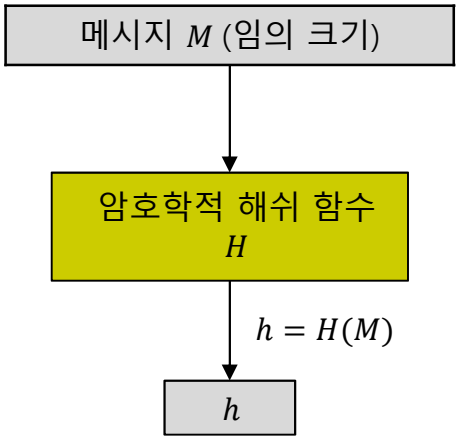
해쉬함수

목차

- 암호학적 해쉬 함수
  - 개념, 실행 예, 응용, 요구사항
- 생일 문제, 생일 역설
- 해쉬 함수 충돌 확률
- Merkle-Damgård scheme
- 해쉬 알고리즘 비교
- 해쉬 알고리즘: SHA-512
- 해쉬 함수 응용: MAC
  - 메시지 변경 확인, 메시지 인증
  - HMAC
  - CMAC
- 암호학적 해쉬 함수 응용: 전자서명
- 암호학적 해쉬 함수 응용: 패스워드 기반 개체 인증

암호학적 해쉬 함수



Hash value
Message digest
Message fingerprint
(고정크기)

암호학적 해쉬함수
• 해쉬 함수는 임의 크기 메시지를 입력 받아 고정 크기 출력을 생성 • 특정 해쉬 값을 생성하는 메시지를 찾는 것이 계산적으로 불가능해야 함 (one-way property) • 동일 해쉬 값을 생성하는 서로 다른 메시지를 찾는 것이 계산적으로 불가능해야 함 (collision-free property)

암호학적 해쉬 함수 실행 예

A.txt: Seoul

B.txt: seoul

C.txt: 한국 미국 독일 영국 서울 부산 울산 제주

Windows 명령프롬프트에서 실행		
명령어	해쉬값	해쉬 크기
certutil -hashfile A.txt MD5	Fd38499c5c04df42d1d78807aa4b7d7d	128비트 (16B)
certutil -hashfile B.txt MD5	D793880d5ba70feb1c2e259d1a373add	128비트 (16B)
certutil -hashfile C.txt MD5	80a7ddb2a3459a9b474c8c77bf9cb7eb	128비트 (16B)
certutil -hashfile A.txt MD5	Fd38499c5c04df42d1d78807aa4b7d7d	128비트 (16B)
certutil -hashfile A.txt SHA1	59f054fa5a2164fb3ae235dd40b71eea61fc005f	160비트 (20B)
certutil -hashfile A.txt SHA256	32b634b8898c07efbeb397da71cdc08819ac384c3a28cb4fcfe483f2cfa7e1d0	256비트 (32B)
certutil -hashfile A.txt SHA512	c057da638de6afda0cae5923dee8e4c8a22d228a2e25d9d705e4377bcad3dfe9a65573bdc92d91a25385f02d4d4177daeb6ae868514ab5fe19932601e37345c8	512비트 (64B)

해쉬 알고리즘

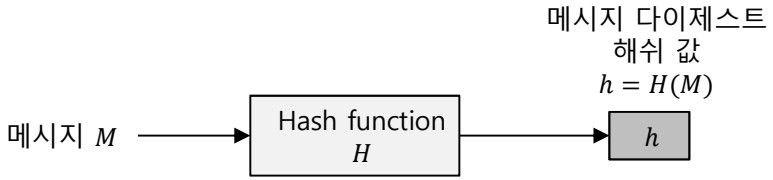
- MD4
- MD5
- SHA-1
- SHA-2 → SHA-224, SHA-256, SHA-384, SHA-512
- SHA-3

암호학적 해쉬 함수: 응용

- 암호학적 해쉬함수
  - 임의 크기 입력에 대해 (암호학적 특성을 갖는) 고정 크기 출력 생성
    - 메시지 다이제스트(message digest), 해쉬 값(hash value), 메시지 지문(message fingerprint)
  - 키가 필요 없음
  - 응용
    - 메시지 인증 (Message authentication)
      - 수신 메시지가 송신측이 보낸 (변경되지 않은 진짜) 메시지인지 확인
    - 디지털 서명 (Digital signatures)
      - 메시지의 해쉬 값에 대해 송신측 개인키로 디지털 서명 수행
    - 패스워드 파일
      - 사용자 패스워드의 해쉬 값을 패스워드 파일에 저장
      - 로그인 시 입력된 패스워드의 해쉬 값을 이전 해쉬 값과 비교
    - 침입 탐지, 바이러스 탐지
      - 시스템 내 각 파일의 해쉬 값 계산하여 별도 보관
      - 이후 각 파일의 해쉬 값 재계산하여 이전 해쉬 값과 비교

암호학적 해쉬 함수: 요구 사항

Reference: (Stallings, 2014; Forouzan, 2008)



해쉬 함수 요구사항	설명
역상저항성 preimage resistance one-way property	• 해쉬 값 h가 주어졌을 때, 아래 수식과 같이 해쉬 값 h를 생성하는 메시지 M을 찾는 것이 계산적으로 불가능할 것 $h = H(M)$ • b 비트를 입력 받아 n 비트를 출력하는 해쉬 함수의 경우($b > n$), 평균적으로 2^{b-n}개 서로 다른 메시지들에 대해 동일한 해쉬 값이 생성됨
제 2 역상저항성 second preimage resistance weak collision resistance	• 메시지 M이 주어졌을 때, 아래 수식과 같이 동일한 해쉬 값 h를 생성하는 다른 메시지 M'을 찾는 것이 계산적으로 불가능할 것 $h = H(M) = H(M'), \quad M \neq M'$
충돌저항성 collision resistance strong collision resistance	• 주어진 정보 없이, 아래 수식과 같이 동일한 해쉬 값 h를 생성하는 서로 다른 두 메시지 M, M'을 찾는 것이 계산적으로 불가능할 것 $h = H(M) = H(M'), \quad M \neq M'$

암호학적 해쉬 함수: 요구 사항

Reference: (Stallings, 2014; Forouzan, 2008)

- 안전한 해쉬 함수가 갖추어야 할 3가지 핵심 특성
  - 해쉬 값 h가 주어졌을 때, h=H(m)을 만족하는 메시지 m을 찾는 것이 계산적으로 불가능해야 함 → **역상저항성**
  - 동일 해쉬 값을 갖는 서로 다른 두 메시지 생성이 불가능해야 함
    - 메시지 m_1이 주어졌을 때, $h=H(m_1)=H(m_2)$를 만족하는 또다른 메시지 m_2의 생성이 계산적으로 불가능해야 함 → **제2역상저항성**
    - 공격자가 자유롭게 $h=H(m_1)=H(m_2)$를 만족하는 서로 다른 두 메시지 m_1, m_2를 생성하는 것이 계산적으로 불가능해야 함 → **충돌저항성**

충돌을 찾는 것이 얼마나 어려운가?

