2018/01/16, 2018 보안 기초 세미나

Network Security Essentials

- Chapter_3 공개키 암호와 메시지 인증 (1) -

최서윤 (seoyun@pel.smuc.ac.kr)

상명대학교 프로토콜공학연구실

목 차

- 메시지 인증 방법
- 안전 해시함수
- 메시지 인증 코드

메시지 인증 방법

- 메시지 인증(Message Authentication)
  - 의의
    - 메시지 내용의 무결성 제공
    - 메시지 출처 검증
    - 시간성 검증
      - 타임스탬프(Timestamp)를 통해 고의적 지연 및 재전송 여부 확인
      - 메시지 순서 변경 여부 확인
  - 방법
    - 메시지에 암호화 적용
      - 송신자와 수신자가 공유하는 대칭키를 통해 인증 가능
    - 메시지에 암호화 비적용
      - 인증 꼬리표(Authentication Tag)를 생성하여 인증 가능

메시지 인증 방법

- 메시지 암호화 없는 메시지 인증(Message Authentication without Message Encryption)
  - 기밀성 없는 메시지 인증이 적합한 경우 사용
    - 예시
      - 동일한 메시지를 브로드캐스트(Broadcast)하는 경우
        - e.g., 네트워크 사용 통지, 통제 센터 경고 신호
        - 비용 절감 및 신뢰성 증가
      - 메시지 교환 중 한 쪽의 과부화로 수신되는 메시지를 복호화 할 시간이 없는 경우
        - 임의로 메시지를 선택하여 인증 수행
      - 평문 상태로 인증되는 컴퓨터 프로그램
        - 매번 복호화 하지 않고 실행 가능

메시지 인증 방법

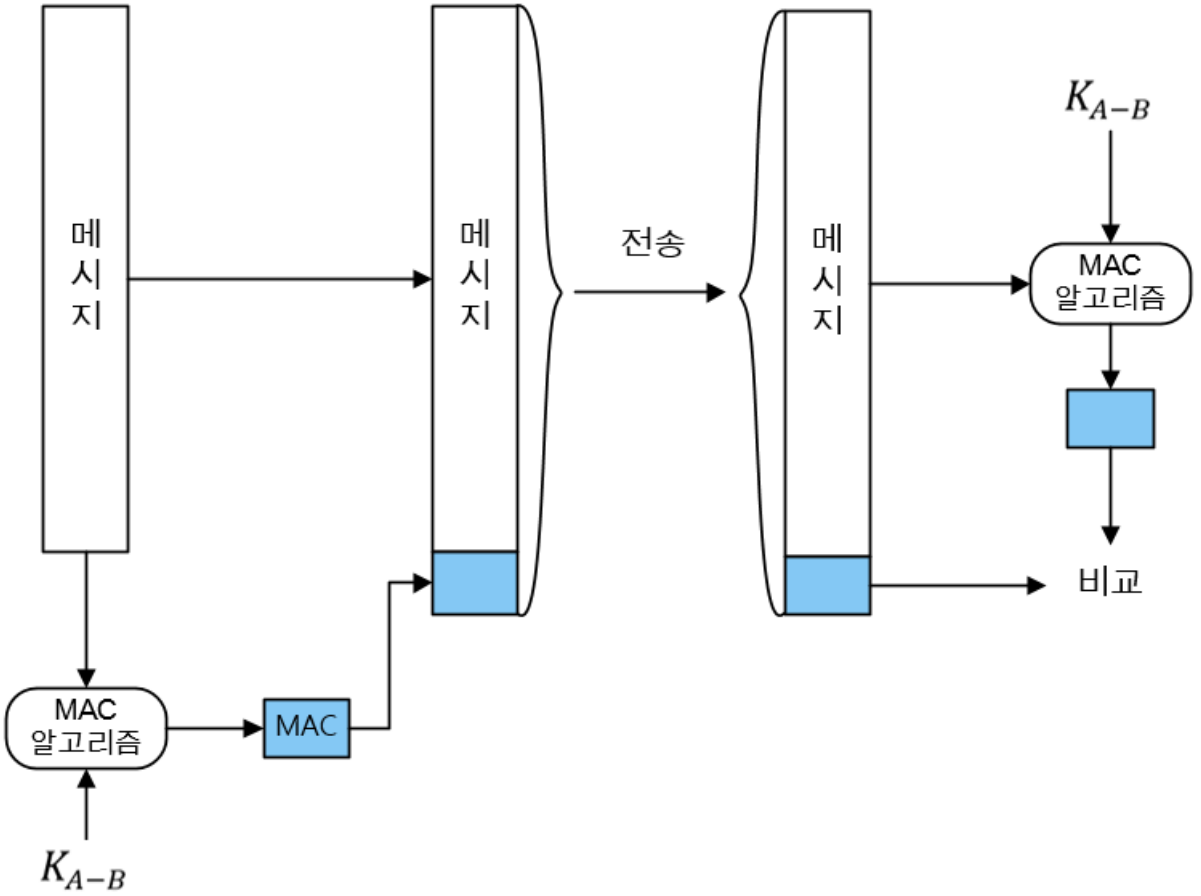
- 메시지 암호화 없는 메시지 인증(Message Authentication without Message Encryption)
  - 보안 요구사항을 충족하도록 인증과 암호 필요
    - 평문이 아닌 인증 수단에 암호화 적용
  - 해결 방안
    - 인증 꼬리 표(Authentication Tag) 생성
      - 메시지 인증 코드(MAC, Message Authentication Code)
        - 메시지 끝에 연접되는 블록(MAC 값) 생성에 비밀키 이용
      - 메시지 다이제스트(Message Digest)
        - 일방향 해시 함수 사용

메시지 인증 방법

- 인증 꼬리표(Authentication Tag)
  - 메시지 인증 코드(MAC: Message Authentication Code)
    - 특징
      - NIST(National Institution of Standards and Technology) 명세 FIPS PUB(Federal Information Processing Standards Publication) 113에서 Code 생성에 DES 사용 권장
        - 메시지를 DES로 암호화하여 암호문의 끝 16 또는 32비트를 코드로 사용
      - 인증 알고리즘에서는 역방향 계산이 불가해야 함

메시지 인증 방법

- 인증 꼬리표(Authentication Tag)
  - 메시지 인증 코드(MAC: Message Authentication Code)



