네트워크 보안 에센셜

3장 메시지 인증과 해시함수

Sein Myung (sein@pel.smuc.ac.kr)

Protocol **E**ngineering **L**ab., **Sangmyung** University

목 차

- 메시지 인증
- 안전 해시 함수
- 메시지 인증 코드

메시지 인증

- **메시지 인증 (Message Authentication)**
  - **암호를 사용하면 소극적 공격을 막을 수 있지만 적극적 공격에 대응하려면 인증 등의 방법이 필요함**
  - **메시지 인증이 완료되면**
    - **데이터가 진짜이고 송신자 라고 주장하는 곳에서 왔음을 확신 할 수 있음**
    - **통신 양측으로 하여금 받은 메시지가 진짜 임을 확인**
    - **시간성도 확인할 수 있음**
  - **인증 방법**
    - **관용암호를 이용한 인증**
    - **암호화 하지 않고 인증**
      - **관용암호 인증**
      - **공개키 인증**
      - **비밀 값 인증**

메시지 인증

- 관용 암호를 이용한 인증
  - 동일한 비밀키에 의해 수신자가 복호화 하여 평문을 얻으면 송신자로부터 옴을 확인

  - 오류 감지 코드
    - 메시지가 변경되지 않음을 확인
  - 순서 번호
    - 메시지의 순서가 틀리지 않음을 확인
  - 타임스탬프 (Timestamp)
    - 메시지가 고의로 지연되지 않음 을 확인

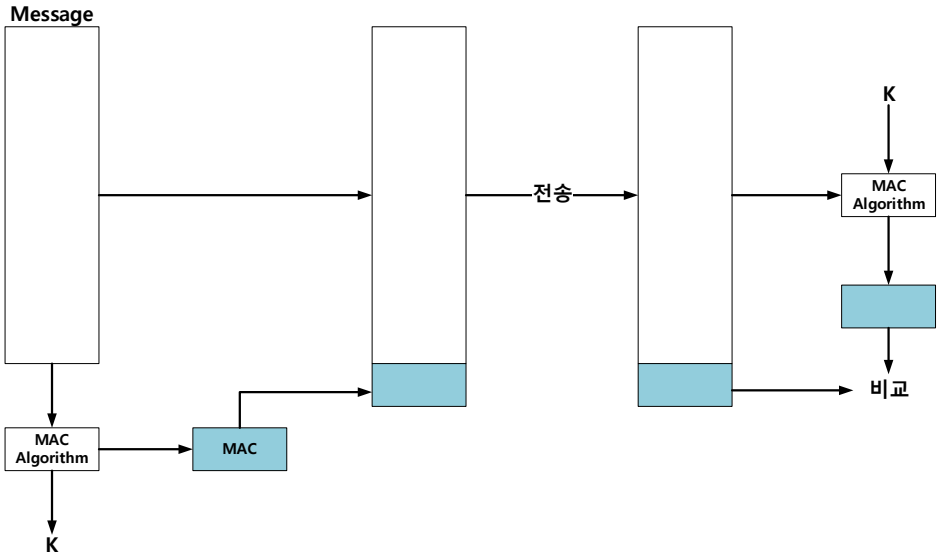
메시지 인증

- **암호화 하지 않고 메시지 인증** **(Message Authentication Without Message Encryption)**
  - **메시지 전체를 암호화 하지 않음**
  - **기밀성은 제공되지 않고 인증 만 제공**
  - **인증 꼬리표 사용 (Authentication Tag)**

  - **기밀성과 인증을 제공하는 관용암호를 사용하지 않는 이유**
    - **연산이 많아짐**
      - **브로드 캐스트**
      - **통신 과부하**
      - **확신이 항상 필요한가?**

메시지 인증

- **암호화 하지 않고 메시지 인증** (Message Authentication Without Message Encryption)
  - 메시지 인증 코드 (MAC: Message Authentication Code)
    - 붙여지는 작은 데이터블록코드를 생성하기 위해 비밀 키를 이용
    - 코드는 무결성, 출처 확인, 시간성을 보장



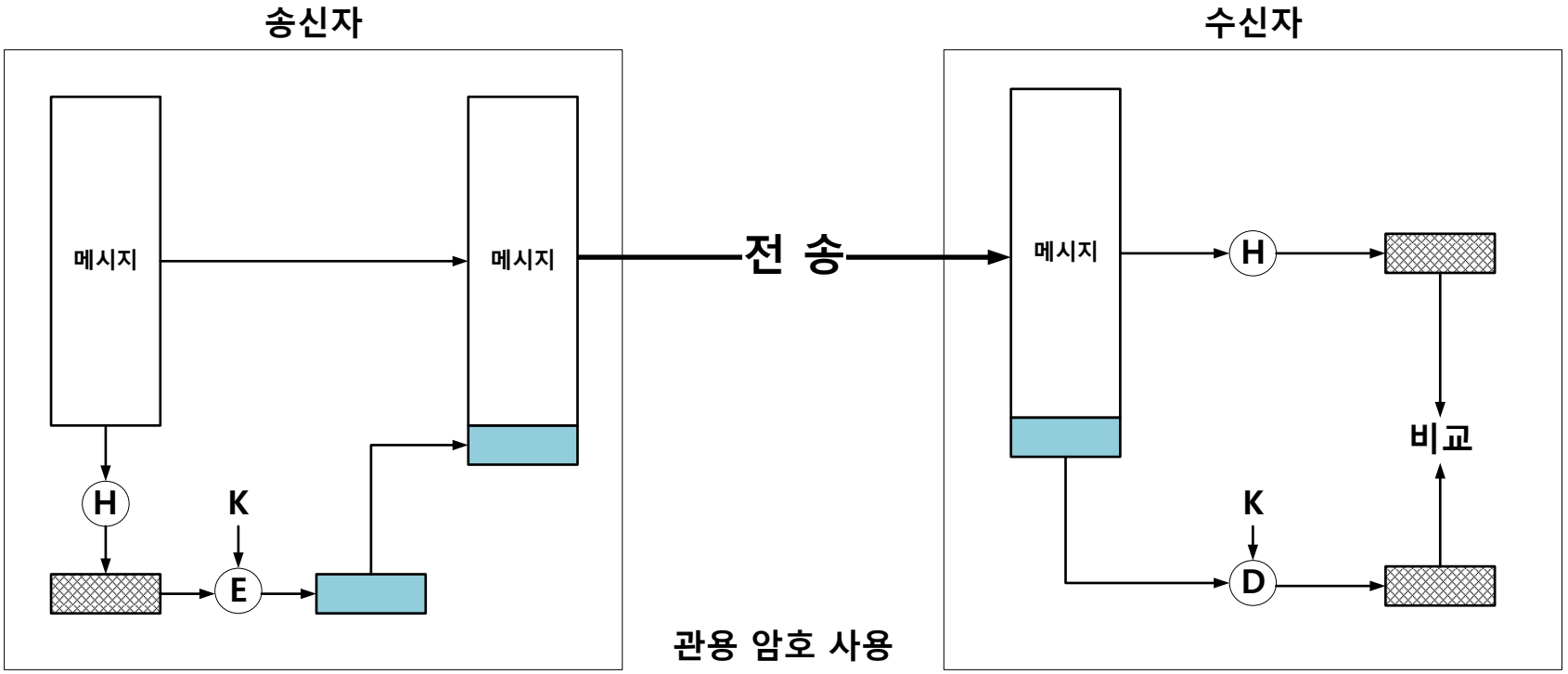
  - NIST의 FIPS PUB 113에서는 DES를 권장
    - 복호화가 (계산적으로)불가능한 DES기반의 다른 버전을 만들어 사용

