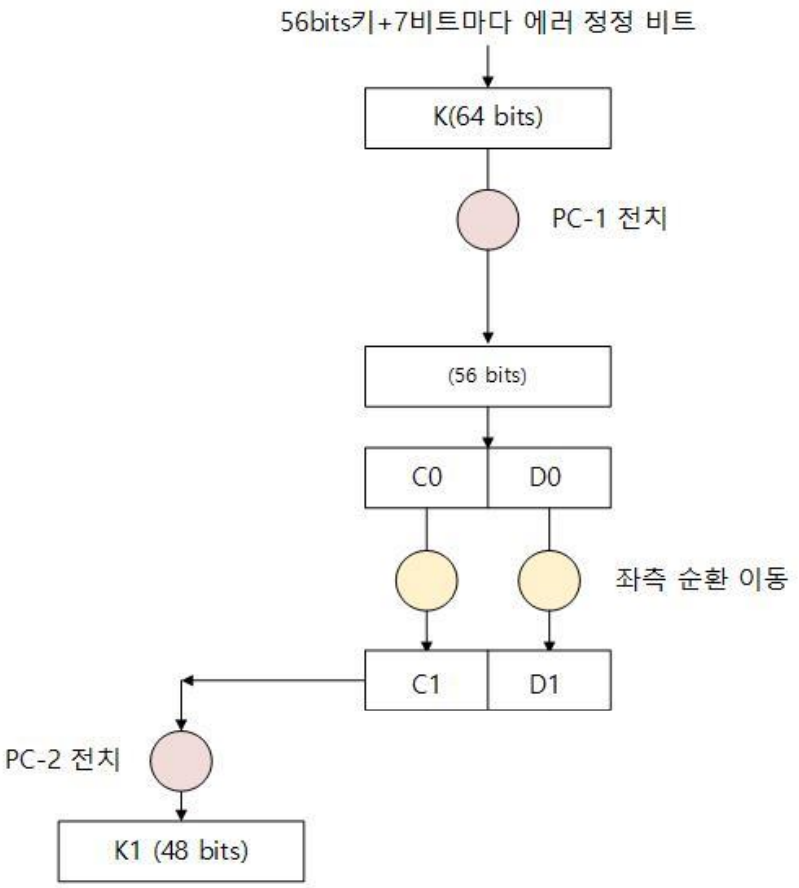


# 보충

- AES에서의 대체/치환
  - 한 라운드의 암호화 과정
    - 바이트 대체
      - S-box라는 표를 이용하여 바이트 단위로 블록을 교환한다
    - 행 이동
      - 행과 행을 치환한다
    - 열 섞기
      - 열에 속한 모든 바이트의 함수로서 열에 있는 각 바이트를 대체하여 변화시킨다
    - 라운드 키 더하기
      - 확장된 키와 현재 블록을 비트별로 XOR하여 대체한다
  - 대체, 치환이 둘 다 적용됨

# 보충

- 페이스텔 암호 구조에서의 Subkey 생성 과정



# 보충

- 페이스텔 구조에서의 Sub Key 생성과정(1/3)
  - 입력 받은 64비트의 마스터 키를 PC1 테이블을 통해 56비트의 새 키로 대체

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

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

# 보충

- 페이스텔 구조에서의 Sub Key 생성과정(2/3)
  - 56비트의 키를 28비트 씩 반으로 나누어 분리
  - 분리한 각 키 배열을 라운드의 수만큼 누적하여 좌측 순환 이동
    - 1, 2, 9, 16 라운드는 1번, 나머지는 2번

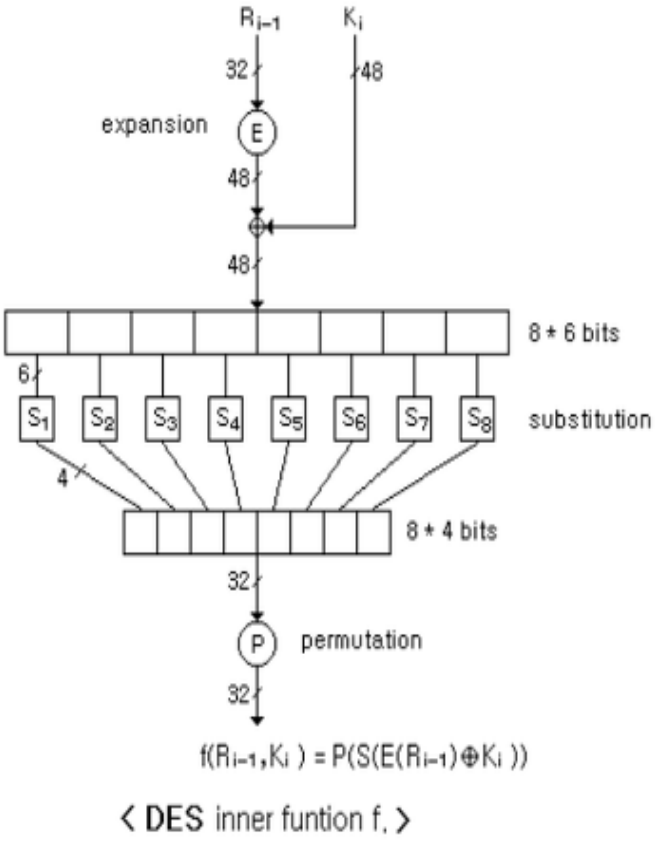
# 보충

- 페이스텔 구조에서의 Sub Key 생성과정(3/3)
  - 각 28비트를 합쳐서 다시 56비트의 키 배열로 만든 후, PC2 테이블을 이용하여 48비트의 새 키 배열을 만듦
  - 만들어진 키 배열이 해당 라운드의 서브키

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 14 | 17 | 11 | 24 | 1 | 5 | 3 | 28 |
| 15 | 6 | 21 | 10 | 23 | 19 | 12 | 4 |
| 26 | 8 | 16 | 7 | 27 | 20 | 13 | 2 |
| 41 | 52 | 31 | 37 | 47 | 55 | 30 | 40 |
| 51 | 45 | 33 | 48 | 44 | 49 | 39 | 56 |
| 34 | 53 | 46 | 42 | 50 | 36 | 29 | 32 |

# 보충

- 페이스텔 구조에서의 라운드 함수
  - 32비트의 입력 값이 48비트로 확장됨
  - 해당 라운드 키와 XOR 연산됨
  - 48비트가 6비트 씩 8개로 나뉨
  - 각 6비트가 S-box를 통해 각 4비트의 값으로 변함
  - 총 32비트의 출력 값으로 나옴



< DES inner funtion f, >

# 보충

- RC4 알고리즘
  - 정의
    - Ron Rivest가 설계한 바이트 단위로 작동되도록 만들어진 다양한 크기의 키를 사용하는 스트림 암호
  - 강도
    - 여러 논문에서 RC4를 공격하는 방법에 대해 분석하고 있지만, 어느 것도 실제로 공격하지 못함

# 보충

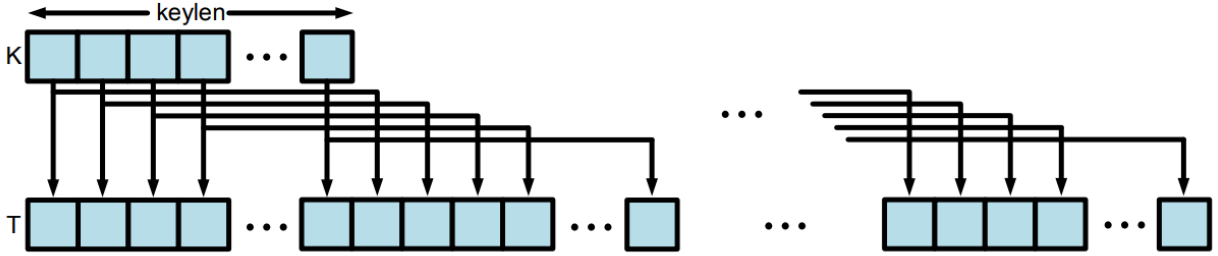
- RC4 알고리즘
  - 과정(1/4)
    - S[0], S[1],…,S[255]를 원소로 갖는 256바이트의 상태 벡터 S를 0부터 255까지 오름차순으로 정렬

```
/*Initialization*/
for i=0 to 255 do
 S[i] = i;
```

# 보충

- RC4 알고리즘
  - 과정(2/4)
    - 키 K의 길이(keylen)만큼 K를 임시 벡터 T에 이동하고, keylen<256이면 T를 채울 때까지 반복하여 K를 T에 복사