- 작동 순서

1. 통신 구성원 A, B 공통 비밀키 (K_{A-B}) 공유

2. A가 전송할 메시지(M)와 키를 이용해 MAC 생성

$$MAC_M = F\{(K_{A-B}), M\}$$

3. B에게 MAC이 연접된 메시지와 키 전송

4. B도 메시지와 키를 이용해 MAC 생성

5. 수신한 MAC과 자신이 생성한 MAC을 비교

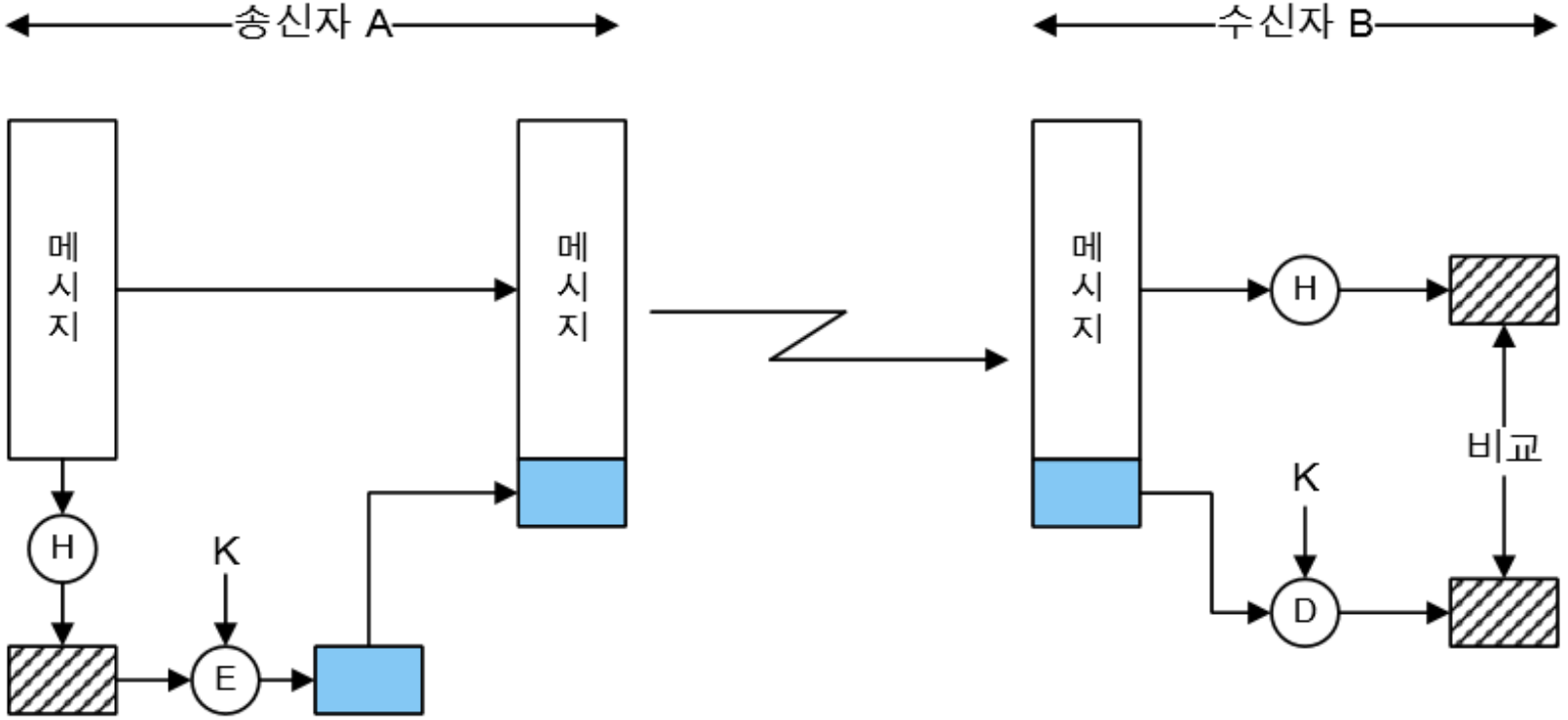
메시지 인증 방법

- 인증 꼬리표(Authentication Tag)
  - 메시지 다이제스트(Message Digest)
    - 의의
      - 해시함수를 통해 임의 크기의 메시지를 고정 크기의 블록으로 압축
    - 일방향 해시함수를 이용한 메시지 인증
      - 메시지 다이제스트에 관용 암호 이용
      - 메시지 다이제스트에 공개키 암호 이용
      - 메시지 다이제스트에 비밀 값 이용
  - 비교

	메시지 인증 코드 (MAC)	메시지 다이제스트
입력 값(Input)	원문 메시지, 비밀키	원문 메시지
보장하는 성질	인증, 무결성	무결성 (전자 서명 사용을 통해 인증 제공)

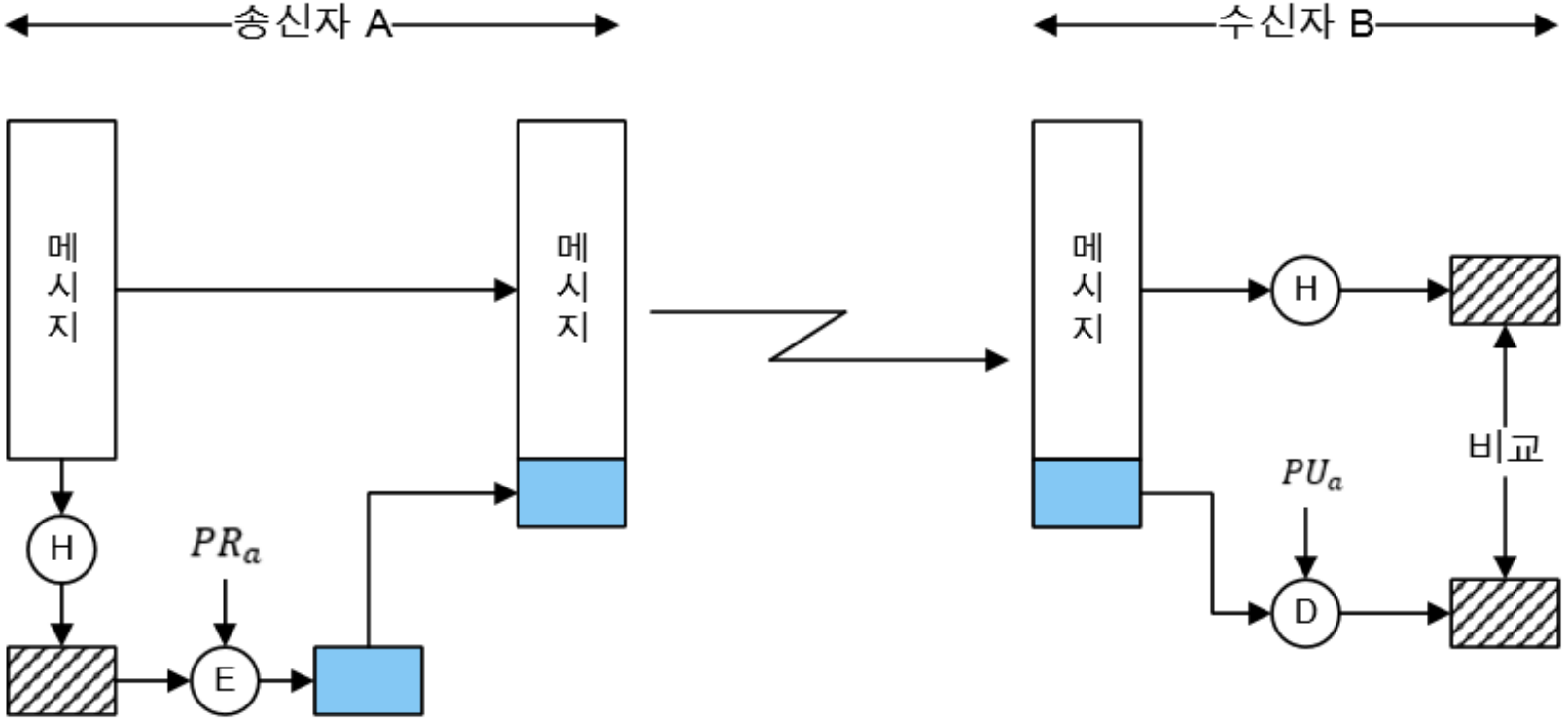
메시지 인증 방법

- 인증 꼬리표(Authentication Tag)
  - 메시지 다이제스트(Message Digest)
    - 일방향 해시함수를 이용한 메시지 인증
      - 메시지 다이제스트에 관용 암호 이용