메시지 인증

- **암호화 하지 않고 메시지 인증** (Message Authentication Without Message Encryption)
  - **일방향 해시 함수 (One-way Hash Function)**
    - **임의 크기의 메시지 M을 입력으로 일정한 크기의 메시지 다이제스트(MD: Message Digest) H(M)를 출력하여 인증을 하는 함수**
    - **MAC과는 달리 비밀키를 해시함수의 입력으로 사용 하지 않음**

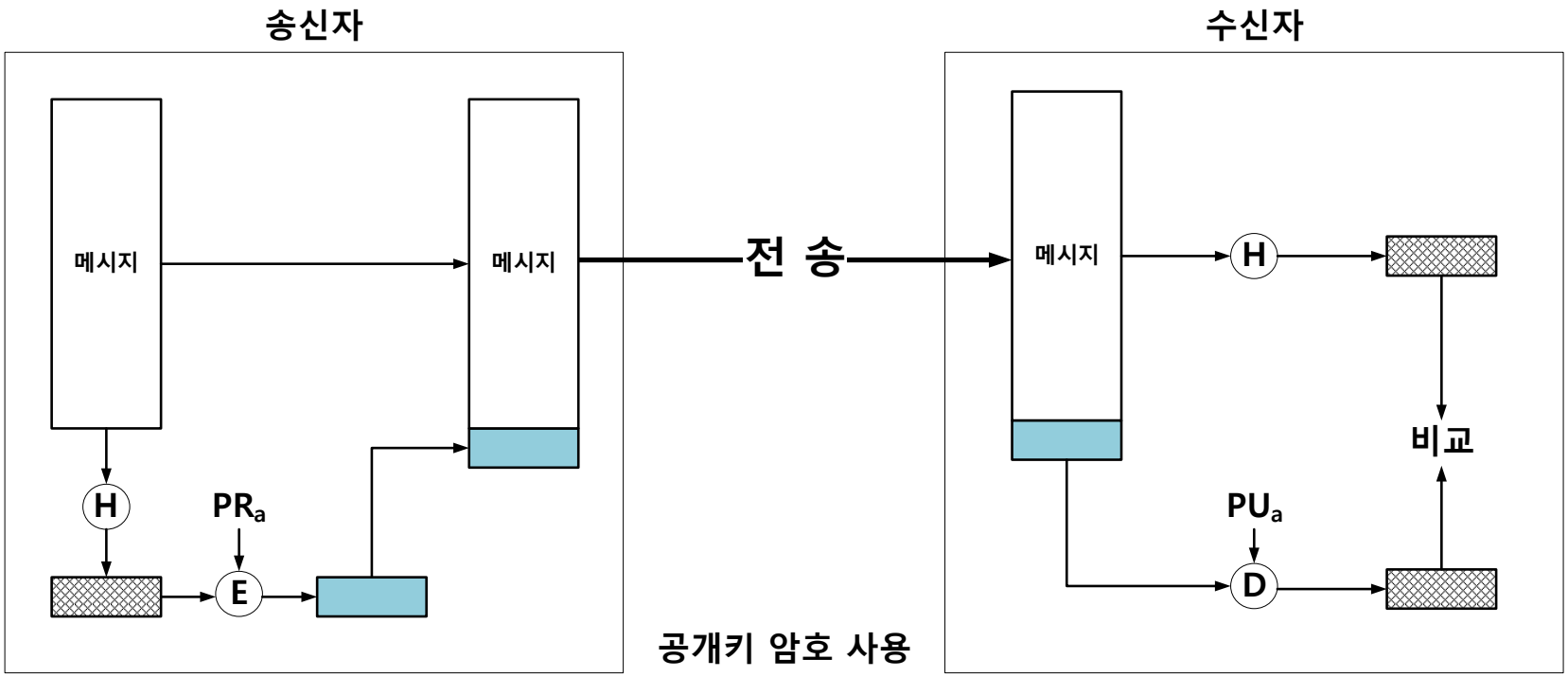
메시지 인증

- **암호화 하지 않고 메시지 인증** (Message Authentication Without Message Encryption)
  - 관용 암호를 사용하는 일방향 해시 함수 인증



메시지 인증

- **암호화 하지 않고 메시지 인증** (Message Authentication Without Message Encryption)
  - 공개키 암호를 사용하는 일방향 해시 함수 인증