- 비둘기집 원리에 의해 해쉬 값의 충돌은 피할 수 없음
  - 비둘기 100마리가 99개 비둘기집에 들어간다면 최소 한 개 비둘기집에는 두 마리 이상 비둘기가 들어가야 한다
- 해쉬 함수의 충돌을 찾는 것이 얼마나 어려운가?
  - 해쉬 함수의 출력 크기가 n 비트인 경우 가능한 해쉬 값은 2^n개
  - 해쉬 함수의 출력 크기가 80 비트인 경우, 충돌을 찾기 위해 2^{80}개의 메시지가 필요한가?
    - 생일 문제(birthday problem)에 따르면 약 2^{40}개의 메시지만 필요

생일문제, 생일역설

Reference: (Stallings, 2014; Forouzan, 2008)

생일 문제

- 생일 문제 → n 명의 사람 중 같은 생일을 가진 사람들이 존재할 확률을 구하는 문제
- n명 중 생일 충돌 없을 확률 $P'(n)$ → n명 모두 생일이 다를 확률

- 1명 중 생일 충돌 없을 확률 $P'(1) = 1$
- 2명 중 생일 충돌 없을 확률 $P'(2) = \left(1 - \frac{1}{365}\right)$
- 3명 중 생일 충돌 없을 확률 $P'(3) = \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right)$
- n명 중 생일 충돌 없을 확률 $P'(n) = \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right) \times \cdots \times \left(1 - \frac{n-1}{365}\right)$

- n명 중 생일 충돌 발견 확률 $P(n)$ → n명 중 생일이 같은 사람이 발견될 확률 → $P(n) = 1 - P'(n)$

$$P(n) = 1 - \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right) \times \cdots \times \left(1 - \frac{n-1}{365}\right)$$

- 23명 중 생일 충돌 발견 확률 $P(23) = 1 - \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right) \times \cdots \times \left(1 - \frac{23-1}{365}\right) \approx 0.507$

- 생일 문제는 해쉬 함수의 충돌 확률을 찾는 것과 동일한 문제임 → 출력 크기 k 비트인 해쉬 함수의 경우는 365 대신 2^k

생일 역설

- 생일 역설 → 임의의 23명 중 같은 생일이 발견될 확률은 50% 이상이다
- 임의 그룹 23명 중 생일 충돌 발견 확률은 약 50% 이상
- 임의 그룹 40명 중 생일 충돌 발견 확률은 약 90% 이상

해쉬 함수 출력 크기와 해쉬함수 충돌 확률

Reference: (Stallings, 2014; Forouzan, 2008)

해쉬함수 충돌 확률

- $0 \sim N-1$ 범위 내 같은 값을, P 이상 확률로, 발견하려면 $0 \sim N-1$ 중 최소 몇 개를 임의 추출해야 하는가?

- 임의 추출된 1개 샘플 내 충돌 없을 확률 $P'(1) = 1$
- 임의 추출된 2개 샘플 내 충돌 없을 확률 $P'(2) = 1 \times \left(1 - \frac{1}{N}\right)$
- 임의 추출된 3개 샘플 내 충돌 없을 확률 $P'(3) = 1 \times \left(1 - \frac{1}{N}\right) \times \left(1 - \frac{2}{N}\right)$
- 임의 추출된 n개 샘플 내 충돌 없을 확률 $P'(n) = 1 \times \left(1 - \frac{1}{N}\right) \times \left(1 - \frac{2}{N}\right) \times \cdots \times \left(1 - \frac{n-1}{N}\right)$

- 임의 추출된 n개 샘플 내 충돌 발견 확률 $P(n) = 1 - P'(n)$

$$P(n) = 1 - 1 \times \left(1 - \frac{1}{N}\right) \times \left(1 - \frac{2}{N}\right) \times \cdots \times \left(1 - \frac{n-1}{N}\right)$$

- $0 \sim N-1$ 범위 내 같은 값을, 50% 이상 확률로, 발견하려면 $0 \sim N-1$ 중 최소 몇 개를 임의 추출해야 하는가?

$$1 - 1 \times \left(1 - \frac{1}{N}\right) \times \left(1 - \frac{2}{N}\right) \times \cdots \times \left(1 - \frac{n-1}{N}\right) \geq \frac{1}{2}$$

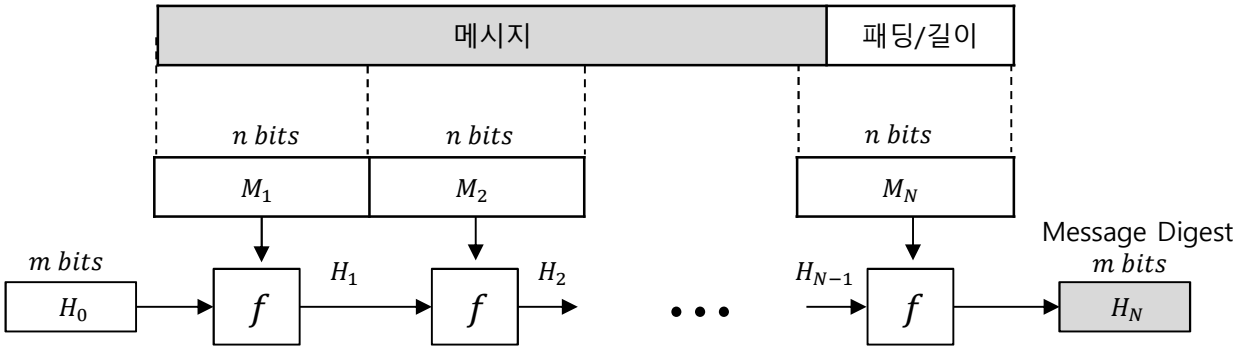
$$n \geq (2 \times \ln 2)^{1/2} \times N^{1/2} \approx \mathbf{1.18} \times \sqrt{\boldsymbol{N}}$$

- **출력 크기 k 비트인 해쉬 함수**의 충돌을, 50%의 확률로, 찾기 위해 필요한 메시지의 수는 **약 $\sqrt{2^k}$**
- **출력 크기 80 비트인 해쉬 함수**의 충돌을, 50%의 확률로, 찾기 위해 필요한 메시지의 수는 **약 $\sqrt{2^{80}} = 2^{40}$**
- **x 비트 보안 수준**을 얻기 위해 필요한 해쉬 함수의 출력 크기는 **$2x$ 비트**이상 이어야 함
- 생일 공격(birthday attack) → 생일 문제의 결과를 활용한 공격

Merkle-Damgård scheme

Reference: (Stallings, 2014; Forouzan, 2008)

- 해쉬 함수 (Hash function)
  - 임의 길이 메시지를 입력으로 받아 고정 길이 출력 생성
- Merkle-Damgård scheme
  - 메시지를 n 비트의 배수가 되도록 패딩 및 길이 추가
  - 패딩/길이 추가된 메시지를 n 비트 단위 블록들로 분할
  - m 비트 출력을 생성하는 함수 f를 각 n 비트 메시지 블록에 반복 적용
    - 함수 f 의 입력:
      - n 비트 메시지 블록
      - 함수 f 의 직전 출력 (최초에는 IV 값 사용)



해쉬 알고리즘

- 해쉬 알고리즘
  - MD4, MD5, SHA-1, SHA-2, SHA-3
- 해쉬 알고리즘의 두 유형
  - 블록 암호 기반 해쉬 함수
  - 전용 해쉬 함수

SHA → Secure Hash Algorithm
MD → Message Digest
MD5 → 출력 128-bit

Truncated Message Digest

- 해쉬 함수의 출력 크기(예: L 비트)보다 짧은 길이(예: K 비트)의 해쉬 값이 필요한 경우 해쉬 함수 출력의 leftmost K 비트를 사용
  (https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-107r1.pdf)

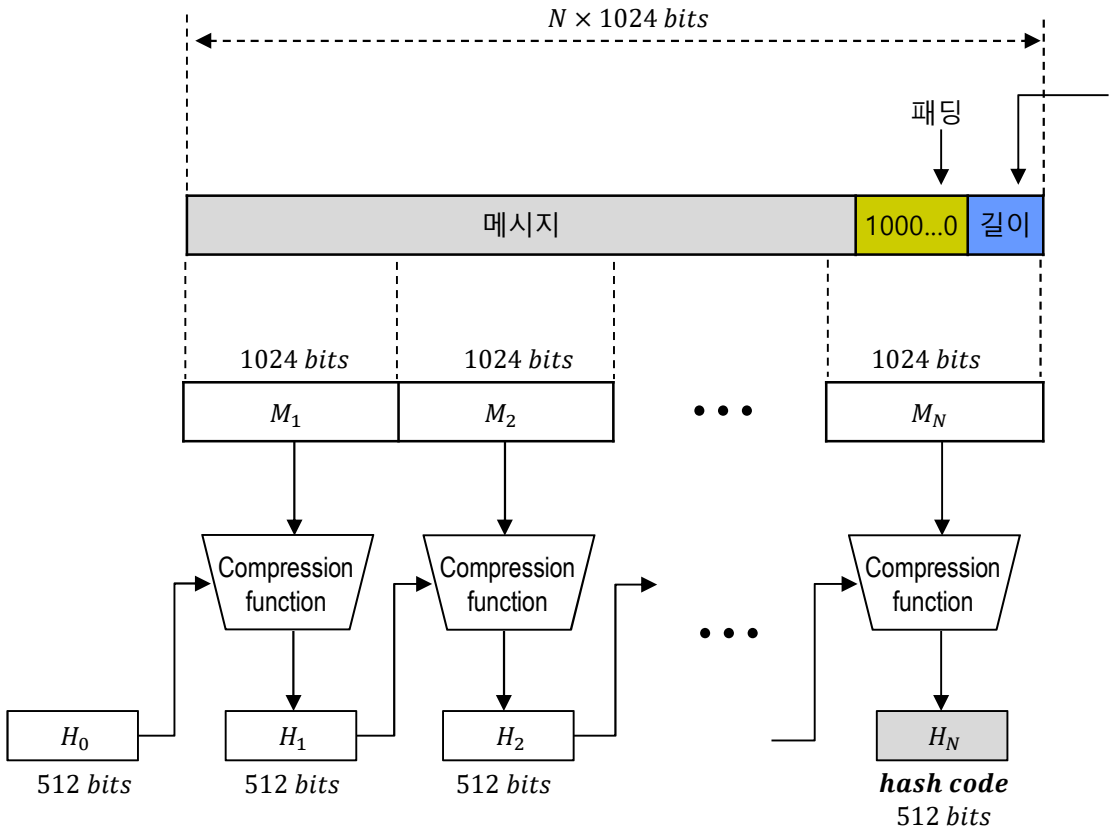
	SHA-1	SHA-2			
		SHA-224	SHA-256	SHA-384	SHA-512
Digest size (bits)	160	224	256	384	512
Maximum message size (bits)	$2^{64}-1$	$2^{64}-1$	$2^{64}-1$	$2^{128}-1$	$2^{128}-1$
Block size (bits)	512	512	512	1024	1024
Number of rounds	80	64	64	80	80
Word size (bits)	32	32	32	64	64