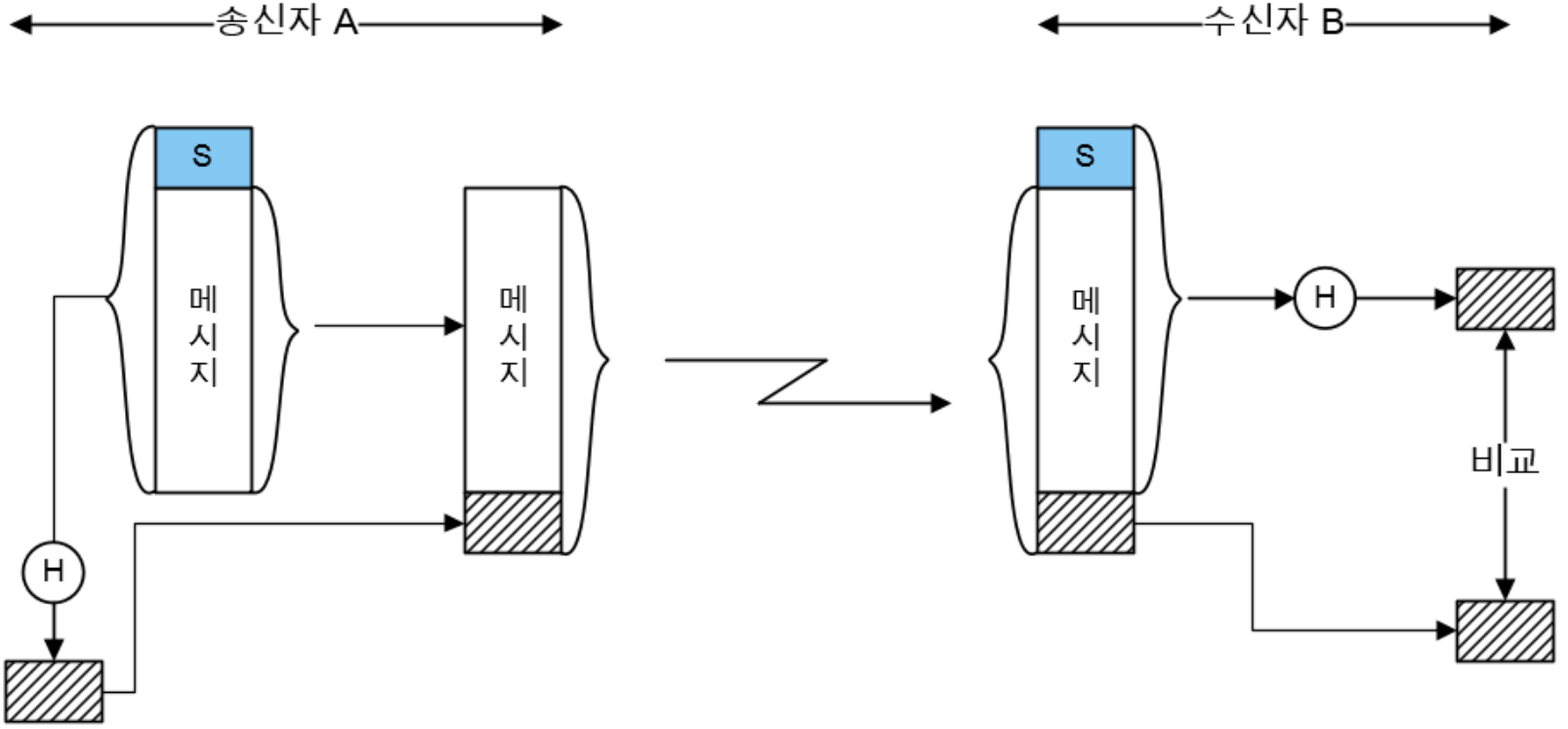
메시지 인증 방법

- 인증 꼬리표(Authentication Tag)
  - 메시지 다이제스트(Message Digest)
    - 일방향 해시함수를 이용한 메시지 인증
      - 메시지 다이제스트에 공개키 암호 이용



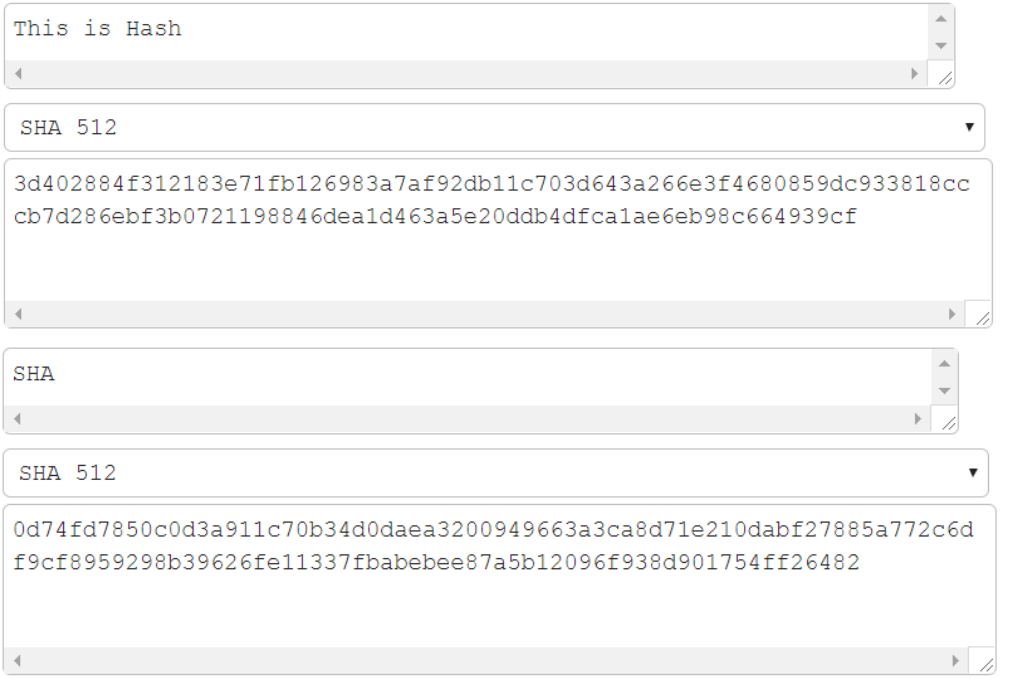
메시지 인증 방법

- 인증 꼬리표(Authentication Tag)
  - 메시지 다이제스트(Message Digest)
    - 일방향 해시함수를 이용한 메시지 인증
      - 메시지 다이제스트에 비밀 값 이용



안전 해시함수

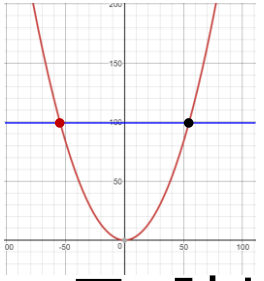
- 안전 해시함수(Secure Hash Function)
  - 의의
    - 임의의 길이의 데이터를 고정된 길이의 데이터로 매핑하는 함수
      - 파일, 메시지, 데이터 블록에 대한 '지문'을 생성
        - 일방향 성질, 고정된 길이
  - 대표적 알고리즘
    - MD5, SHA



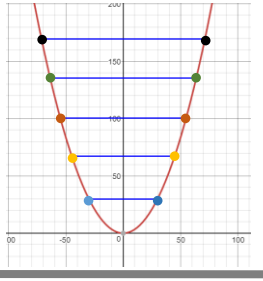
안전 해시함수

- 해시함수의 요건

1. 어떠한 크기의 데이터 블록에도 적용될 수 있어야 함
2. 일정한 길이의 출력을 생성해야 함
3. $H(x)$는 어떤 x 에 대해서도 계산이 쉬워야 함
4. $H(x) = h$가 성립되는 x를 계산하는 것이 불가능해야 함
5. 주어진 블록 x에 대해 $H(x) = H(\mathrm{y})$를 만족하는 $\mathrm{y}(\neq x)$를 찾는 것이 불가능해야 함
   (약한 충돌 저항성)



1. $H(x) = H(\mathrm{y})$를 만족하는 쌍 (x, y)를 찾는 것이 계산적으로 불가능해야 함
   (강한 충돌 저항성)



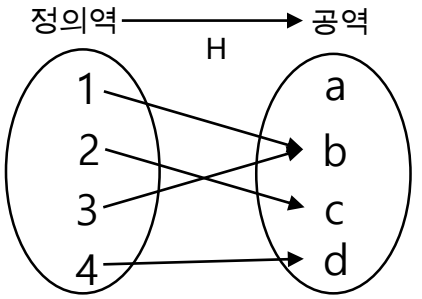
안전 해시함수

- 해시함수의 요건

1. 어떠한 크기의 데이터 블록에도 적용될 수 있어야 함
2. 일정한 길이의 출력을 생성해야 함
3. H(x)는 어떤 x에 대해서도 계산이 쉬워야 함
4. H(x) = h가 성립되는 x를 계산해내는 것이 불가능해야 함
5. 주어진 블록 x에 대해 H(x)=H(y)를 만족하는 y(≠x)를 찾는 것이 불가능해야 함
6. H(x)=H(y)를 만족하는 쌍(x,y)를 찾는 것이 계산적으로 불가능해야 함