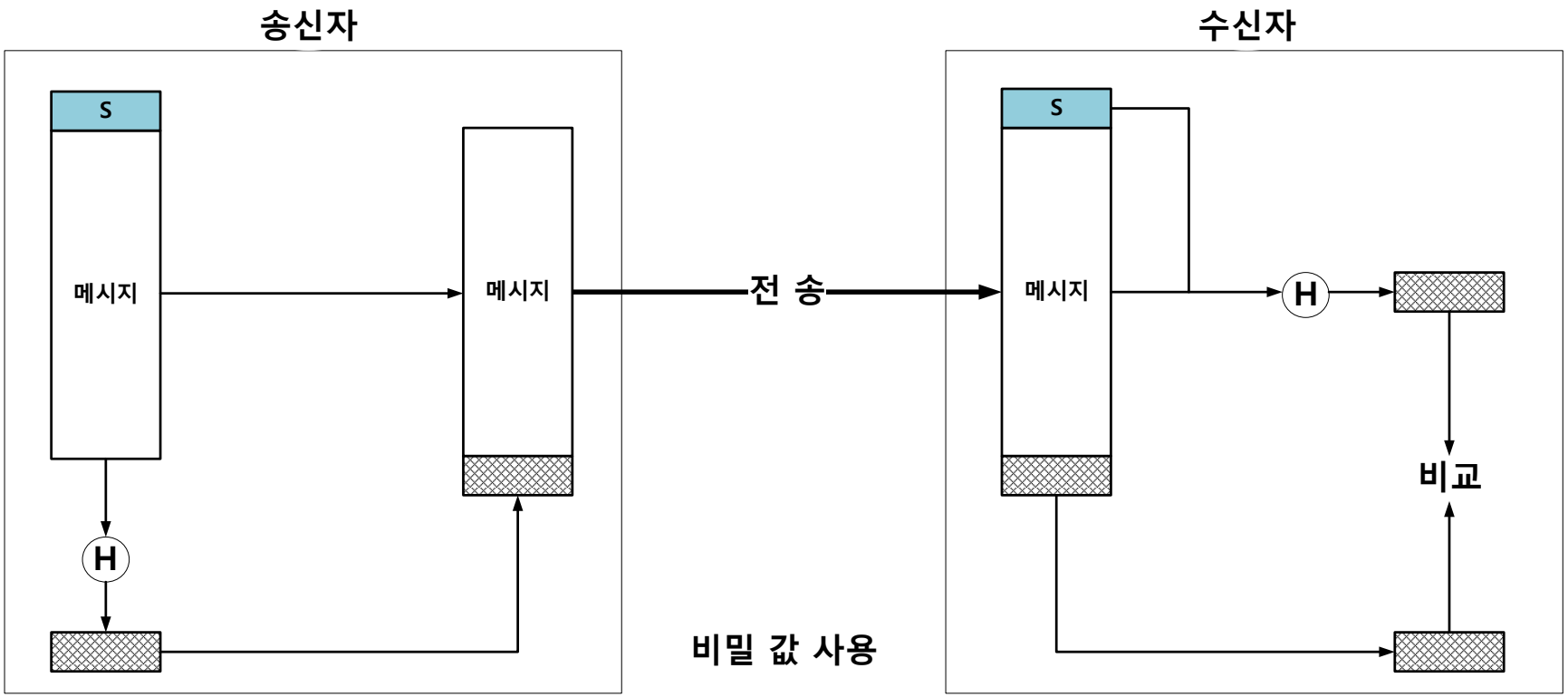


메시지 인증

- 암호화 하지 않고 메시지 인증 (Message Authentication Without Message Encryption)
  - 관용 암호와 공개키 암호를 사용하는 일방향 해시 함수 인증
    - 이 두가지 방법은 전체 메시지를 암호화 하는 방법보다 계산의 양을 줄였지만 암호화를 전혀 하지 않는 기술을 생각
      - 암호 소프트웨어가 느림
        - 메시지 개별에 대해서는 매우 빠르지만 전체적으로 입출력 되는 양을 처리하는 시간과 비용
      - 암호 장비의 값
        - 암호화가 구현된, 값은 싸지만 네트워크의 모든 노드에 설치 해야 하는 칩의 총 가격
      - 암호 장비는 대용량 처리에 적합
        - 작은 데이터 블록에 대한 초기화/오버헤드에 상당한 시간이 소비
      - 암호 알고리즘은 수출에 제약
    - 해시함수를 사용하고 메시지 인증에 암호화 기술이 적용되지 않는 비밀값 사용법

메시지 인증

- **암호화 하지 않고 메시지 인증** (Message Authentication Without Message Encryption)
  - 비밀 값을 사용하는 일방향 해시 함수 인증



안전 해시 함수

- 안전 해시 함수
  - 일방향 해시함수, 안전 해시함수(Secure Hash Function)는 메시지 인증 뿐만 아니라 디지털 서명(Digital Signature)에도 매우 중요한 함수임

    디지털 서명 : 공용 키 시스템에서 송신자의 신원을 증명하는 방법

- 메시지 인증에 사용되는 해시 함수의 요건
  - 해시 함수의 목적은 데이터 블록에 대한 지문(Fingerprint)을 생성

  - 해시함수 H의 요건
    1. 자유로운 길이의 입력에 대한 처리
    2. 일정한 길이의 출력
    3. 계산이 쉽고 실제 구현이 가능해야 함
    4. 일방향 성질
    5. 약한 충돌 저항성(Weak Collision Resistance)
    6. 강한 충돌 저항성(Strong Collision Resistance)

안전 해시 함수

- 메시지 인증에 사용되는 해시 함수의 요건
  - 해시함수 H의 기본 요건 (1,2,3)
    - 메시지 인증에 해시함수를 응용하기 위해 필요한 사항이며 해시 함수의 특성
  - 일방향 성질(One-way Property)을 갖는 해시함수 (4)
    - 주어진 출력값 h에 대해 H(x)=h가 성립하는 x찾는 것(역함수)이 계산적으로 불가능
    - 이 성질이 없다면 비밀값 사용시 실제 비밀값은 전송되지 않지만, 공격자가 비밀값을 찾을수 있게됨
  - 충돌 저항성 (5,6)
    - 주어진 블록 x에 대해 H(x)=H(y)가 성립하는 y(x와 다른)값을 찾지 못하는 약한 충돌 저항성 (Weak Collision Resistance)
    - H(x)=H(y)를 만족하는 쌍(x, y)를 찾지 못하는 강한 충돌 저항성 (Strong Collision Resistance)
      생일공격(Birthday Attack)에 대응 할 수 있음
  - 기본 요건과 프리이미지(Preimage:4,5) 저항성 조건을 만족하면 약한 해시함수
  - 모든 조건을 만족하면 강한 해시함수

안전 해시 함수

- 해시 함수 보안
  - 생일 공격
    - 생일은 1년 중 하루, 365가지가 있을 수 있음
    - 366명이 모이면 100% 확률로 생일이 같은 사람이 생김
    - $P(n) = 1 - \frac{365!}{365^n(365-n)!}$