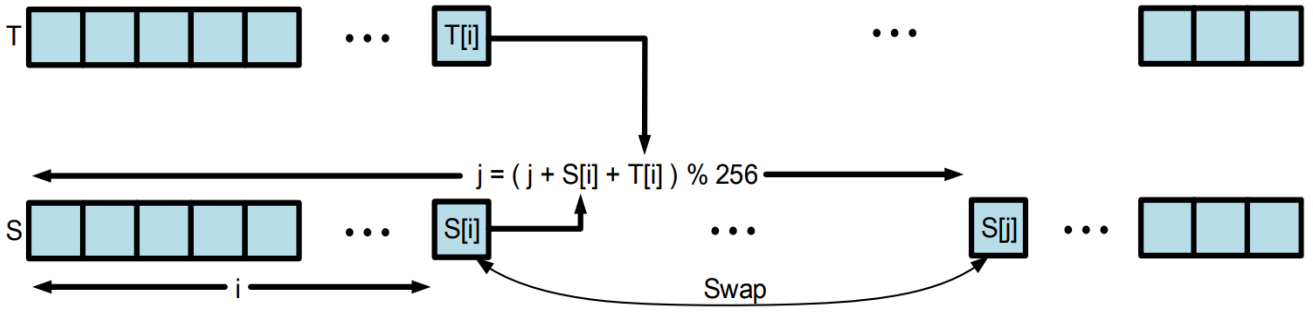
```
/*Initialization*/
for i=0 to 255 do
 S[i] = i;
 T[i] = K[i mod keylen];
```



# 보충

- RC4 알고리즘
  - 과정(3/4)
    - 각 S[i]를 T[i]를 이용한 구조에 따라서 S의 다른 바이트와 교환

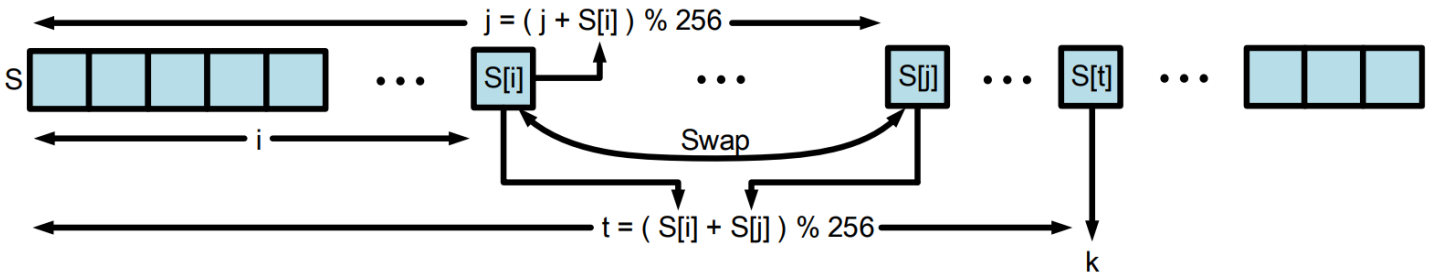
```
/*Initial Permutation of S*/
j=0;
for i = 0 to 255 do
 j=(j+S[i]+T[i]) mod 256;
 Swap (S[i], S[j]);
```



# 보충

- RC4 알고리즘
  - 과정(4/4)
    - S[0]에서 처음부터 S의 원소를 Swap하며 S[i]와 S[j]의 구조에 따라 생성된 키 k를 평문의 다음 바이트와 XOR

```
/*Stream Generation*/
i, j=0;
while (true)
 i = (i+1) mod 256;
 j = (j+S[i]) mod 256;
 Swap (S[i], S[j]);
 t = (S[i]+S[j]) mod 256;
 k = S[t];
```



# 보충

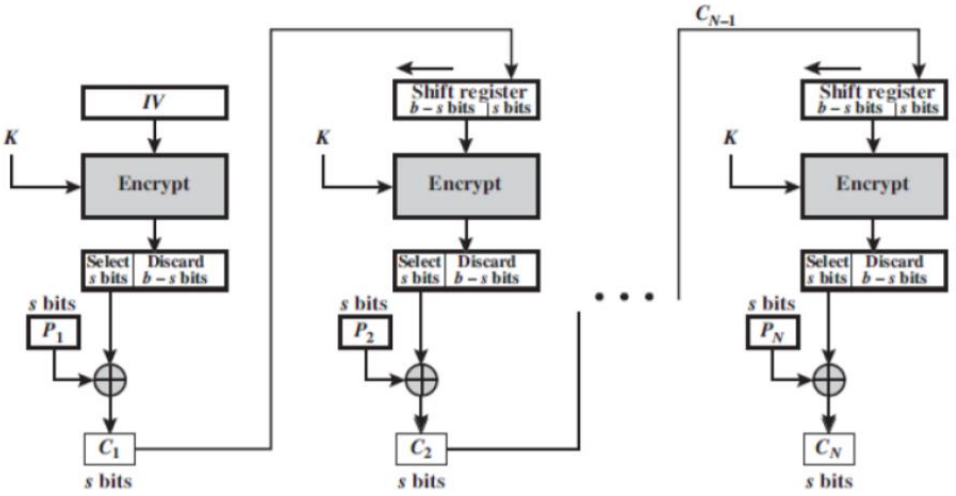
- 암호 블록 운용 모드
  - 정의
    - 블록 암호를 다양하게 응용하기 위해 NIST에서 정의한 5가지 운용모드
  - 종류
    - 전자 코드북 모드(ECB: Electronic Codebook Mode)
    - 암호 블록 체인 모드(CBC: Cipher Block Chaining Mode)
    - 암호 피드백 모드 (CFB: Cipher Feedback Mode)
    - 출력 피드백 모드 (OFB: Output Feedback Mode)
    - 카운터 모드(CTR: Counter Mode)

# 보충

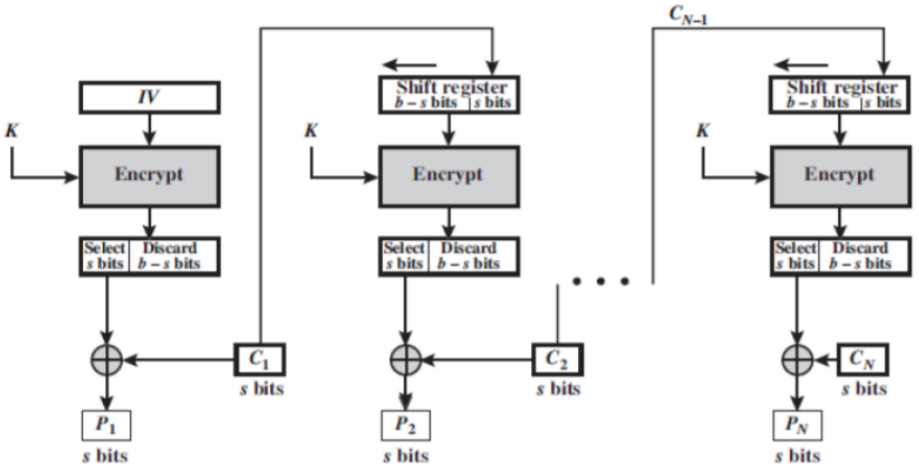
- 암호 블록 운용 모드
  - ECB(Electronic Codebook) 모드
    - 평문을 같은 크기의 블록으로 나누고 각 블록을 동일한 키로 암호화하는 방식
  - CBC(Cipher Block Chaining) 모드
    - 현재의 평문 블록과 바로 직전의 암호 블록을 XOR한 결과를 알고리즘의 입력으로 사용하는 방식