조건	특징
1~3	• 메시지 인증에 응용하기 위해 필요
4	• 일방향 성질(One-Way Property) • 프리이미지 저항성(Preimage Resistance) • 비밀값을 이용하는 경우 매우 중요
5	• 2차 프리이미지 저항성(Second-Preimage Resistance) • 약한 충돌 저항성(Weak Collision Resistance)
6	• 강한 충돌 저항성(Strong Collision Resistance) • 생일 공격(Birthday Attack) 방지
1~5 만족	• 약한 해시함수(Weak Hash Function)
1~6 만족	• 강한 해시함수(Strong Hash Function)

- 프리이미지 (원상)
  - 어떤 함수에 대한 공역에 대응하는 정의역
    - e.g.,



일 때,
1과 2는 b의 프리이미지
3은 c의 프리이미지
4는 d의 프리이미지

안전 해시함수

- 해시함수 보안
  - 전수 공격(Brute Force Attack)에 대한 해시함수의 강도는 해시코드의 길이에 의존

공격 이용 성질	(n-비트에 대하여) 최대 계산 수
프리이미지 저항성	2^n
2차 프리이미지 저항성	2^n
충돌 저항성	$2^{n/2}$

단순 해시함수

- 단순 해시함수(Simple Hash Function)
  - 세로 덧붙임 검사(Longitude Redundancy Check)
    - $C_1 = b_{i1} \oplus b_{i2} \oplus \cdots \oplus b_{im}$
      - C_1 = 해시코드의 i번째 비트, 1≤ i ≤n
      - m = 입력의 n비트 블록의 수
      - $b_{ij} = j$번째 블록의 i번째 비트
      - $\oplus$ = XOR 연산

	비트1	비트2	…	비트n
블록1	b_{11}	b_{21}		b_{n1}
블록2	b_{12}	b_{22}		b_{n2}
…	…	…	…	…
블록m	b_{1m}	b_{2m}		b_{nm}
해시 코드	C_1	C_2		C_n

단순 해시함수

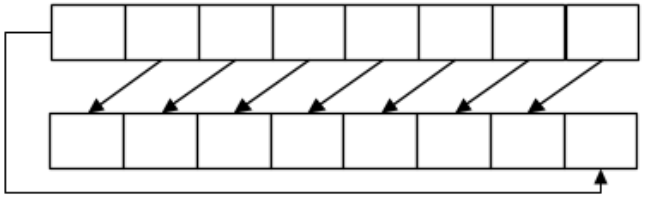
- 단순 해시함수(Simple Hash Function)
  - 세로 덧붙임 검사(Longitude Redundancy Check)
    - e.g.,

11100111
11011101
00111001

00000011 ← LRC

- 무결성 검사에 효과적
- 데이터 형식을 예측할 수 있는 경우 해시함수 효과가 떨어짐

1. n비트(각 블록의 마지막 비트) 해시 값을 계속 0으로 초기화
2. 뒤에 이어지는 n비트 데이터 블록에 대해 현재 해시 값을 왼쪽으로 한 비트씩 이동
3. 블록을 해시 값에 XOR



안전 해시함수

- SHA(Secure Hash Algorithm) 안전 해시함수
  - 미국 국가안보국(National Security Agency)에서 처음 설계
  - 가장 널리 사용되는 해시함수
  - MD4, MD5와 비슷한 구조를 가지나 big 엔디안 방식 사용
  - 종류
    - SHA-1(1995년 출판)
      - SHA-0(1993년 출판) 방식에 비트 회전 연산을 하나 추가
      - 160비트의 해시 값 생성
    - SHA-2(2002년 출판)
      - 생성하는 해시 값의 길이에 따라 SHA-224, SHA-256, SHA-384, SHA-512

안전 해시함수

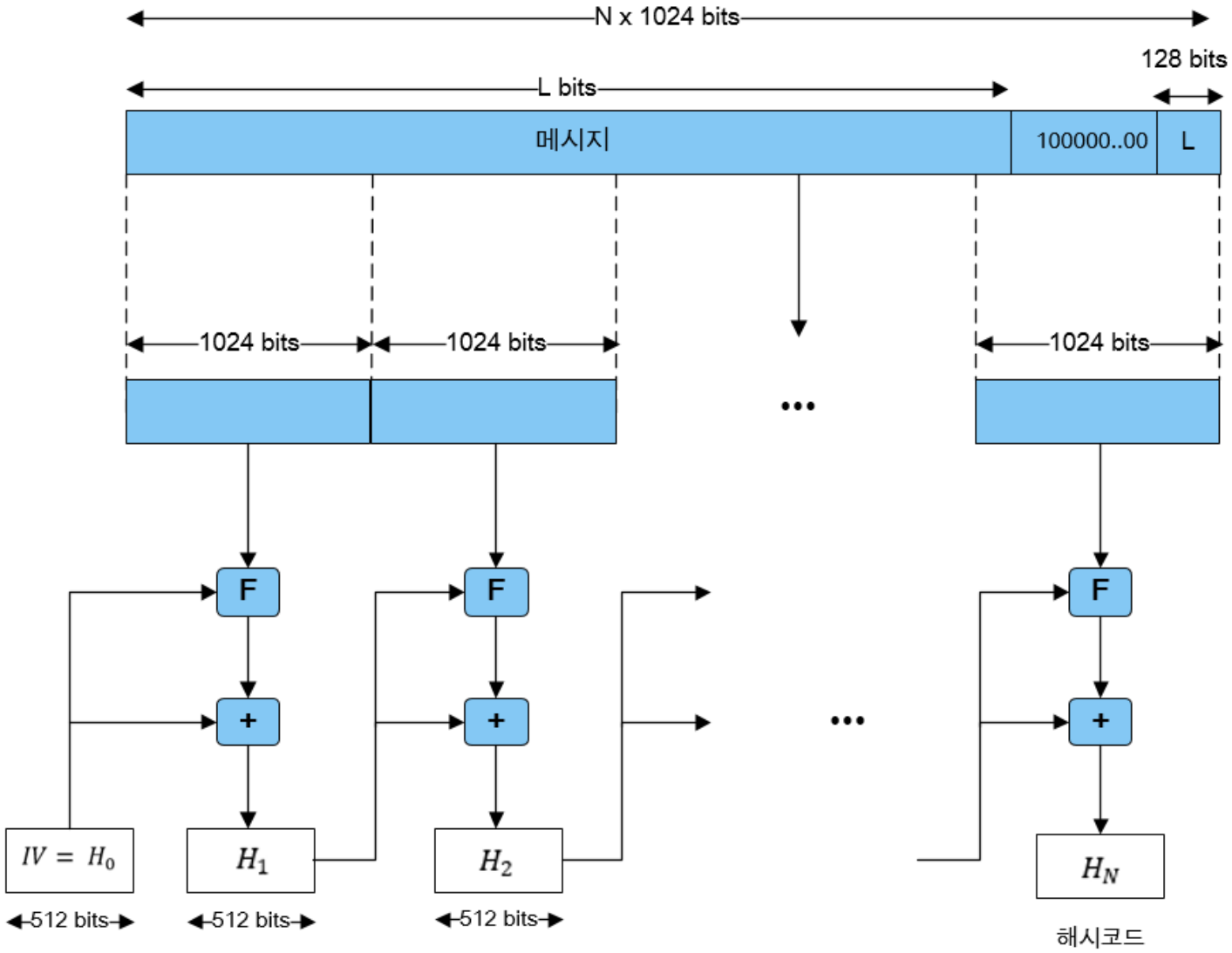
- SHA 비교

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
메시지 다이제스트 길이 (bit)	160	224	256	384	512
메시지 길이 (bit)	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
블록 길이 (bit)	512	512	512	1024	1024
워드 길이 (bit)	32	32	32	64	64
단계 수	80	64	64	80	80
보안	2^{80}	2^{112}	2^{128}	2^{192}	2^{256}

- 보안
  - 길이가 n인 메시지 다이제스트에 대해 생일공격에서 충돌이 한 번 나타나는데 필요한 최대 계산 수 $(2^{n/2})$

안전 해시함수

- SHA 512



- 동작 과정

1. 패딩 비트 붙이기 (Appending Padding Bits)

