2023/02/02, 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

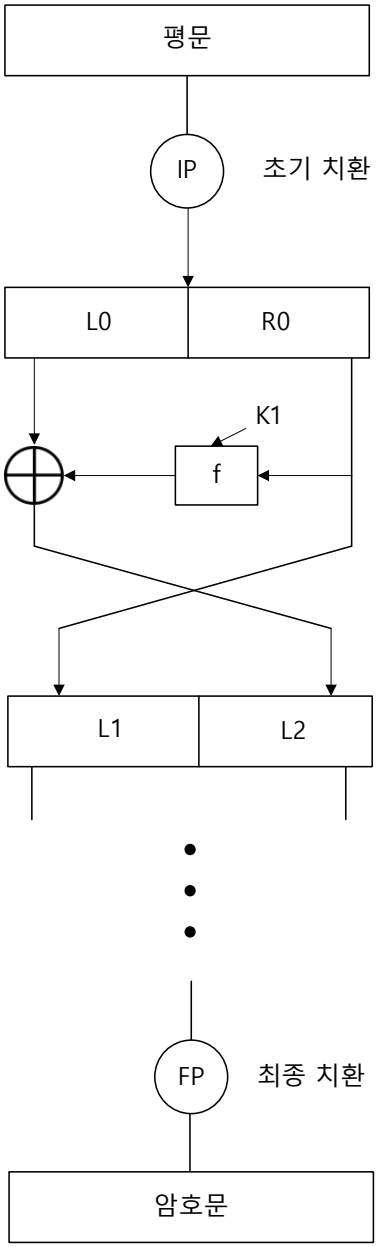
- 보충
  - DES, AES에서의 대체/치환
  - 페이스텔 구조에서의 Sub Key 생성과정
  - 페이스텔 구조에서의 F함수
  - 스트림 암호와 RC4
  - 암호 블록 운용 모드
- 메시지 인증 방법
- 해시 함수
- 메시지 인증 코드

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
- 해시 함수
- 메시지 인증 코드

보충

- DES에서의 대체/치환
  - 대체
    - 한 라운드에서 왼쪽 데이터를 오른쪽 데이터에 라운드 함수를 적용한 것과 XOR한 것으로 대체
  - 치환
    - 초기 치환(Initial Permutation)
    - 한 라운드에서 양쪽 데이터를 치환
    - 최종 치환(Final Permutation)

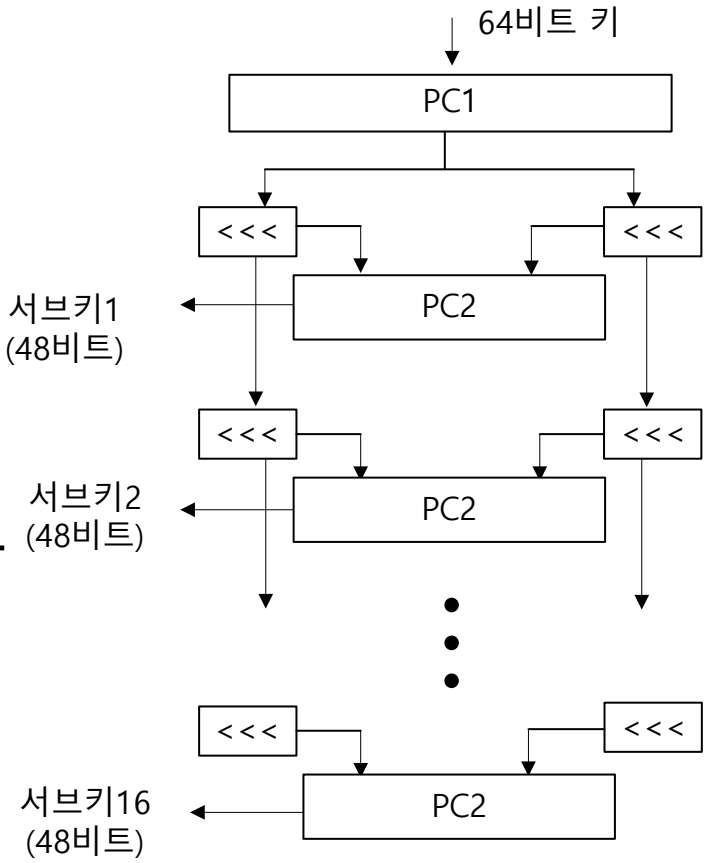


보충

- AES에서의 대체/치환
  - 대체
    - 바이트 대체
      - S-box라는 표를 이용하여 바이트 단위로 블록을 대체
    - 열 섞기
      - 열에 있는 각 바이트를 고정행렬을 사용하여 대체
    - 라운드 키 더하기
      - 확장된 키와 현재 블록을 비트별로 XOR하여 대체
  - 치환
    - 행 이동
      - 행과 행을 이동(shift)하여 치환

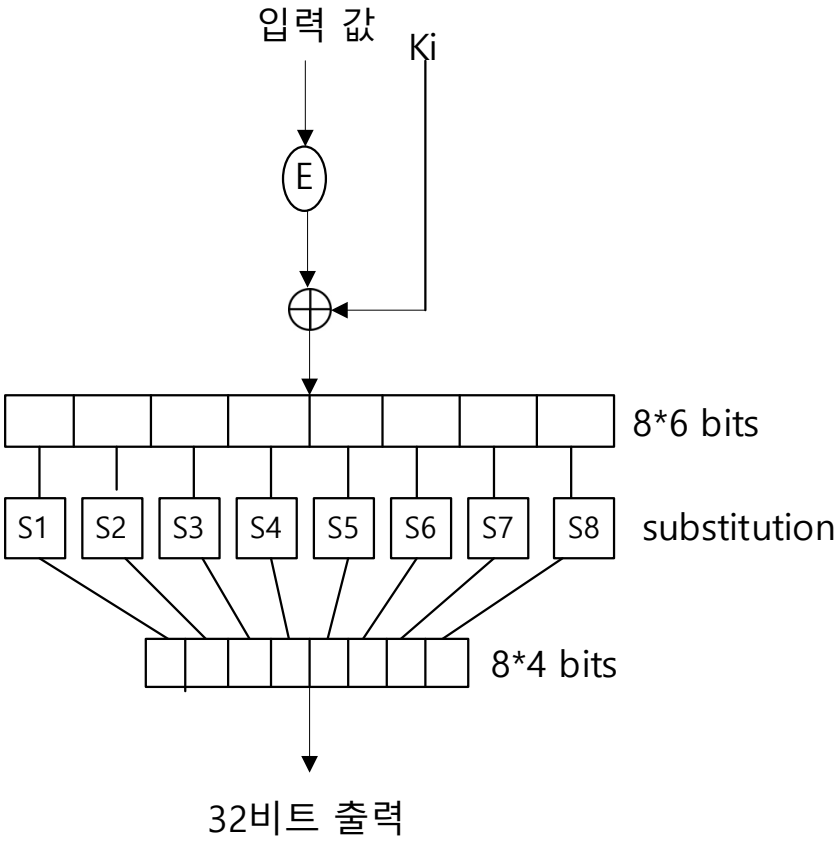
보충

- 페이스텔 암호 구조에서의 Subkey 생성 과정
  - 입력 받은 64비트의 마스터 키를 패리티 비트 제거 후, PC1 테이블을 통해 56비트의 새 키로 대체
  - 56비트의 키를 28비트씩 반으로 나누어 분리
  - 분리한 각 키 배열을 라운드의 수만큼 누적하여 좌측 순환 이동
    - 1,2,9,16 라운드는 1번, 나머지는 2번
  - 각 28비트를 합친 후 PC2 테이블을 이용하여 48비트의 새 키 배열 생성



보충

- 페이스텔 구조에서의 라운드 함수
  - 32비트의 입력 값이 확장 순열 (Expansion P-Box)을 통해서 48비트로 확장됨
  - 해당 라운드 키와 XOR 연산됨
  - 48비트가 6비트씩 8개로 나뉨
  - 각 6비트가 S-box를 통해 각 4비트의 값으로 압축됨
  - 총 32비트의 값이 출력됨



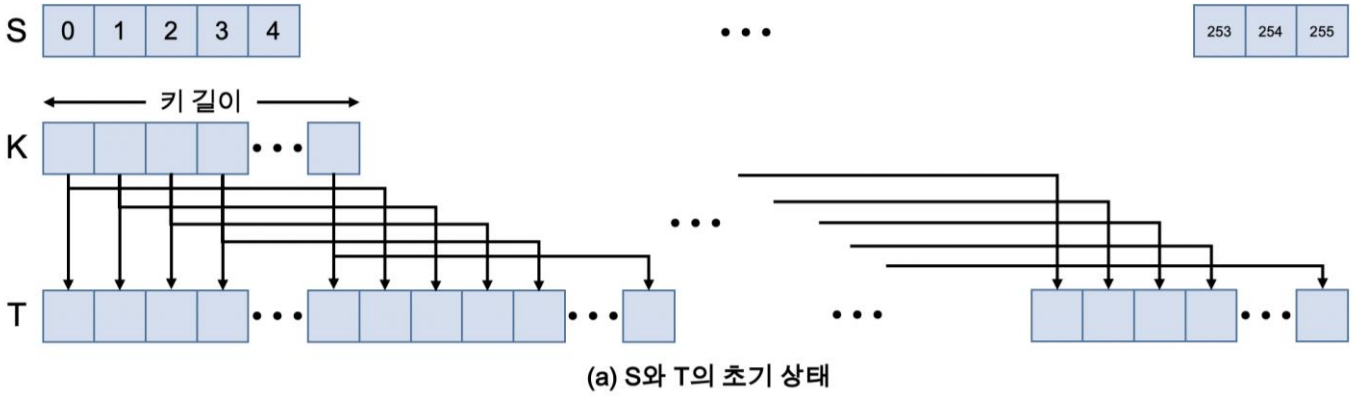
보충

- RC4 알고리즘
  - 정의
    - Ron Rivest가 설계한 바이트 단위로 작동되도록 만들어진 다양한 크기의 키를 사용하는 스트림 암호
  - 특징
    - 상태(State) 개념을 사용
    - KSA(Key Scheduling Algorithm)을 통해 256 바이트의 배열을 치환
    - PRGA(Pseudo Random Generation Algorithm)을 통해 평문을 암호화