# 보충

- 암호 블록 운용 모드
  - CFB(Cipher Feedback) 모드
    - 어떤 블록 암호도 스트림 암호로 바꿀 수 있는 방식



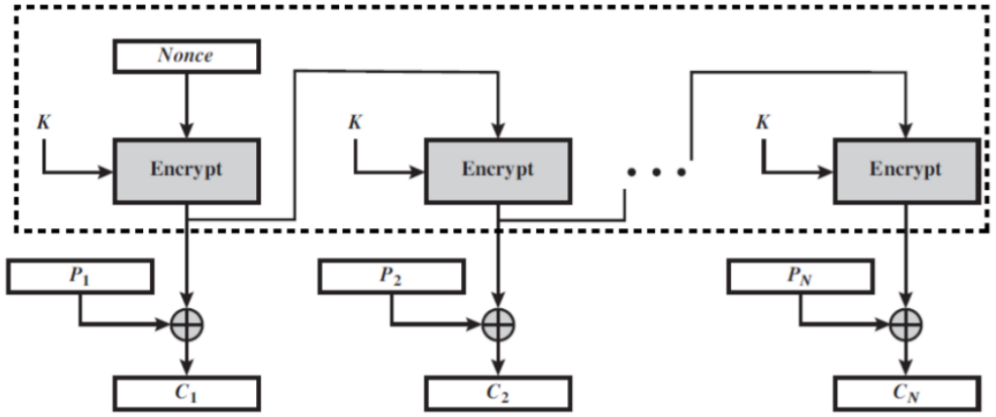

암호화



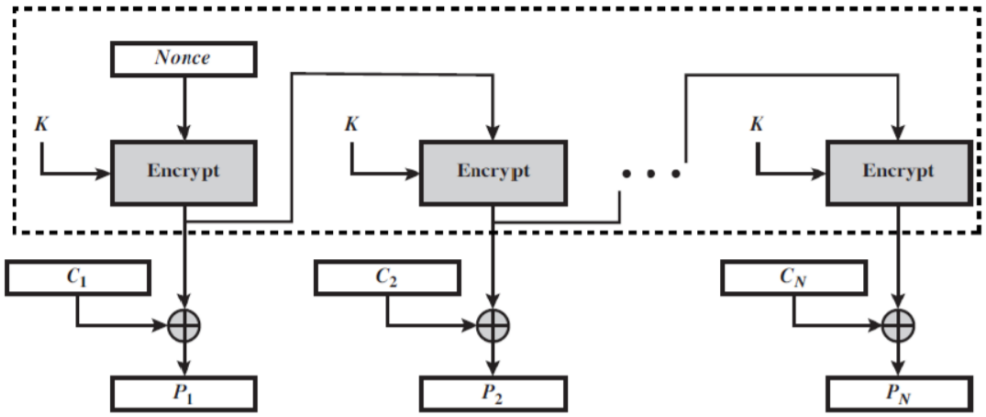

복호화

# 보충

- 암호 블록 운용 모드
  - OFB(Output Feedback) 모드
    - 어떤 블록 암호도 스트림 암호로 바꿀 수 있는 방식이며, CFB의 변형 방식



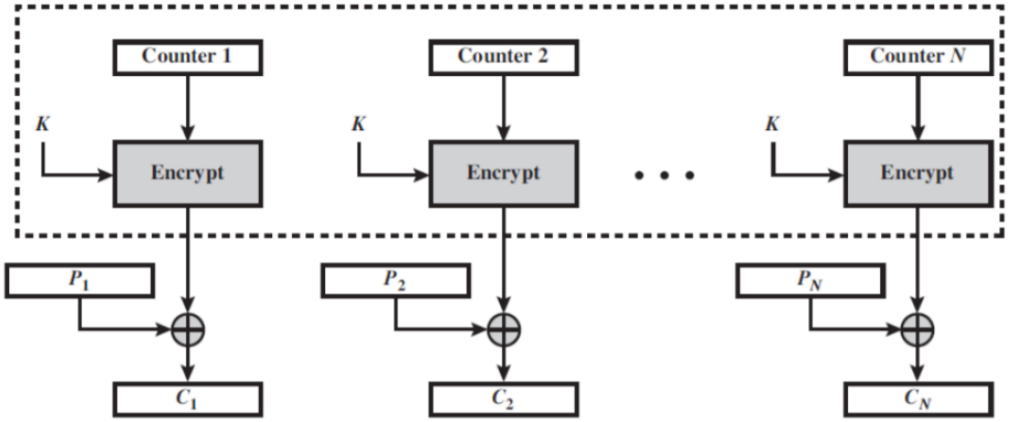

암호화



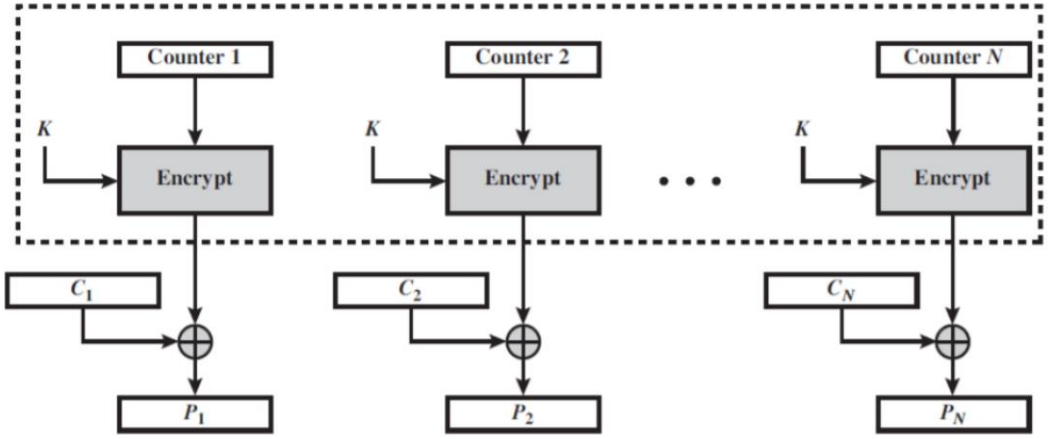

복호화

# 보충

- 암호 블록 운용 모드
  - CTR(Counter) 모드
    - 각 평문 블록마다 +1씩 커지는 카운터를 암호화하여 평문 블록과 XOR하는 방식




암호화




복호화

# 목 차

- 보충
  - DES, AES에서의 대체/치환
  - 페이스텔 구조에서의 Sub Key 생성과정
  - 페이스텔 구조에서의 F함수
  - 스트림 암호와 RC4
  - 암호 블록 운용 모드
- 메시지 인증 방법
- 안전 해시 함수
- 메시지 인증 코드

# 메시지 인증 방법

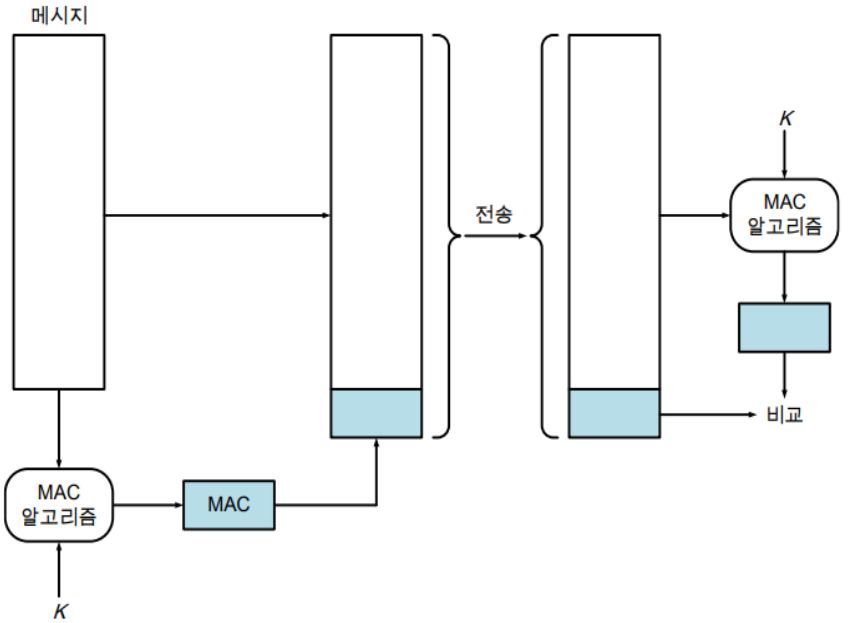
- 메시지 인증
  - 정의
    - 통신 양측으로 하여금 메시지의 내용이 전송 도중 불법적으로 변경되지 않고 안전하게 전송됨을 보증하는 것
  - 두가지 중요한 측면
    - 메시지 내용이 변경되지 않음을 확인하는 것
    - 송신자를 확인하는 것

# 메시지 인증 방법

- 메시지 인증
  - 인증 방법
    - 관용 암호를 이용하여 인증하기
      - 송수신자가 공유하는 비밀 키를 이용해 메시지를 암/복호화 할 수 있음
    - 메시지 암호화 없이 메시지 인증하기
      - 인증 꼬리표를 생성하여 메시지에 붙여서 보내기
      - 적합한 세 가지 경우
        - 동일한 메시지를 다수의 수신자에게 브로드캐스팅할 때
        - 메시지 교환 시 송수신자 중 한 쪽에 부하가 걸려 메시지 복호화가 힘들 때
        - 컴퓨터 프로그램을 평문인 채로 인증하려 할 때

