2020/01/20 @ 상명대학교 2020보안기초세미나

Network Security Essential

- Chapter 3 공개키 암호와 메시지 인증(1) -

박 재 형(jaehyoung@pel.smuc.ac.kr)

상명대학교 프로토콜공학연구실

목 차

- 메시지 인증 방법
- 안전 해시 함수
- 메시지 인증 코드

메시지 인증 방법

- 메시지 인증 (Message Authentication)
  - 정의
    - 통신을 할 때 메시지 내용의 출처나 무결성을 보장하는 방법
  - 특징
    - MAC를 이용한 방식
    - 해시 함수를 이용한 방식

메시지 인증 방법

- 대칭키 암호를 이용한 인증
  - 송신자와 수신자가 동일한 키를 가지고 있다고 가정
    - 송신자가 메시지에 MAC를 포함 시킴으로써 수신자에게 자신이 송신자 임을 인증하는 것

메시지에 포함된 것	수신자가 확인 할 수 있는 것
오류 감지 코드	메시지의 변경 여부
순서 번호	메시지의 순서
타임스탬프	메시지의 고의적 시간지연

메시지 인증 방법

- 메시지 암호화 없는 메시지 인증 (Message authentication without message encryption)
  - 메시지 자체는 암호화 하지 않고 MAC를 생성하여 메시지에 붙여서 보내는 것
    - 기밀성이 보장되지 않음
    - 메시지 인증 코드 (MAC: Message Authentication Code)
      - 메시지 끝에 연결되는 메시지 인증을 위한 코드
      - $MAC_M = F(K_{AB}, M)$
      - M = 메시지, F = 인증코드 함수, K_{AB} = A와 B의 공유키
    - 메시지 다이제스트 (Message Digest)
      - 메시지를 일방향 해시 함수의 입력으로 사용 할 때의 출력 값

메시지 인증 방법

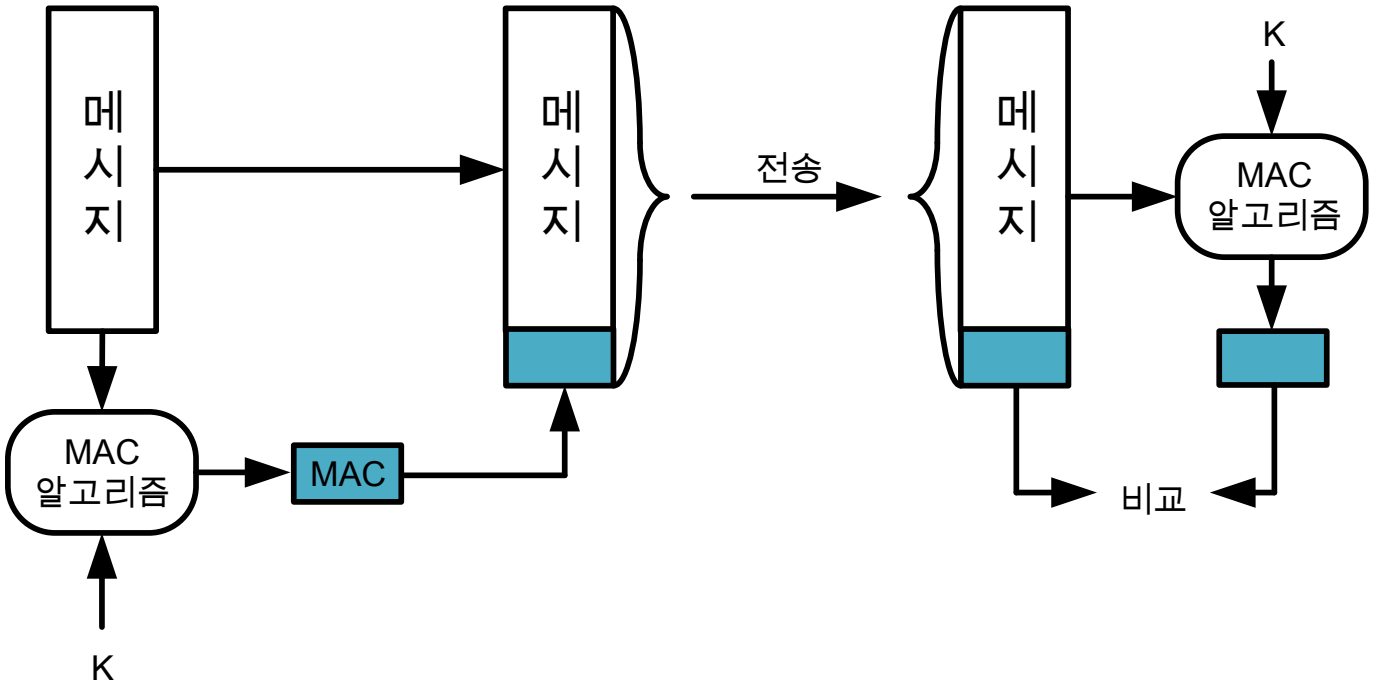
- 메시지 암호화 없는 메시지 인증을 사용하는 경우
  - 브로드 캐스팅 하는 경우
    - e.g.,네트워크 경고신호를 사용자에게 보낼 때
  - 메시지 교환시 과부하가 걸렸을 때
    - 테스트를 위해 임의의 메시지를 선택해 인증 수행
  - 컴퓨터 프로그램을 평문 자체로 인증
    - 프로그램을 매번 복호화 하지 않고 실행
    - 확신이 필요할 경우 무결성 확인

메시지 인증 방법

- 메시지 암호화 없이 메시지 인증 방법
  - 메시지 인증 코드 (MAC: Message Authentication Code)
    - 계산된 코드와 수신된 코드가 동일하다면
      - 수신자는 메시지가 변경되지 않았다는 것을 확신
      - 수신자는 메시지가 송신자라고 주장하는 근원지로부터 송신되었다는 것을 확신
  - 일방향 해시 함수 (One-way hash function)
    - 메시지를 입력 받아 메시지 다이제스트 (Message digest)을 출력하는 함수
      - 비밀키를 입력으로 사용하지 않음
    - 메시지 다이제스트를 메시지와 함께 전송

메시지 인증 방법

- 메시지 암호화 없이 메시지 인증 방법
  - 메시지 인증 코드 (MAC: Massage Authentication Code)
    - 메시지의 무결성과 데이터의 출처 확인을 보장하는 방법
    - $MAC_M = F(K_{AB}, M)$
    - M = 메시지, F = 인증코드 함수, K_{AB} = A와 B의 공유키

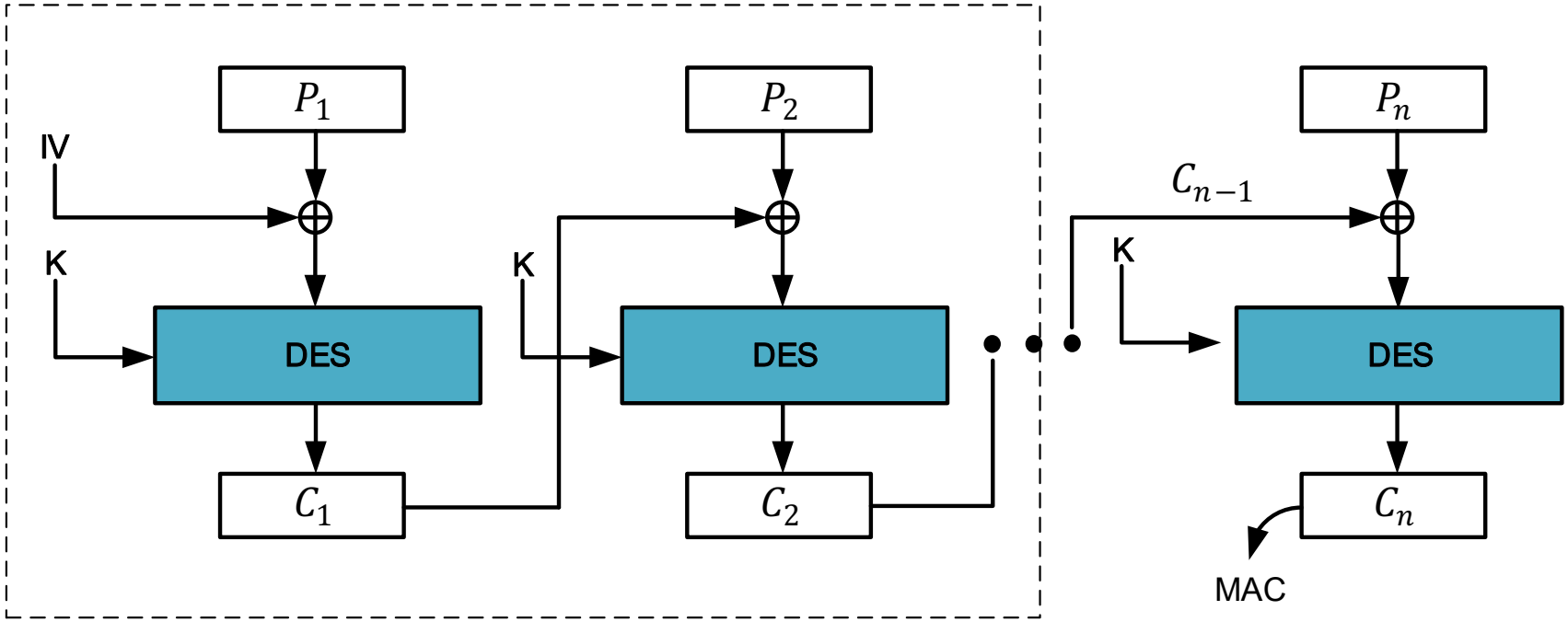


메시지 인증 방법

- 메시지 암호화 없이 메시지 인증 방법
  - MAC알고리즘 종류
    - DES, 3DES, AES, CBC모드 등 여러 알고리즘을 통해 MAC를 생성할 수 있음
      - NIST명세 FIPS PUB 113은 DES사용 권장
        - DES를 이용해서 메시지를 암호화 하고 암호문의 끝부분에 있는 여러 비트를 코드로 사용