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;
```

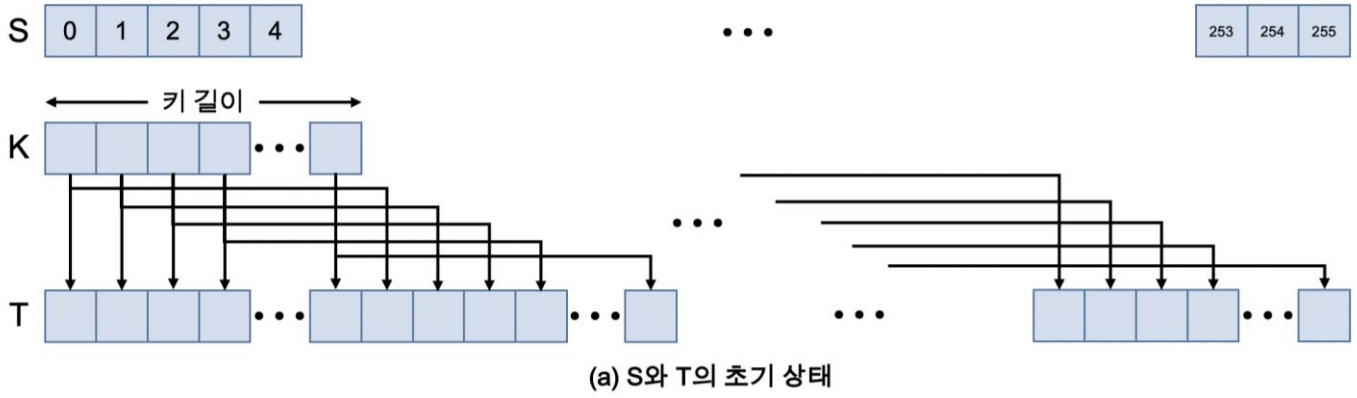


(a) S와 T의 초기 상태

보충

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

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

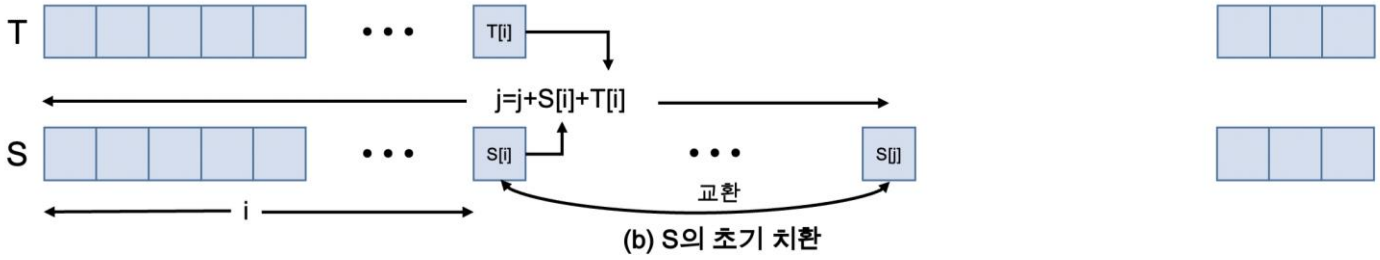


(a) S와 T의 초기 상태

보충

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

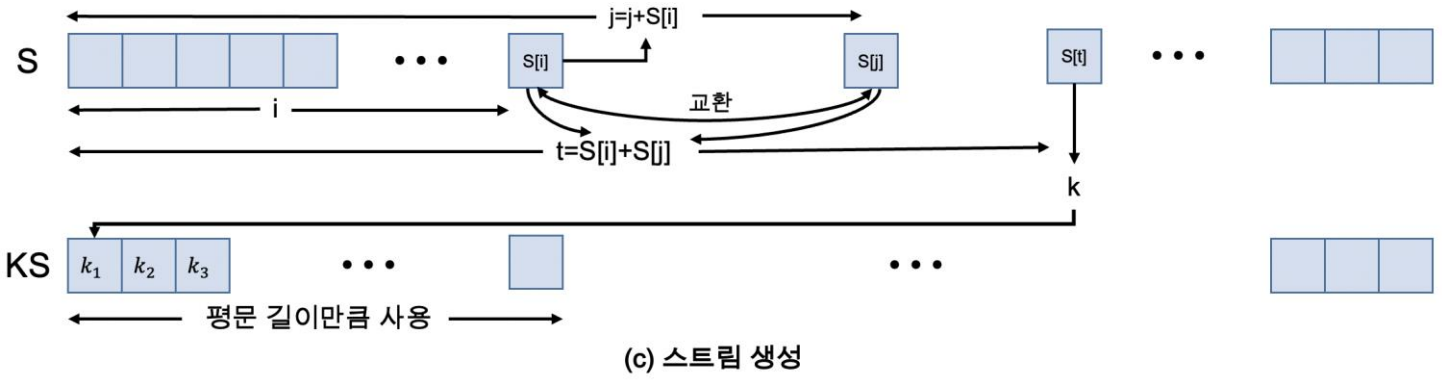
```
/*Initial Permutation of S*/
j=0;
for i = 0 to 255 do
 j=(j+S[i]+T[i]) mod 256;
 Swap (S[i], S[j]);
```



보충

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