# 메시지 인증 방법

- 메시지 암호화 없는 메시지 인증
  - 메시지 인증 코드
    1) 송수신자는 공통 비밀 키 $K_{AB}$ 를 가짐
    2) 송신자는 메시지와 키를 이용해 메시지 인증 코드를 계산 $[MAC_M = F(K_{AB}, M)]$
    3) 송신자는 메시지에 인증 코드를 붙여 수신자에게 전달
    4) 수신자는 동일한 방법으로 인증 코드를 계산한 뒤, 수신된 코드와 같은지 비교

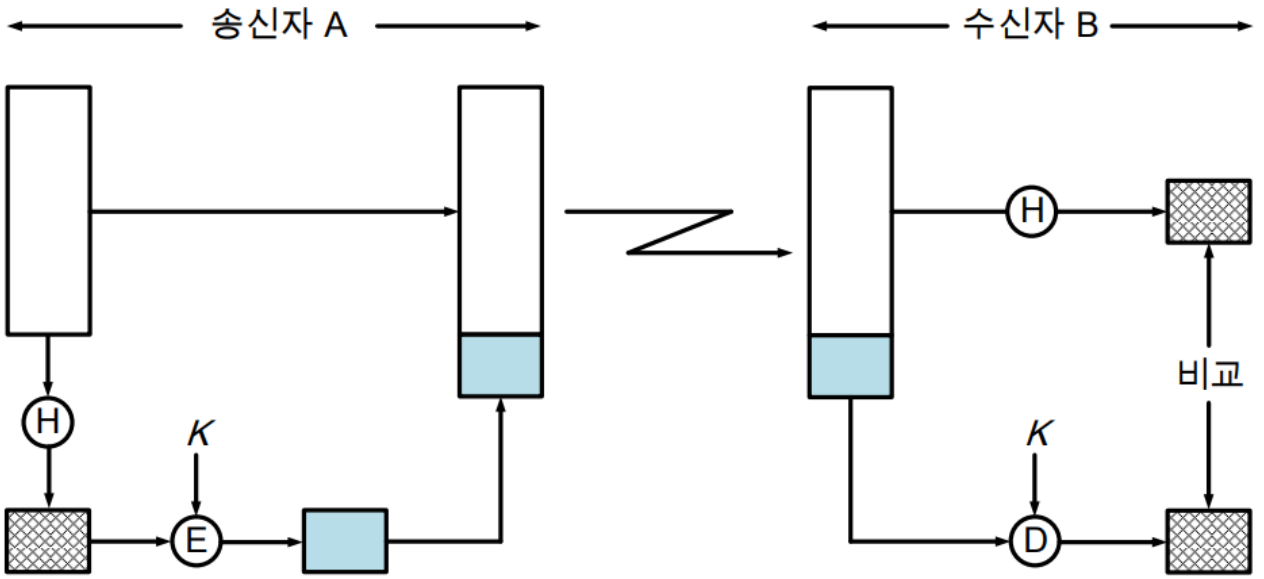


# 메시지 인증 방법

- 메시지 암호화 없는 메시지 인증
  - 일방향 해시 함수
    - 정의
      - 임의 크기의 메시지를 입력으로 받아들여 일정한 크기의 메시지 다이제스트를 출력하는 함수
    - 특징
      - 비밀 키를 입력으로 사용하지 않음
      - 인증된 상태의 메시지 다이제스트를 메시지와 함께 전송함
    - 세가지 인증 방법
      - 관용 암호를 사용한 인증
      - 공개 키 암호를 사용한 인증
      - 비밀 값을 사용한 인증

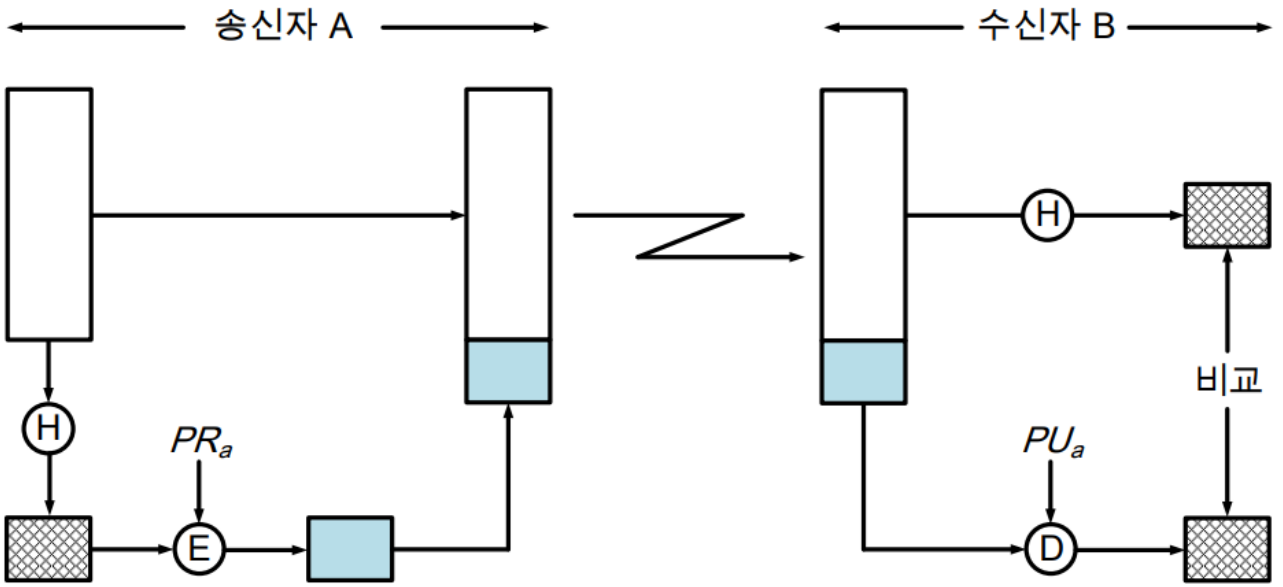
# 메시지 인증 방법

- 메시지 암호화 없는 메시지 인증
  - 일방향 해시 함수
    - 관용 암호를 사용한 인증
      - 송신자는 메시지를 해시 함수에 입력하여 나온 출력 값을 비밀키로 암호화하여 인증 코드를 만들고 메시지와 함께 수신자에게 전송
      - 수신자는 송신자와 같은 계산을 통해 계산된 인증 코드와 수신된 인증 코드가 같은지 확인



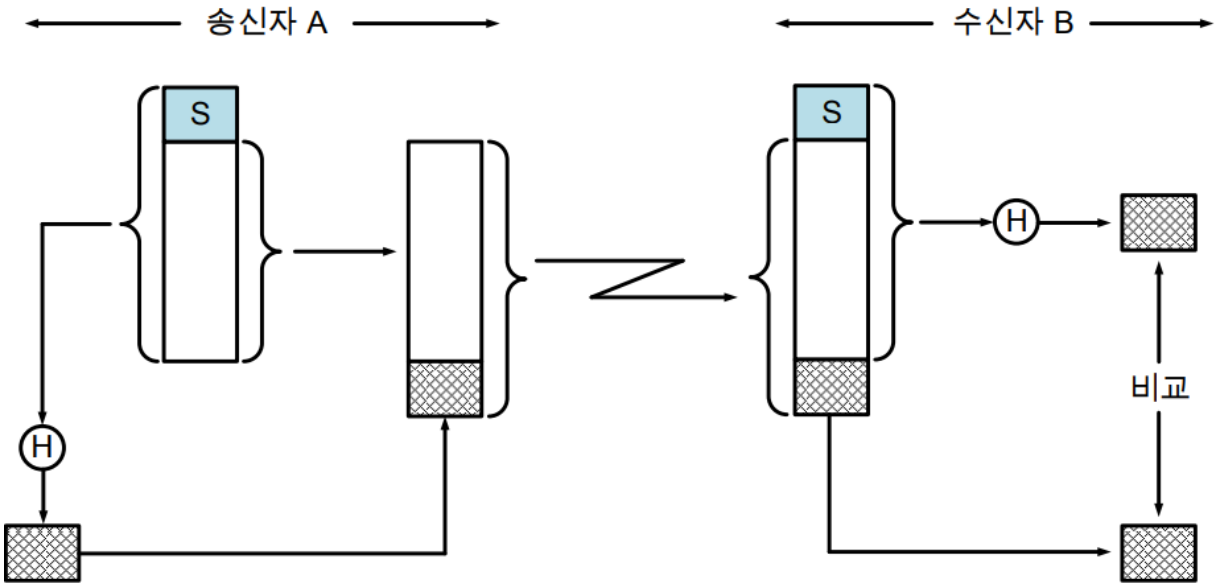
# 메시지 인증 방법

- 메시지 암호화 없는 메시지 인증
  - 일방향 해시 함수
    - 공개 키 암호를 사용한 인증
      - 송신자는 메시지를 해시 함수에 입력하여 나온 출력값을 자신의 키로 암호화하여 인증 코드를 만들고 메시지와 함께 수신자에게 전송
      - 수신자는 자신의 키를 이용하여 송신자와 같은 방법을 통해 계산된 인증 코드와 수신된 인증 코드가 같은지 확인



