제 8 장
일방향 해시 함수

박 종 혁 교수

Tel: 970-6702

Email: jhpark1@seoultech.ac.kr

1절 일방향 해시 함수

2절 일방향 해시 함수의 응용 예

3절 일방향 해시 함수의 예

4절 일방향 해시 함수 SHA-1

5절 일방향 해시 함수 SHA-512

6절 일방향 해시 함수에 대한 공격

7절 어떤 일방향 해시 함수를 사용하면 좋은가?

8절 일방향 해시 함수로 해결할 수 없는 문제

제1절 일방향 해시 함수

1.1 파일의 진위

1.2 일방향 해시 함수란?

1.3 일방향 해시 함수의 성질

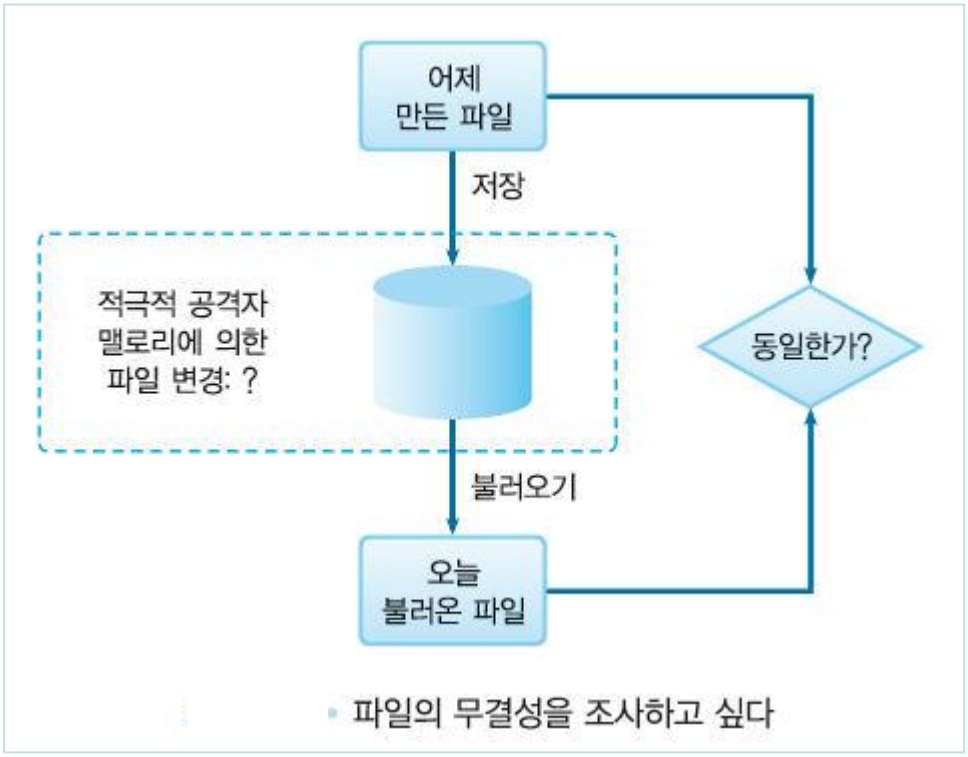
1.4 해시 함수 관련 용어

1.1 파일의 진위

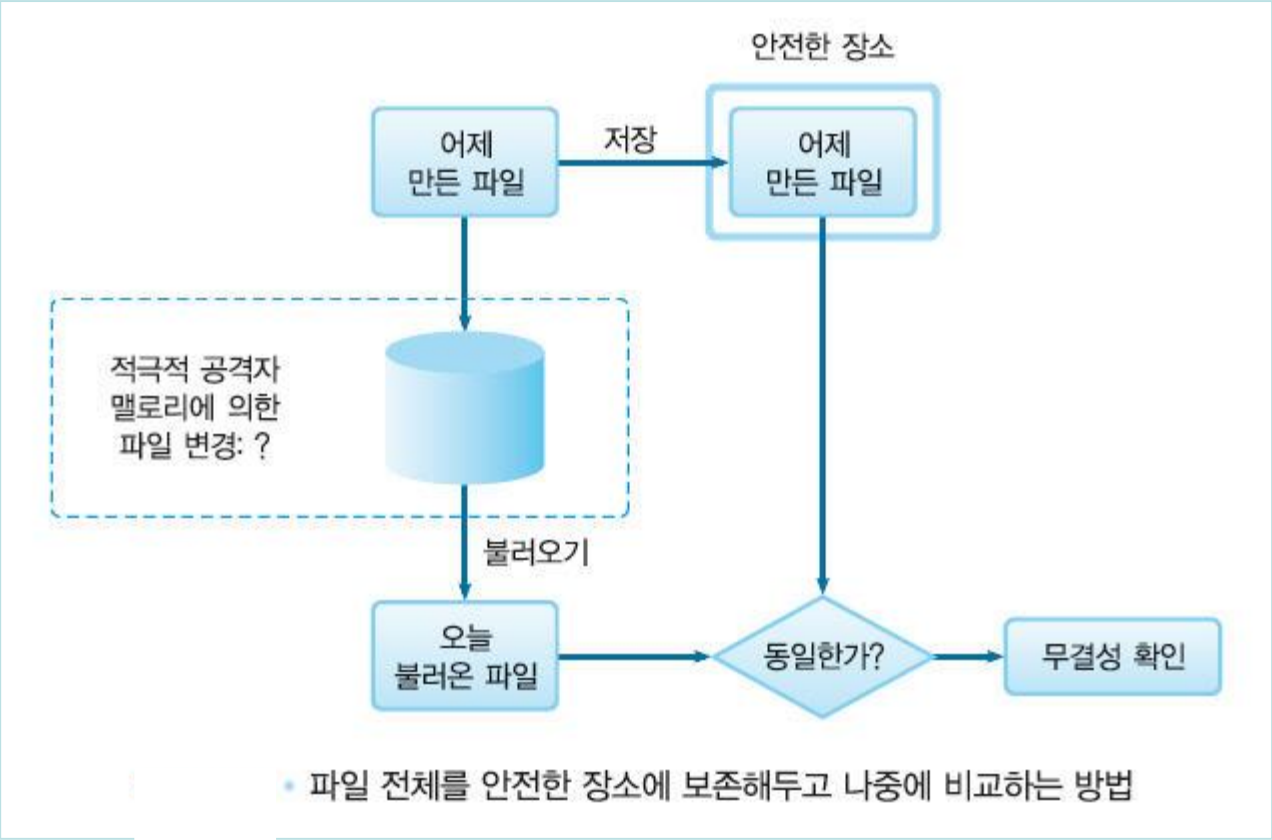
- **어제 저장한 파일과 오늘의 파일 비교**
  - 밤새 맬로리가 파일을 변경했는지 어떤지를 조사하고 싶다