메시지 인증 방법

- 메시지 암호화 없이 메시지 인증 방법
  - CBC모드를 이용한 MAC생성과정
    - 마지막 블록 제외하고 나머지블록은 폐기(복호화 필요 없음)



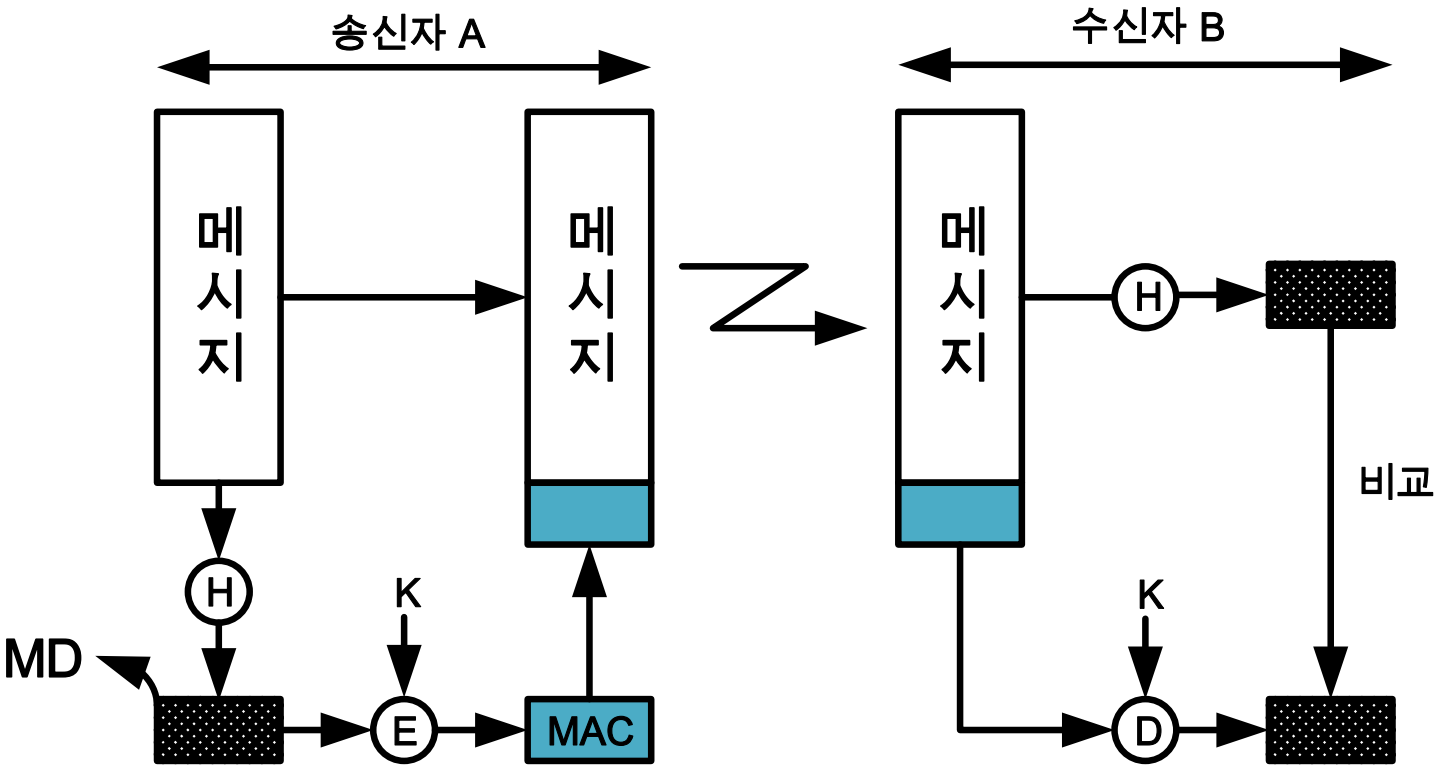
- 암호화에 DES사용

메시지 인증 방법

- 메시지 암호화 없이 메시지 인증 방법
  - 일방향 해시 함수 (One-way hash function)
    - 해시 함수 (Hash Function)
      - 임의의 길이 메시지를 입력 받아 고정된 길이의 해시 값을 출력하는 함수
      - 같은 입력에 대해서는 같은 값 출력
      - 비밀 키를 사용하지 않음
    - 일방향 해시 함수 사용으로 메시지를 인증하는 3가지 방법
      - 대칭키 암호 사용
      - 공개키 암호 사용
      - 비밀 값 사용

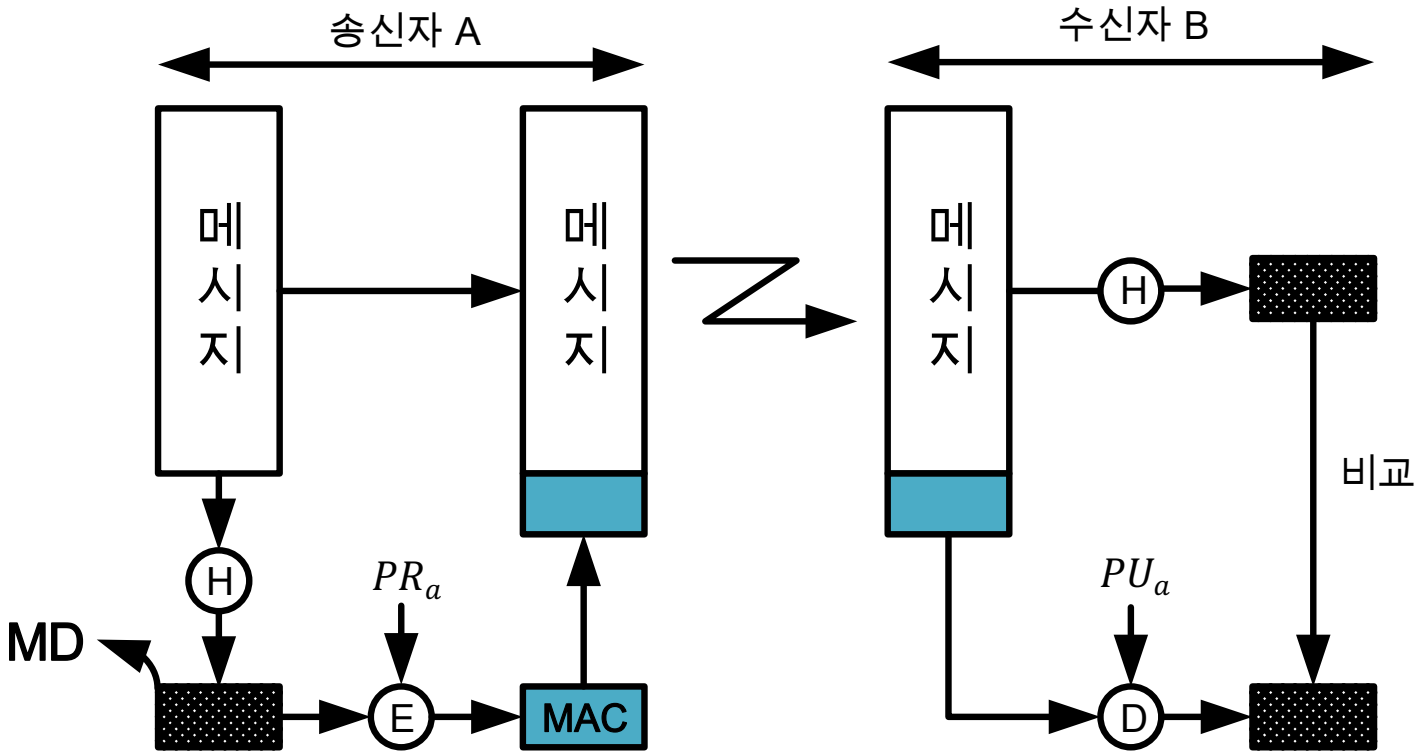
메시지 인증 방법

- 일방향 해시 함수 (One-way hash function)
  - 메시지를 인증하는 3가지 방법
    - 대칭키 암호 사용
      - 송신자와 수신자만이 동일한 키를 소유한다고 가정