# 메시지 인증 방법

- 메시지 암호화 없는 메시지 인증
  - 일방향 해시 함수
    - 비밀 값을 사용한 인증
      - 송수신자는 공통 비밀 값 $S_{AB}$를 가짐
      - 송신자는 비밀 값에 메시지를 붙인 뒤에 해시 함수를 적용하여 얻은 인증 코드를 메시지와 함께 수신자에게 전송
      - 수신자는 송신자와 같은 방법으로 계산한 인증 코드와 수신된 인증 코드가 같은지 비교



# 메시지 인증 방법

- 암호화를 전혀 하지 않는 기술이 필요한 이유
  - 암호 소프트웨어는 속도가 매우 느림
  - 암호 장비 값이 저렴하지 않음
  - 암호 장비를 사용하면 상당 시간이 소비됨
  - 암호 알고리즘 수출이 금지되어 있을 수 있음

# 목 차

- 보충
  - DES, AES에서의 대체/치환
  - 페이스텔 구조에서의 Sub Key 생성과정
  - 페이스텔 구조에서의 F함수
  - 스트림 암호와 RC4
  - 암호 블록 운용 모드
- 메시지 인증 방법
- 안전 해시 함수
- 메시지 인증 코드

# 안전 해시 함수

- 해시 함수
  - 정의
    - 특정 입력 데이터에서 고정 길이 값 또는 해시 값을 생성하는 알고리즘
  - 해시 함수가 되기 위한 조건
    - 어떠한 크기의 데이터 블록에도 적용될 수 있어야 함
    - 일정한 길이의 출력을 생성해야 함
    - 계산이 쉬워야 하고, 하드웨어적이나 소프트웨어적으로 구현을 실제로 할 수 있어야 함
    - $H(x) = h$가 성립되는 $x$ 를 찾는 것이 계산적으로 불가능해야 함(일방향 성질)
    - $H(x) = H(y)$를 만족하는 $x \neq y$를 찾는 것이 계산적으로 불가능해야 함 (약한 충돌 저항성)
    - $H(x) = H(y)$를 만족하는 쌍 $(x, y)$를 찾는 것이 계산적으로 불가능해야 함 (강한 충돌 저항성)

# 안전 해시 함수

- 해시 함수 보안
  - 길이가 n비트인 해시 코드에 대한 공격 난이도

| 일방향 성질 | $2^n$ |
|---|---|
| 약한 충돌 저항성 | $2^n$ |
| 강한 충돌 저항성 | $2^{n/2}$ |

  - 해시 코드 길이가 128비트인 해시 함수에서, 24시간 만에 충돌이 발견되어 안전하지 않음이 드러남
  - 현 시점에서 256, 384, 512비트 길이의 해시 함수가 가장 많이 쓰임

# 안전 해시 함수

- 단순 해시 함수
  - 입력은 연속된 n비트 블록으로 간주
  - 각 비트의 자리별로 패리티를 계산하는 세로 덧붙임 검사를 함
  - $C_i = b_{i1} \oplus b_{i2} \oplus \bullet\bullet\bullet \oplus b_{im}$

| | 비트 1 | 비트 2 | ••• | 비트 n |
|---|---|---|---|---|
| 블록 1 | $b_{11}$ | $b_{21}$ | | $b_{n1}$ |
| 블록 2 | $b_{12}$ | $b_{22}$ | | $b_{n2}$ |
| • | • | • | • | • |
| • | • | • | • | • |
| • | • | • | • | • |
| 블록 m | $b_{1m}$ | $B_{2m}$ | | $b_{nm}$ |
| 해시 코드 | $C_1$ | $C_2$ | ••• | $C_n$ |

$C_i$ = 해시 코드의 i번째 비트
$m$ = 입력의 n 비트 블록의 수
$b_{ij}$ = j번째 블록의 i번째 비트

# 안전 해시 함수

- 안전 해시 알고리즘(SHA: Secure Hash Algorithm)
  - 정의
    - NIST가 개발한 암호학적 해시 함수들의 모음
  - 종류
    - SHA-1
    - SHA-2
      - SHA-224
      - SHA-256
      - SHA-384
      - SHA-512
    - SHA-3

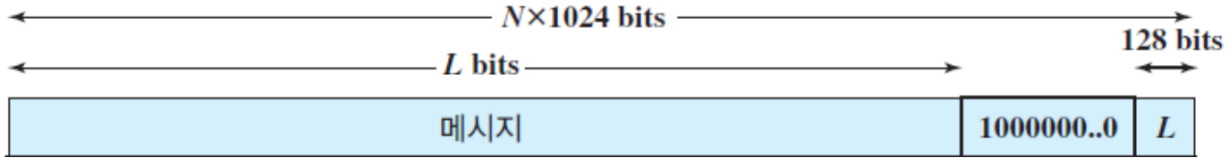
# 안전 해시 함수

- 안전 해시 알고리즘(SHA: Secure Hash Algorithm)
  - SHA 매개 변수 비교

| | SHA-1 | SHA-224 | SHA-256 | SHA-384 | SHA-512 |
|---|---|---|---|---|---|
| 메시지 다이제스트 비트 길이 | 160 | 224 | 256 | 384 | 512 |
| 메시지 비트 길이 | $< 2^{64}$ | $< 2^{64}$ | $< 2^{64}$ | $< 2^{128}$ | $< 2^{128}$ |
| 블록 비트 길이 | 512 | 512 | 512 | 1024 | 1024 |
| 단어 비트 길이 | 32 | 32 | 32 | 64 | 64 |
| 단계 수 | 80 | 64 | 64 | 80 | 80 |

# 안전 해시 함수

- 안전 해시 알고리즘(SHA: Secure Hash Algorithm)
  - SHA-512 알고리즘의 동작 과정(1/6)
    - 패딩 비트 붙이기(Appending Padding Bits)
      - 총 길이를 896 (mod 1024)가 되도록 메시지 패딩을 추가
      - 패딩을 구성하는 비트는 첫번째 비트가 1, 나머지는 0
  - SHA-512 알고리즘의 동작 과정(2/6)
    - 길이 붙이기(Append length)
      - 128비트 블록을 메시지에 추가하여 전체 메시지의 길이를 1024의 배수가 되게 만듦
        - 원래 메시지의 길이를 포함하는 블록