사람수	확률
10	12%
20	41%
30	70%
50	97%

안전 해시 함수

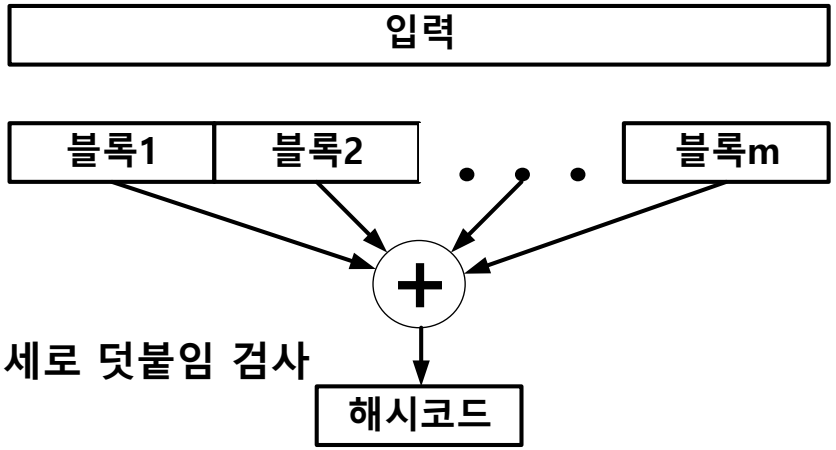
- 해시 함수 보안
  - 전수 공격에 대한 해시 함수의 강도는 출력 값인 해시 코드의 길이에 의존
  - 해시코드의 길이 n에 대하여

해시함수 요건	공격 복잡도
프리 이미지 저항성	2^n
2차 프리 이미지 저항성	2^n
충돌 저항성	$2^{n/2}$

  - Van Oorschot과 Wiener가 128bit-MD5(Message-Digest Algorithm 5)에서 충돌을 찾아내기 위한 천만 달러 알고리즘으로 24시간만에 충돌을 찾아냄 (128은 안전하지 않음)
  - 32bit열 단위로 처리한다면 160bit가 다음 길이지만 현대 컴퓨팅 기술로도 안전을 보장할 수 없음 (위의 기술로는 4000년이 걸림)

안전 해시 함수

- **단순 해시함수 (Simple Hash Function)**
  - **전체 입력 데이터를 연속적인 n-bit블록 입력으로 간주**



  - **이 방법은 각 비트별로 패리티를 계산하는 세로 덧붙임 검사 (Longitudinal Redundancy Check)**
  - **임의의 데이터 무결성 검사에 아주 효과적**

    n-bit의 모든 해시 값이 나타날 확률은 동일

    한 데이터의 오류가 해시 값에 변화를 초래하지 않을 확률 : 2^{-n}

    데이터의 형식을 예측 할 수 있을경우 효과가 떨어짐 128비트 해시 값의 각 byte의 첫 비트가 0일 경우, 2^{-128} -> 2^{-112}

    각 바이트 연산시에 회전shift로 더욱 난해하게 할 수 있음

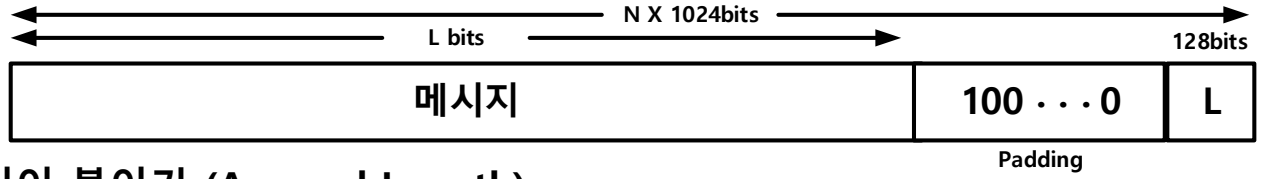
안전 해시 함수

- SHA 안전 해시 함수(Secure Hash Algorithm)
  - 1993년 NIST의 FIPS PUB 180으로 소개, 최초 약점에 대한 수정판이 1995년에 나옴
  - MD4에 기초하여 만들어짐 (Message Digest Algorithm 4)
  - 앞서 사용하던 대부분의 해시 함수가 치명적인 암호학적 약점이 발견됨
  - 하부 구조는 모두 동일

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

안전 해시 함수

- **SHA-512 안전 해시 함수**
  - **패딩 비트 붙이기 (Appending Padding Bits)**
    - **메시지 패딩을 추가하여 총 길이를 896(mod 1024)가 되도록 만듦**
    - **이미 원하는 길이여도 항상 추가함**
    - **패딩시 첫 비트는 1, 나머지는 0 으로 채운다**