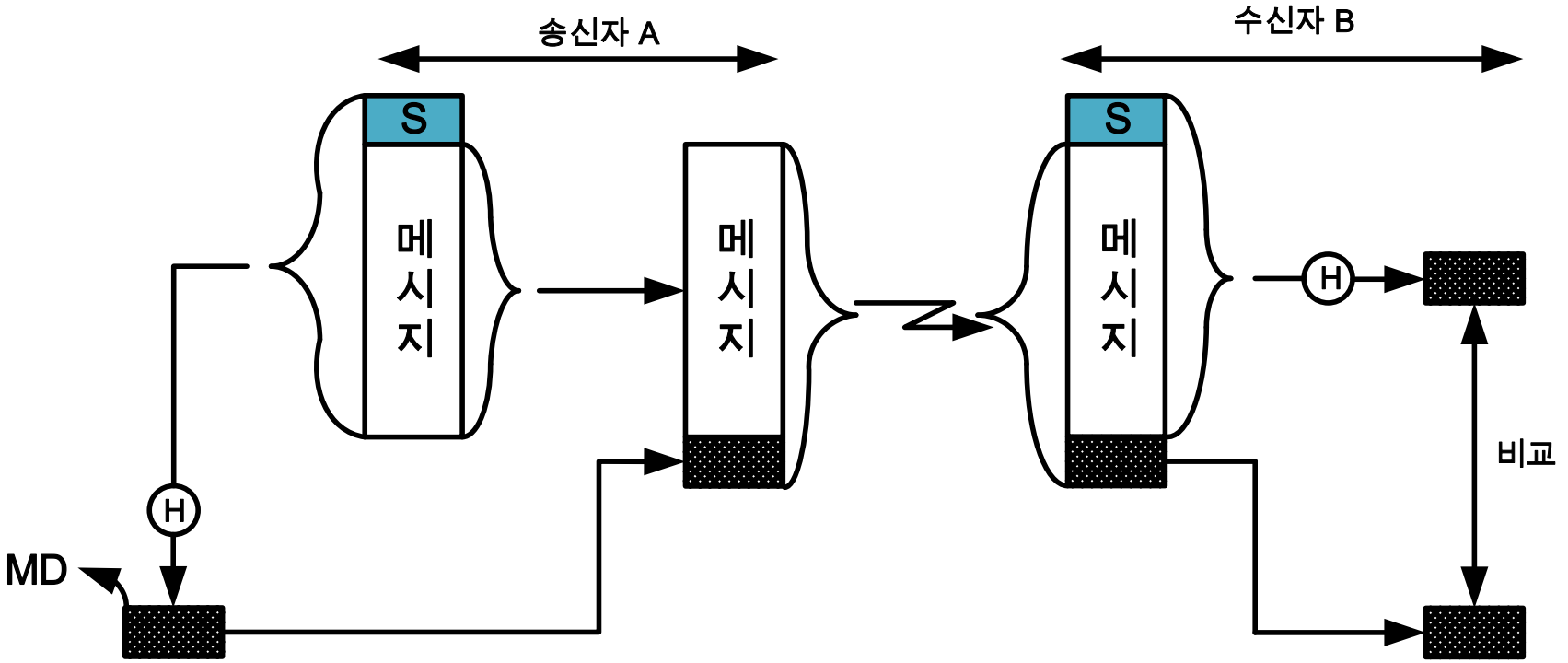
메시지 인증 방법

- 일방향 해시 함수 (One-way hash function)
  - 메시지를 인증하는 3가지 방법
    - 공개키 암호 사용



메시지 인증 방법

- 일방향 해시 함수 (One-way hash function)
  - 메시지를 인증하는 3가지 방법
    - 비밀 값 사용
      - 송/수신자가 비밀 값을 가지고있다는 가정



안전 해시 함수

- 안전 해시 함수(Secure Hash Function)
  - 해시 값으로부터 원래 입력 값과의 관계를 찾기 어려운 성질을 가진 해시 함수

- 해시 함수 (Hash Function)요건
  - 어떠한 크기의 데이터 블록에도 적용될 수 있어야 함
  - 일정한 길이의 출력을 생성
  - $\boldsymbol{H}(x)$는 어떠한 x에 대해서도 계산이 쉽고 하드웨어나 소프트웨어적으로 구현할 수 있어야 함
  - 일방향 성질(One-Way Property)을 가져야 함
    - 주어진 값 $\boldsymbol{h}$에 대해서 $\boldsymbol{H}(x) = \boldsymbol{h}$가 성립되는 x를 찾는 것이 불가능 해야 함

안전 해시 함수

- 안전 해시 함수(Secure Hash Function)
  - 해시 값으로부터 원래 입력 값과의 관계를 찾기 어려운 성질을 가진 해시 함수

- 해시함수 (Hash Function)요건
  - 약한 충돌 저항성 (Weak Collision Resistance)
    - 어떤 블록 x가 주어졌을 때, $\boldsymbol{H}(x) = \boldsymbol{H}(y)$를 만족하는 $y(\neq x)$를 찾는 것이 불가능 해야 함
  - 강한 충돌 저항성 (Strong Collision Resistance)
    - $\boldsymbol{H}(x) = \boldsymbol{H}(y)$를 만족하는 두 입력 (x, y)를 찾는 것이 불가능 해야 함

안전 해시 함수

- 해시 함수 보안
  - 전수 공격에 대한 해시 함수의 강도는 해시코드의 길이에 달려 있음
  - 길이가 n – 비트인 해시코드에 대한 공격 난이도
    - 프리이미지 저항성
      - H(m)만을 가지고 또 다른 메시지 m을 찾기 어려워야 함
    - 2차 프리이미지 저항성 (약한 충돌 저항성)
      - 메시지를 위조 할 수 없다고 가정, 메시지m과 해시 값H(m)이 있을 때, H(m) = H(M)을 만족하는 M을 찾기 어려워야 함
    - 충돌 저항성 (강한 충돌 저항성)
      - 동일한 해시 값을 가지는 2개의 메시지를 구하지 못하는 성질(아무 정보 없이)

프리이미지 저항성	2^n
2차 프리이미지 저항성	2^n
충돌 저항성	$2^{n/2}$

안전 해시 함수

- 단순 해시 함수 (Simple Hash Function)
  - 해시함수의 간단한 형태
  - 전체 입력 데이터를 연속적인 n-비트 블록 으로 간주
    - 세로 덧붙임 검사 (Longitudinal Redundancy Check)
      - 각 비트별로 패리티를 계산하는 방법
      - $C_i = b_{i1} \oplus b_{i2} \oplus \bullet\bullet\bullet \oplus b_{im}$

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

안전 해시 함수

- 단순 해시 함수 (Simple Hash Function)
  - $C_i = b_{i1} \oplus b_{i2} \oplus \bullet\bullet\bullet \oplus b_{im}$
    - e.g., Data : 11100111, 11011101, 00111001, 10101001
      LRC : 10101010

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

안전 해시 함수

- SHA 안전 해시함수 (SHA: Secure Hash Algorithm)
  - 1993년 디지털 서명을 위해 NIST가 개발한 알고리즘
  - SHA-0
    - MD5의 취약점(암호방식이 MD5라면 복호화가 쉬움)개선을 위해 개발된 알고리즘
      - MD5과정
        - 패딩 비트 붙이기
        - 길이 붙이기
        - MD 버퍼 초기화
        - 16 Word 블록 메시지 처리
        - 출력
      - 1993년 NIST에 의해 안전한 해시 표준 (Secure Hash Standard, FIPS PUB 180)으로 출판

안전 해시 함수

- SHA 안전 해시함수 (SHA: Secure Hash Algorithm)
  - 1993년 디지털 서명을 위해 NIST가 개발한 알고리즘
  - SHA-1
    - SHA-0압축함수에 비트회전 연산을 추가한 알고리즘
    - 160 bits의 해시 값 출력
      - 1995년 개정된 알고리즘 (FIPS PUB 180-1)을 새로 출판
  - SHA-2
    - 다양한 길이의 해시 값을 갖기 위한 알고리즘
      - 2002년 해시 값의 길이가 더 긴 4개의 변형을 FIPS PUB 180-2로 출판
        - SHA-224, SHA-256, SHA-384, SHA-512

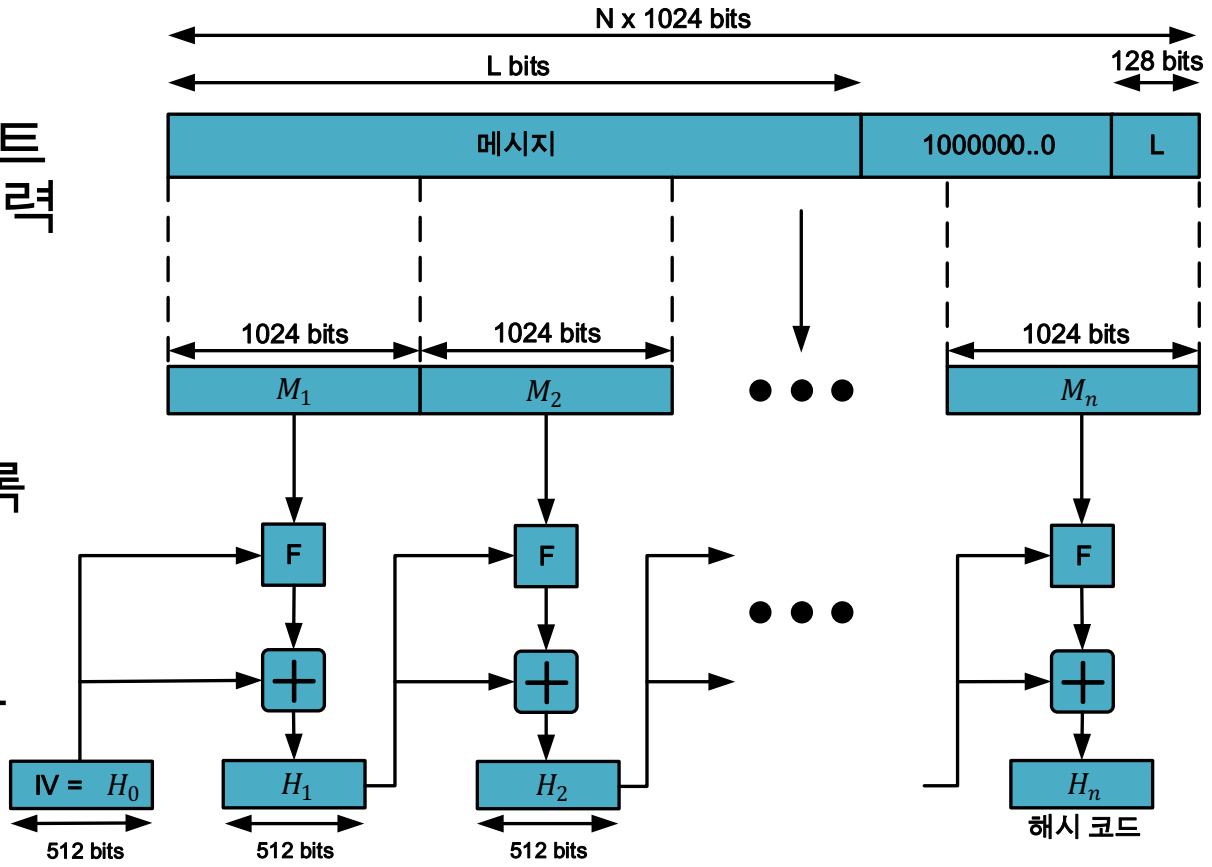
안전 해시 함수

- 안전 해시 알고리즘
  - SHA 매개변수 비교 표
    - 모든 길이의 단위는 비트

	SHA-0	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
메시지 다이제스트 길이	160	160	224	256	384	512
최대 메시지 길이	2^{64}	2^{64}	2^{64}	2^{64}	2^{128}	2^{128}
블록 길이	512	512	512	512	1024	1024
단어 길이	32	32	32	32	64	64
라운드 수	80	80	64	64	80	80