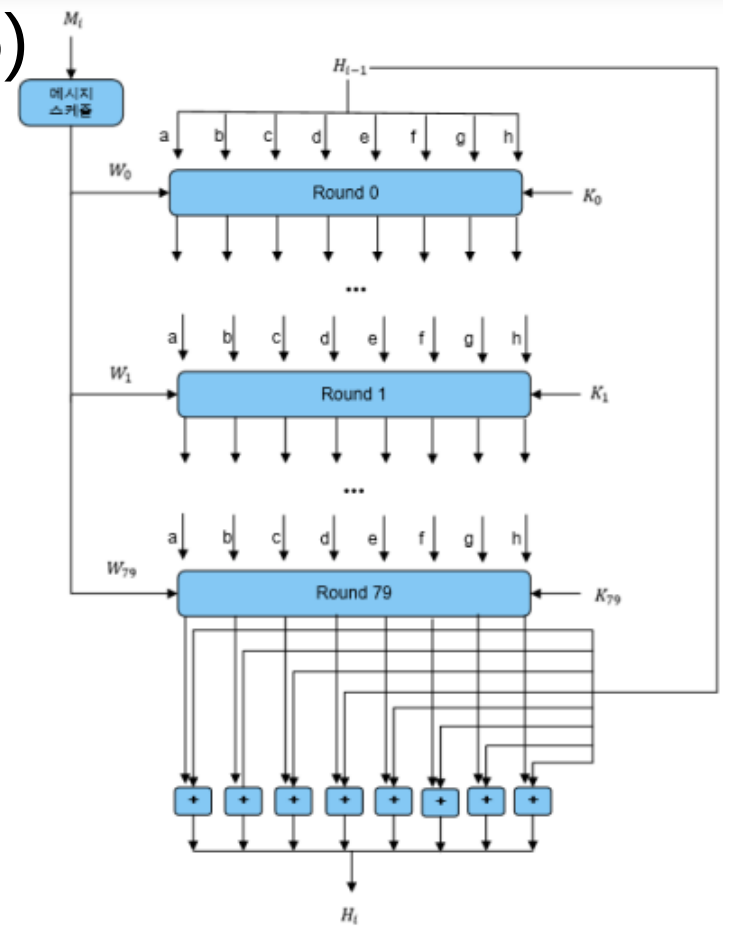
# 안전 해시 함수

- 안전 해시 알고리즘(SHA: Secure Hash Algorithm)
  - SHA-512 알고리즘의 동작 과정(3/6)
    - MD 버퍼 초기화(Initialize MD Buffer)
      - 64비트 버퍼 8개를 다음 64비트 정수로 초기화

| | |
|---|---|
| a = 6A09E667F3BCC908 | e = 510E527FADE682D1 |
| b = BB67AE8584CAA73B | f = 9B05688CEB3E6C1F |
| c = 3C6EF372FE94F82B | g = 1F83D9ABFB41BD6B |
| d = A54FF53A5F1D36F1 | h = 5BE0CDI9137E2179 |

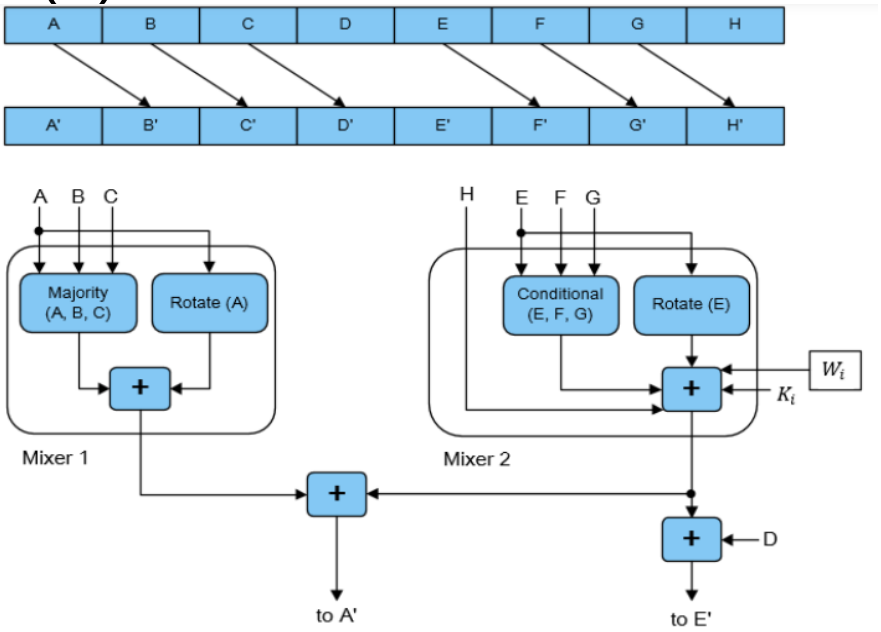
# 안전 해시 함수

- 안전 해시 알고리즘(SHA: Secure Hash Algorithm)
  - SHA-512 알고리즘의 동작 과정(4/6)
    - 1024 비트 블록 메시지 처리 (Process Message in 1024 Bit)
      - 첫번째 라운드에서 입력 시, 버퍼는 $H_{i-1}$값을 갖게 됨
      - 각각의 라운드 t에서는 그 순간에 처리되는 1024 비트 블록인 $M_i$ 로부터 구한 64 비트 값인 $W_t$와 64비트 덧셈상수 $K_t$와 직전 8개의 64비트 버퍼 값을 라운드 함수에 입력하여 버퍼의 내용을 갱신
      - 이 라운드가 80번 반복됨
      - 마지막 라운드가 끝난 후 출력 블록을 라운드가 시작되기 전의 입력 블록과 모듈로 덧셈을 진행하고 512비트의 출력을 내보냄



# 안전 해시 함수

- 안전 해시 알고리즘(SHA: Secure Hash Algorithm)
  - SHA-512 알고리즘의 동작 과정(5/6)
    - 라운드 함수
      - B', C', D'블록은 각각 A, B, C블록의 값을 대입
      - F', G', H'블록은 각각 E, F, G블록의 값을 대입
      - Mixer1 = Majority(A, B, C)+Rotate(A)
      - Mixer2 = Majority(E, F, G)+Rotate(E)+H
      - A'=Mixer1+Mixer2
      - E'=Mixer2+D




Majority (x, y, z)

(x AND y) ⊕ (y AND z) ⊕ (z AND x)

Conditional (x, y, z)

(x AND y) ⊕ (NOT x AND z)

Rotate (A)

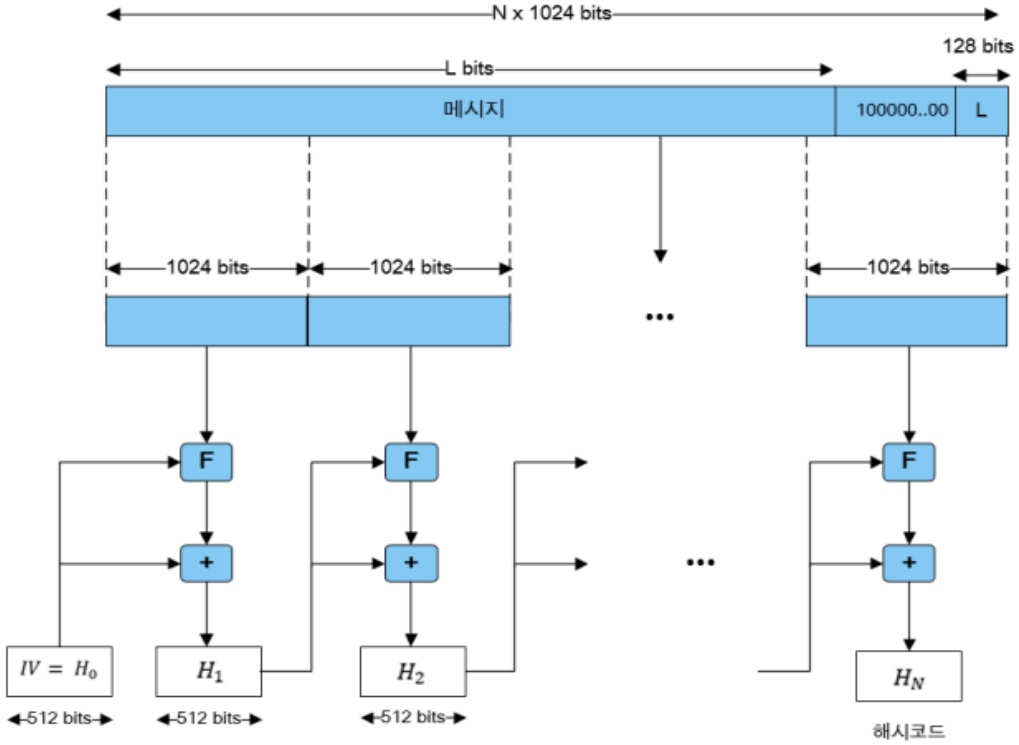
$ROTR_{28}(x) \oplus ROTR_{34}(x) \oplus ROTR_{39}(x)$

+ : mod $2^{64}$ 로 워드 별 합산

$ROTR_i(x)$ : X비트 만큼 오른쪽으로 shift

# 안전 해시 함수

- 안전 해시 알고리즘(SHA: Secure Hash Algorithm)
  - SHA-512 알고리즘의 동작 과정(6/6)
    - 출력(Output)
      - N개의 1024 비트 블록 모두가 처리된 뒤에 N번째 단계에서 512 비트 메시지 다이제스트를 얻음