해쉬 알고리즘: SHA-512

Reference: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

SHA-512: Message Digest Generation

Reference: (Forouzan, 2008)



- 메시지 길이 L
- 패딩 내 비트 0의 개수 Z

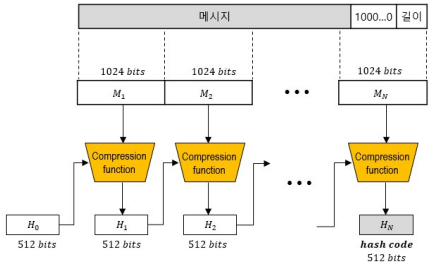
$$L + 1 + Z \equiv 896\ (mod\ 1024)$$

- 예) SHA-512에서 2590 비트 크기의 원본 메시지에 대해 생성되는 패딩 내 비트 0의 개수는 353개
- 예) SHA-512에서 24 비트 크기의 원본 메시지에 대해 생성되는 길이 필드(128 비트)의 비트열은 $000 \ldots 011000$

Reference: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

SHA-512: Compression function

Reference: (Forouzan, 2008)



Round 상수 80개 $K_0 \sim K_{79}$
- 각 라운드에 64 비트 크기 K_i 입력
- 80개 상수 값 → 이후 슬라이드 참조

Digest 512 비트
- 64 비트 워드 8개로 구성

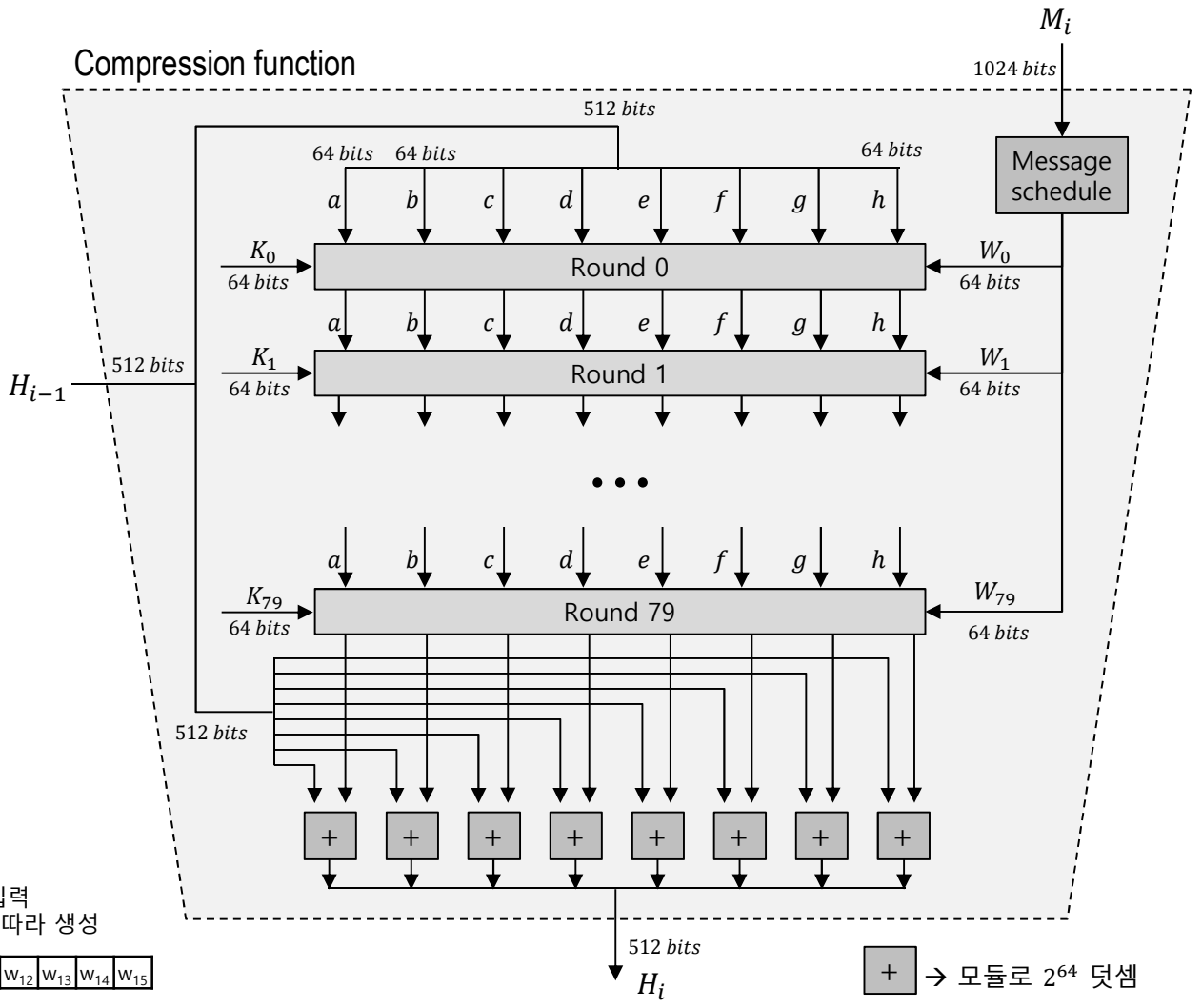
a	b	c	d	e	f	g	h

- 메시지 다이제스트 초기값

```
a => 6a09e667f3bcc908
b => bb67ae8584caa73b
c => 3c6ef372fe94f82b
d => a54ff53a5f1d36f1
e => 510e527fade682d1
f => 9b05688c2b3e6c1f
g => 1f83d9abfb41bd6b
h => 5be0cd19137e2179
```

Message block 1024 비트
- 64 비트 워드 16개로 구성
- Round 0~15까지 각 라운드에 한 워드 입력
- Round 16~79의 입력 워드 W_i는 규칙에 따라 생성

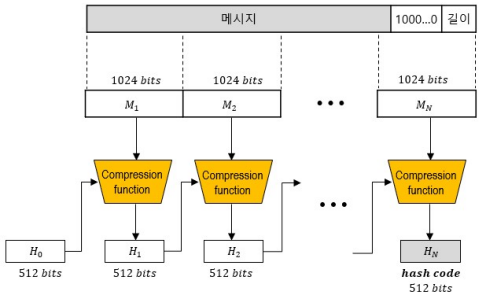
w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}	w_{12}	w_{13}	w_{14}	w_{15}