```
/*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];
```



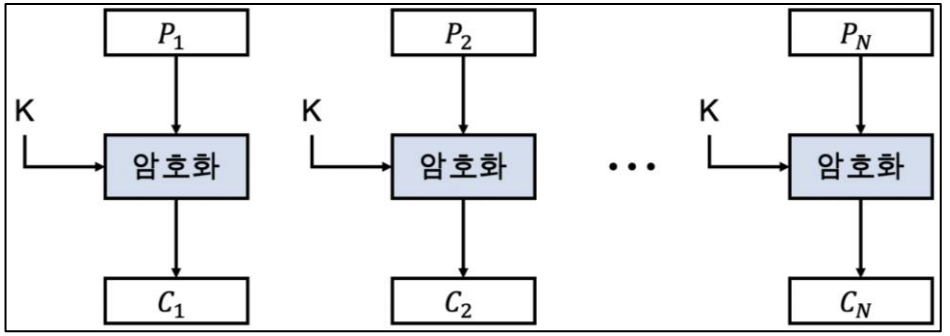
(c) 스트림 생성

보충

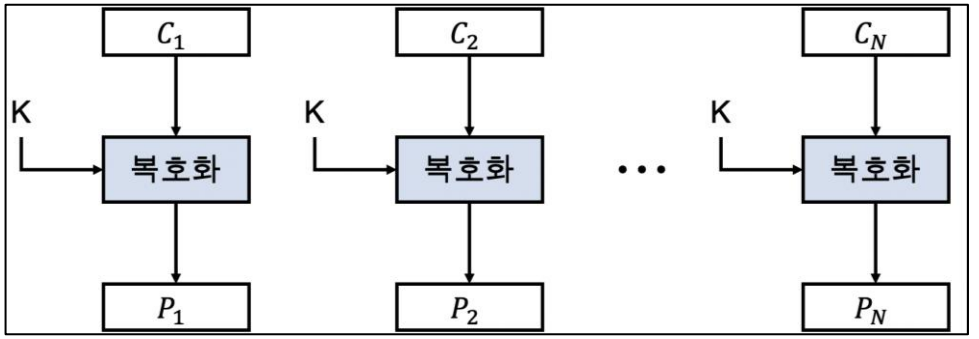
- 암호 블록 운용 모드
  - 정의
    - 블록 암호를 다양하게 응용하기 위해 NIST에서 정의한 5가지 운용모드
  - 종류
    - 전자 코드북 모드(ECB, Electronic Codebook Mode)
    - 암호 블록 체인 모드(CBC, Cipher Block Chaining Mode)
    - 암호 피드백 모드 (CFB, Cipher Feedback Mode)
    - 출력 피드백 모드 (OFB, Output Feedback Mode)
    - 카운터 모드(CTR, Counter Mode)

보충

- 암호 블록 운용 모드
  - ECB(Electronic Codebook) 모드
    - 평문을 같은 크기의 블록으로 나누고 각 블록을 동일한 키로 암호화하는 방식



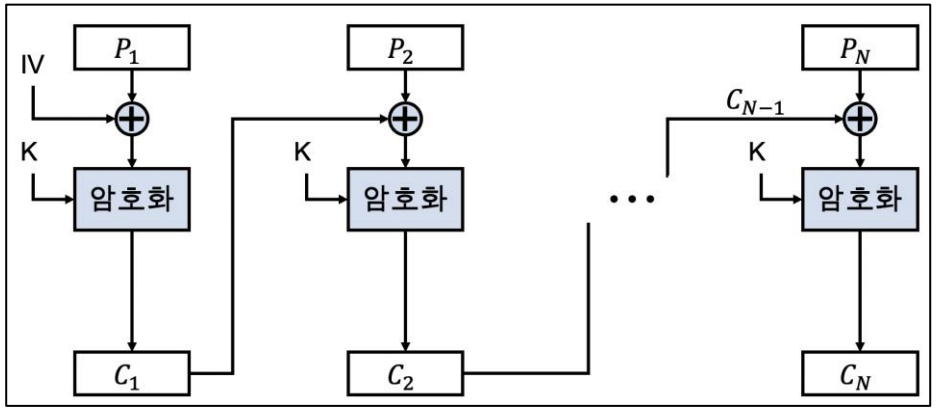
암호화



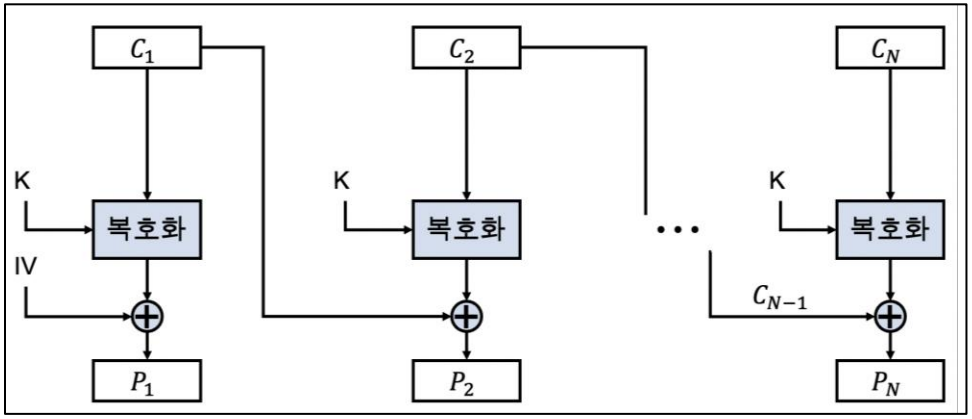
복호화

보충

- 암호 블록 운용 모드
  - CBC(Cipher Block Chaining) 모드
    - 현재의 평문 블록과 바로 직전의 암호 블록을 XOR한 결과를 알고리즘의 입력으로 사용하는 방식



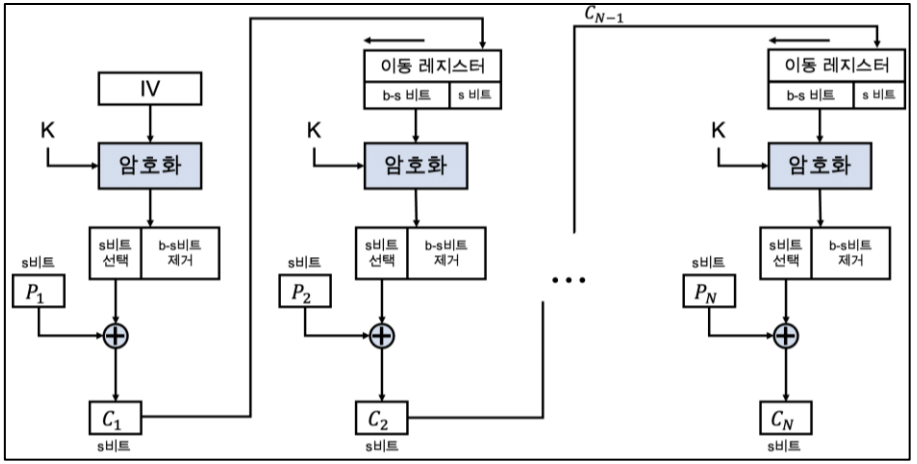
암호화



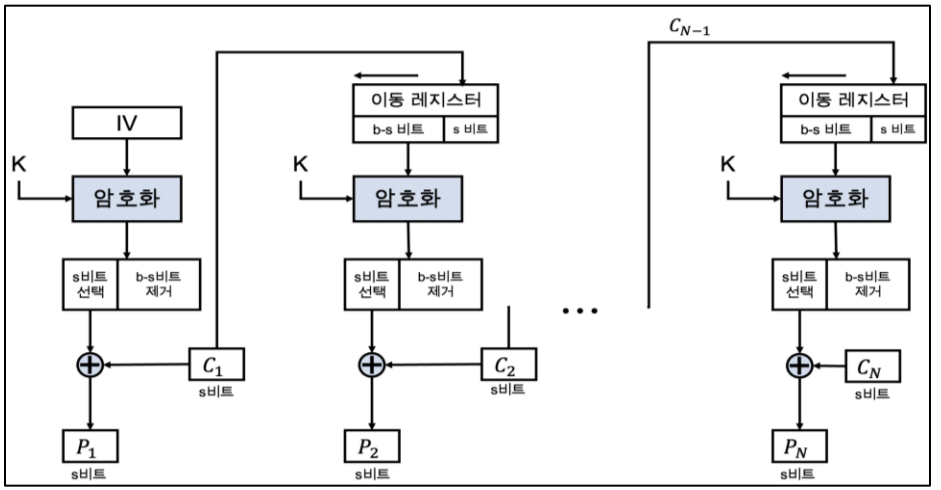
복호화

보충

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



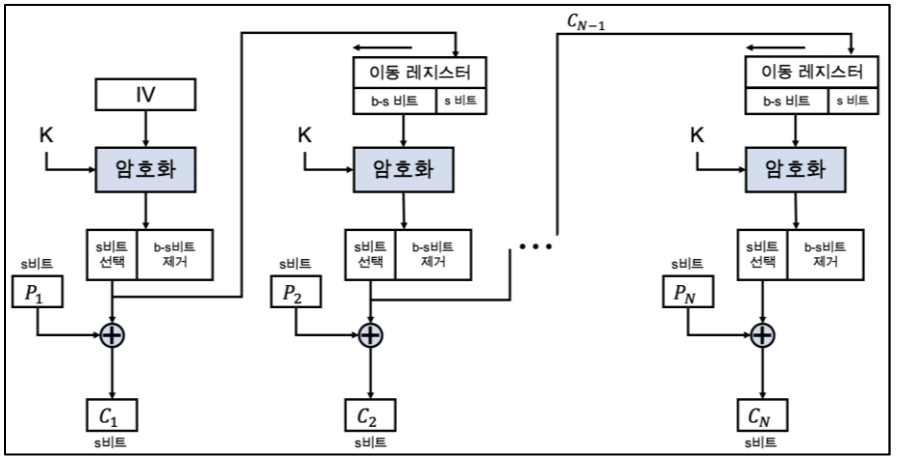
암호화



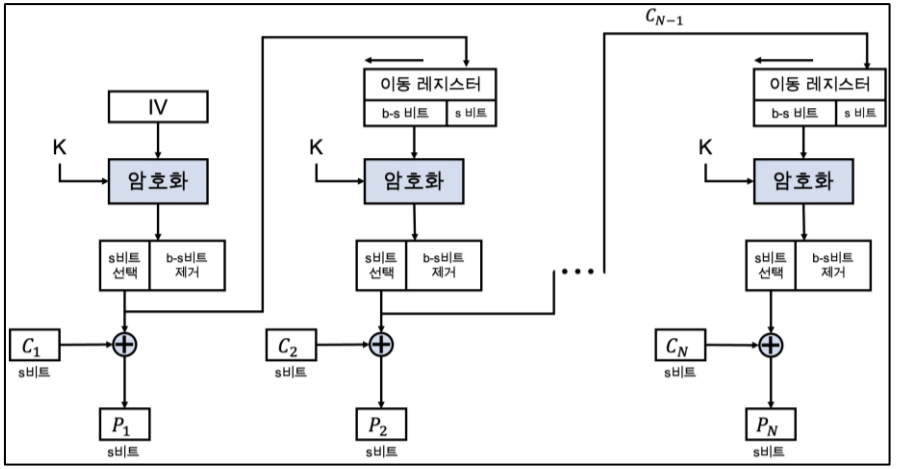
복호화

보충

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



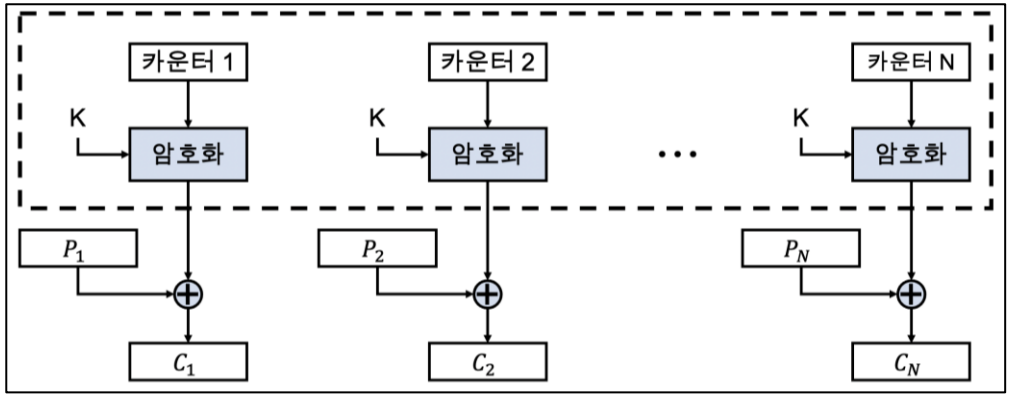
암호화



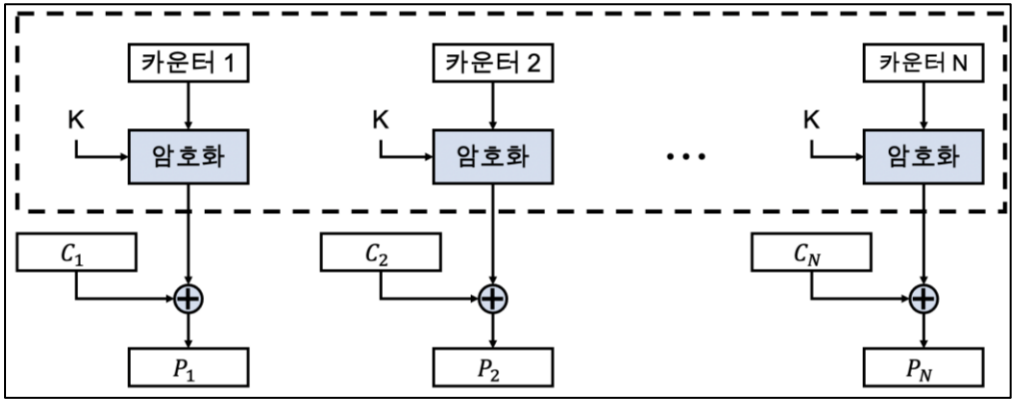
복호화

보충

- 암호 블록 운용 모드
  - CTR(Counter) 모드
    - 각 평문 블록마다 값이 다른 카운터를 암호화하여 평문 블록과 XOR하는 방식



암호화



복호화

보충

- 암호 블록 운용 모드