  - **L길이 붙이기 (Append Length)**
    - **부호 없는 128비트 정수의 패딩전 메시지 길이를 의미**

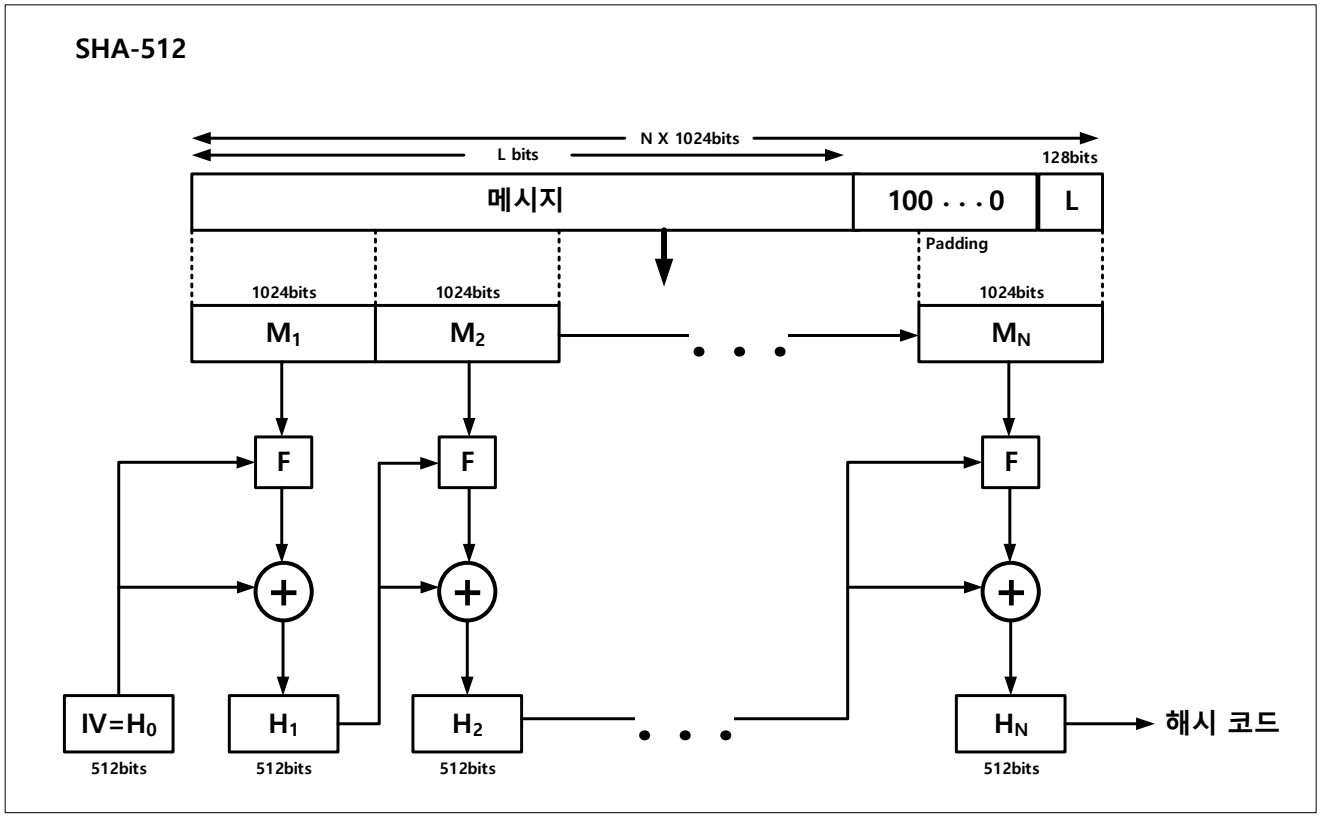
MD 버퍼 초기화

a = 6A09E667F3BCC908	e = 510E527FADE682D1
b = BB67AE8584CAA73B	f = 9B05688CEB3E6C1F
c = 3C6EF372EF94F82B	g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1	h = 5BE0CD19137E2179

  - **MD 버퍼 초기화 IV (Initialize MD buffer)**
    - **512bit 버퍼를 해시함수에서 사용**
    - **처음 8개 소수의 세제곱근 소수점 이하 64비트값을 8개 레지스터로 초기화**