Reference: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

SHA-512: Round constants

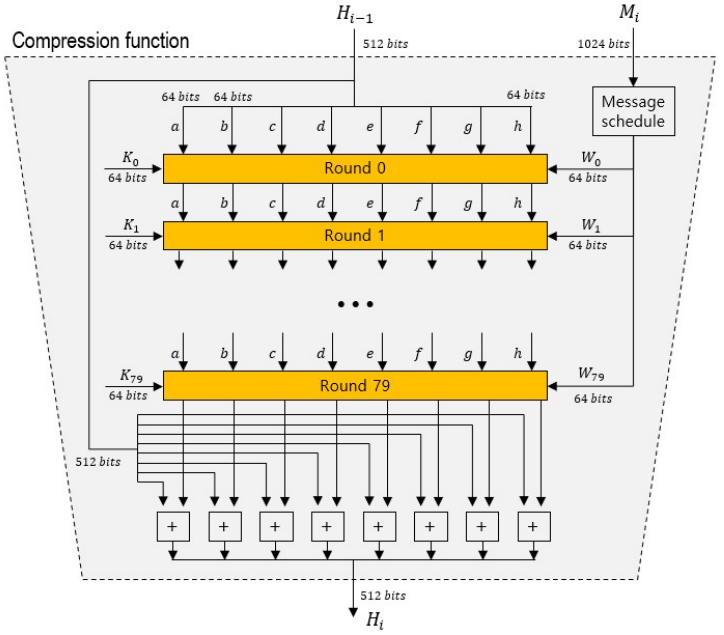
Reference: (Forouzan, 2008)



Round 상수 80개 $K_0 \sim K_{79}$

- 각 라운드에 64 비트 크기 K_i 입력
- 80개 상수 값 $K_0 \sim K_{79}$ (왼쪽에서 오른쪽으로)

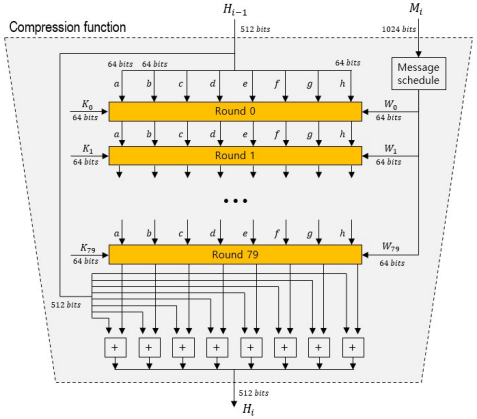
```
428a2f98d728ae22 7137449123ef65cd b5c0fbcfec4d3b2f e9b5dba58189dbbc
3956c25bf348b538 59f111f1b605d019 923f82a4af194f9b ab1c5ed5da6d8118
d807aa98a3030242 12835b0145706fbe 243185be4ee4b28c 550c7dc3d5ffb4e2
72be5d74f27b896f 80deb1fe3b1696b1 9bdc06a725c71235 c19bf174cf692694
e49b69c19ef14ad2 efbe4786384f25e3 0fc19dc68b8cd5b5 240ca1cc77ac9c65
2de92c6f592b0275 4a7484aa6ea6e483 5cb0a9dcbd41fbd4 76f988da831153b5
983e5152ee66dfab a831c66d2db43210 b00327c898fb213f bf597fc7beef0ee4
c6e00bf33da88fc2 d5a79147930aa725 06ca6351e003826f 142929670a0e6e70
27b70a8546d22ffc 2e1b21385c26c926 4d2c6dfc5ac42aed 53380d139d95b3df
650a73548baf63de 766a0abb3c77b2a8 81c2c92e47edaee6 92722c851482353b
a2bfe8a14cf10364 a81a664bbc423001 c24b8b70d0f89791 c76c51a30654be30
d192e819d6ef5218 d69906245565a910 f40e35855771202a 106aa07032bbd1b8
19a4c116b8d2d0c8 1e376c085141ab53 2748774cdf8eeb99 34b0bcb5e19b48a8
391c0cb3c5c95a63 4ed8aa4ae3418acb 5b9cca4f7763e373 682e6ff3d6b2b8a3
748f82ee5defb2fc 78a5636f43172f60 84c87814a1f0ab72 8cc702081a6439ec
90befffa23631e28 a4506cebde82bde9 bef9a3f7b2c67915 c67178f2e372532b
ca273eceea26619c d186b8c721c0c207 eada7dd6cde0eb1e f57d4f7fee6ed178
06f067aa72176fba 0a637dc5a2c898a6 113f9804bef90dae 1b710b35131c471b
28db77f523047d84 32caab7b40c72493 3c9ebe0a15c9bebc 431d67c49c100d4c
4cc5d4becb3e42b6 597f299cfc657e2a 5fcb6fab3ad6faec 6c44198c4a475817
```

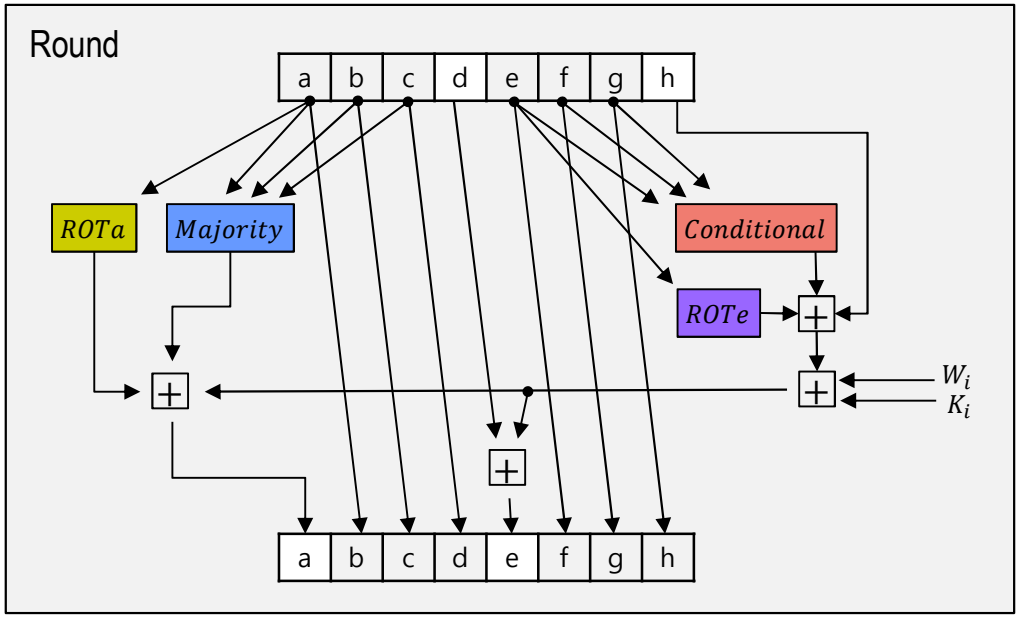


Reference: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

SHA-512: 라운드

Reference: (Forouzan, 2008)





Digest 512 비트

- 64 비트 워드 8개로 구성

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*

- 메시지 다이제스트 초기값

```
a => 6a09e667f3bcc908
b => bb67ae8584caa73b
c => 3c6ef372fe94f82b
d => a54ff53a5f1d36f1
e => 510e527fade682d1
f => 9b05688c2b3e6c1f
g => 1f83d9abfb41bd6b
h => 5be0cd19137e2179
```

$$ROTa = ROR^{28}(a) \oplus ROR^{34}(a) \oplus ROR^{39}(a)$$
$$Majority = (a\ AND\ b) \oplus (a\ AND\ c) \oplus (b\ AND\ c)$$

$$ROTe = ROR^{14}(e) \oplus ROR^{18}(e) \oplus ROR^{41}(e)$$
$$Conditional = (e\ AND\ f) \oplus (NOT\ e\ AND\ g)$$

$ROR^n(x)$ → 64비트 x를 n 비트 오른쪽 회전
W_i → 1024비트 입력 블록으로부터 생성된 총 80개 워드(64비트) 중 하나
K_i → 64비트 상수