  - 운용 모드 비교 표

암호 운용 모드	병렬 처리	초기화 벡터	패딩
ECB	O	X	O
CBC	복호화만	O	O
CFB	복호화만	O	X
OFB	X	O	X
CTR	O	X	X

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
- 해시 함수
- 메시지 인증 코드

메시지 인증 방법

- 메시지 인증(Message Authentication)
  - 정의
    - 통신 양측으로 하여금 메시지의 내용이 전송 도중 불법적으로 변경되지 않고 안전하게 전송됨을 보증하는 것
  - 특징
    - 메시지의 무결성 보호
    - 메시지 송신자의 검증
    - 부인 방지

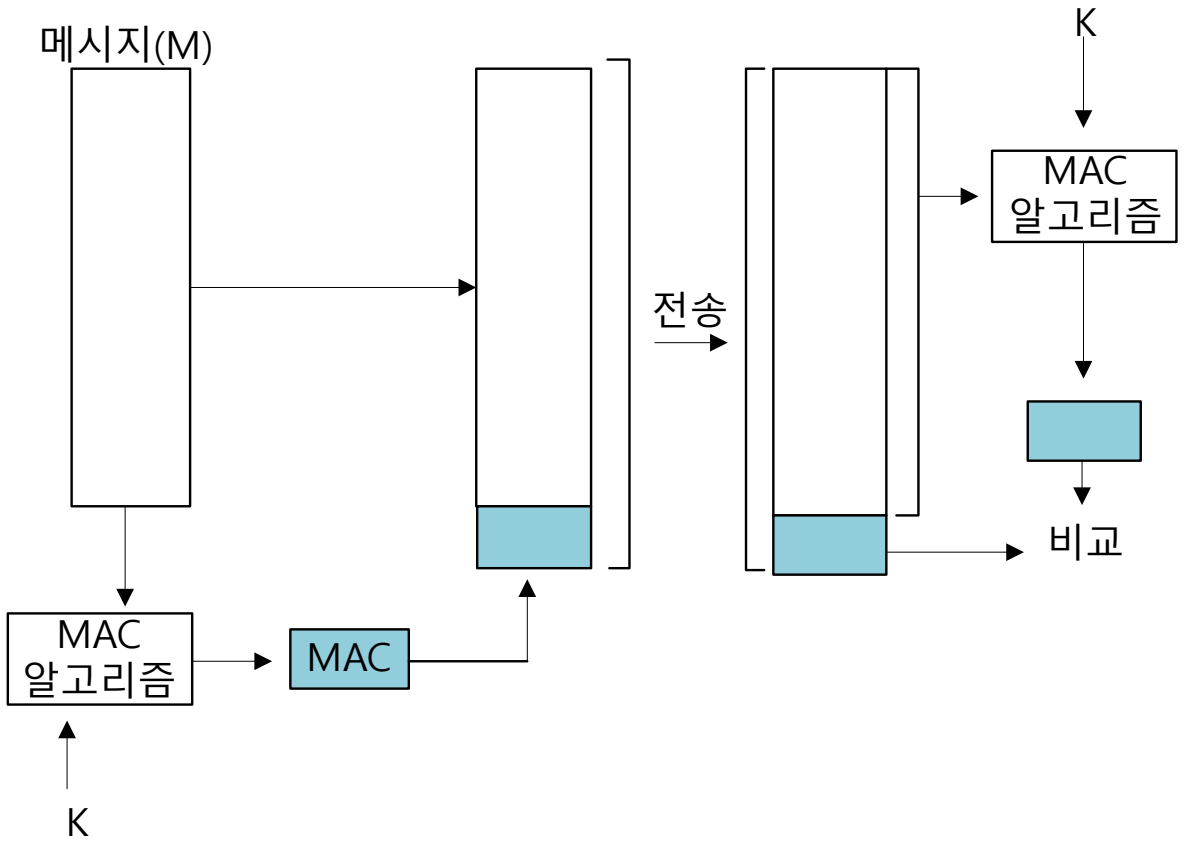
메시지 인증 방법

- 메시지 인증(Message Authentication)
  - 인증 방법
    - 대칭 키 암호를 사용한 인증
      - 송수신자가 공유하는 비밀 키를 이용해 메시지를 암/복호화 할 수 있음
      - 오류 감지 코드를 통해 수신자는 메시지의 변경 여부 확인 가능
      - 순서 번호를 통해 수신자는 메시지의 순서가 맞는지 확인 가능
      - 타임스탬프를 통해 수신자는 메시지의 고의적 시간 지연이 있었는지 확인 가능
    - 메시지 암호화 없는 메시지 인증
      - 인증 꼬리표를 생성하여 메시지에 붙여서 보냄
      - e.g.,
        - 동일한 메시지를 다수의 수신자에게 브로드캐스팅하는 경우
        - 메시지 교환 시 송수신자 중 한 쪽에 부하가 걸려 메시지 복호화가 힘든 경우
        - 컴퓨터 프로그램을 평문인 채로 인증하는 경우

메시지 인증 방법

- 메시지 암호화 없는 메시지 인증
  - 메시지 인증 코드(MAC, Message Authentication Code)를 이용한 메시지 인증

$$[MAC_M = F(K,\ M)]$$



목 차

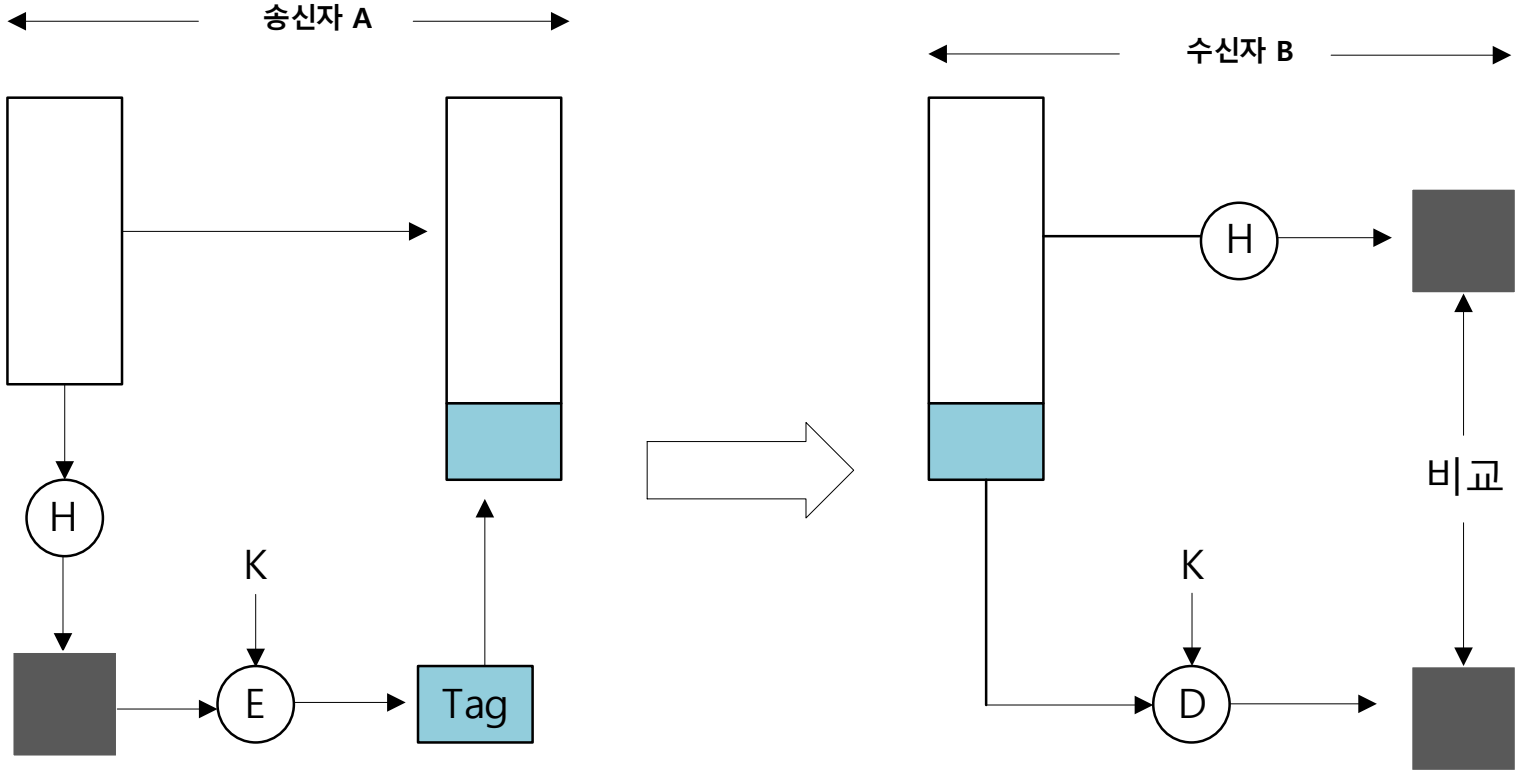
- 보충
  - DES, AES에서의 대체/치환
  - 페이스텔 구조에서의 Sub Key 생성과정
  - 페이스텔 구조에서의 F함수
  - 스트림 암호와 RC4
  - 암호 블록 운용 모드
- 메시지 인증 방법
- 해시 함수
- 메시지 인증 코드

해시 함수

- 일방향 해시 함수
  - 정의
    - 특정 입력 데이터에서 고정 길이 값 또는 해시 값을 생성하며, 역으로 수행될 수 없는 알고리즘
  - 특징
    - 비밀 키를 입력으로 사용하지 않음
    - 인증된 상태의 메시지 다이제스트를 메시지와 함께 전송함