# 목 차

- 보충
  - DES, AES에서의 대체/치환
  - 페이스텔 구조에서의 Sub Key 생성과정
  - 페이스텔 구조에서의 F함수
  - 스트림 암호와 RC4
  - 암호 블록 운용 모드
- 메시지 인증 방법
- 안전 해시 함수
- 메시지 인증 코드

# 메시지 인증 코드

- HMAC(Hashed Mac)
  - 정의
    - 해시 함수와 비밀 키를 수반하는 메시지 인증 코드
  - 설계 목표
    - 소프트웨어에서 잘 실행되게 함
    - 무료로 코드를 제공할 수 있게 함
    - 기존의 해시 함수를 쉽게 바꿀 수 있게 함
    - 기능 저하를 유발하지 않게 함
    - 키를 보다 쉽게 다루고자 함

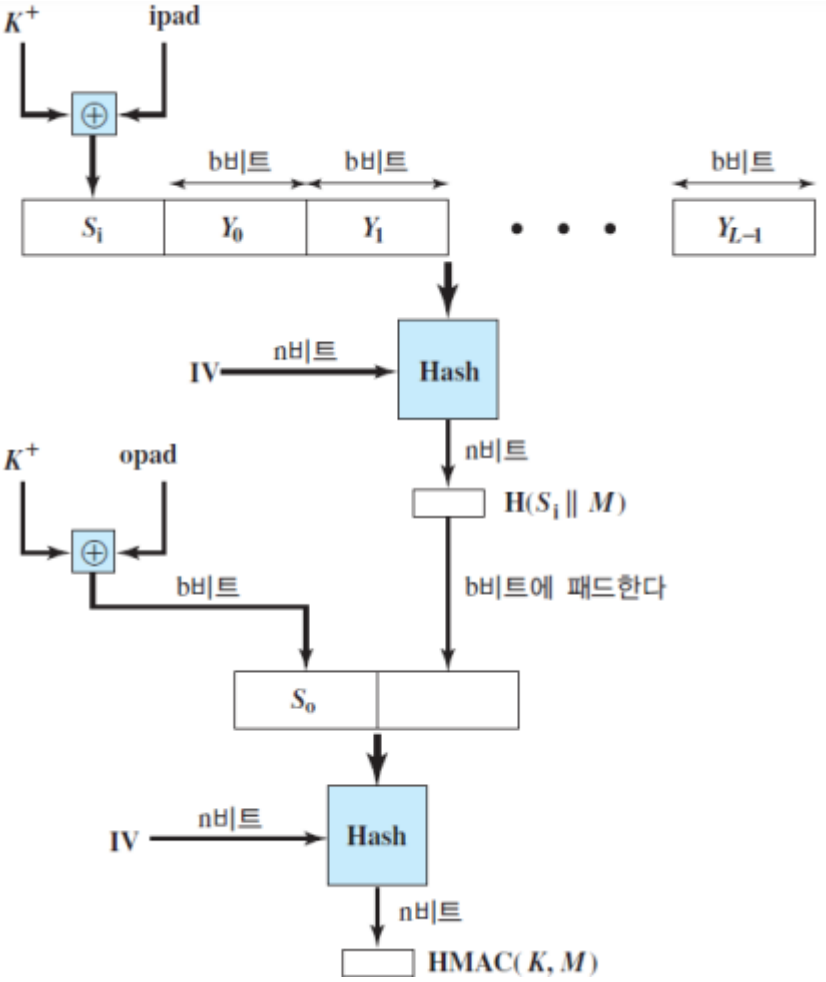
# 메시지 인증 코드

- HMAC(Hashed Mac)
  - 용어

| 용어 | 뜻 |
|---|---|
| H | 해시 함수 |
| M | HMAC의 입력 메시지 |
| $Y_i$ | M의 i번째 블록 |
| L | M의 블록 수 |
| b | 블록의 비트 수 |
| n | 내장된 해시 함수에 의해 생성된 해시 코드의 길이 |
| K | 비밀 키 |
| $K^+$ | K의 왼쪽에 0을 붙여서 길이가 b비트가 되도록 한 것 |
| ipad | 00110110를 b/8번 반복한 2진 수열 |
| opad | 01011100를 b/8번 반복한 2진 수열 |

# 메시지 인증 코드

- HMAC(Hashed Mac)
  - 동작 과정
    1) b비트 스트링 $K^+$를 만들기 위해 K의 왼쪽에 0을 붙임
    2) b비트 블록 $S_i$를 생성하기 위해 $K^+$ 와 ipad를 XOR
    3) $S_i$에 M을 붙임
    4) 3단계에서 생성된 스트림에 해시함수를 적용
    5) b 비트 블록 $S_0$를 생성하기 위해 $K^+$ 와 opad를 XOR
    6) 4단계에서 얻은 해시 결과를 $S_0$에 붙임
    7) 6단계에서 생성된 스트림에 해시함수를 적용해서 출력



# 메시지 인증 코드

- 암호기반 메시지 인증코드 (CMAC: Cipher based Mac)
  - 정의
    - 블록 암호 기반 메시지 인증 코드
  - 운용 모드
    - AES
      - 키 길이는 128, 192, 또는 256비트
      - 블록 길이는 128비트
    - 3DES
      - 키 길이는 112 또는 168비트
      - 블록 길이는 64비트

# 메시지 인증 코드

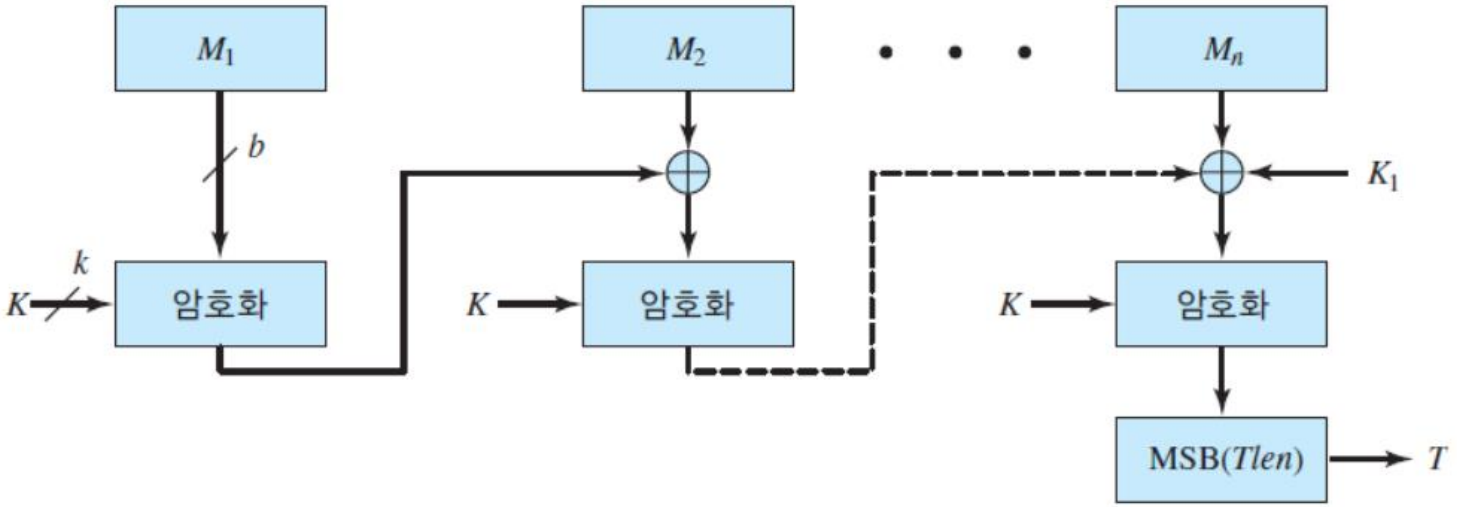
- 암호기반 메시지 인증코드 CMAC
  - 계산식

$$C_1 = \mathrm{E}(\mathrm{K}, M_1)$$
$$C_2 = \mathrm{E}(\mathrm{K}, [M_2 \oplus C_1])$$
$$C_3 = \mathrm{E}(\mathrm{K}, [M_3 \oplus C_2])$$
$$\vdots$$
$$C_n = \mathrm{E}(\mathrm{K}, [M_N \oplus C_{n-1} \oplus K_1])$$
$$T = MSB_{T_{len}}(C_n)$$

| 용어 | 뜻 |
|---|---|
| T | 메시지 인증 코드, 또는 태그(tag)라고 함 |
| $T_{len}$ | T의 비트 길이 |
| $MSB_s(X)$ | 비트열 X의 왼쪽부터 s개 비트 |
| K | k비트 암호키 |
| $K_1$ | 결과로 나온 암호문을 한 비트 왼쪽으로 이동한 것 |
| $K_2$ | $K_1$를 한 비트 왼쪽으로 이동한 것 |