2. 길이 붙이기(Append Length)

3. MD버퍼 초기화(Initialize MD Buffer)

4. 1024 비트 (64 워드) 블록 메시지 처리 (Process Message in 1024 Bits (128 Words) Blocks)

5. 출력(Output)

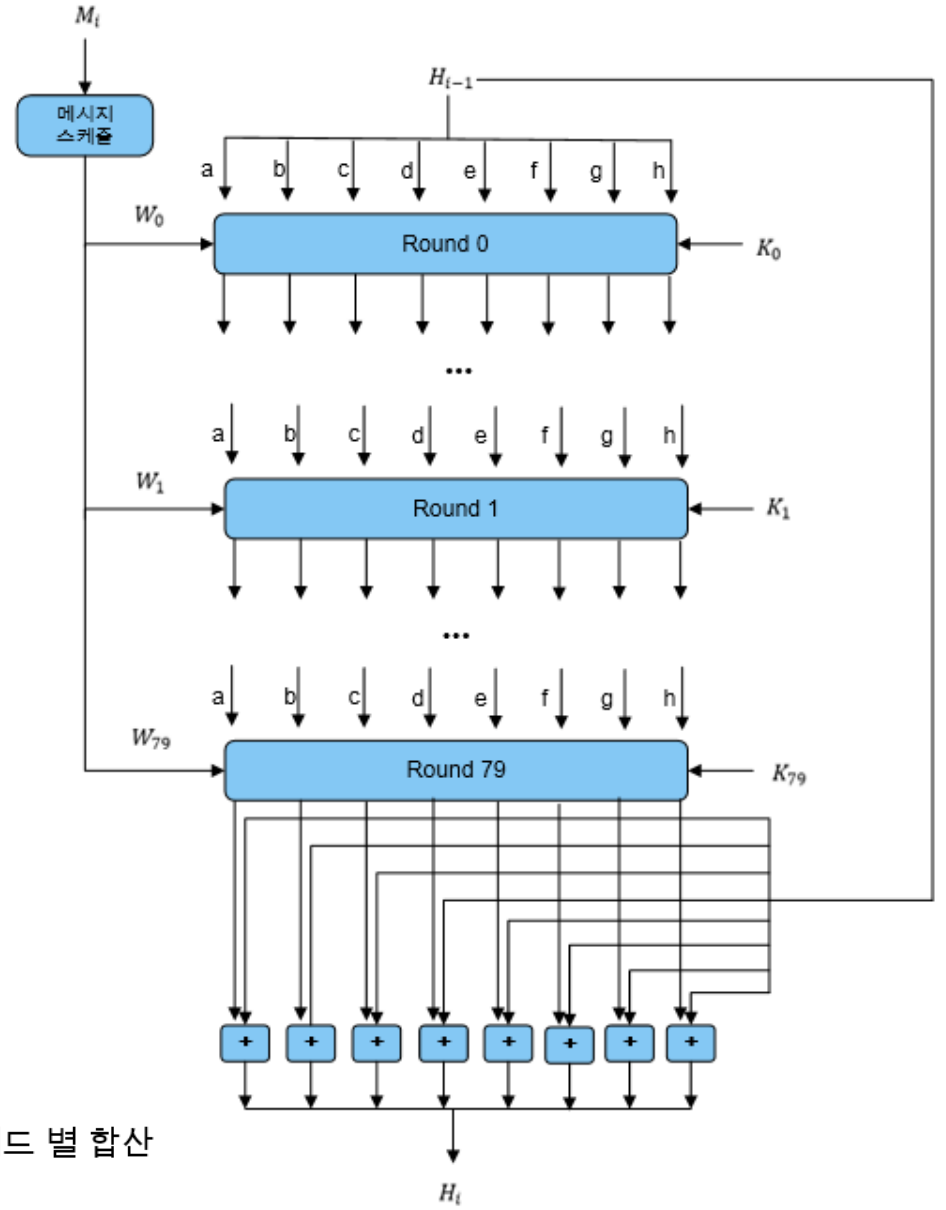
안전 해시함수

- SHA 512
  - 단계 3
    - MD버퍼 초기화
      - 512비트 버퍼를 해시함수의 중간 값과 최종 값 저장에 사용
      - 8개의 64비트 레지스터 a, b, c, d, e, f, g, h로 나뉨
      - 각 레지스터는 처음 소수 8개의 제곱근 소수점 이하 64비트를 16진수로 나타낸 것

레지스터	초기화 값
a	6A09E667F3BCC908
b	BB67AE8584CAA73B
c	3C62F372FE94F82B
d	A54FF53A5F1D36F1
e	510E527FADE682D1
f	9B05688CEB326C1F
g	1F83D9ABFB41BD6B
h	5BE0CDI9137E2197

안전 해시함수

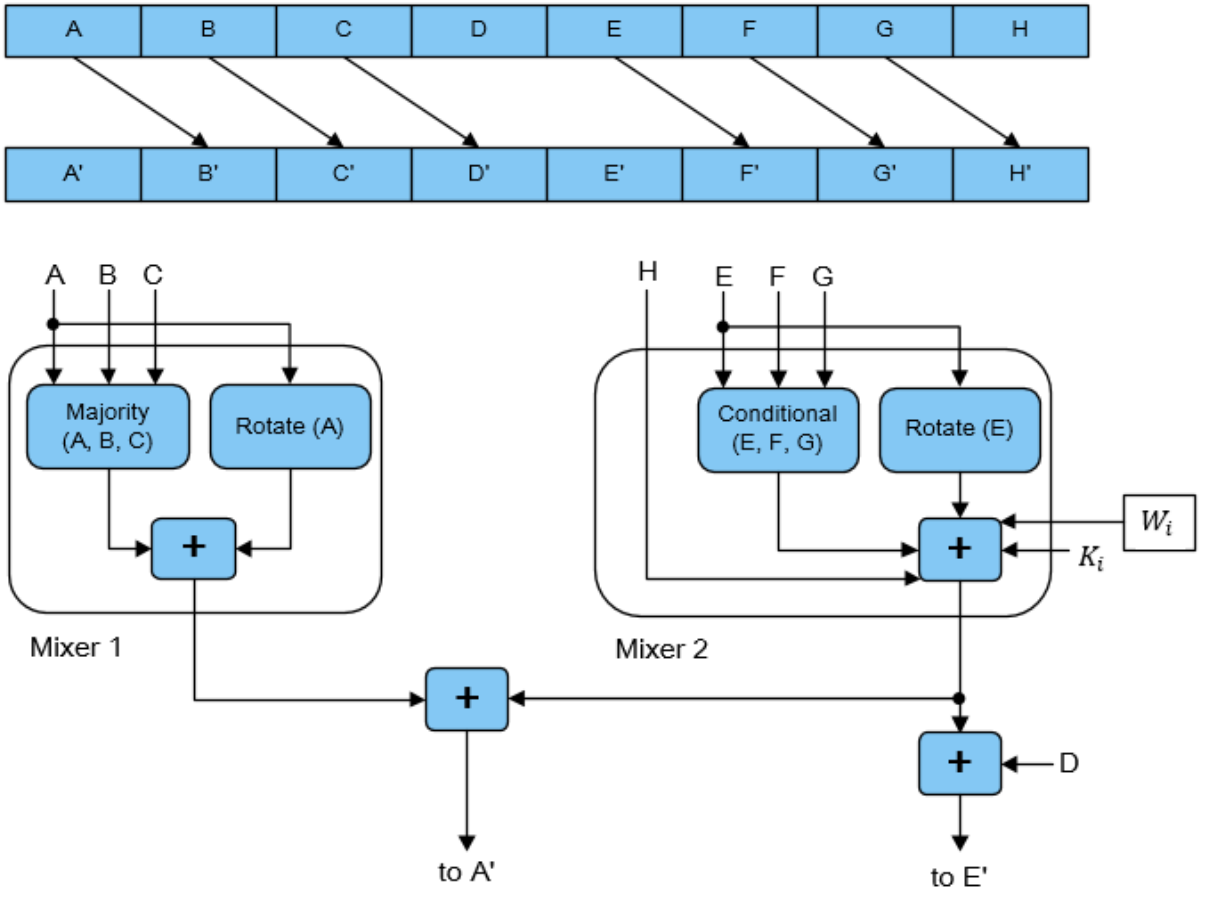
- SHA-512
  - F 와 +
    - 1024 비트 블록 메시지 처리
      - Round 0~79까지 총 80라운드
        - 하드웨어 상에서 동일한 회로 연속적으로 80번 사용 가능
      - 해시 길이에 따라 라운드 수 다름
      - 덧셈상수 K는 80개의 소수 세제곱근을 2진수로 바꾼 것의 최초 64비트
      - W_t는 1024비트 블록인 M_i로부터 구한 64비트 값