    - 데이터의 무결성을 검증하는 데에 사용됨
  - 세 가지 인증 방법
    - 관용 암호를 사용한 인증
    - 공개 키 암호를 사용한 인증
    - 비밀 값을 사용한 인증

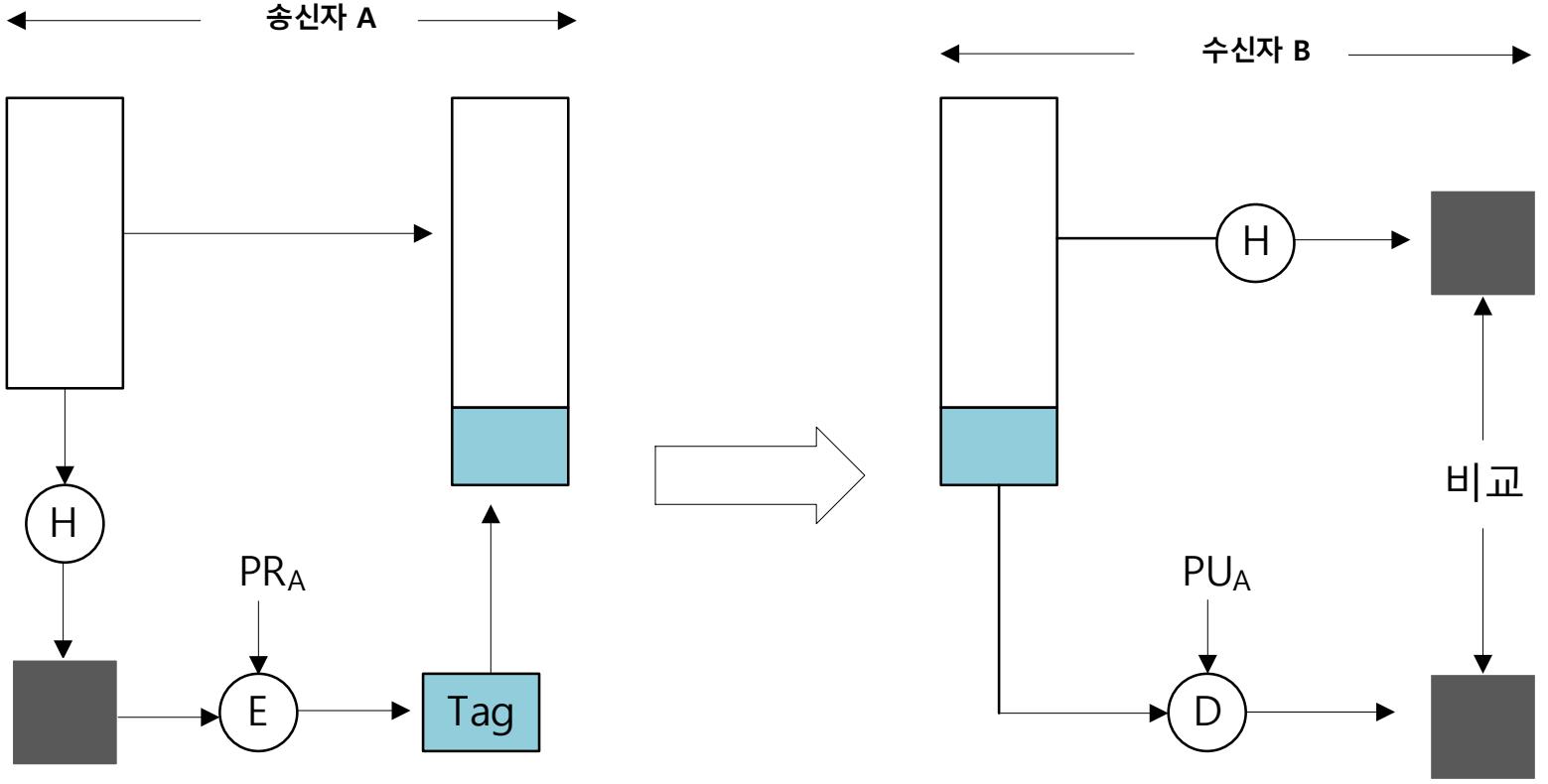
해시 함수

- 일방향 해시 함수
  - 관용 암호를 사용한 인증
    - 송수신자는 공유하는 비밀 키를 사용



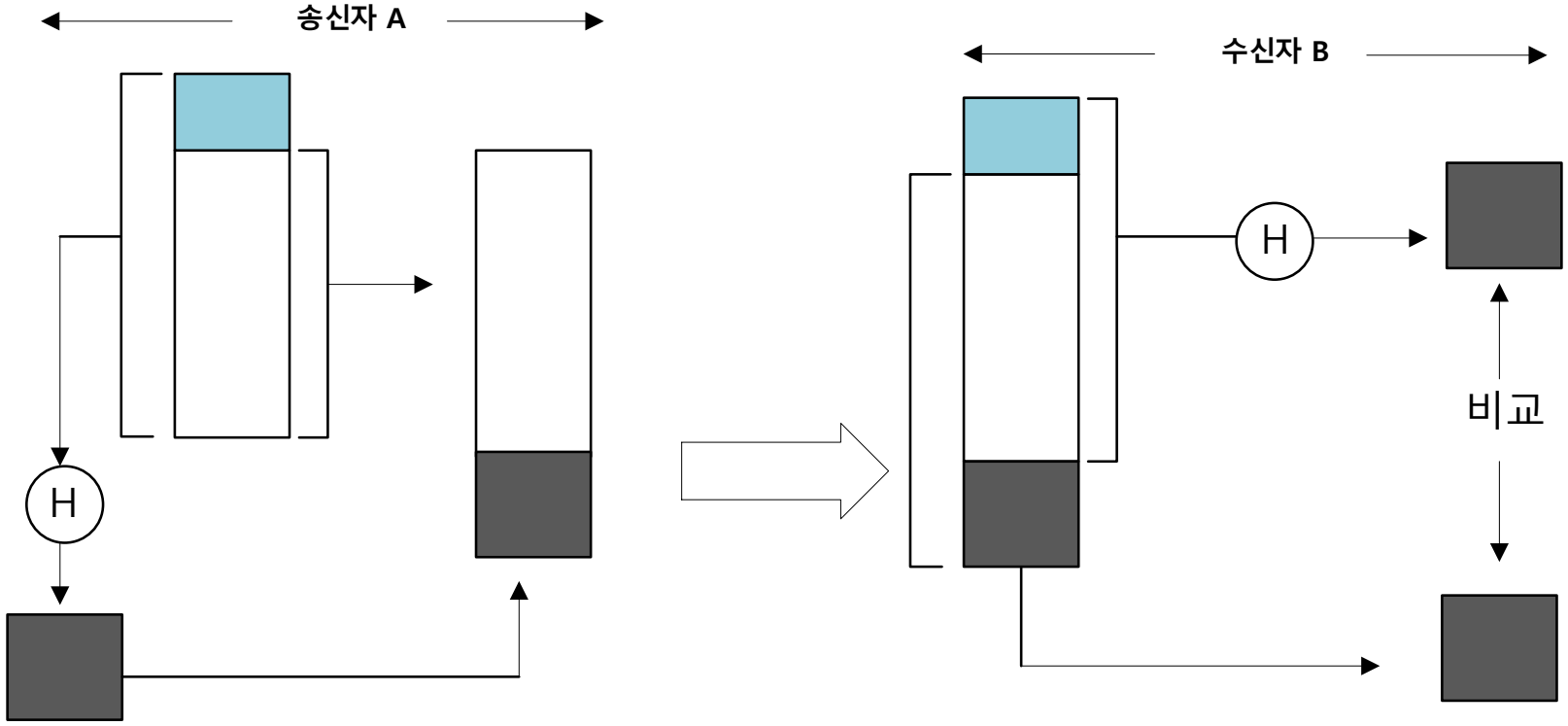
해시 함수

- 일방향 해시 함수
  - 공개 키 암호를 사용한 인증
    - 송신자의 개인 키를 통해 암호화, 송신자의 공개 키를 통해 복호화



해시 함수

- 일방향 해시 함수
  - 비밀 값을 사용한 인증
    - 송수신자는 공유하는 비밀 값을 사용



해시 함수

- 해시 함수
  - 해시 함수 필수 조건
    - 어떠한 크기의 데이터 블록에도 적용될 수 있어야 함
    - 일정한 길이의 출력 값을 생성해야 함
    - 계산이 쉬워야 하고, 구현을 실제로 할 수 있어야 함
  - 해시 함수 성질
    - 일방향 성질
      - $H(x) = h$가 성립되는 x 를 찾는 것이 어려워야 함
    - 약한 충돌 저항성
      - $H(x) = H(y)$를 만족하는 $x \neq y$를 찾는 것이 어려워야 함
    - 강한 충돌 저항성
      - $H(x) = H(y)$를 만족하는 쌍 (x, y)를 찾는 것이 어려워야 함

해시 함수

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

해시 함수 성질	공격을 위해 찾아야하는 메시지의 개수
일방향 성질	2^n
약한 충돌 저항성	2^n
강한 충돌 저항성	$2^{n/2}$

- 현 시점에서 256, 384, 512비트 길이의 해시 함수가 가장 많이 쓰임
  - 128비트 해시 함수의 경우, 24시간만에 충돌이 발견됨

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

해시 함수

- 안전 해시 알고리즘(SHA, Secure Hash Algorithm)
  - 정의
    - NIST가 개발한 암호학적 해시 함수들의 모음
  - 종류
    - SHA-0
    - SHA-1
    - SHA-2
      - SHA-224
      - SHA-256
      - SHA-384
      - SHA-512
    - SHA-3

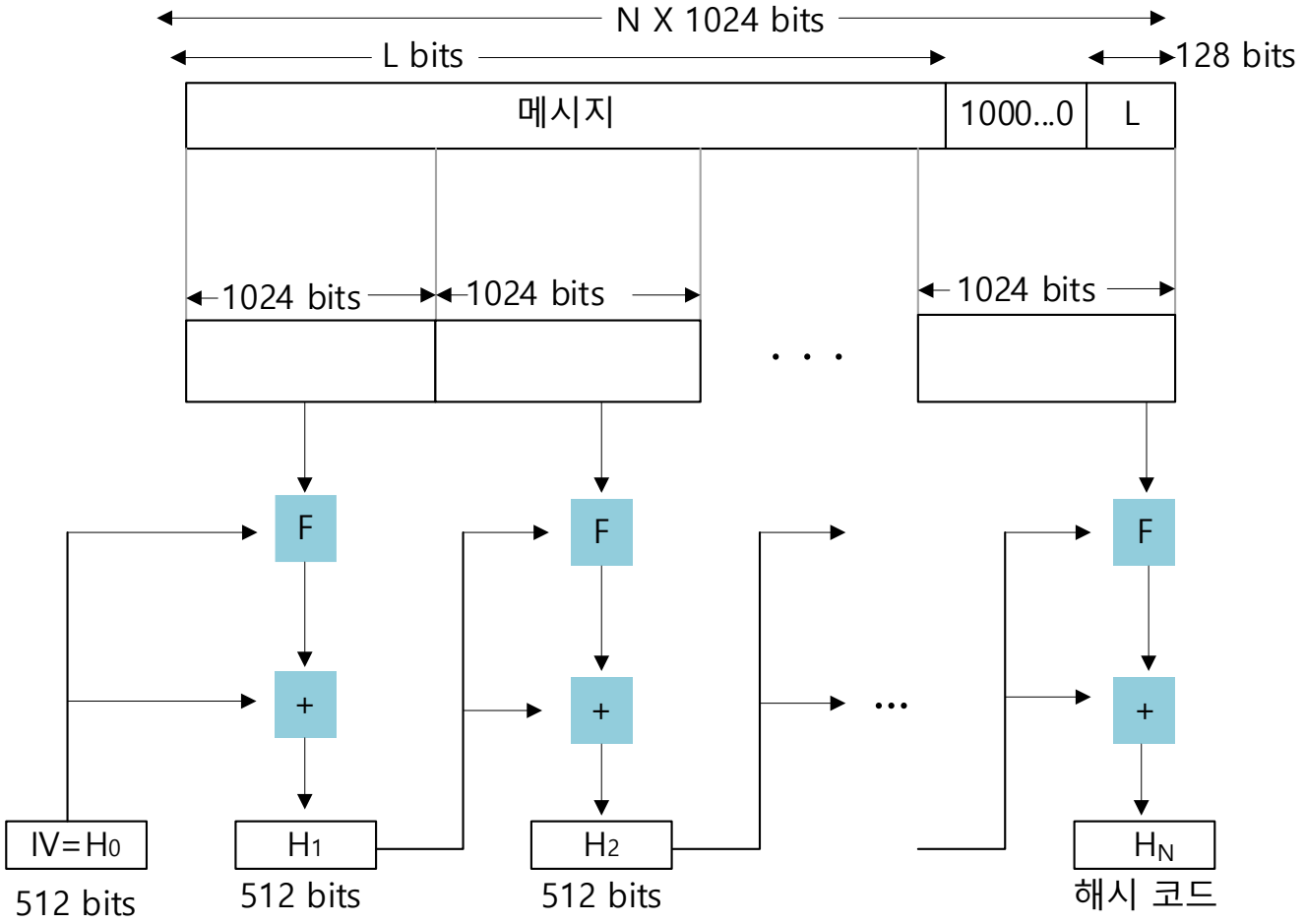
해시 함수

- 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