안전 해시 함수

- 안전 해시 알고리즘
  - SHA-512 구조
    - 과정 (1/2)
      1. 최대 길이가 2^{128}비트 이하인 메시지를 입력으로 사용
      2. 입력 데이터를 길이가 1024 비트인 블록으로 나누어 처리
      3. 512 bits의 IV초기화



안전 해시 함수

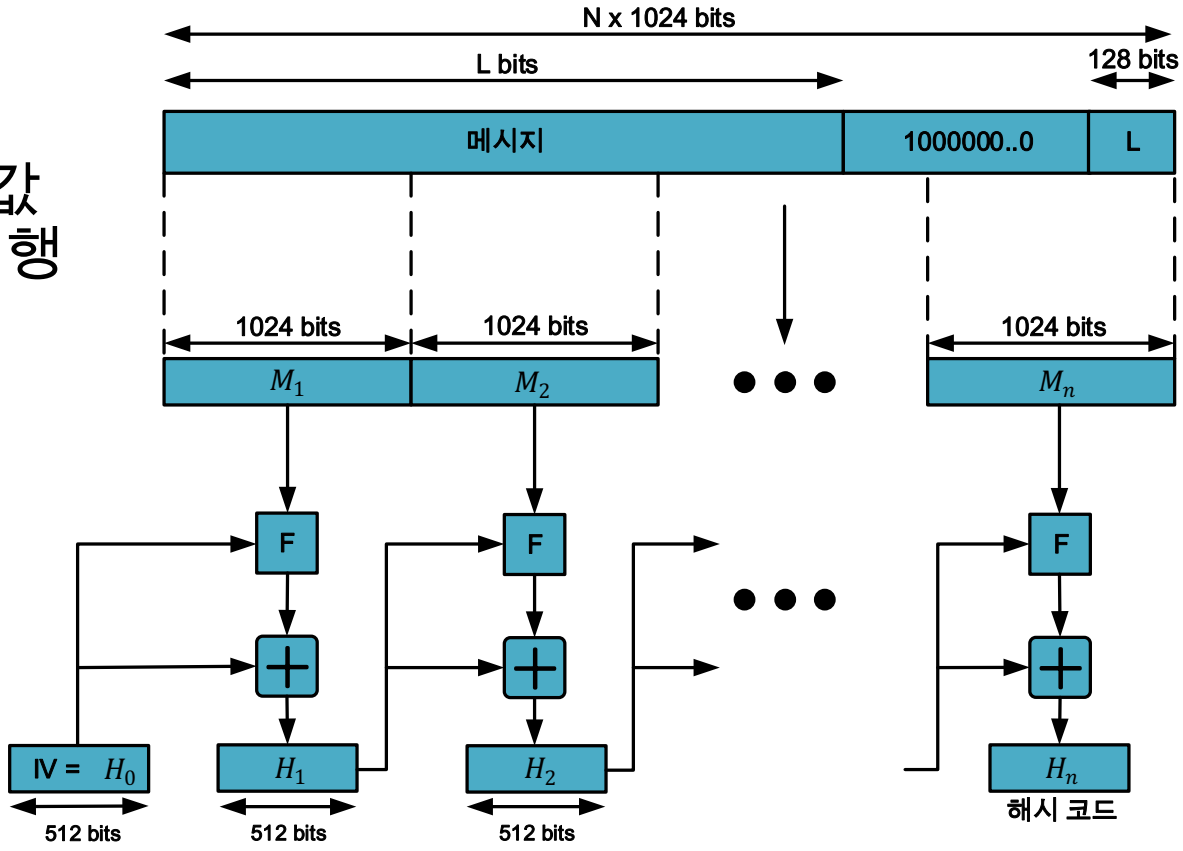
- 안전 해시 알고리즘
  - SHA-512 구조
    - 과정 (2/2)

4. M_1과IV의 입력으로 F함수통과 후 출력 값과 IV의 모듈연산 실행

5. 출력 값 H_1 생성

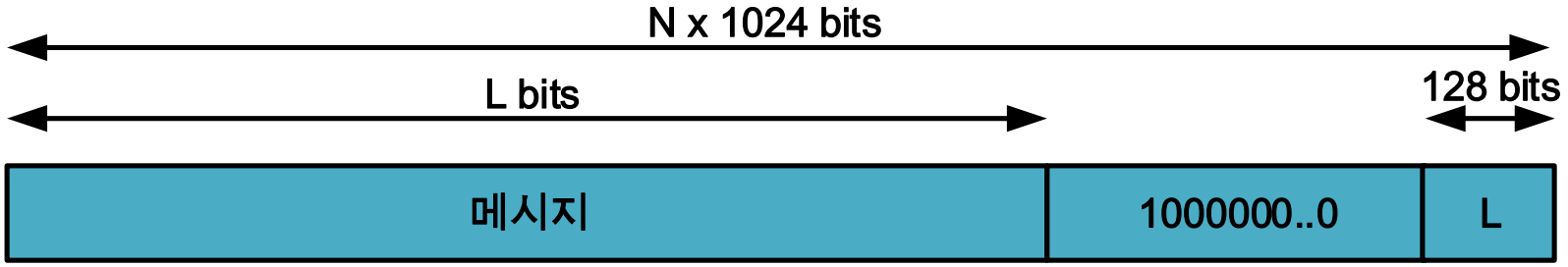
6. 4, 5단계를 반복 함으로써 N bits 처리

7. 생성된 출력 값들이 해시코드가 됨 $(H_0 \sim H_n)$



안전 해시 함수

- 안전 해시 알고리즘
  - SHA-512 구조
    1. 패딩 비트 붙이기 (Appending Padding bits)
       - 메시지에 패딩을 추가하여 총 길이를 896 bits (mod 1024) 가 되도록 만듦
       - 메시지의 길이가 이미 원하는 길이여도 패딩은 항상 추가
       - 패딩 시 첫 비트는 1, 나머지 비트는 0으로 채움
    2. 길이 붙이기 (Append Length)
       - 부호 없는 128비트 정수를 붙임으로 패딩 전 메시지 길이 표현



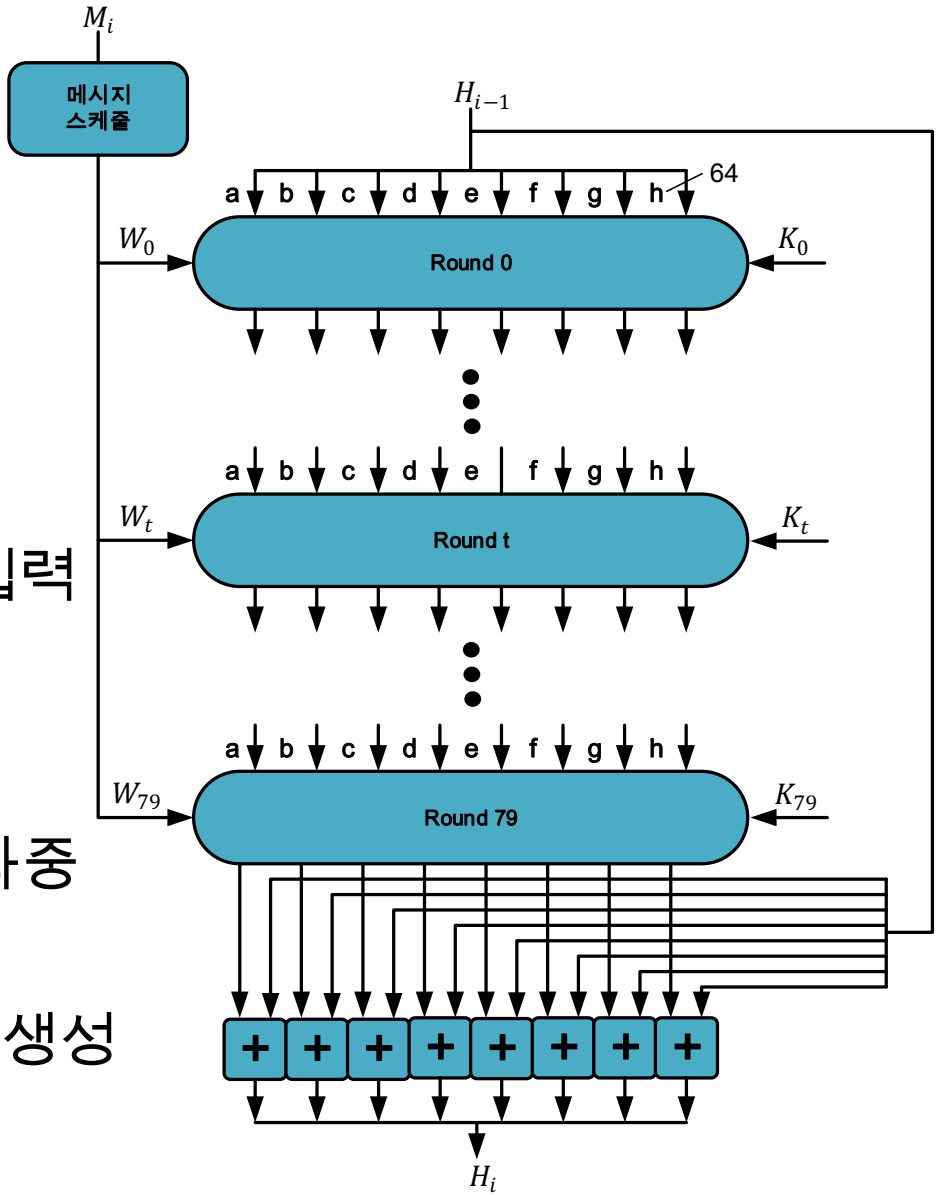
안전 해시 함수

- 안전 해시 알고리즘
  - SHA-512구조
    3. MD버퍼 초기화 (Initialize MD buffer)
       - 512 비트 해시 값 출력을 위해 사용
       - 버퍼를 8개의 64 비트 레지스터 (a, b, c, d, e, f, g, h)로 나타냄
         - 레지스터의 초기값은 16 진수로 초기화됨