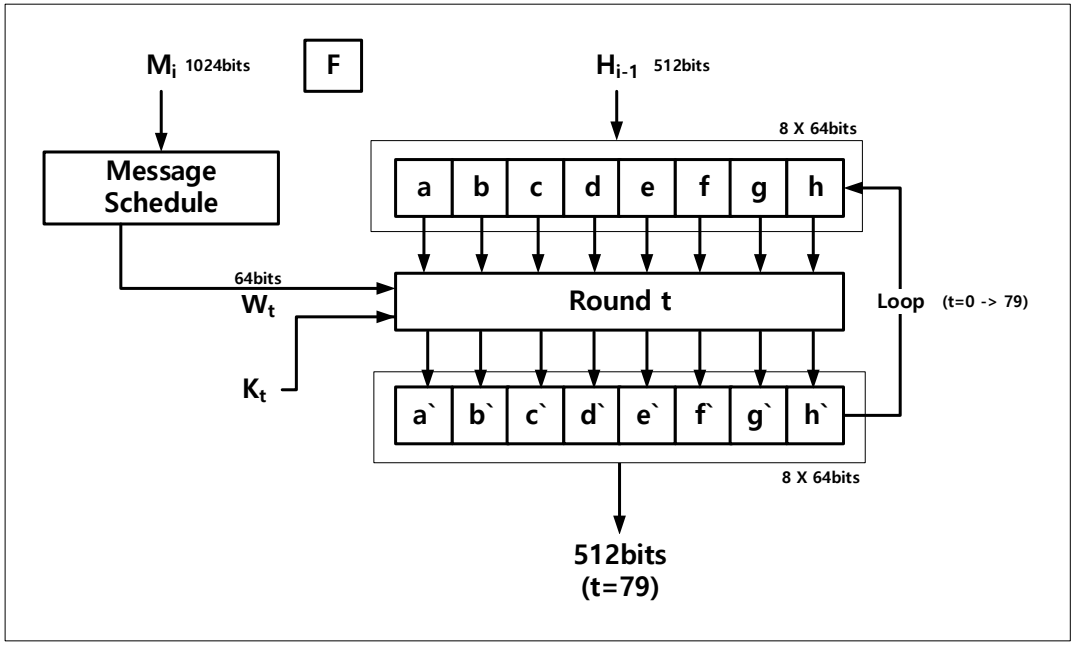
안전 해시 함수

- **SHA-512 안전 해시 함수**
  - **블록 분할**
    - **1024비트 블록의 배수 가 되어 해시 함수의 입력**



안전 해시 함수

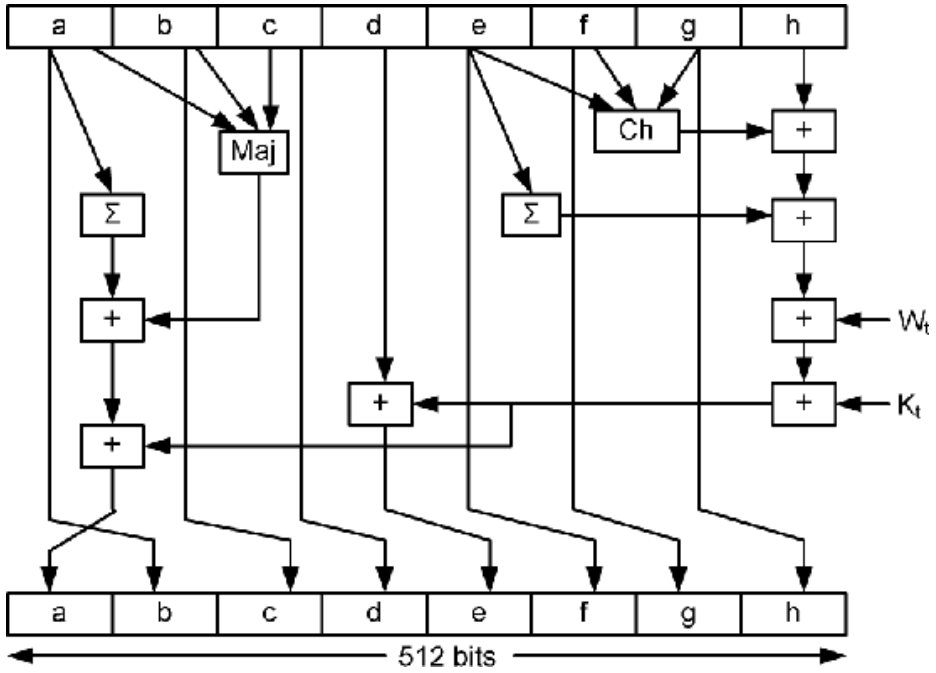
- **SHA-512 안전 해시 함수**
  - **1024bits 블록 메시지 처리**
    - **라운드 함수의 입력으로 512bits 버퍼, 64bits 스케줄된 메시지 워드, 덧셈 상수K**
    - **총 80 라운드 수행 후 중간 값 출력**



안전 해시 함수

- **SHA-512 안전 해시 함수**
  - **1024bits 블록 메시지 처리**
    - **Round 함수**
      - **Maj, Ch, Σ, XOR, 연산등을 이용하여 입력된 버퍼, 워드 상수를 처리**

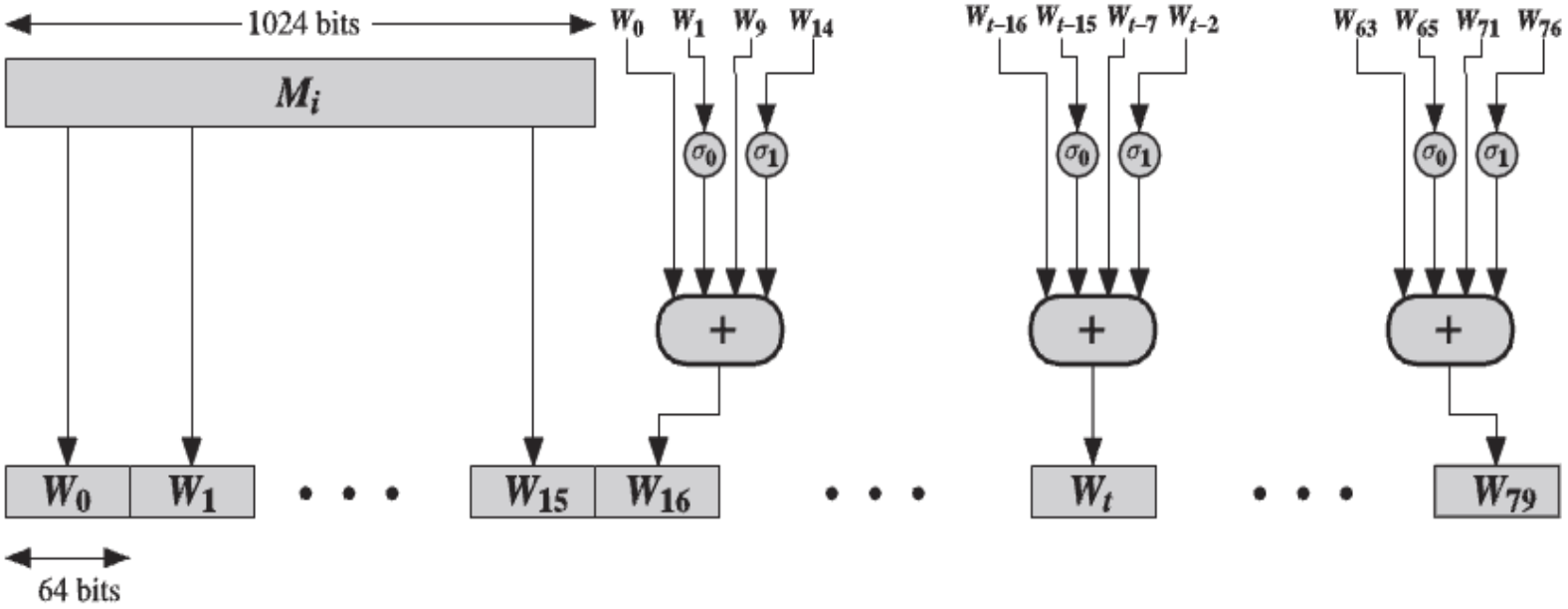
```
Maj(x, y, z) {
        return (x & y)^(y & z)^(z & x);
}
Ch(x, y, z) {
        return (x & y)^(!x & z);
}

R_R(x,n) { 우측 순환 쉬프트
        return x>>n;
}

ΣA(a) {
        return R_R(a,28) ^ R_R(a,34) ^ R_R(a,39);
}

ΣE(e) {
        return R_R(e,14) ^ R_R(e,18) ^ R_R(e,41);
}
```



안전 해시 함수

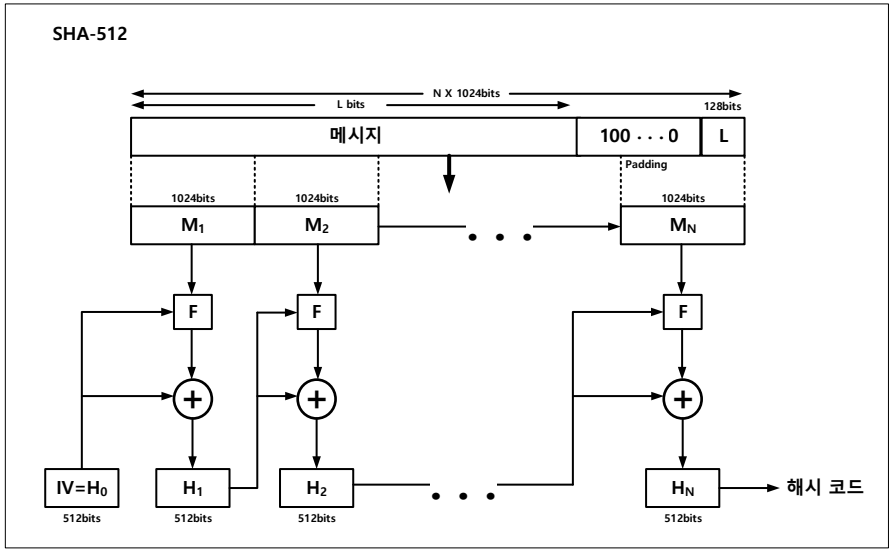
- **SHA-512 안전 해시 함수**
  - **1024bits 블록 메시지 처리**
    - **메시지 스케줄**
      - **라운드함수에 쓰일 64bits 워드를 스케줄**

```
R_R(x,n) { 우측 순환 쉬프트
        return x>>n;
}

σ0(x) {
        return R_R(x,1) ^ R_R(x,8) ^ R_R(x,7);
}

σ1(x) {
        return R_R(x,19) ^ R_R(x,61) ^ R_R(x,6);
}
```