해시 함수

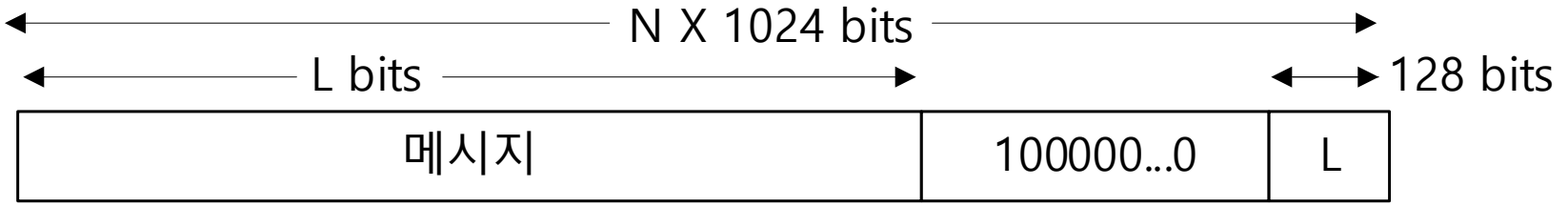
- 안전 해시 알고리즘(SHA: Secure Hash Algorithm)
  - SHA-512 알고리즘의 동작 과정



해시 함수

- 안전 해시 알고리즘(SHA: Secure Hash Algorithm)
  - SHA-512 알고리즘의 동작 과정
    1. 패딩 비트 붙이기(Appending Padding Bits)
       - 총 길이를 896 (mod 1024)가 되도록 메시지 패딩을 추가
       - 패딩을 구성하는 비트는 첫번째 비트가 1, 나머지는 0

    2. 길이 붙이기(Append Length)
       - 128비트 블록을 메시지에 추가하여 전체 메시지의 길이를 1024의 배수가 되게 만듦
         - 원래 메시지의 길이를 포함하는 블록



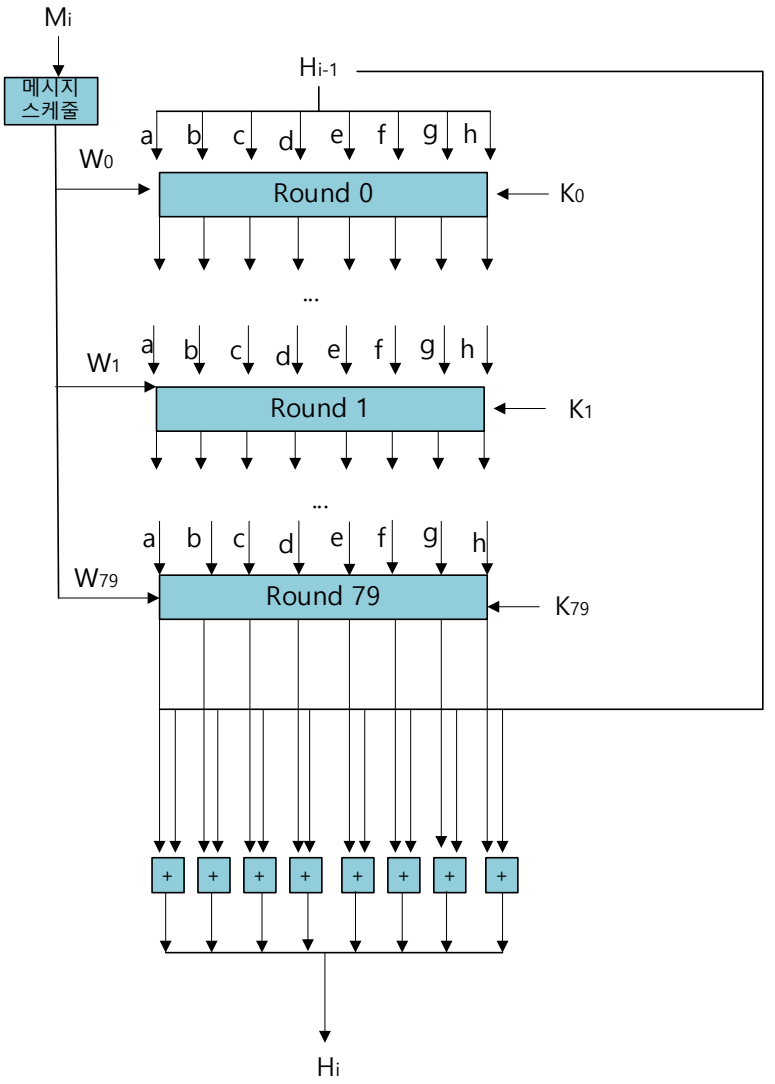
해시 함수

- 안전 해시 알고리즘(SHA: Secure Hash Algorithm)
  - SHA-512 알고리즘의 동작 과정
    3. MD 버퍼 초기화(Initialize MD Buffer)
       - 64비트 버퍼 8개를 각각의 고정된 64비트 정수로 초기화
       - 512비트 해시 값 출력을 위해 사용됨

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

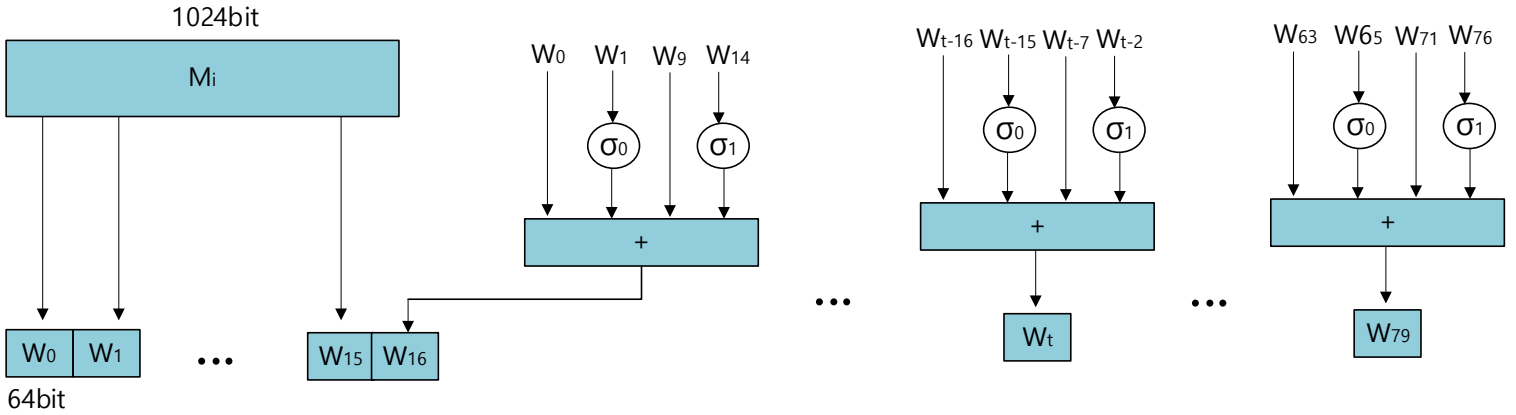
해시 함수

- 안전 해시 알고리즘(SHA: Secure Hash Algorithm)
  - SHA-512 알고리즘의 동작 과정
    4. 1024 비트 블록 메시지 처리 (Process Message in 1024 Bit)
       - 총 80라운드로 구성
       - 첫 라운드에서 버퍼는 H_{i-1}값을 가짐
       - 라운드마다 8개의 64비트 버퍼 값을 갱신
       - 마지막 라운드가 끝난 후 1024 비트 블록을 512 블록으로 압축



해시 함수

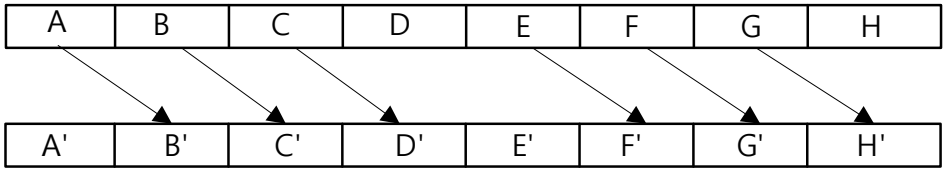
- 안전 해시 알고리즘(SHA: Secure Hash Algorithm)
  - SHA-512 알고리즘의 동작 과정
    - 메시지 스케줄
      - 16개의 워드를 80개로 확장



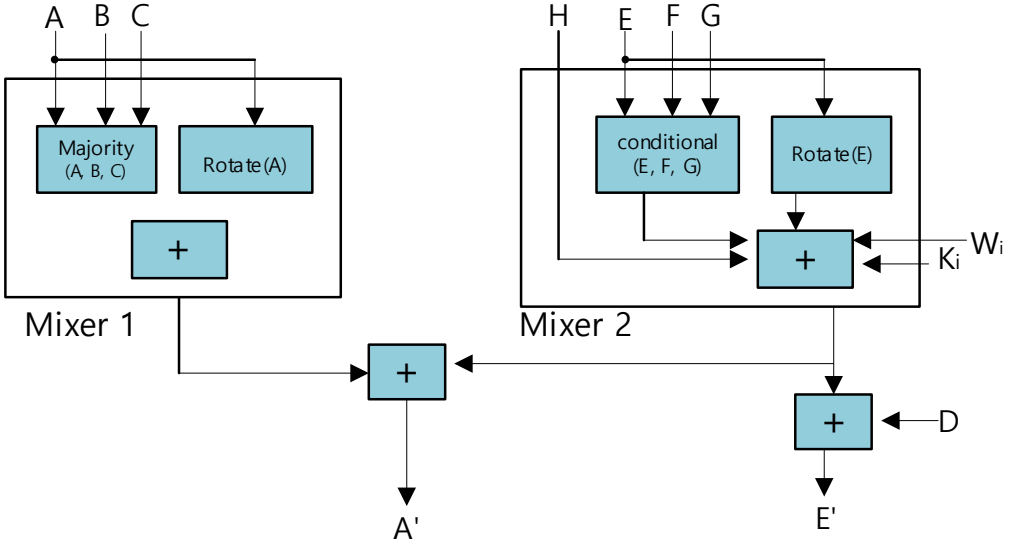
$\sigma_0(x) = ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x)$