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

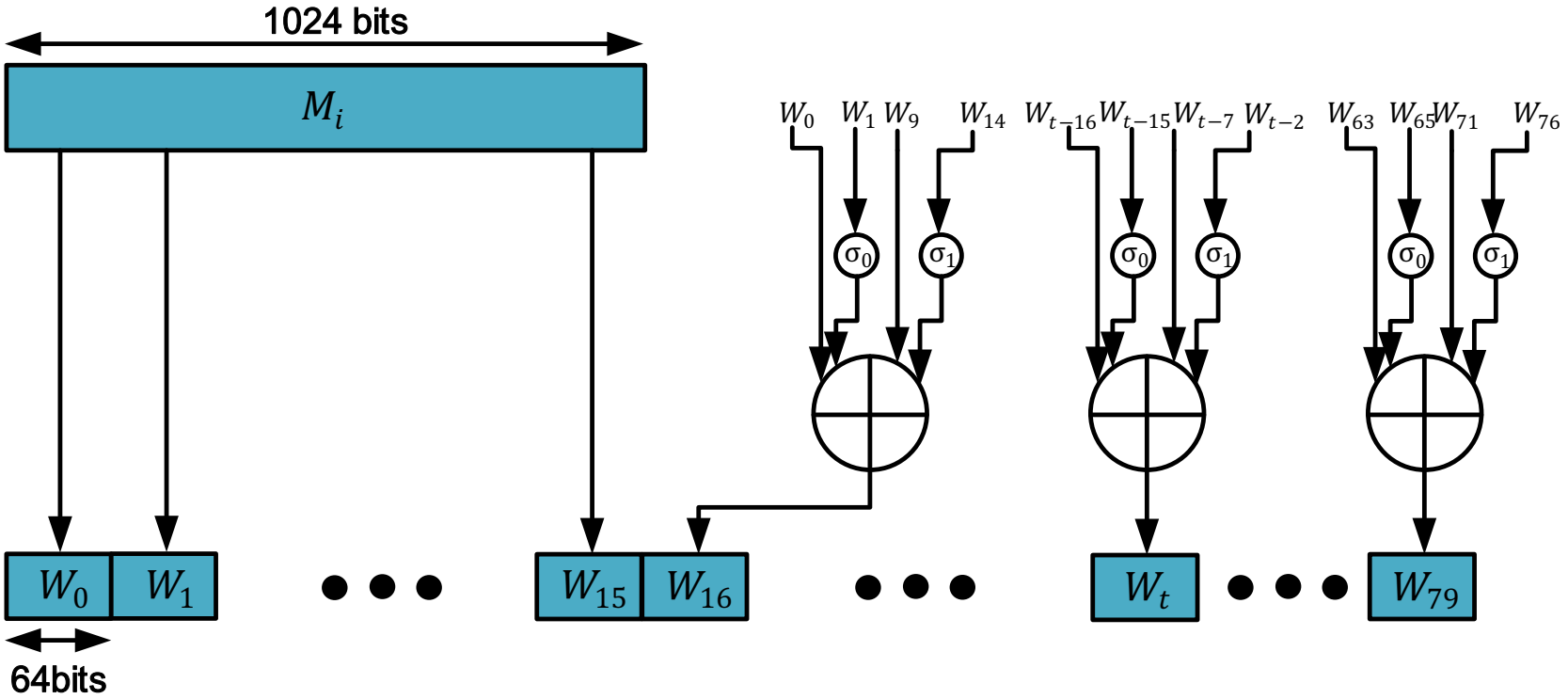
안전 해시 함수

- 안전 해시 알고리즘
  - SHA -512구조
    4. 1024비트 비트 블록 메시지 처리
       - 각 블록 처리는 80라운드
       - 8개의 64 bits의 버퍼 값, 덧셈 상수, 스케줄된 메시지워드를 입력으로 사용
       - 버퍼의 내용을 라운드마다 갱신
       - 각 블록의 길이가 1024 bits인 다중 블록메시지를 512 bits로 압축
       - 512bits (8개의 64bits 워드)MD 생성



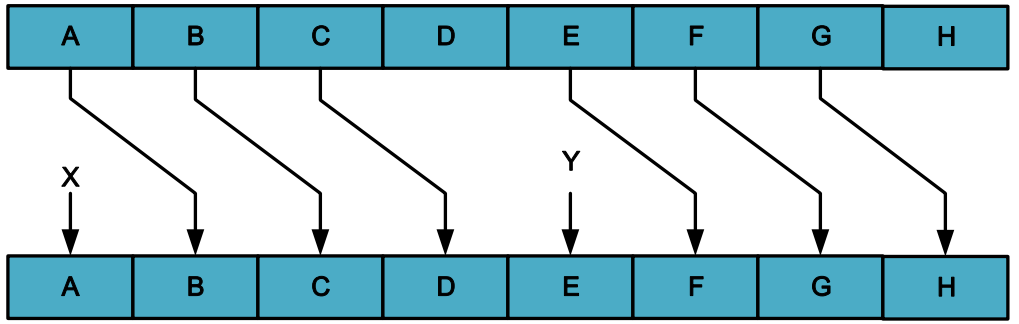
안전 해시 함수

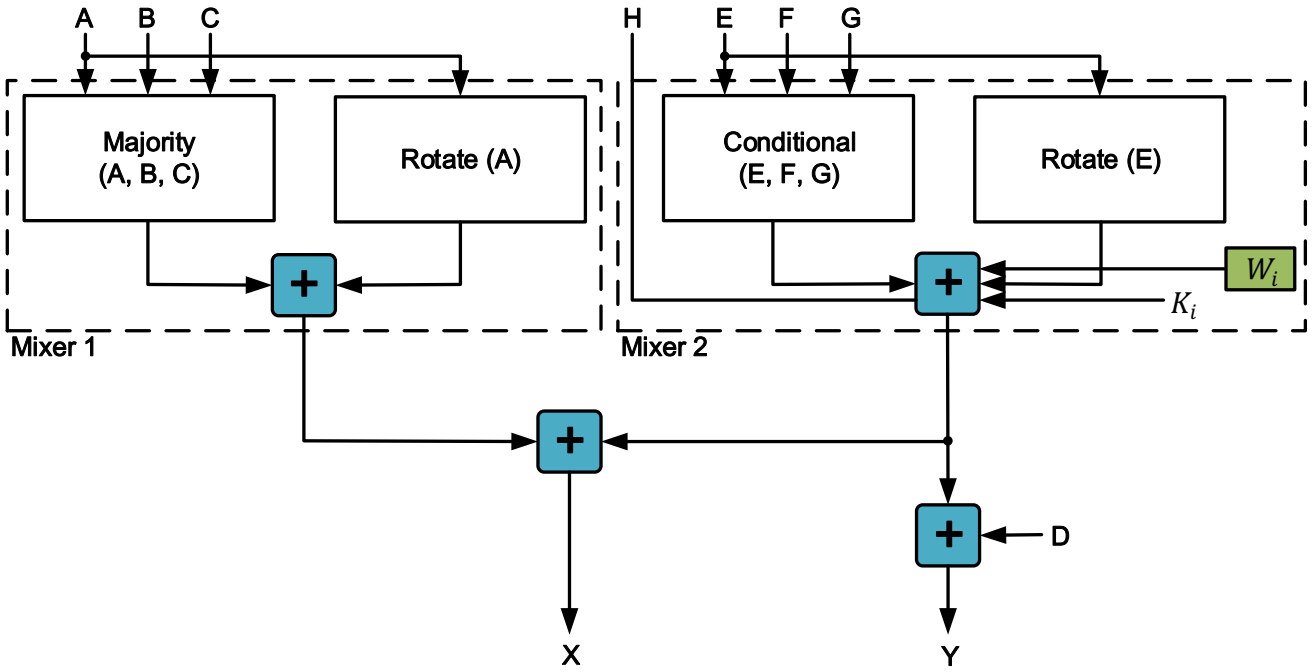
- 안전 해시 알고리즘
  - SHA-512 구조
    - 메시지 스케줄 구조
      - 16개의 워드 블록을 80개로 확장