+ : mod 2^{64} 로 워드 별 합산

안전 해시함수

- SHA-512
  - F
    - 각 라운드의 동작 과정



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) ⊕ $ROTR_{34}$ (x) ⊕ $ROTR_{39}$ (x)

+ : mod 2^{64} 로 워드 별 합산

$ROTR_i$ (x) : X비트 만큼 오른쪽으로 shift

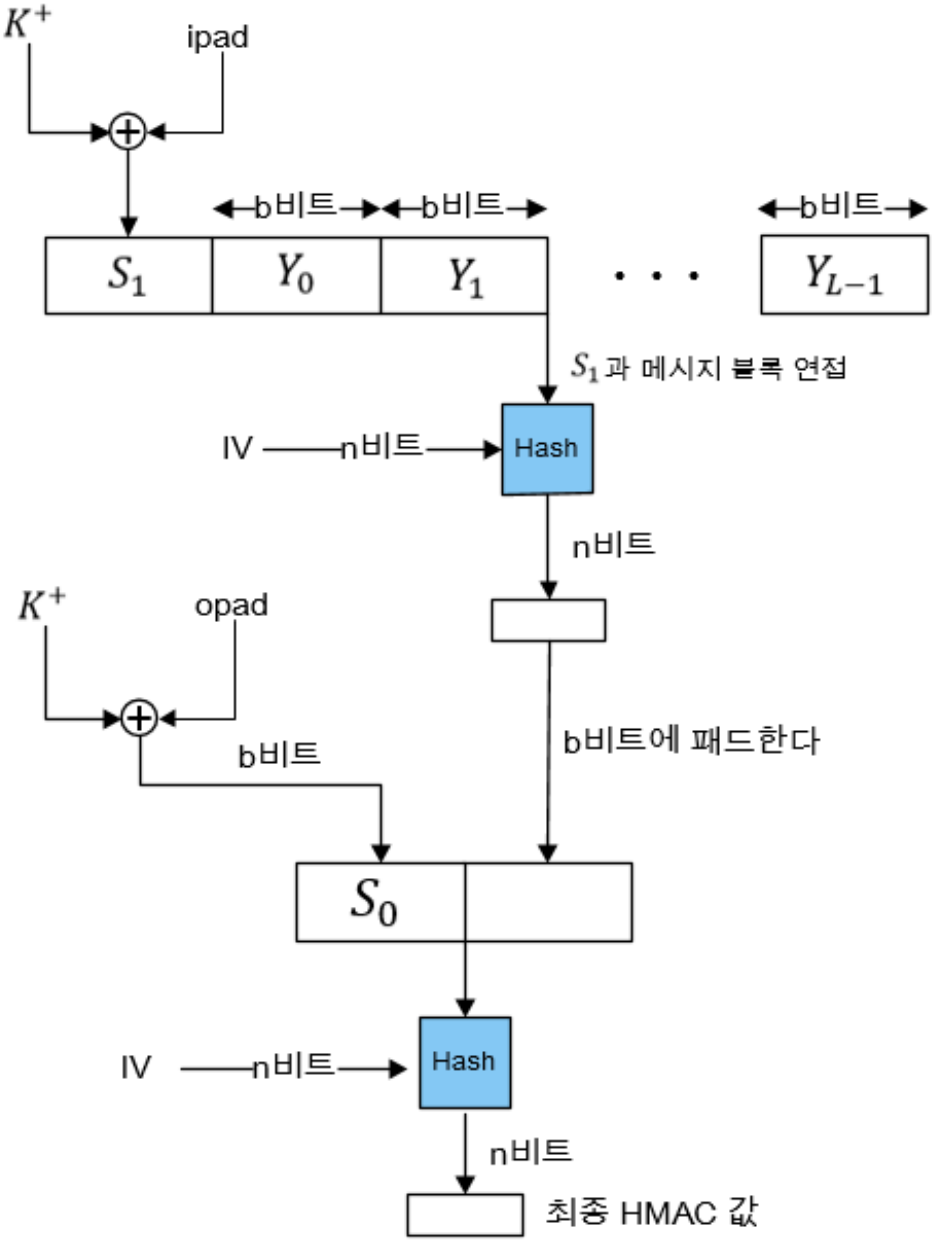
메시지 인증 코드

- HMAC(Hash-based MAC)
  - 의의
    - 암호화된 해시코드를 사용해 MAC을 생성하는 알고리즘
  - 설계 목표
    - 수정하지 않고 사용할 수 있는 해시함수여야 함
    - 소프트웨어에서 잘 작동되어야 하며 무료로 배포되어야 함
    - 더 빠르고 안전한 해시함수 개발 시 기존의 해시함수가 쉽게 대체되어야 함
      - 해시함수를 HMAC 구현에 있어 모듈로 사용

메시지 인증 코드

- HMAC(Hash-based MAC)
  - 동작 과정

용어	정의
H	내장된 해시함수 (e.g., SHA-1)
M	HMAC의 입력 메시지 (내장 해시함수에 필요한 패딩까지 포함)
Y_1	M의 i번째 블록, 0 ≤ i ≤ (L-1)
L	M의 블록 수
b	블록의 비트 수
n	내장된 해시함수에 의해 생성된 해시코드의 길이
K	비밀키, 키의 길이가 b보다 길면 n비트 키를 생성하는 해시함수에 입력으로 사용됨
K^+	K의 왼쪽에 0을 붙여서 길이가 b비트가 되도록 한 것
ipad	00110110(16진수 36)을 b/8번 반복한 2진 수열
opad	01011100(16진수 5C)을 b/8번 반복한 2진 수열



메시지 인증 코드

- 블록암호 기반 MAC
  - 암호기반 메시지 인증코드(Cipher-based MAC, CMAC)
    - 의의
      - 대칭키와 결합된 블록암호를 사용하여 메시지 인증 코드를 계산
    - 사용하는 암호 알고리즘
      - AES, 3DES
    - 구분
      - 메시지가 암호 블록 길이 b의 n배 정수 길이인 경우
        - k비트 암호키 K와 n(나눈 블록 수)비트 키 K_1
        - AES의 경우 b=128, k는 128 또는192 또는 256
        - 3DES의 경우 b=64, k는 112또는 168
      - 메시지가 암호 블록 길이의 정수배가 아닌 경우
        - 마지막 블록의 오른쪽에 패딩을 붙여 b비트가 되도록 함
        - K_1대신 K_2 사용

메시지 인증 코드

- 블록암호 기반 MAC
  - 메시지가 암호블록 길이 b의 n배 정수 길이인 경우
    - 계산 과정
      - $C_1 = \mathrm{E}(\mathrm{K}, M_1)$

        $C_2 = \mathrm{E}(\mathrm{K}, [M_2 \oplus C_1])$

        $C_3 = \mathrm{E}(\mathrm{K}, [M_3 \oplus C_2])$

        .

        .

        .