$\sigma_1(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x)$
$ROTR^n(x)$= x를 n비트만큼 오른쪽으로 순환 이동한 것
$SHR^n(x)$= x를 n비트만큼 왼쪽으로 이동하고 오른쪽을 0으로 채움

해시 함수

- 안전 해시 알고리즘(SHA: Secure Hash Algorithm)
  - SHA-512 알고리즘의 동작 과정
    - 라운드 함수
      - 상수 $K_0, K_1, \ldots, K_{79}$
        - 처음 80개 소수 세제곱근의 소수점 이하 64비트

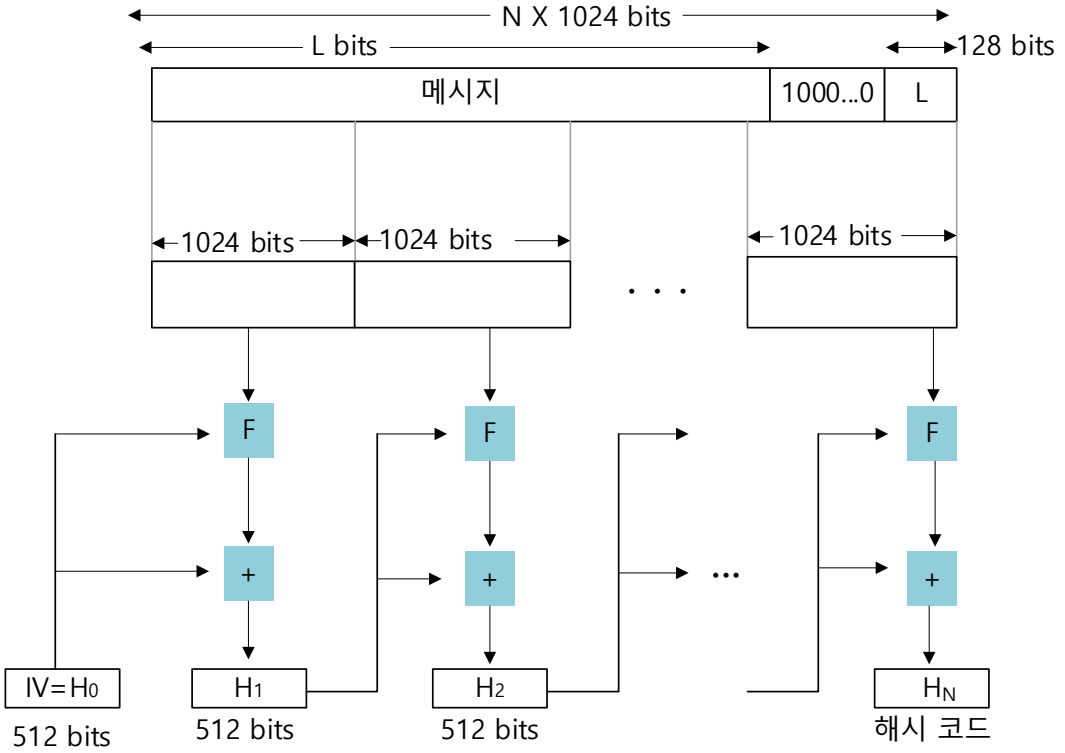


Majority(x,y,z):
(x AND y) ⊕ (y AND z) ⊕ (z AND x)
Conditional (x, y, z):
(x AND y) ⊕ (NOT x AND z)
Rotate (A):
$ROTR_{28}(x) \oplus ROTR_{34}(x) \oplus ROTR_{39}(x)$
$ROTR_i(x)$: x를 i비트 만큼 오른쪽으로 shift



해시 함수

- 안전 해시 알고리즘(SHA: Secure Hash Algorithm)
  - SHA-512 알고리즘의 동작 과정
    5. 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
- 해시 함수
- 메시지 인증 코드

메시지 인증 코드

- HMAC(Hashed MAC)
  - 정의
    - 해시 함수와 비밀 키를 수반하는 메시지 인증 코드
  - 특징
    - 해싱 기법을 사용하여 메시지의 위변조 방지
    - 기존 해시 함수 변경 가능
    - 기능 저하가 유발되지 않도록 설계됨

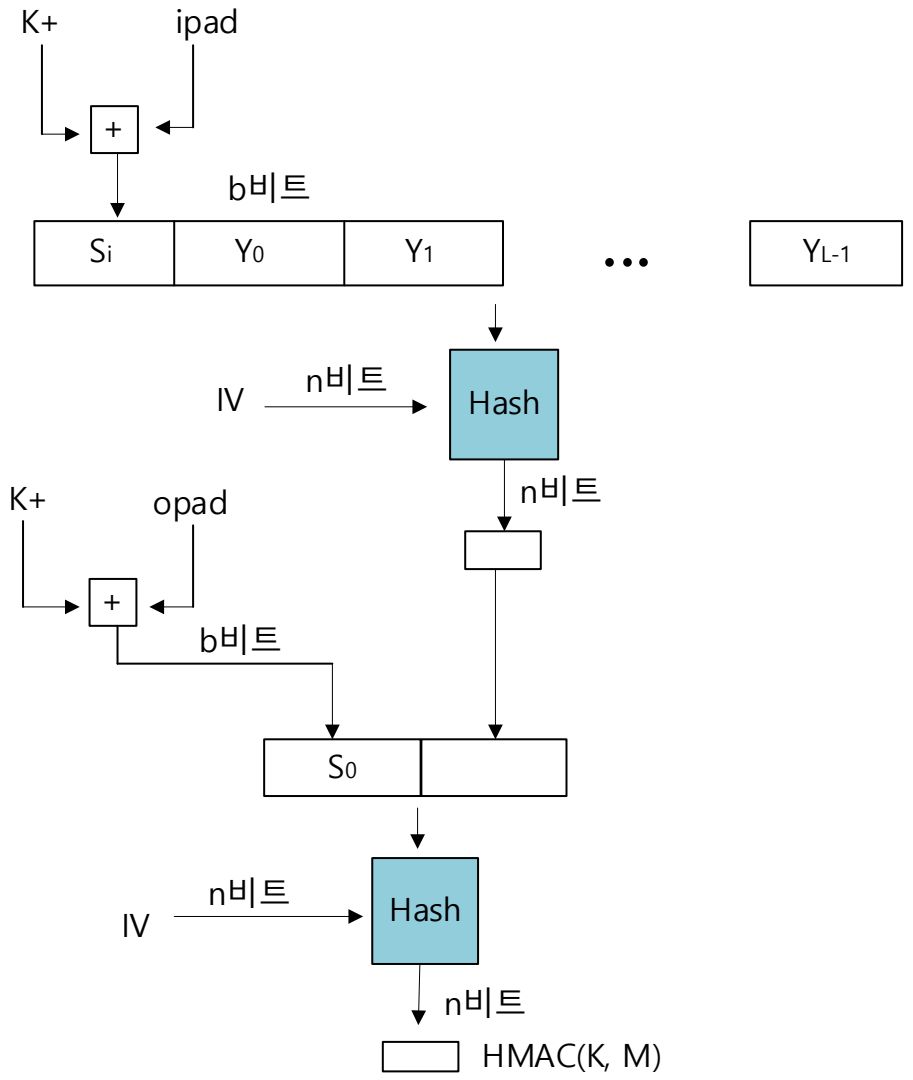
메시지 인증 코드

- 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^+생성
    2) K^+ 와 ipad를 XOR연산하여 b비트 블록 S_i 생성
    3) S_i에 M을 붙임
    4) 3)에서 생성된 스트림에 해시함수를 적용
    5) K^+ 와 opad를 XOR연산하여 b비트 블록 S_0 생성
    6) 4)에서 얻은 해시 결과를 S_0에 붙임
    7) 6)에서 생성된 스트림에 해시함수를 적용해서 출력



메시지 인증 코드

- CMAC(Cipher based MAC)
  - 정의
    - 블록 암호 기반 메시지 인증 코드
  - 특징
    - AES, 3DES가 사용됨