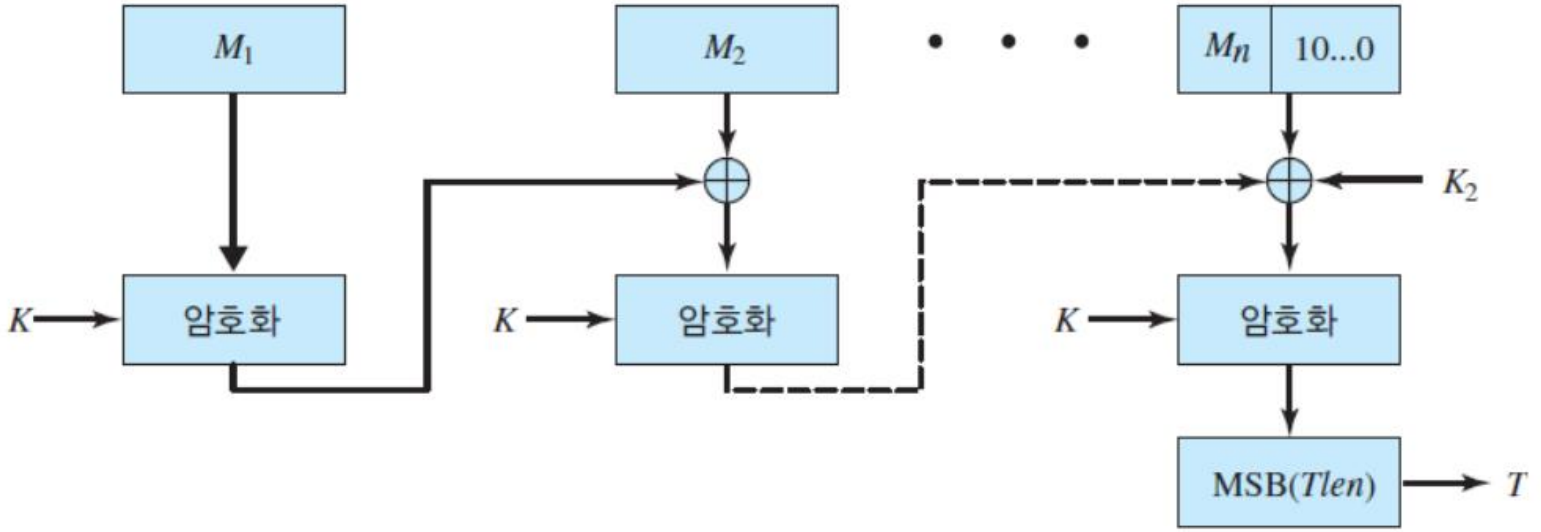
# 메시지 인증 코드

- 암호기반 메시지 인증코드 CMAC
  - 동작 과정
    - 메시지 길이가 블록 길이의 정수배일 때
      - k비트 암호 키 K와 b비트 키 $K_1$를 사용



# 메시지 인증 코드

- 암호기반 메시지 인증코드 CMAC
  - 동작 과정
    - 메시지 길이가 블록 길이의 정수배가 아닐 때
      - 마지막 평문 블록의 오른쪽에 패딩을 붙여서 블록 길이를 b비트로 만듦
      - k비트 암호 키 K와 b비트 키 $K_2$를 사용

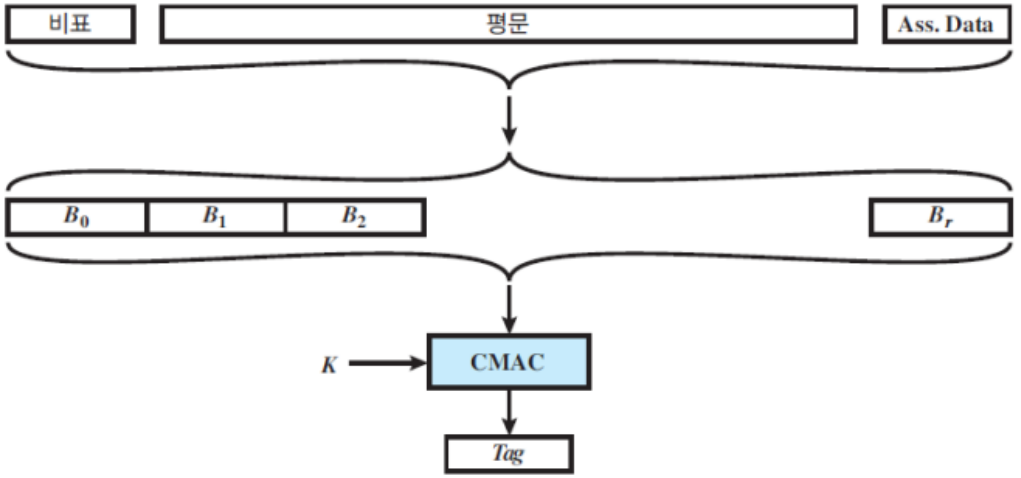


# 메시지 인증 코드

- 암호 블록 체인 카운터-메시지 인증 코드 CCM(Counter With Cipher Block Chaining MAC)
  - 정의
    - 하나의 키 아래에서 블록 암호를 반복적으로 안전하게 이용하게 하는 절차
  - 핵심적인 알고리즘 요소
    - AES 암호 알고리즘
    - CTR 운용 모드
    - CMAC 인증 알고리즘

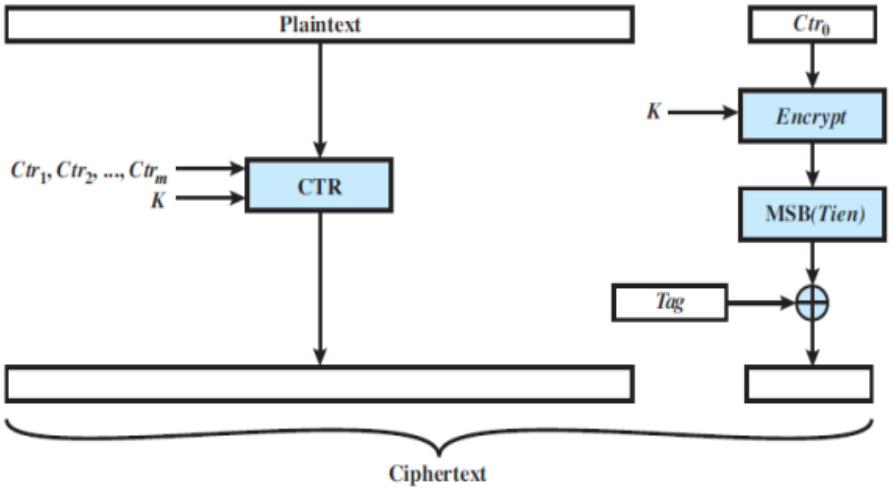
# 메시지 인증 코드

- 암호 블록 체인 카운터-메시지 인증 코드 CCM(Counter With Cipher Block Chaining MAC)
  - 인증 과정
    - 비표(N), 유관 데이터(A), 평문(P)를 입력
    - 입력 값을 $B_0$부터 $B_r$까지 블록 열 형식으로 나타냄
      - 첫 번째 블록에는 N, A와 P의 길이를 나타내는 형식 비트가 추가됨
    - 블록 열을 CMAC 알고리즘의 입력값으로 사용



# 메시지 인증 코드

- 암호 블록 체인 카운터-메시지 인증 코드 CCM(Counter With Cipher Block Chaining MAC)
  - 암호화 과정
    - 비표와 독립적으로 카운터 열 생성
    - 인증 태그를 $Ctr_0$를 이용하여 CTR모드로 암호화
    - 출력된 비트 중 Tlen개의 유효 비트를 태그와 XOR해서 암호화된 태그 생성
    - 나머지 카운터는 평문을 CTR 모드로 암호화 할 때 사용
    - 암호화된 평문을 암호화된 태그와 이어 붙여서 암호문으로 출력



암호화

# Thanks!

강 민 채 (minchae@pel.sejong.ac.kr)