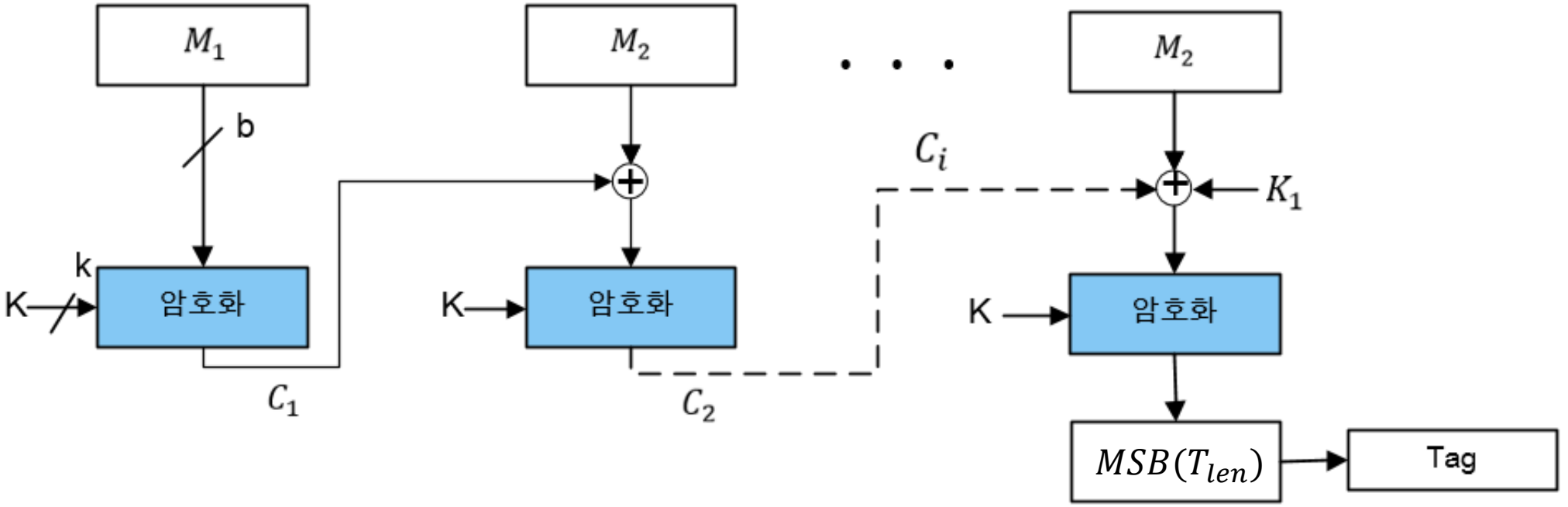
        $C_n = \mathrm{E}(\mathrm{K}, [M_N \oplus C_{n-1} \oplus K_1])$

        $T = MSB_{T_{len}}(C_n)$

용어	정의
T	메시지 인증 코드 또는 태그(tag)라고도 함
T_{len}	T의 비트 길이 (AES에서는 128비트, 3DES에서는 64비트)
$MSB_s(X)$	비트열 X의 왼쪽부터 s개 비트
K	k비트 암호키
K_1	n비트 키 (메시지 길이가 n의 정수 배 일 때) • 모든 비트가 0으로 이루어진 블록에 블록 암호 적용 • 그 결과로 나온 암호문을 한 비트씩 왼쪽으로 이동
K_2	n비트 키 (메시지 길이가 정수 배 아닐 때) • K_1에 블록암호 적용 • 그 결과로 나온 결과값을 한 비트씩 왼쪽으로 이동

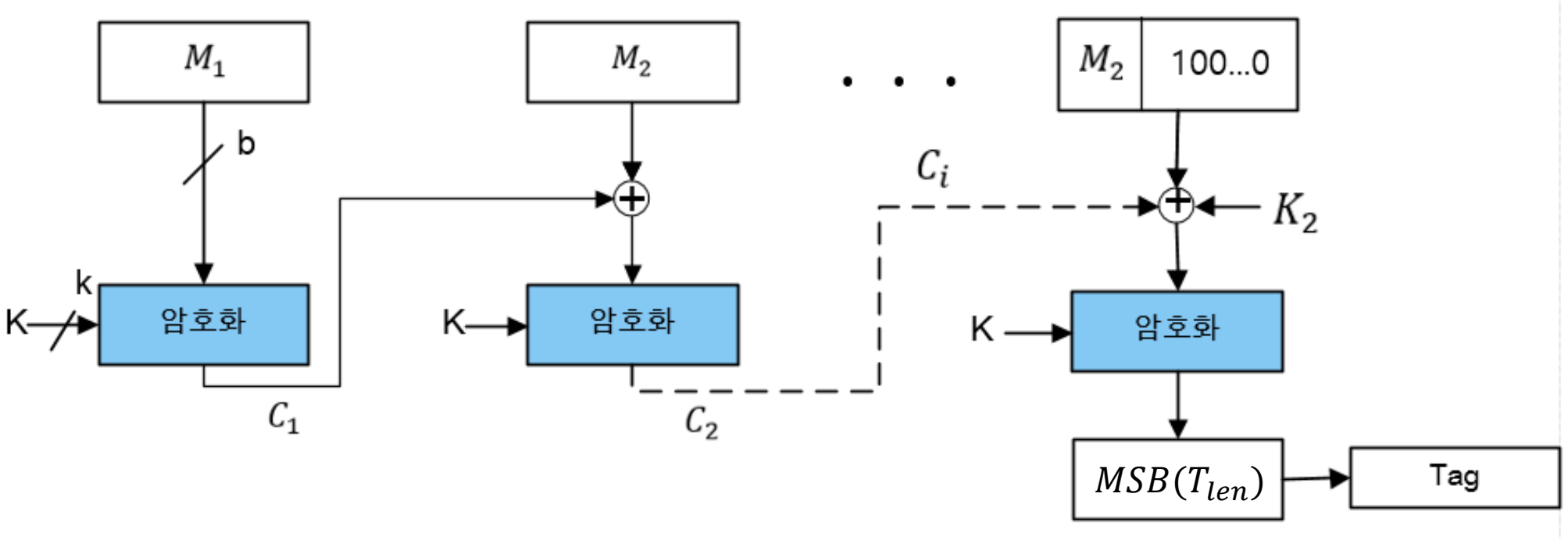
메시지 인증 코드

- 블록암호 기반 MAC
  - 메시지가 암호블록 길이 b의 n배 정수 길이인 경우
    - 작동 과정



메시지 인증 코드

- 블록암호 기반 MAC
  - 메시지가 암호 블록 길이의 정수배가 아닌 경우
    - 작동 과정



메시지 인증 코드

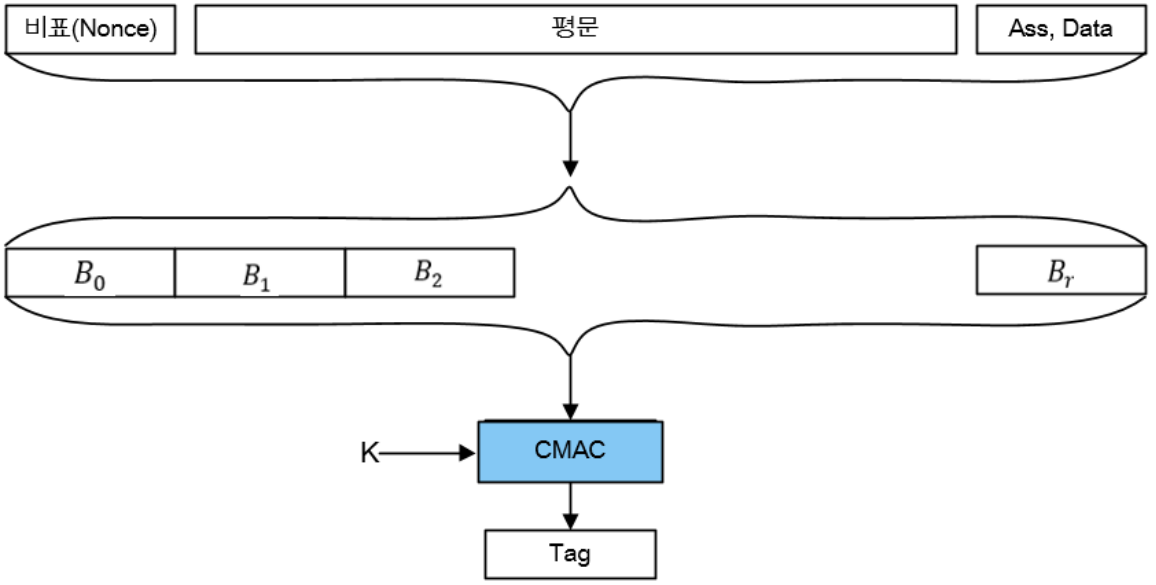
- 암호 블록 체인 카운터-메시지 인증 코드(Counter with Cipher Block Chaining MAC, CCM)
  - 운용 모드
    - 인증된 암호화 모드 (Authentication Encryption)
      - 통신 기밀성(암호화), 인증(무결성)을 동시에 보호하는 암호 시스템
      - 두 가지 서비스가 별개로 설계되어 제공
  - 핵심 알고리즘 요소
    - 인증
      - CMAC 알고리즘
    - 암호화
      - AES 알고리즘, CTR 운용 모드