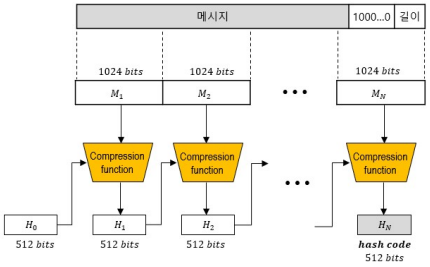
⊞ → 모듈로 2^{64} 덧셈

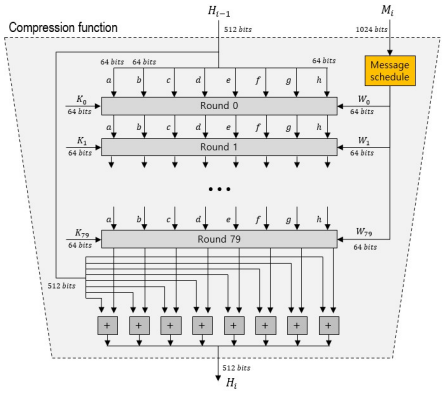
Reference: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

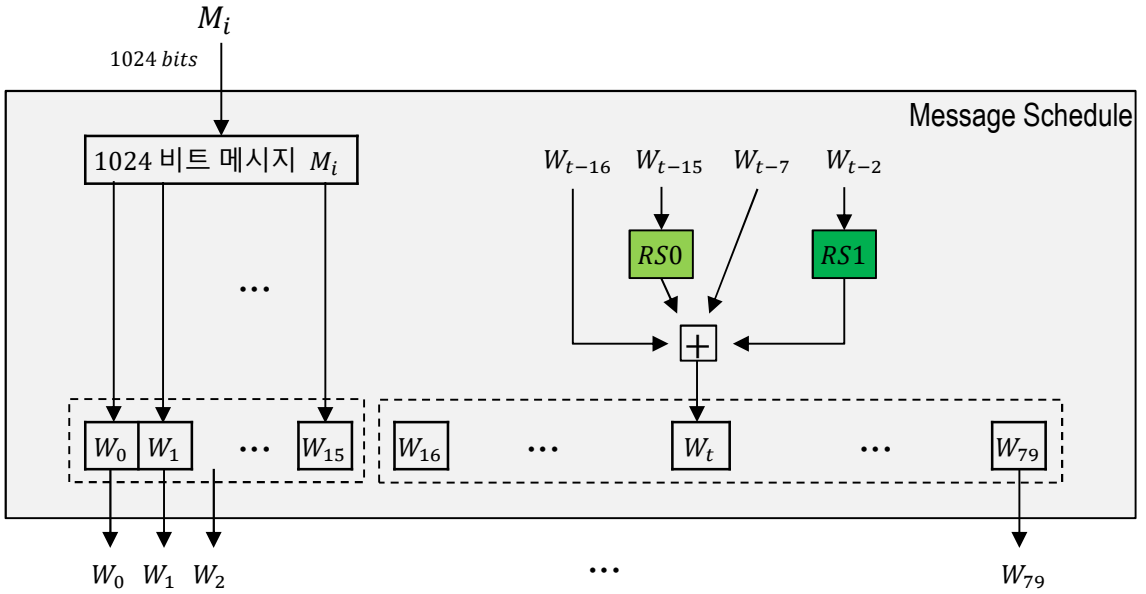
SHA-512:

1024-bit 메시지로부터 80개 워드(64-bit) 생성

Reference: (Forouzan, 2008)







W_0부터 W_{15}까지 16개 워드들은 1024비트 메시지로부터 그대로 복사
W_{16}부터 W_{79}까지 64개 각 워드 W_t는 이전 4개 워드들 W_{t-16} , W_{t-15} , W_{t-7} , W_{t-2} 로부터 생성

$$RS0(x) = ROR^1(x) \oplus ROR^8(x) \oplus SHR^7(x)$$
$$RS1(x) = ROR^{19}(x) \oplus ROR^{61}(x) \oplus SHR^6(x)$$

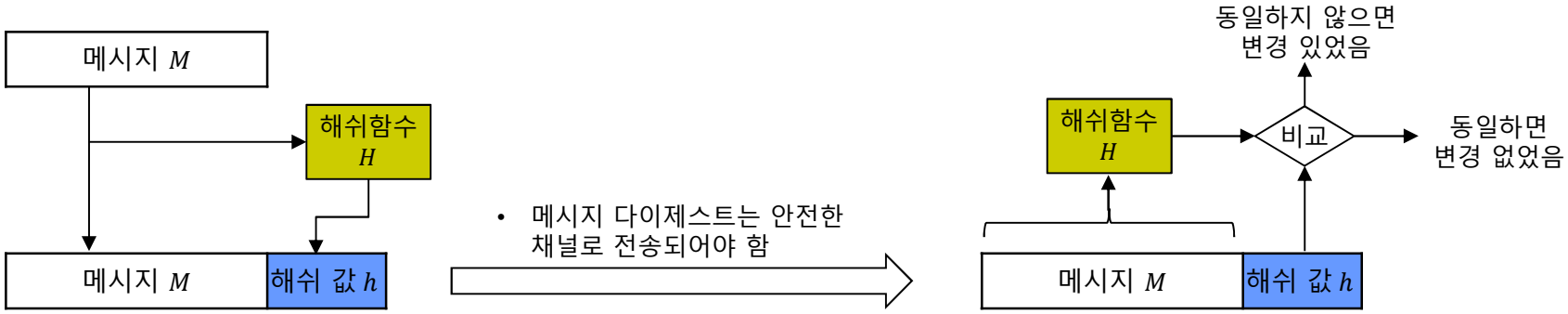
$ROR^n(x)$ → 64비트 x를 n 비트 오른쪽 회전
$SHR^n(x)$ → 64비트 x를 n 비트 오른쪽 쉬프트 (왼쪽 빈 곳은 0으로 채움)
W_i → 1024비트 입력 블록으로부터 생성된 총 80개 워드(64비트) 중 하나

$\boxplus$ → 모듈로 2^{64} 덧셈

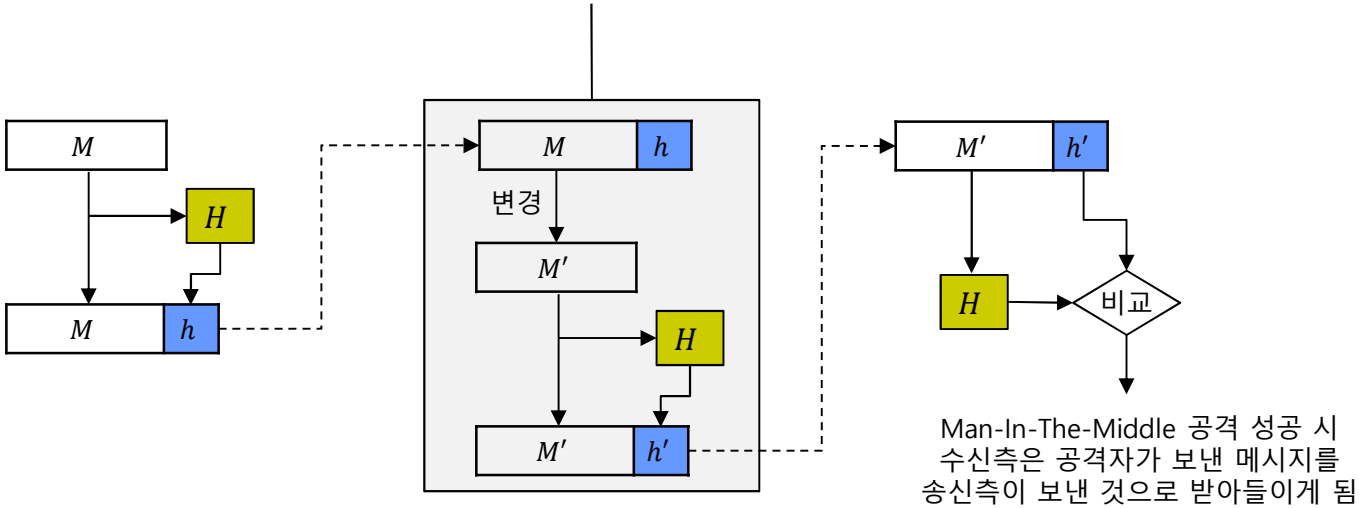
해쉬 함수 응용: MAC

해쉬 함수 응용: 메시지 변경 확인

Reference: (Stallings, 2014; Forouzan, 2008)