안전 해시 함수

- 안전 해시 알고리즘
  - SHA-512 구조
    - 라운드 함수 구조
      - 80개의 상수인 K_0~K_{79}가 사용





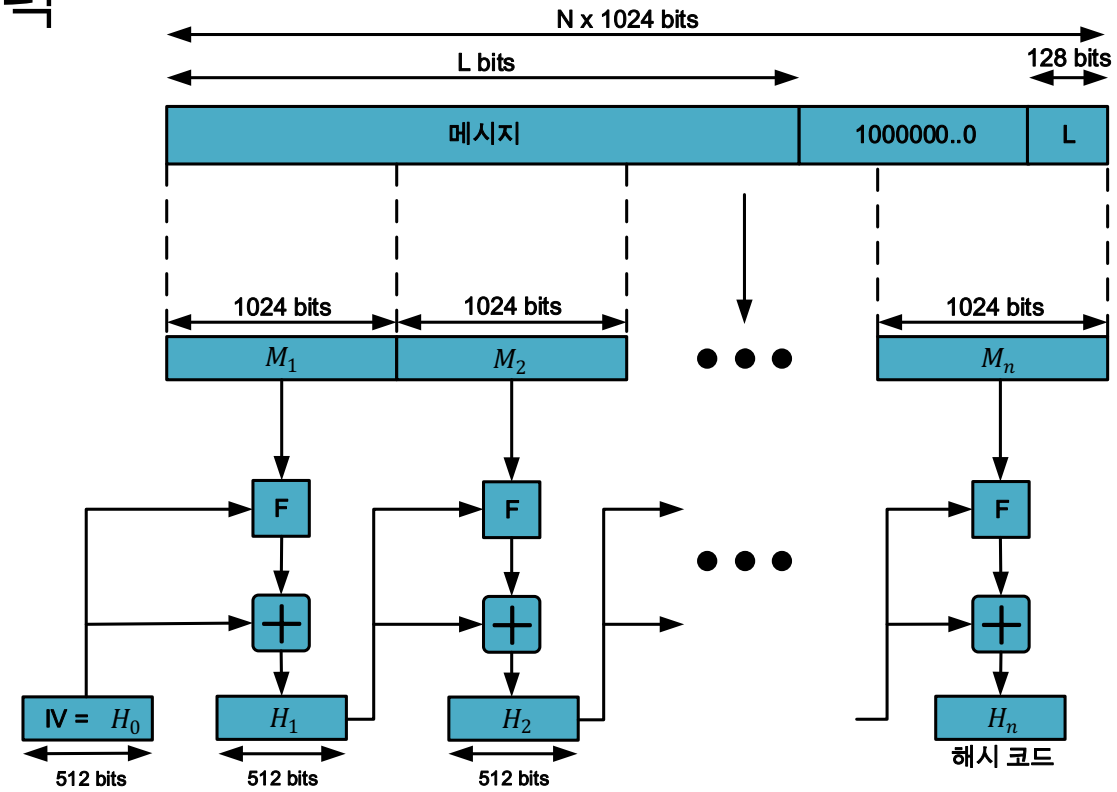
안전 해시 함수

- 안전 해시 알고리즘
  - SHA-512 구조
    - 라운드 함수 구조
      - 80개의 상수인 K_0~K_{79}가 사용
      - 상수 생성
        - 소수(2, 3, 5, 7, 11, 13, 17, 19)의 세제곱근 사용
        - $\sqrt[3]{2}$= 1.25992104989…. 소수점 이하 64 bits 사용

Majority (x, y, z)	(x AND y) $\oplus$ (y AND z) $\oplus$ (z AND x)
Rotate (x)	$RotR_{28}(x) \oplus RotR_{34}(x) \oplus RotR_{39}(x)$
Conditional (x, y, z)	(x AND y) $\oplus$ (NOT x AND z)
+	Addition modulo 2^{64}

안전 해시 함수

- 안전 해시 알고리즘
  - SHA-512 구조
    5. 출력 (Output)
       - N개의 1024 bits 블록 모두가 처리된 뒤에 n번째 단계에서 512 bits MD를 출력



메시지 인증 코드

- 메시지 인증코드 (Message Authentication Code)
  - HMAC (Hashed MAC)
    - 해시 함수를 적용하여 메시지의 위/변조를 방지하는 기법
    - 설계 목표
      - 수정하지 않고 쓸 수 있는 해시 함수를 만듦
      - 더 빠르고 안전 해시함수가 있다면 기존 해시 함수를 쉽게 바꿀 수 있도록 함
      - 기능 저하 없이 해시 함수의 원래 성능 유지
      - 키를 보다 쉽게 다루고자 함

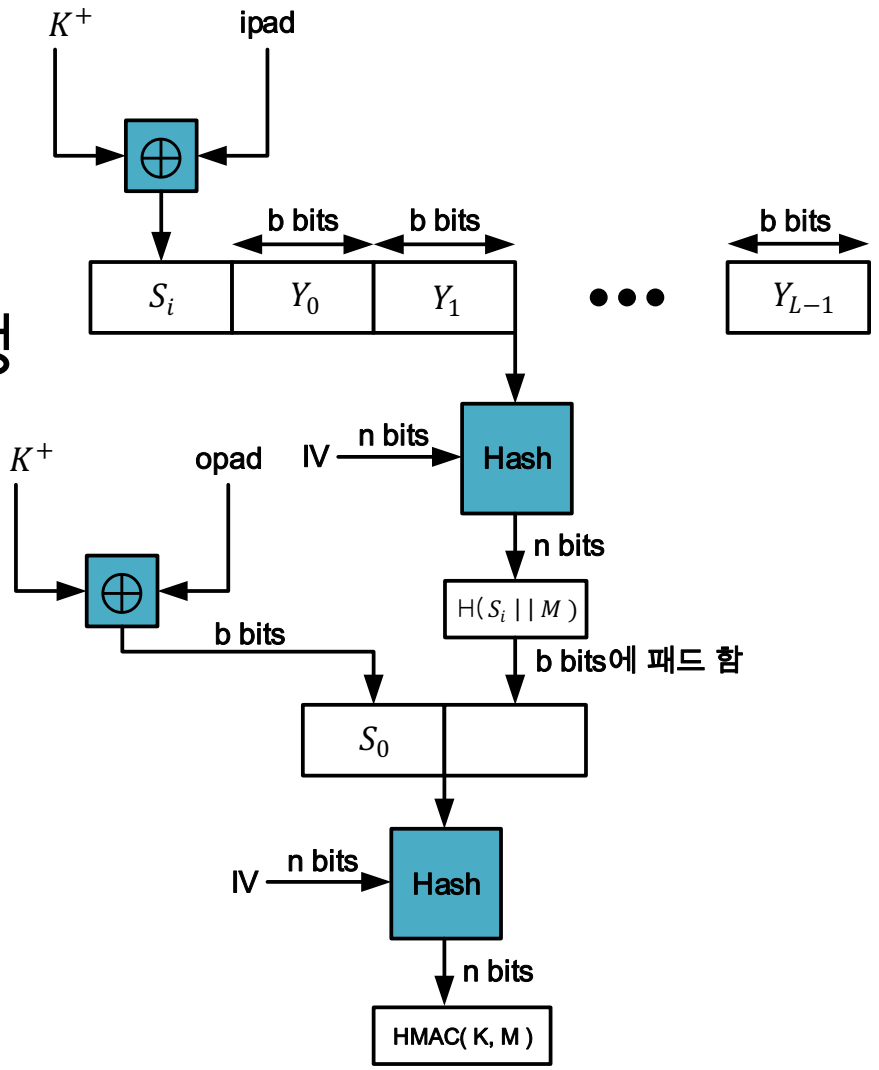
메시지 인증 코드

- HMAC
  - 용어

용어	정의
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	비밀키, 키의 길이가 b보다 길면 n bits 키를 생성하는 해시 함수에 입력으로 사용
K^+	K의 왼쪽에 0을 붙여서 길이가 b 비트가 되도록 한 것
ipad	00110110 (16진수 36)을 b/8번 반복한 2진수열
opad	01011100 (16진수 5C)을 b/8번 반복한 2진수열

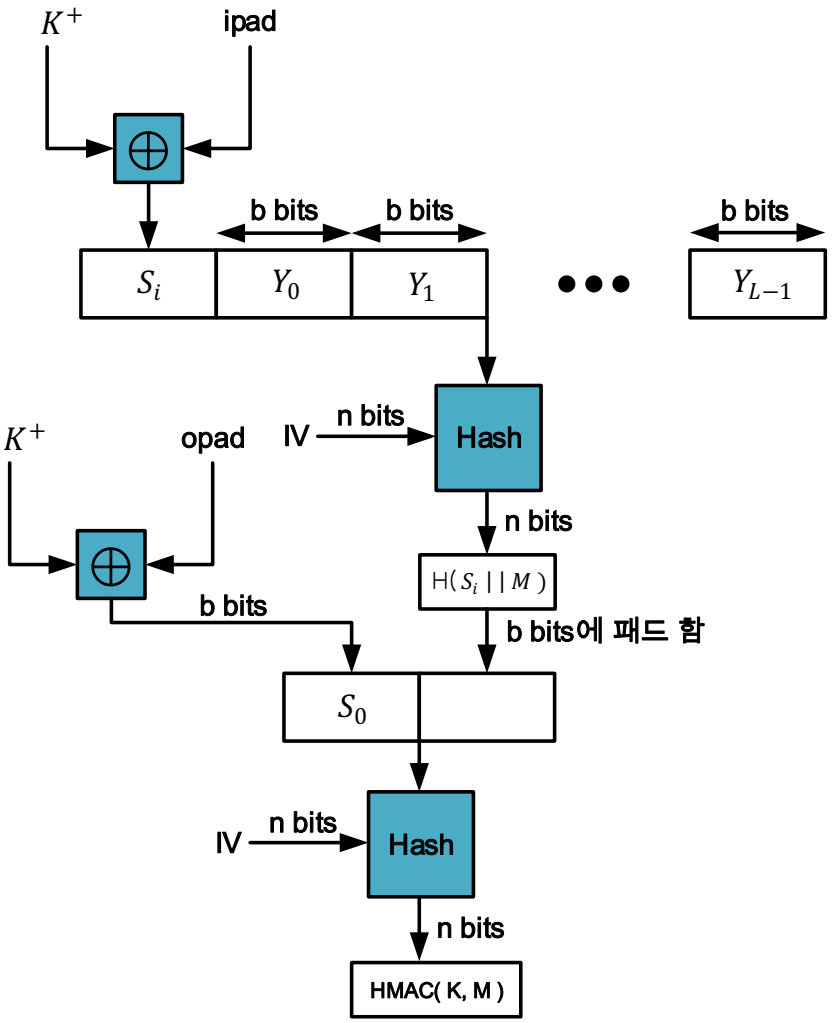
메시지 인증 코드

- HMAC
  - 구조
    1. 메시지 길이를 b bits인 블록으로 분리
    2. 비밀 키 K를 패딩하여 K^+ 생성
    3. K^+와 상수 ipad (input pad)를 XOR 연산하여 b bits S_i 생성
    4. S_i를 메시지 맨 앞에 붙이고 n bits IV와 해시 함수에 입력
       - 출력 값으로 생성된 n bits 다이제스트를 중간 HMAC라고 부름