안전 해시 함수

- **SHA-512 안전 해시 함수**
  - **해시코드**
    - **N단계 수행후 512메시지 다이제스트를 얻음**
    - **두 메시지가 유사하더라도 같은 해시코드일 가능성은 매우 적음**
    - **동일한 두 MD(Message Digest)를 갖는 두 메시지를 찾을 난이도는 2^{256}번의 연산이 필요**
    - **주어진 MD와 같은 MD를 갖는 메시지를 찾기 위해선 2^{512}번의 연산이 필요**

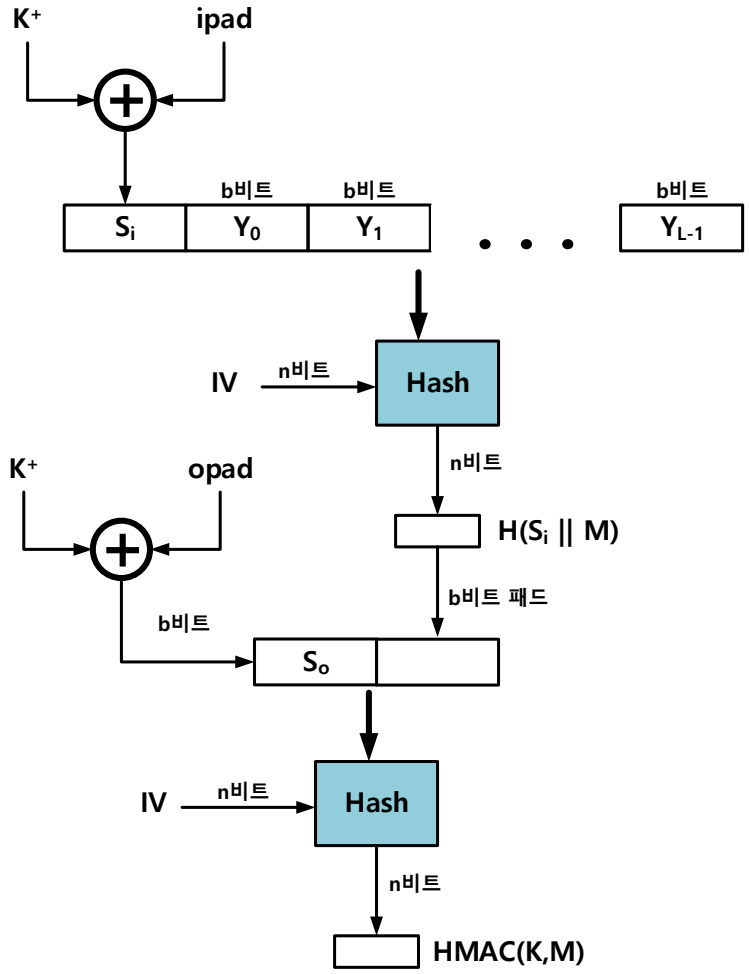


메시지 인증 코드

- **HMAC (Hashed MAC)**
  - **HMAC의 설계목표**
    - **소프트웨어구현 가능, 오픈 소스가 목표**
    - **기능저하를 유발하지 않고 원래의 성능을 유지**
    - **HMAC을 블랙박스로 간주**
      - 내부 구조보다는 입력과 출력의 관계를 중요시 하는 개념
      - 독립적인 하나의 구성 요소로 취급하는 HMAC 구현의 모듈(module)로 사용 하므로 수정이 쉬움
    - **IP Security, TLS (Transport Layer Security), SET (Secure Electronic Transaction)등의 인터넷 프로토콜에서 사용됨**

메시지 인증 코드

- **HMAC (Hashed MAC)**
  - **HMAC의 동작**
    1. 비밀키를 b비트의 K^+로 만듦 (0을 패딩)
    2. K^+와 ipad를 XOR 하여 b비트 S_i 블록을 만듦
    3. S_i에 메시지M을 붙이고 IV와 해시 함수에 입력
    4. K^+와 opad를 XOR 하여 b비트 S_o 블록을 만듦
    5. 3단계의 출력값을 So에 붙이고 IV와 해시 함수에 입력
    6. 결과값 출력



HMAC 용어

H = 해시함수
M = HMAC의 패딩된 입력 메시지
Y_i = M의 i번째 블록
L = M의 블록 수
b = 블록의 비트 수
n = 해시 코드의 길이
K = 비밀키, 키의 길이가 b보다 크면 n-비트 키를 생성하는 해시 함수의 입력으로 사용
K^+ = K의 왼쪽에 0을 붙여 길이가 b비트가 되게 한것
ipad = 00110110(0x36)을 b/8번 반복한 2진 수열
opad= 01011100(0x5C)을 b/8번 반복한 2진 수열

$HMAC(K, M) = H[(K^+ \wedge opad) \| H[(K^+ \wedge ipad) \| M]]$;

- ipad, opad와 XOR하면 비트의 반수가 뒤집히지만 적용 대상이 다름
- S_i 와 S_o에 적용된 해시 알고리즘의 결과값은 의사 랜덤한 키가 됨