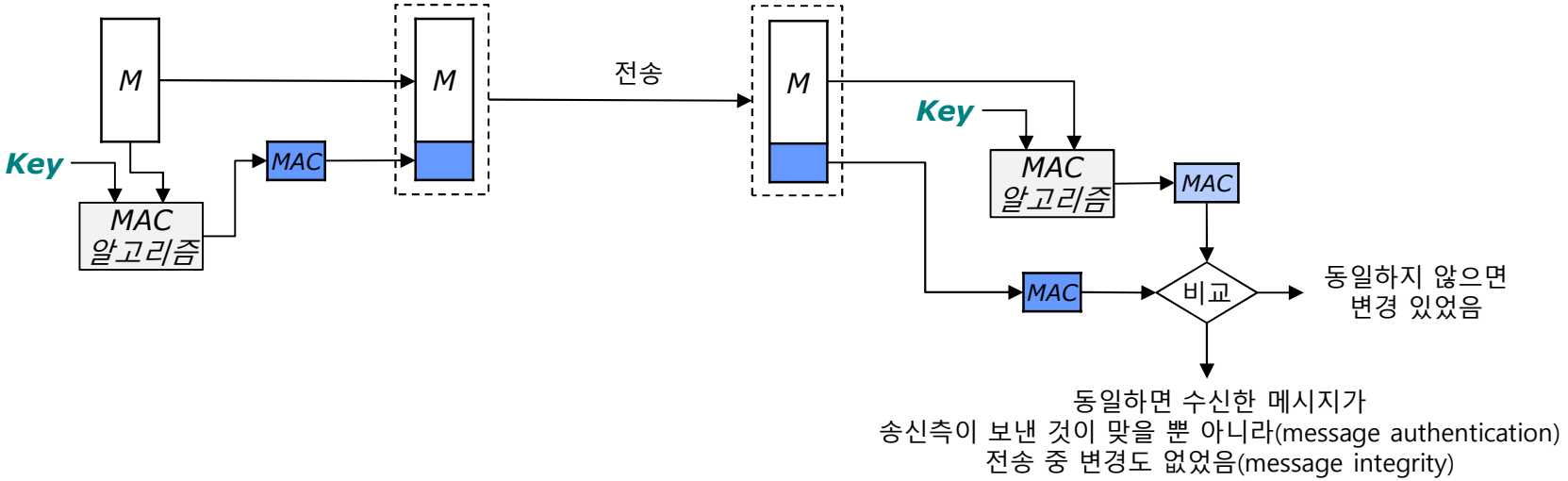
• 안전하지 않은 채널로 전송되는 경우 Man-in-the-middle attack 가능



Man-In-The-Middle 공격 성공 시
수신측은 공격자가 보낸 메시지를
송신측이 보낸 것으로 받아들이게 됨

해쉬 함수 응용: 메시지 인증, MAC

Reference: (Stallings, 2014; Forouzan, 2008)



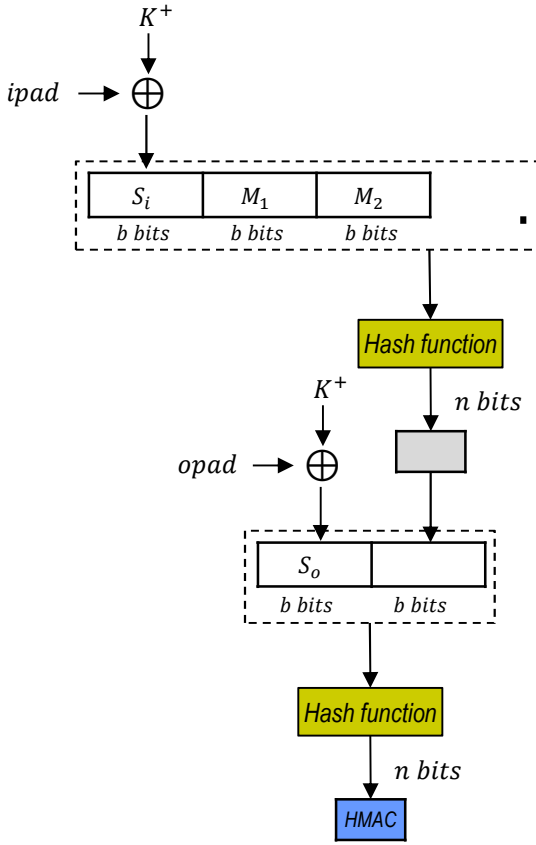
MAC (Message Authentication Code, 메시지 인증 코드)

- 송신측은 메시지와 공유키를 이용하여 MAC을 생성한 후 메시지와 함께 송신
- 수신측은, 수신된 메시지와 공유키로부터 MAC을 생성한 후, 수신된 MAC과 동일한지 비교
- 수신된 메시지가 송신측이 보낸 진짜 메시지가 맞는지 확인하기 위해 사용되는 코드
- 메시지 인증(message authentication)과 메시지 무결성(message integrity) 제공
- 송수신측 간 공유 대칭키 필요
- 부인 방지 제공 불가
- 안전한 MAC 알고리즘 사용 필요
- 암호학적 해쉬 함수를 이용하는 MAC 알고리즘 예: HMAC
- 블록 암호를 이용하는 MAC 알고리즘 예: CMAC

Reference: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.198-1.pdf

HMAC (Hash-based MAC)

Reference: (Stallings, 2014; Forouzan, 2008)



$$HMAC(K, M) = H\left((K^+ \oplus opad) \parallel H\left((K^+ \oplus ipad) \parallel M\right)\right)$$

HMAC (Hashed MAC)

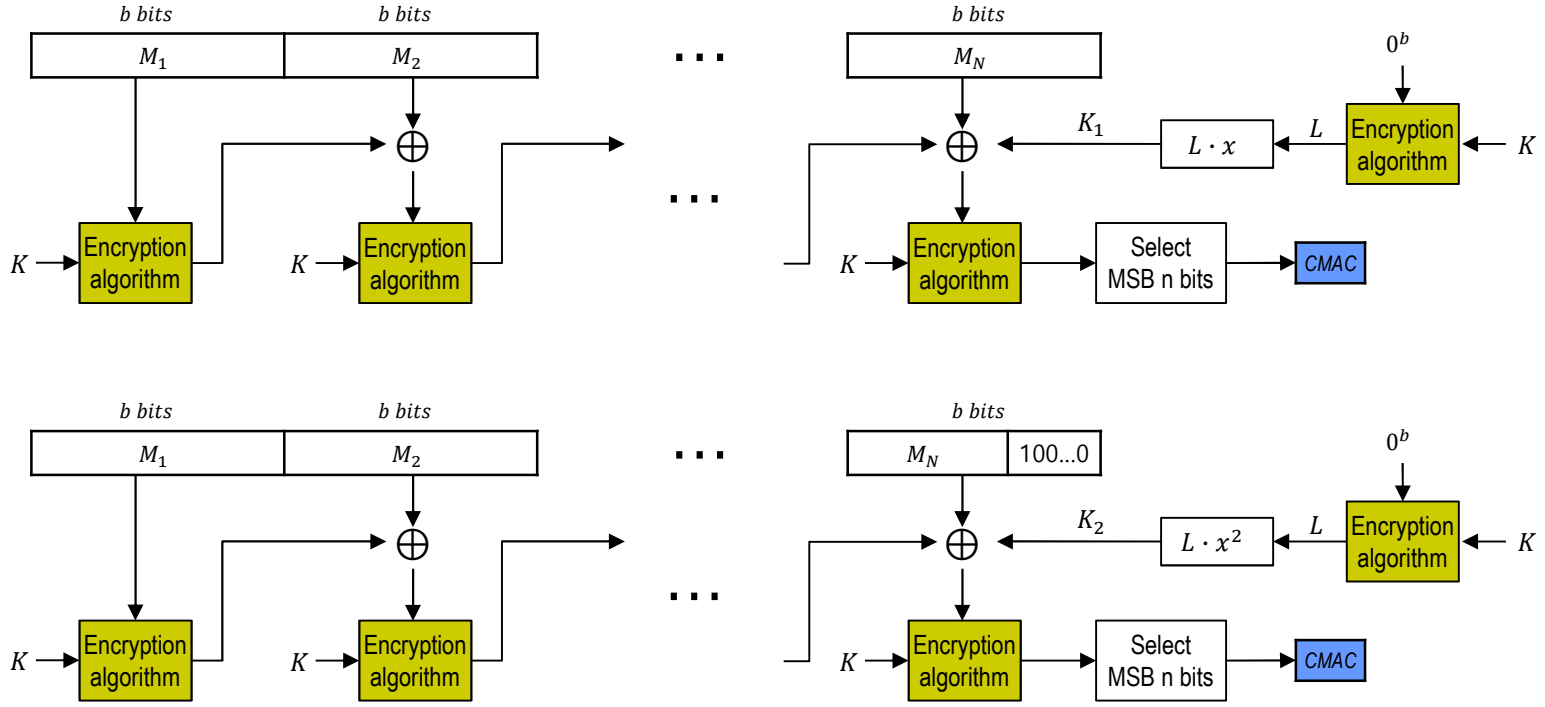
- HMAC에서는 기존 해쉬 함수를 수정 없이 블랙박스처럼 사용 가능 → 해쉬함수의 보안 결함이 발견되면 더 안전한 해쉬함수로의 교체 용이
- RFC-2104, NIST FIPS 198로 공표됨
- IP security, SSL 등에 사용됨