    - 메시지는 일정 길이의 블록으로 나뉨
    - 2개의 키가 사용됨

메시지 인증 코드

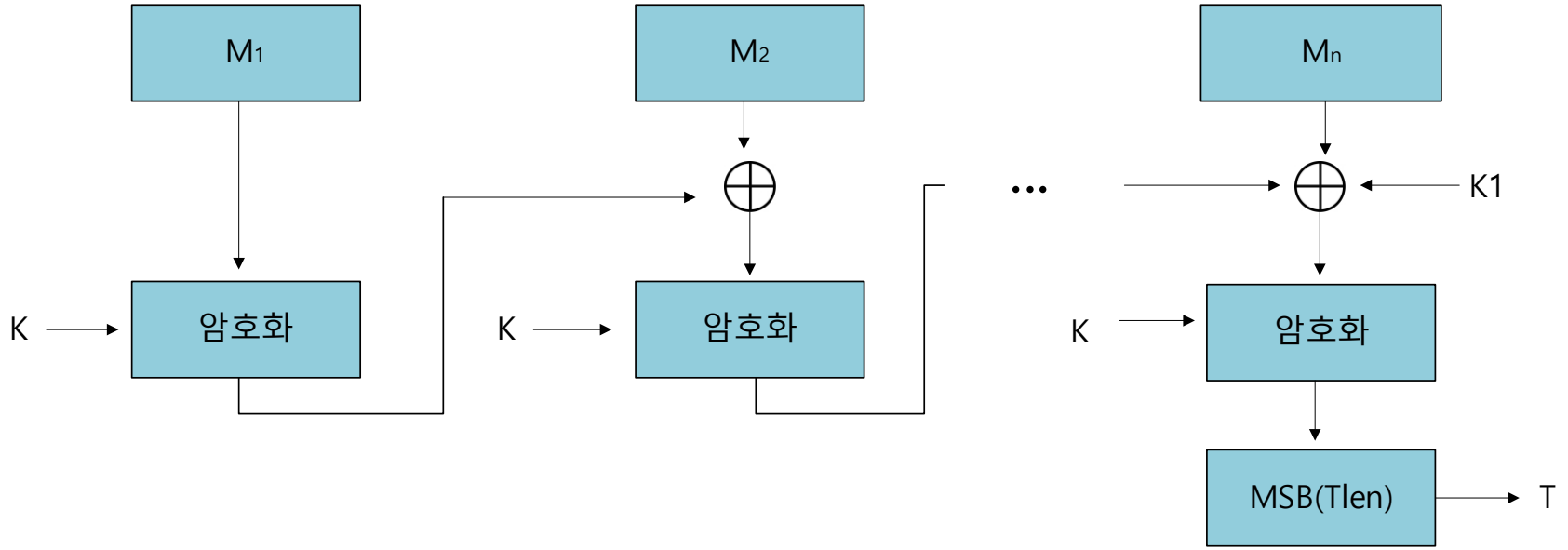
- CMAC(Cipher based MAC)
  - 계산식

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

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	E(0, K) <<1
K_2	E(0,K_1) <<1

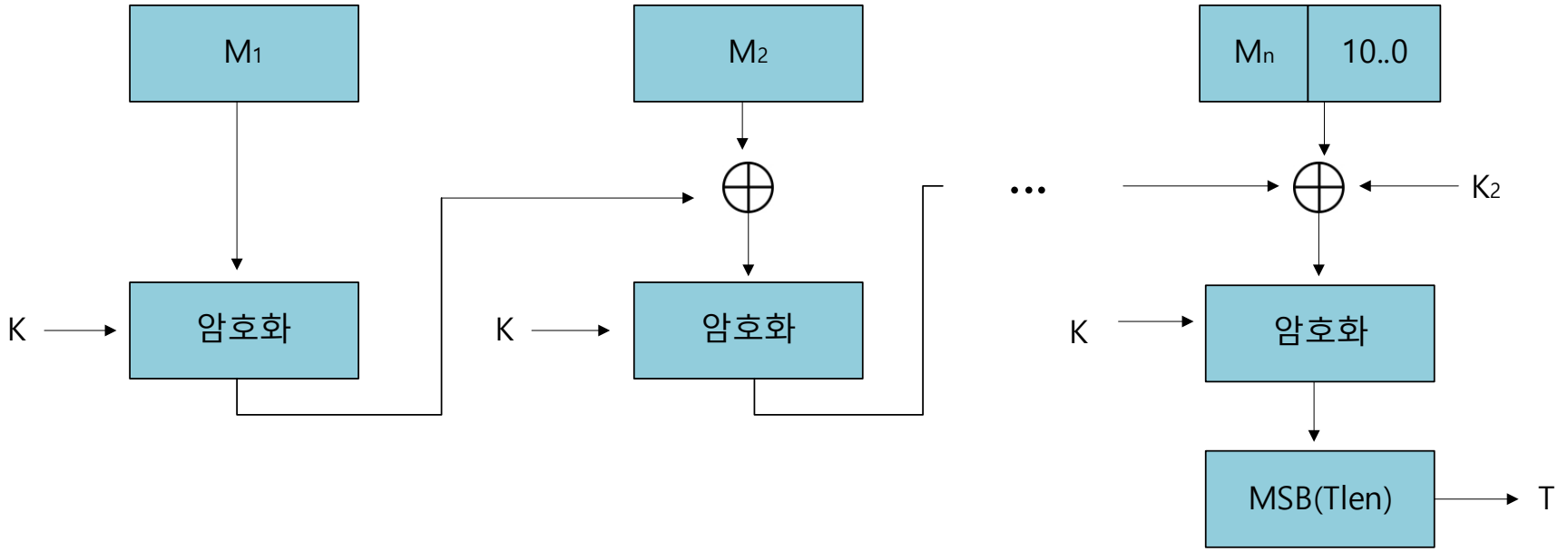
메시지 인증 코드

- CMAC(Cipher based MAC)
  - 동작 과정
    - 메시지 길이가 블록 길이의 정수배인 경우
      - k비트 암호 키 K와 b비트 키 K_1를 사용



메시지 인증 코드

- CMAC(Cipher based MAC)
  - 동작 과정
    - 메시지 길이가 블록 길이의 정수배가 아닌 경우
      - 마지막 평문 블록을 패딩 처리하여 블록 길이를 b비트로 만듦
      - k비트 암호 키 K와 b비트 키 K_2를 사용

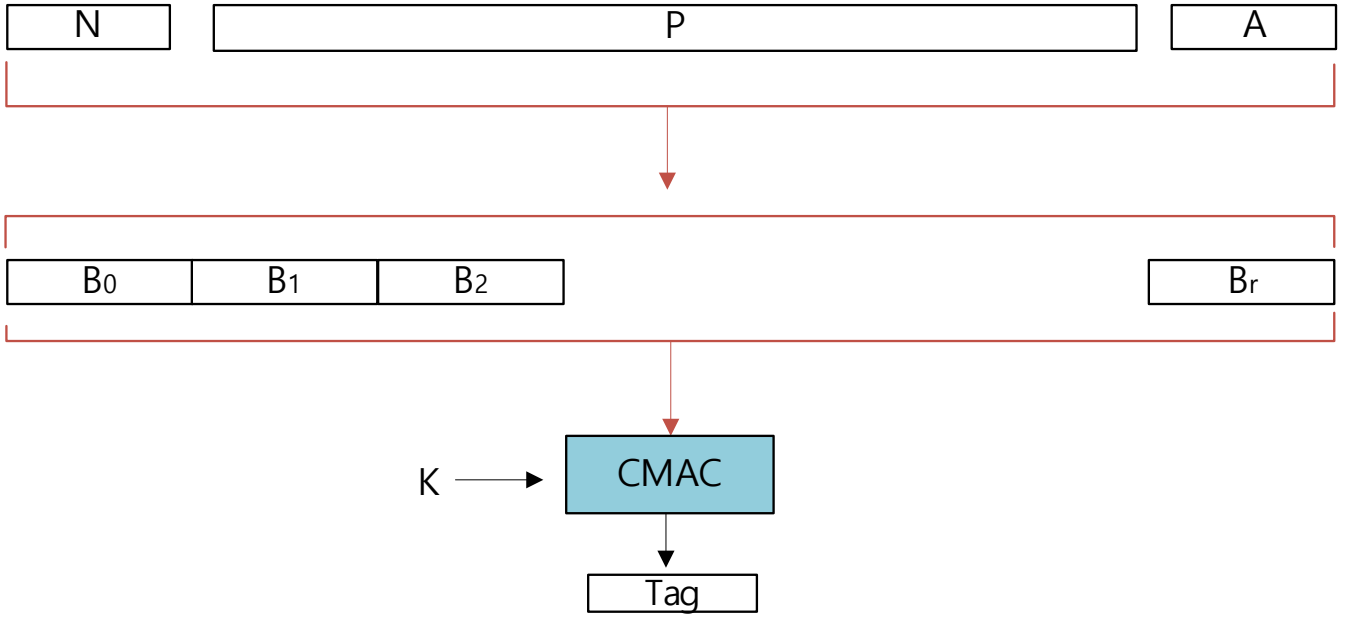


메시지 인증 코드

- CCM(Counter With Cipher Block Chaining MAC)
  - 정의
    - 하나의 키 아래에서 블록 암호를 반복적으로 안전하게 이용하게 하는 메시지 인증 코드
  - 특징
    - AES 알고리즘, CTR 운용 모드, CMAC 알고리즘 사용
    - 암호화와 인증에 동일한 키를 사용
    - 인증과 기밀성을 모두 제공하도록 설계됨

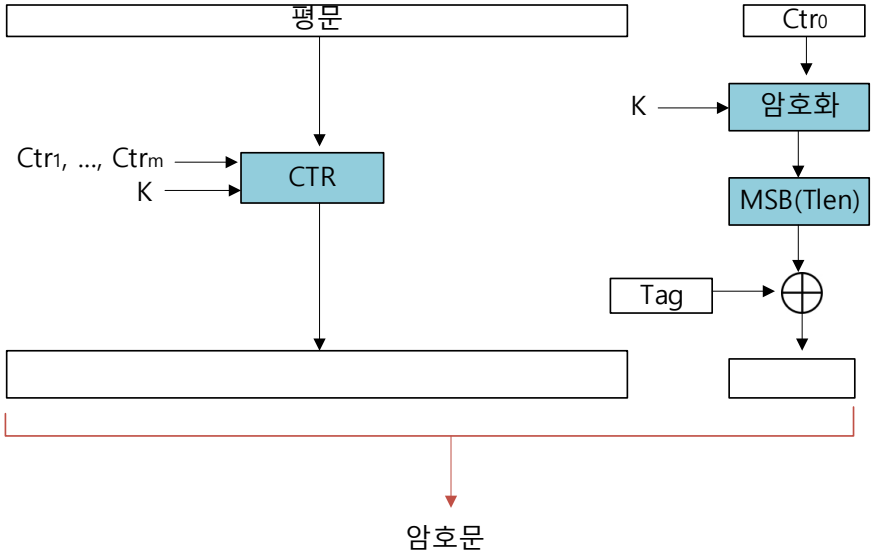
메시지 인증 코드

- CCM(Counter With Cipher Block Chaining MAC)
  - 인증 과정
    - 비표(N), 유관 데이터(A), 평문(P)를 입력
    - 입력 값을 B_0부터 B_r까지 블록 열 형식으로 나타냄
      - 첫 번째 블록에는 N, A와 P의 길이를 나타내는 형식 비트가 추가됨
    - 블록 열을 CMAC 알고리즘의 입력값으로 사용



메시지 인증 코드

- CCM(Counter With Cipher Block Chaining MAC)
  - 암호화 과정
    - 비표와 독립적으로 카운터 열 생성
    - 인증 태그를 Ctr_0를 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



Thanks!

강 민 채 (minchae@pel.sejong.ac.kr)