메시지 인증 코드

- HMAC
  - 구조
    6. 중간 HMAC를 패딩하여 b bits 블록 생성
    7. opad (output pad)입력으로 2,3 단계를 반복하여 b bits S_0블록 생성
    8. S_0 블록을 중간 HMAC 앞에 붙임
    9. 동일한 해시함수 사용으로 최종 n bits HMAC을 생성



메시지 인증 코드

- 블록 암호 기반 MAC
  - 암호 기반 메시지 인증 코드 (CMAC, Copher-based Message Authentication Code)
    - 대칭키 암호를 n번 사용하여 n개의 평문 블록으로부터 하나의 MAC를 생성
    - 운용 모드는 AES와 3DES를 사용
      - AES
        - 암호 블록 길이 : 128 bits
        - 키의 길이 : 128, 192 또는 168 bits
      - 3DES
        - 암호 블록 길이 : 64 bits
        - 키 길이 : 112 또는 168 bits
    - 메시지는 n개의 블록으로 나뉨
    - n bits K_1을 사용

메시지 인증 코드

- 블록 암호 기반 MAC
  - 암호 기반 메시지 인증 코드
    - 계산식과 용어

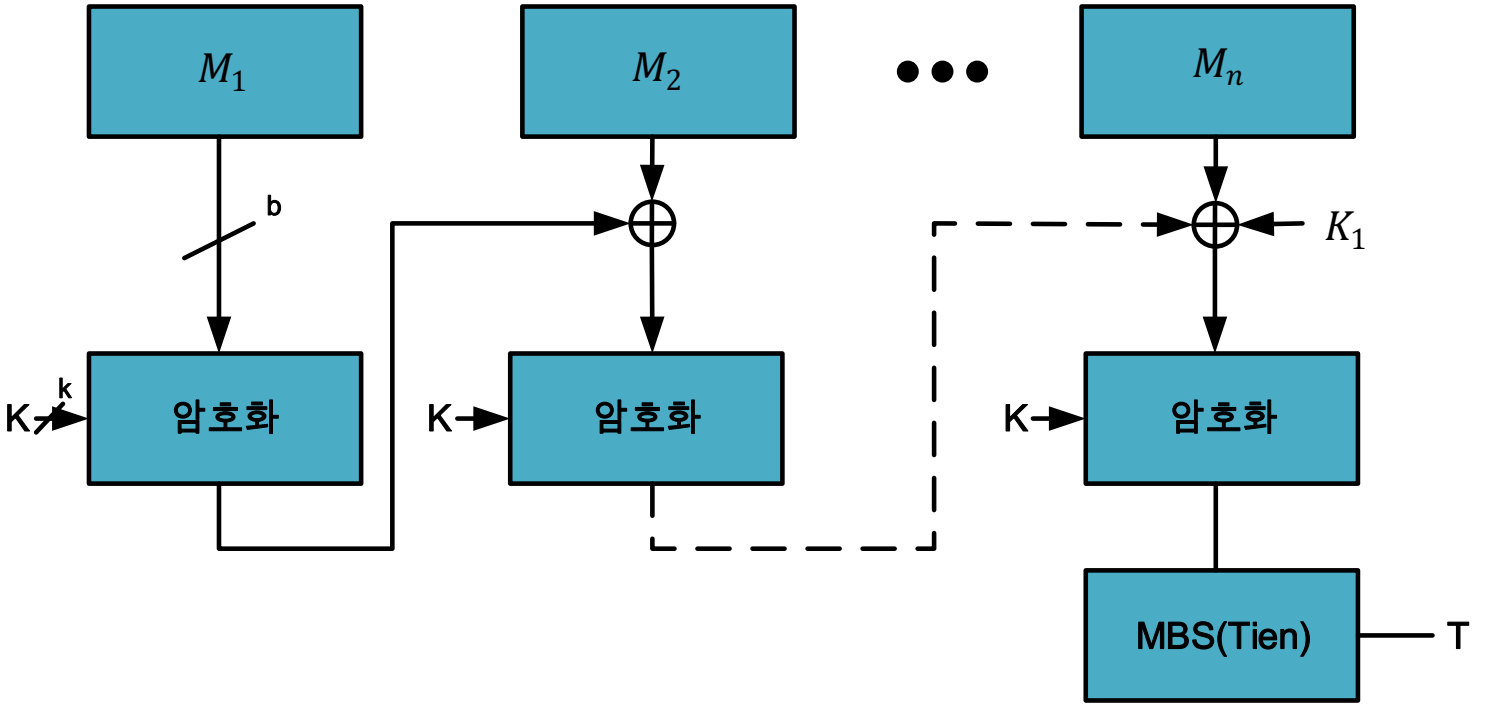
계산식
$C_1 = \mathrm{E}(K, M_1)$
$C_2 = \mathrm{E}(K, [M_2 \oplus C_1])$
$C_3 = \mathrm{E}(K, [M_3 \oplus C_2])$
• • •
$C_n = \mathrm{E}(K, [M_n \oplus C_{n-1} \oplus K_1])$
$T = MSB_{Tlen}(C_n)$

용어	정의
T	메시지 인증 코드, "태그 (Tag)"
Tlen	T의 비트 길이
MSB_s (X)	비트열 X의 왼쪽부터 S개 비트
K_1	결과로 나온 암호문을 한 bits왼쪽으로 이동시킨 값
K_2	K_1을 한 비트 왼쪽으로 이동 시킨 값

- MSB (Most Significant Bit) : 최상위 비트

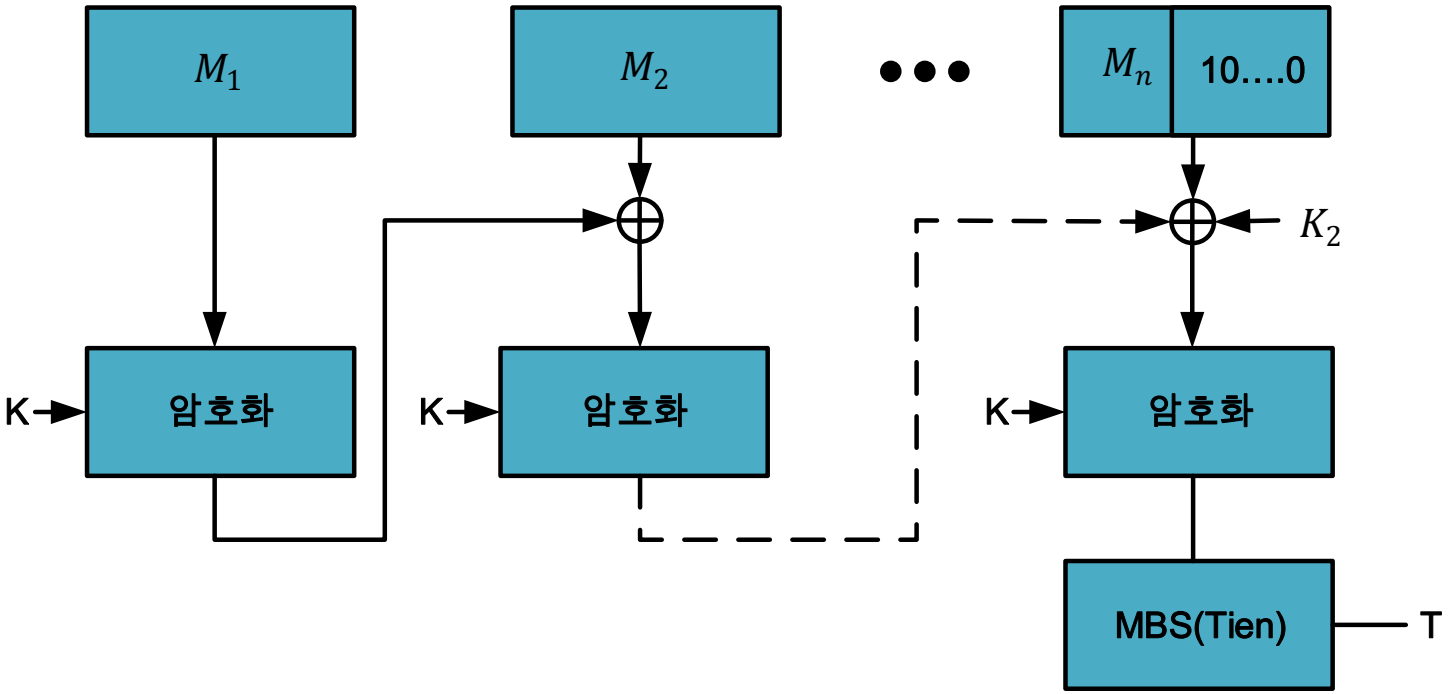
메시지 인증 코드

- 블록 암호 기반 MAC
  - 암호 기반 메시지 인증 코드
    - 메시지 길이가 블록 길이의 정수 배일 때
      - K 비트 키와 b bits 서브키 K_1을 사용



메시지 인증 코드

- 블록 암호 기반 MAC
  - 암호 기반 메시지 인증 코드
    - 메시지 길이가 블록 길이의 정수배가 아닐 때
      - 마지막 블록에 패딩을 붙여 블록의 길이가 b bits가 되게 함
      - k bits 키와 b bits 서브키 K_2를 사용