HMAC 생성 절차

- 입력 메시지 M을 N개의 b 비트 블록들로 분할 (해쉬함수에서 명시된 패딩 포함)
- 키 K의 크기가 블록 크기 b 보다 작다면 키 K에 비트 0을 추가하여 b 비트 크기의 K^+ 생성. 키 K의 크기가 블록 크기 b 보다 크다면 키를 해쉬한 결과(즉 $H(K)$)에 비트 0을 추가하여 b 비트 크기의 K^+ 생성.
- K^+ 와 $ipad$를 XOR하여 b 비트 블록 S_i 생성 ($ipad$는 00110110을 $b/8$ 회 반복하여 얻은 비트열)
- K^+ 와 $opad$를 XOR하여 b 비트 블록 S_o 생성 ($opad$는 01011100을 $b/8$ 회 반복하여 얻은 비트열)
- 해쉬함수의 n 비트 출력을 HMAC으로 출력

CMAC (Cipher-based MAC)

Reference: (Stallings, 2014; Forouzan, 2008)



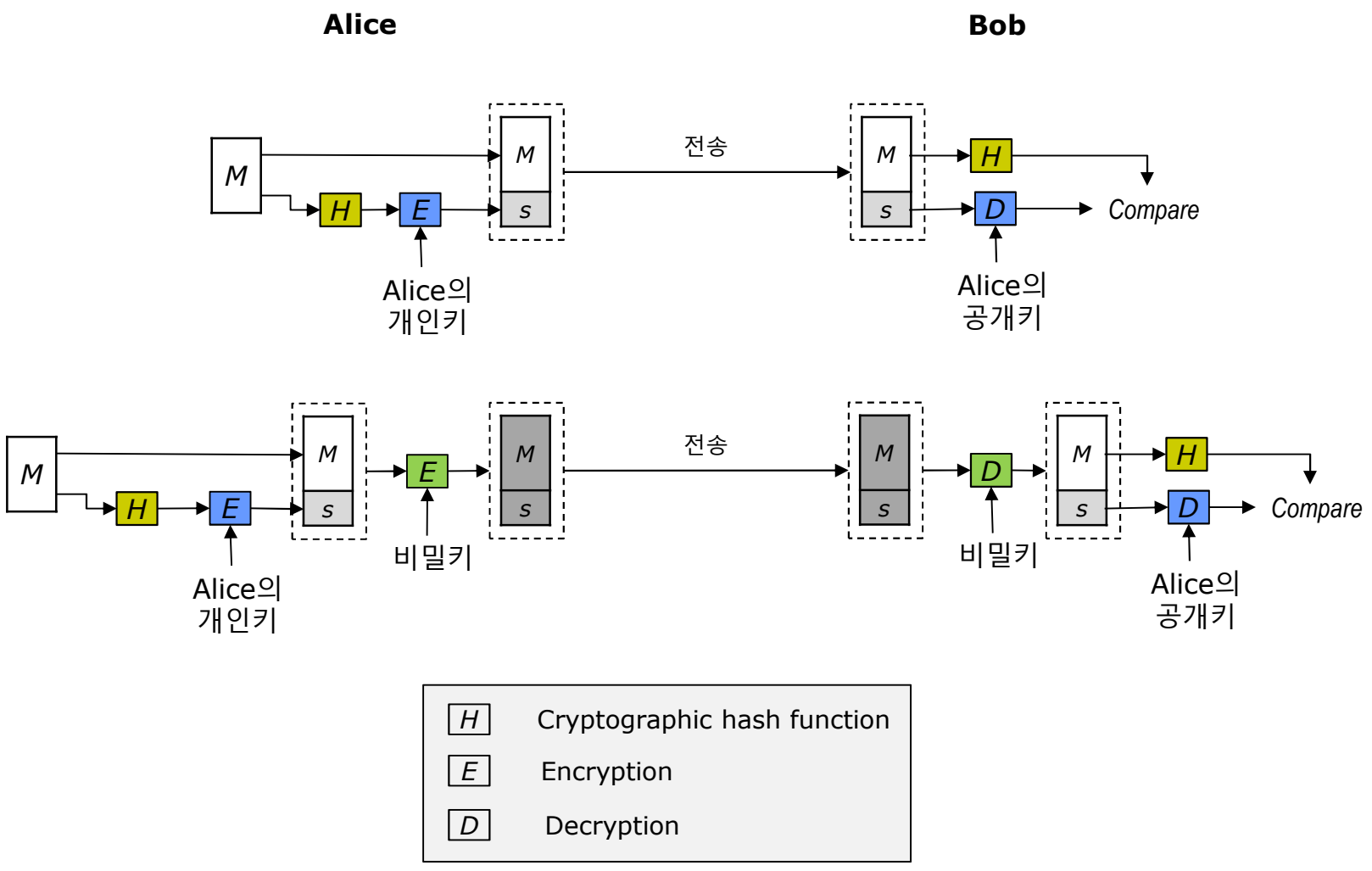
CMAC (Cipher-based MAC)

- 블록 암호화 모듈(AES인 경우 위 그림에서 b는 128 bits, K는 128, 192, 256 bits 중 하나)을 이용하여 MAC 생성
- 입력 메시지 M을 N개의 b 비트 블록들로 분할 (마지막 블록이 블록 크기 배수 아니면 비트 100...0 형식의 패딩 추가)
- N개 메시지 블록에 대해 동일한 키 K를 사용하여 체인 방식의 블록암호화 N번 적용
- M_2부터 M_{N-1}까지 각 메시지 블록의 경우 직전 생성된 암호블록과 XOR한 결과와 키 K를 이용하여 블록암호화 수행
- 마지막 블록 M_N 의 경우 마지막 블록에 패딩이 적용되었는지 유무에 따라 다른 처리 수행
- 위 그림의 $L \cdot x$ 및 $L \cdot x^2$는 $GF(2^b)$에서의 다항식 곱셈임. $x = 00 \ldots 0010$이며, $x^2 = 00 \ldots 0100$임. 기약다항식은 $b = 64$인 경우 $x^{64} + x^4 + x^3 + x + 1$ 이고 $b = 128$ 인 경우 $x^{128} + x^7 + x^2 + x + 1$
- 마지막 블록암호화 결과 비트열에서 왼쪽 n 비트를 CMAC으로 출력