메시지 인증 코드

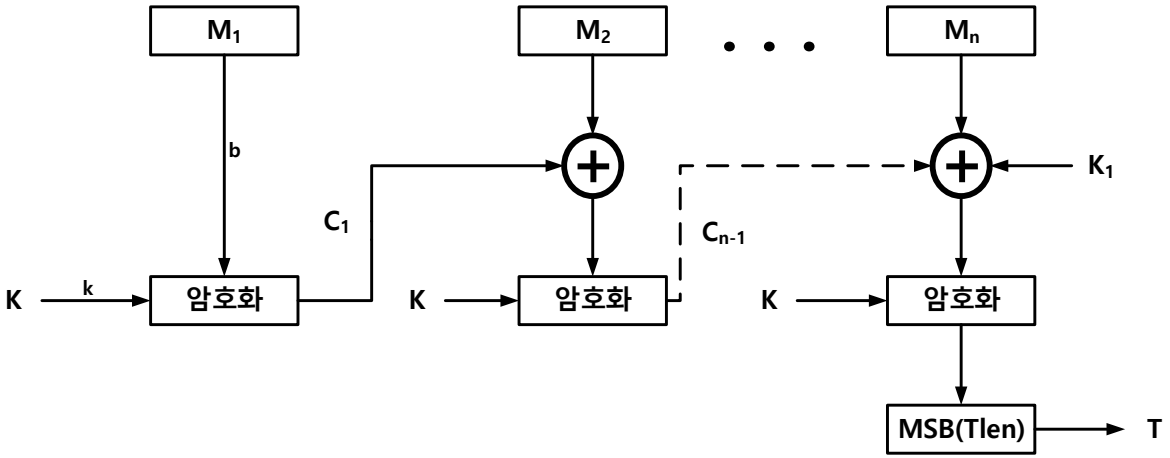
- 블록 암호기반 MAC
  - 암호기반 메시지 인증 코드 CMAC (Cipher-based Message Authentication Code)
    - 이 운용 모드는 AES와 3DES용 이며, NIST 특별문서 800-38B에 명시되어 있음
    - 메시지가 암호블록 길이b(AES=128, 3DES=64)의 n배 정수이면 메시지를 n 개의 블록으로 나눔
    - k비트 키 K와 b비트 키 K_1을 이용 (키의 길이도 AES, 3DES인 경우가 다름)
    - 패딩 하는 경우

$C_1 = E(K, M_1);$
$C_2 = E(K, (M_2 \wedge C_1));$

• • •
$C_n = E(K, (M_n \wedge C_{n-1}));$

$T = MSB_{Tlen}(C_n);$
$K_1 = E(0,K) << 1;$
$K_2 = E(0,k_1) << 1;$

T : MAC, 또는 Tag
Tlen : T의 비트열 길이
$MSB_s(X)$: 비트열 X의 왼쪽부터 s개 비트 반환



메시지 인증 코드

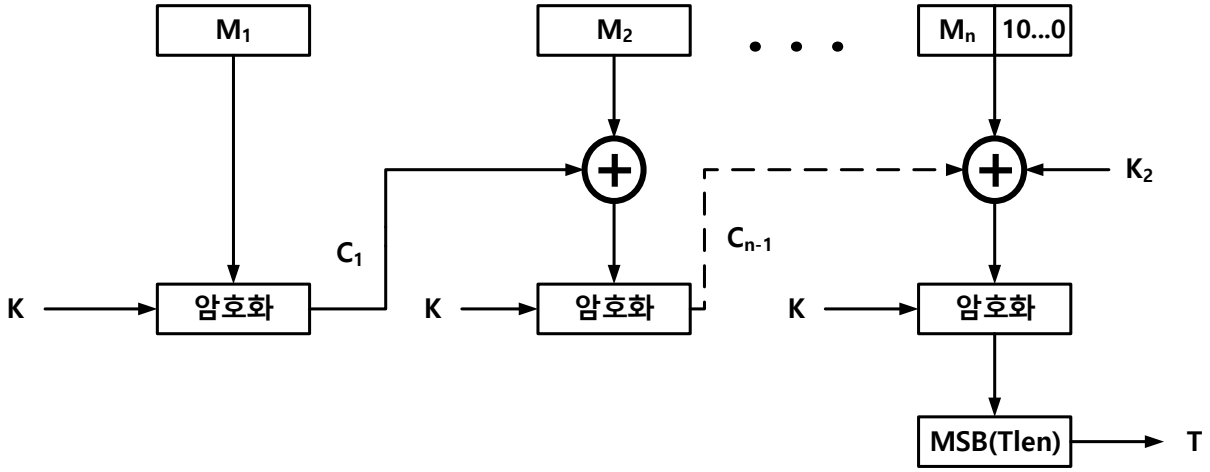
- 블록 암호기반 MAC
  - 암호기반 메시지 인증 코드 CMAC (Cipher-based Message Authentication Code)
    - 패딩을 할 경우
    - 마지막 블록 오른쪽에 패딩을 붙여 블록의 길이가 b비트가 되게 함
    - 마지막 연산에서 K_2를 사용함

$C_1 = E(K, M_1);$
$C_2 = E(K, (M_2 \wedge C_1));$

• • •
$C_n = E(K, (M_n \wedge C_{n-1}));$

$T = MSB_{Tlen}(C_n);$
$K_1 = E(0,K) << 1;$
$K_2 = E(0,k_1) << 1;$

T : MAC, 또는 Tag
Tlen : T의 비트열 길이
$MSB_s(X)$: 비트열 X의 왼쪽부터 s개 비트 반환



메시지 인증 코드

- 블록 암호기반 MAC
  - 암호블록 체인 카운터-메시지 인증 코드 CBC_MAC

    (Counter With Cipher Block Chaining-MAC (CBC_MAC))
    - 인증된 암호화 모드(Authentication Encryption) 라고도 함

      통신시 기밀성과 인증(무결성)을 동시에 보호하는 암호 시스템
    - AES 암호, CTR 모드, CMAC 사용
    - 암호화와 인증에 동일한 하나의 키 K를 사용
    - 비표와 유관데이터를 포함
      - 비표(Nonce): 페이로드와 유관 데이터에 할당되는 비표, 프로토콜 연관이 있는 모든 순간에 달라지는 유일 값, 이 값으로 재전송 공격이나 기타 공격을 막을 수 있음
      - 유관 데이터: 인증은 하지만 암호화는 하지 않는 데이터 (프로토콜 동작이 제대로 이뤄지기 위해 평문 상태로 전달되어야 하지만 프로토콜 헤더의 경우 인증이 필수)

메시지 인증 코드

- 블록 암호기반 MAC
  - 암호블록 체인 카운터-메시지 인증 코드 CBC_MAC

(Counter With Cipher Block Chaining-MAC (CBC_MAC))