메시지 인증 코드

- 암호 블록 체인 카운터-메시지 인증 코드(Counter with Cipher Block Chaining MAC, CCM)
  - 입력 요소
    - 인증하고 암호화 할 데이터(평문 메시지 데이터 블록 P)
    - 인증은 하지만 암호화는 하지 않는 유관 데이터 A (Associated Data)
      - e.g., 프로토콜 헤더
    - 페이로드와 유관 데이터에 할당되는 비표 N(Nonce)

메시지 인증 코드

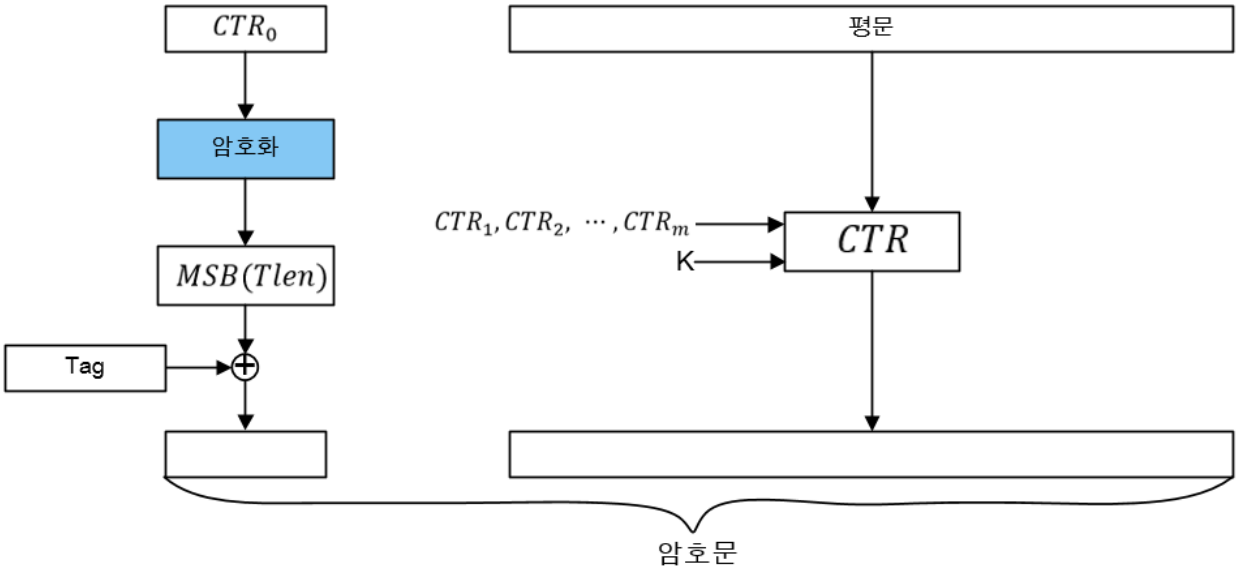
- 암호 블록 체인 카운터-메시지 인증 코드(Counter with Cipher Block Chaining MAC, CCM)
  - 동작 과정
    - 인증



1. 비표(Nonce), 유관 데이터(Associated Data), 평문(Plaintext) 입력
   - 비표
     통신이 재사용될 수 없도록 한 번만 사용될 수 있는 임의의 수
   - 유관 데이터
     인증은 필요하지만 암호화는 되지 않는 데이터
2. 입력 값을 B_0부터 B_r까지 블록 열 형식으로 나타냄
   - 첫 번째 블록에는 각 N, A, P의 길이를 나타내는 형식비트 추가
3. 뒤에 0또는 A를 포함하는 여러 개의 블록 연결
4. 뒤에 0또는 P를 포함하는 여러 개의 블록 연결
5. 블록 열을 CMAC 알고리즘에 입력하면 길이가 T_{len}인 MAC값 생성 (T_{len} ≤ 블록 길이)

메시지 인증 코드

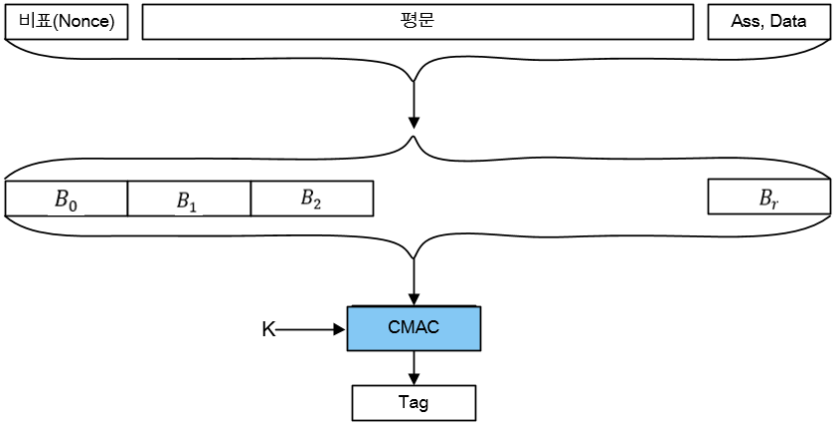
- 암호 블록 체인 카운터-메시지 인증 코드(Counter with Cipher Block Chaining MAC, CCM)
  - 동작 과정
    - 암호화



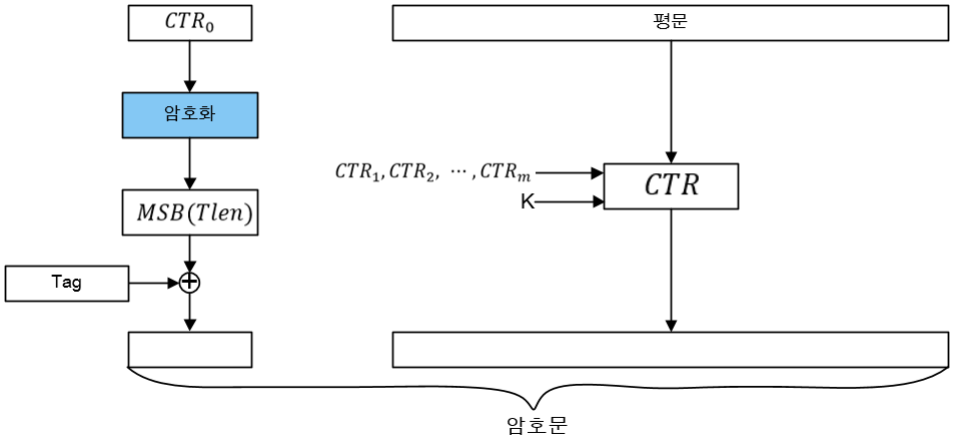
1. 비표와 독립적으로 카운터 열 생성
2. 인증 태그를 Ctr_0를 이용해 CTR모드로 암호화
3. Tlen 개 만큼의 비트를 태그와 XOR하여 암호화 태그 생성
4. 남은 Ctr_i을 사용하여 평문을 CTR모드로 암호화
5. 암호화된 평문을 암호화된 태그에 이어 붙여 암호문으로 출력

메시지 인증 코드

- 암호 블록 체인 카운터-메시지 인증 코드(Counter with Cipher Block Chaining MAC, CCM)
  - 전체 동작 과정



<인증>



<암호화>

감사합니다!