해쉬 함수 응용: 전자서명

전자 서명: 서명과 검증

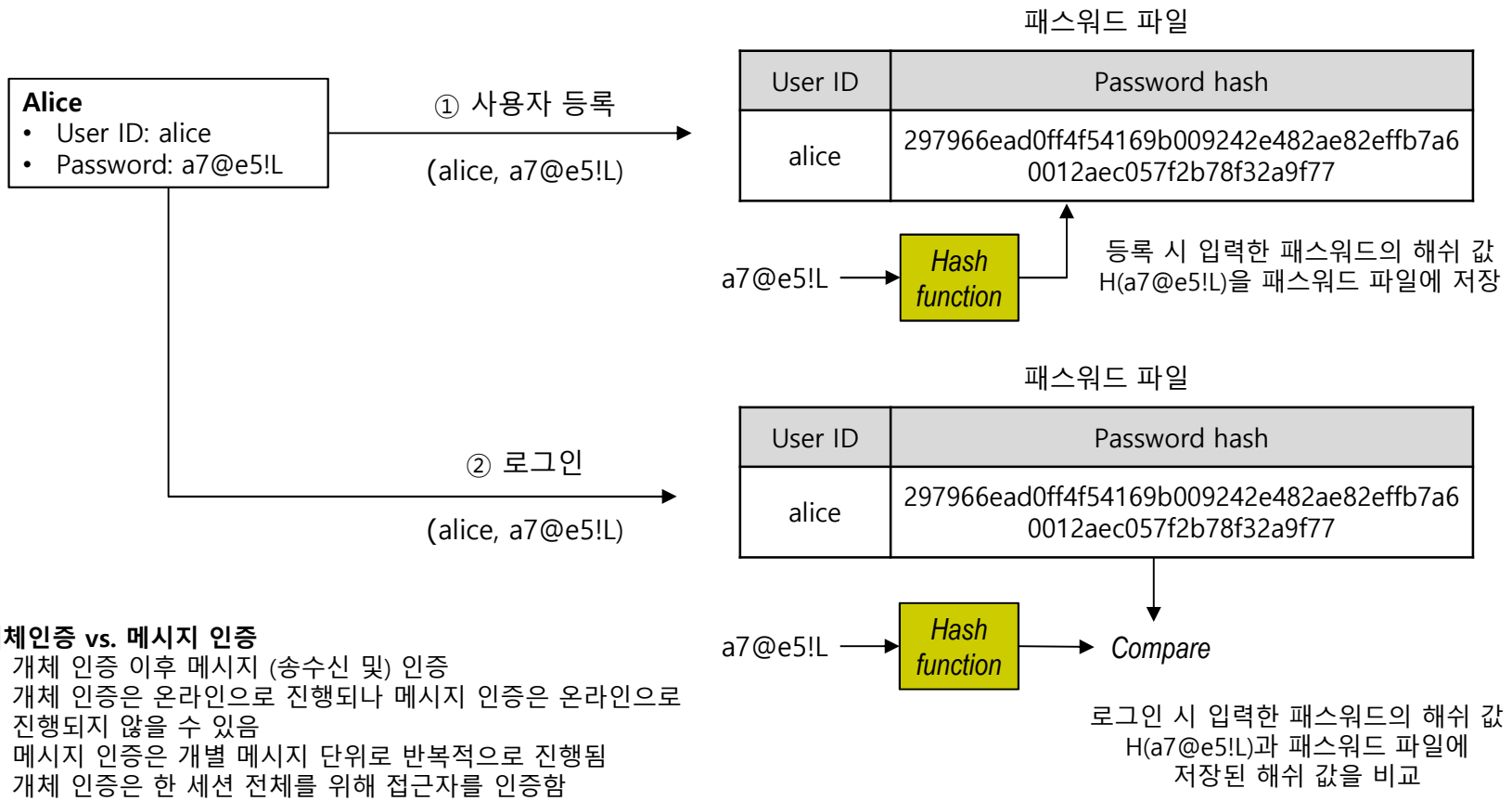
Reference: (Stallings, 2014)



해쉬 함수 응용:
패스워드 기반 개체 인증

해쉬 함수 응용: 패스워드 기반 개체 인증

Reference: (Forouzan, 2008)



개체인증 vs. 메시지 인증

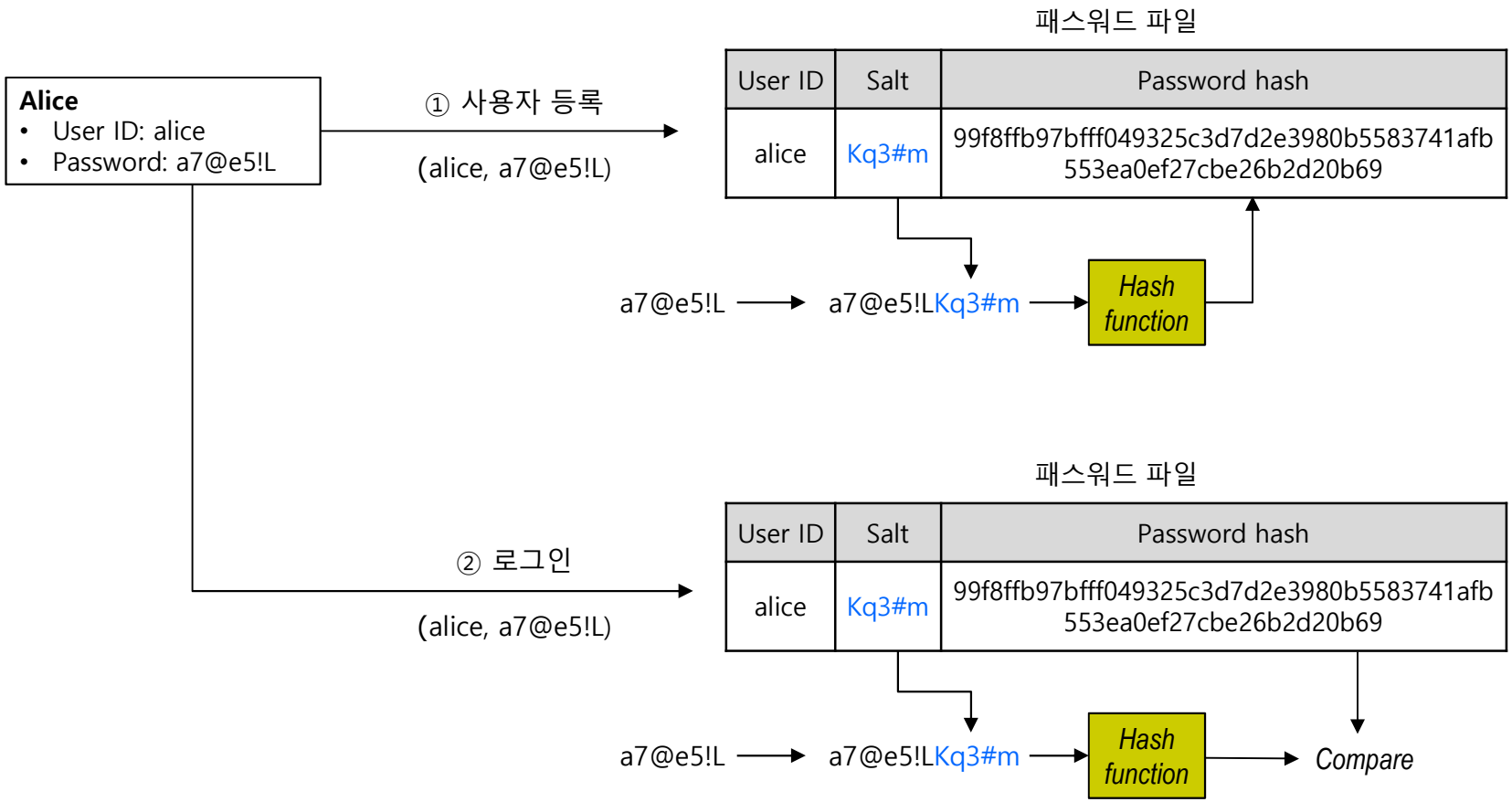
- 개체 인증 이후 메시지 (송수신 및) 인증
- 개체 인증은 온라인으로 진행되나 메시지 인증은 온라인으로 진행되지 않을 수 있음
- 메시지 인증은 개별 메시지 단위로 반복적으로 진행됨
- 개체 인증은 한 세션 전체를 위해 접근자를 인증함

개체 인증을 위한 개체 증명 정보 유형

- Something known → 비밀번호, PIN 등
- Something possessed → 여권, 신분증, 운전면허증, 신용카드 등
- Something inherent → 지문, 홍채, 얼굴 인식 등

해쉬 함수 응용: 패스워드 기반 개체 인증 (Salt 추가)

Reference: (Forouzan, 2008)



References

- Behrouz A. Forouzan, Cryptography and Network Security, McGraw-Hill, 2008
- William Stallings, Cryptography and Network Security: Principles and Practice, Sixth Edition, Prentice Hall, 2014
- Christof Paar, Jan Pelzl, Understanding Cryptography: A Textbook for Students and Practitioners, Springer, 2010
- 김명환, 수리암호학개론, 2019
- 정민석, 암호수학, 경문사, 2017
- 최은미, 정수와 암호론, 북스힐, 2019
- 이민섭, 정수론과 암호론, 교우사, 2008