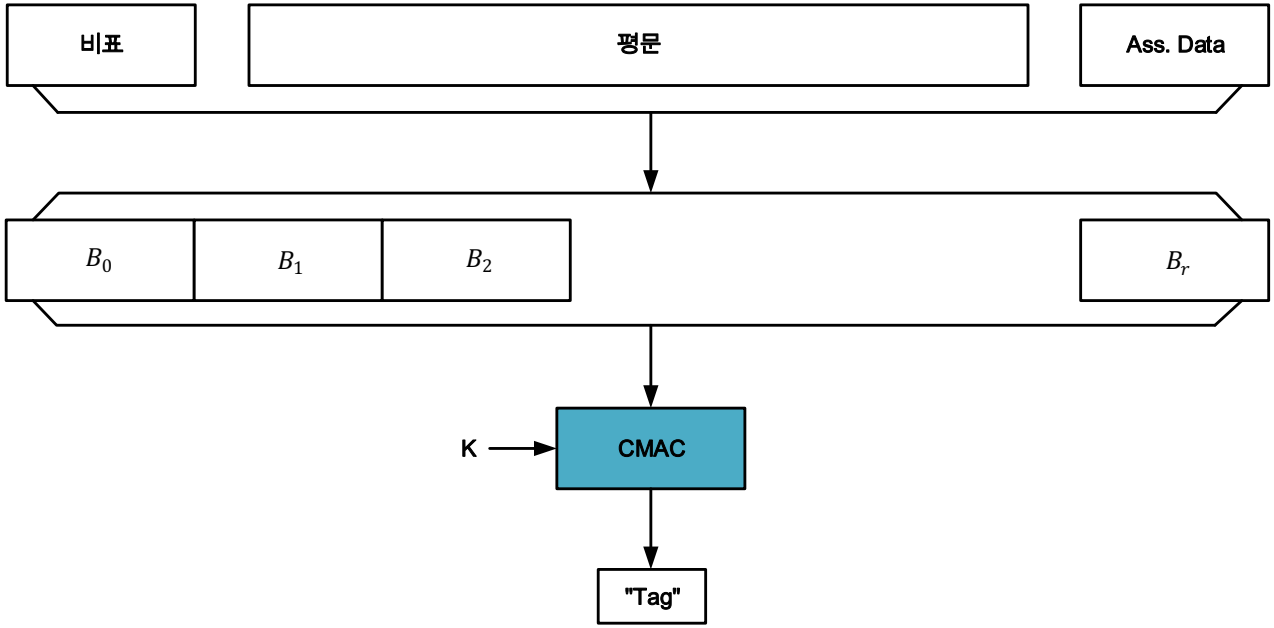


메시지 인증 코드

- 블록 암호 기반 MAC
  - 인증된 암호화 운용 모드 (Authenticated encryption mode)
    - 통신상 기밀성(암호화)과 무결성(인증)을 동시에 보호하는 시스템
    - AES 암호, CTR 운용 모드, CMAC를 사용
    - 암호화와 인증에 동일한 키 K를 사용
    - CCM 암호화 과정에 입력되는 3가지 요소
      - 인증하고 암호화할 데이터
      - 인증을 하는 유관 데이터
      - 비표

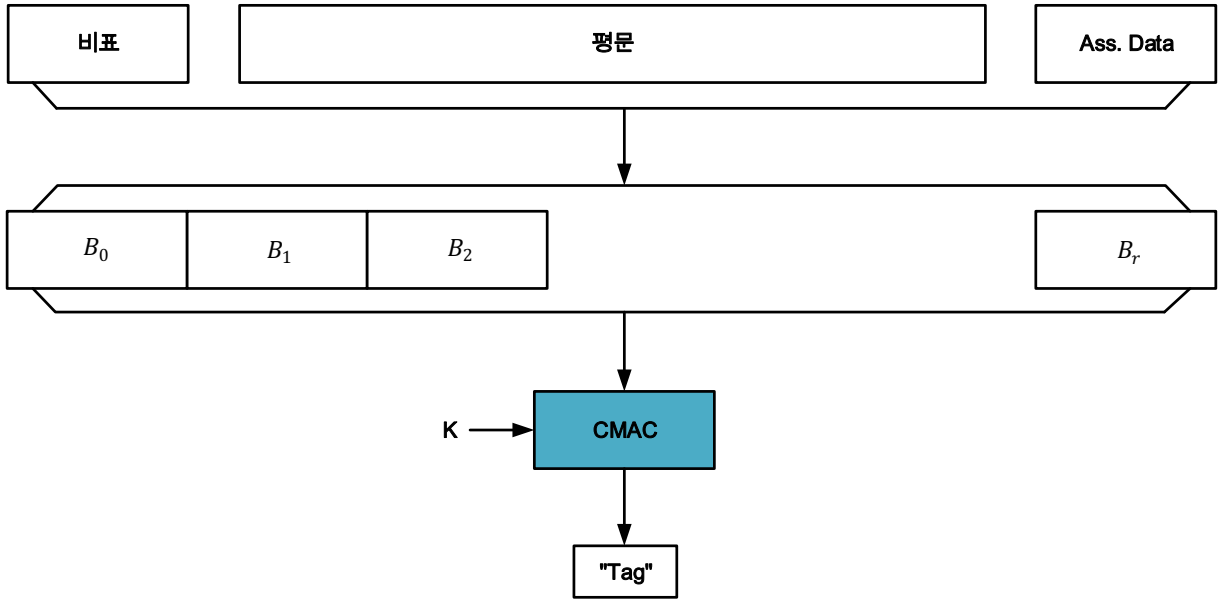
메시지 인증 코드

- 블록 암호 기반 MAC
  - 암호 블록체인 카운터
    - 구조
      - 인증
        - 입력 값을 $B_0 \sim B_r$까지 나타냄
        - CMAC에 입력 하여 길이가Tien인 MAC생성



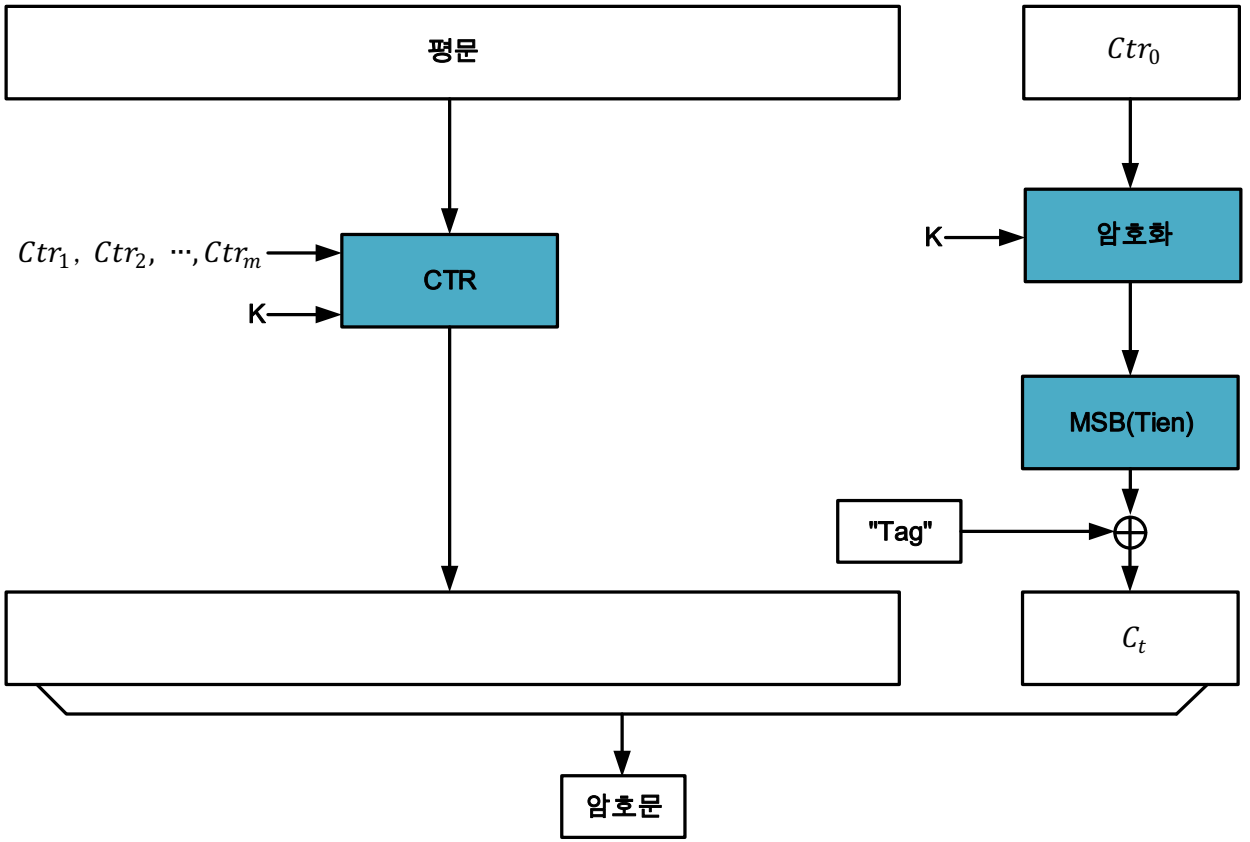
메시지 인증 코드

- 블록 암호 기반 MAC
  - 암호 블록체인 카운터
    - 구조
      - 인증
        - 비표 = 프로토콜에 따라 변하는 값
        - Ass. Data = 프로토콜 동작을 위한 평문
        - CMAC = CBC - MAC



메시지 인증 코드

- 블록 암호 기반 MAC
  - 암호 블록체인 카운터
    - 구조
      - 암호화
        - 카운터 열 생성
        - 하나의 카운터로 CTR암호화
        - 출력 값 중 Tien 개의 bits를 Tag 와 XOR연산C_t생성
        - 암호문 생성



Thanks!

박 재 형 (jaehyoung@pel.smuc.ac.kr)