- **무결성(integrity)**
  - 파일이 변경되지 않았음

파일의 무결성을 조사하고 싶다



파일 전체를 안전한 장소에 보존해 두고, 나중에 비교하는 방법

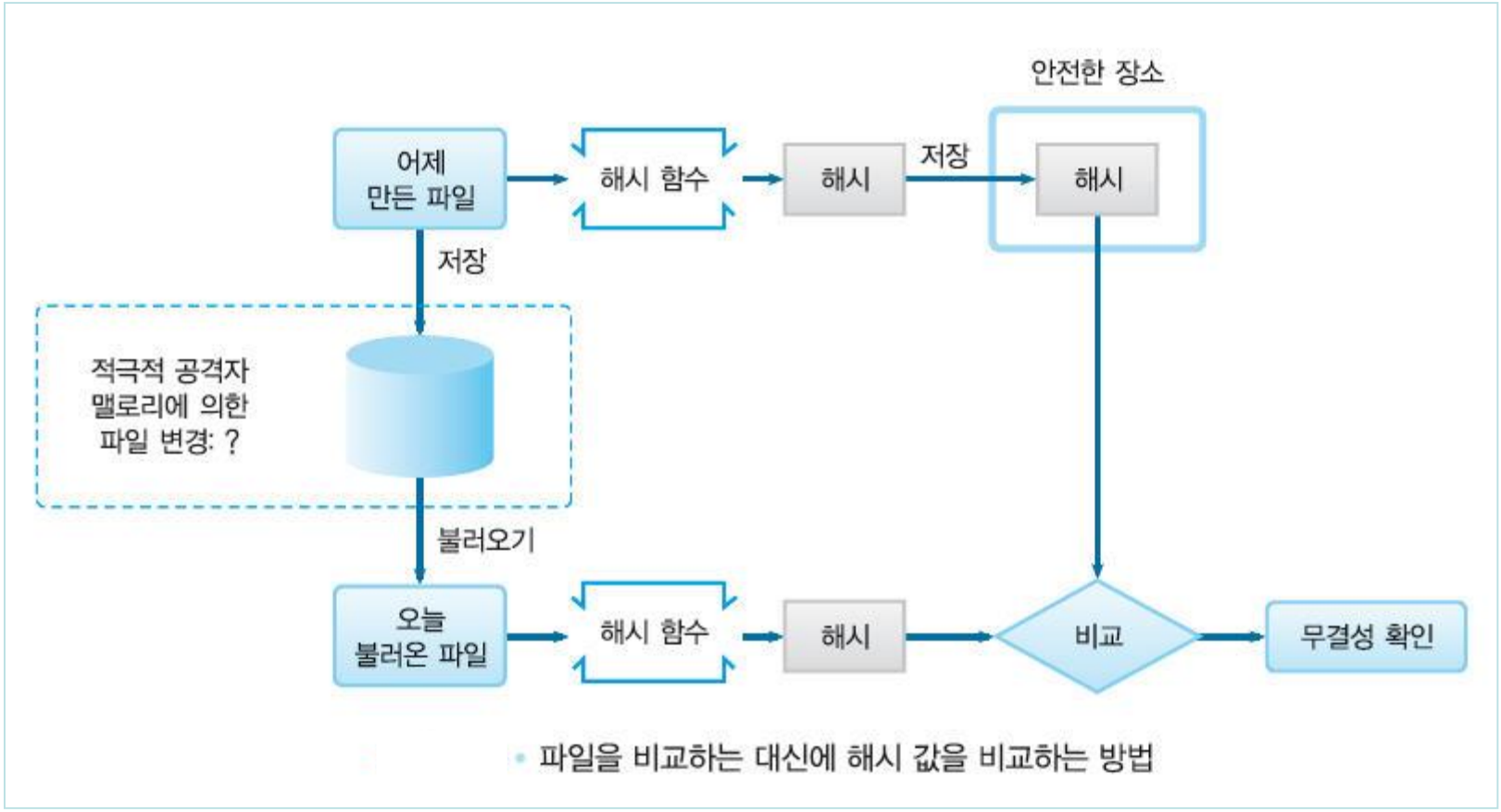


- 파일 전체를 안전한 장소에 보존해두고 나중에 비교하는 방법

파일의 지문

- 범죄 수사에서 지문을 채취하는 것과 마찬가지로 앨리스가 만든 파일의 「지문」을 채취할 수는 없을까?
- 파일 전체를 비교하는 대신에 작은 지문만을 비교하는 것만으로도 무결성을 확인할 수 있다면 매우 편리

파일을 비교하는 대신에 해시 값을 비교하는 방법



파일을 비교하는 대신에 해시 값을 비교하는 방법

1.2 일방향 해시 함수란?

- 일방향 해시 함수는 바로 파일의 지문을 채취하는 기술
- 일방향 해시 함수가 만들어내는 「해시 값」은 메시지의 지문에 해당

일방향 함수의 예

- 입력: 임의의 숫자
- 처리: 입력되는 숫자를 23으로 나누는 메커니즘
- 출력: 그 몫을 소수로 표시했을 때 소숫점 이하 7자리부터 10자리까지 4자리 숫자

실제 적용

- 입력: 345689
- 처리: 345689 를 23으로 나누어 보자
- 출력: ?

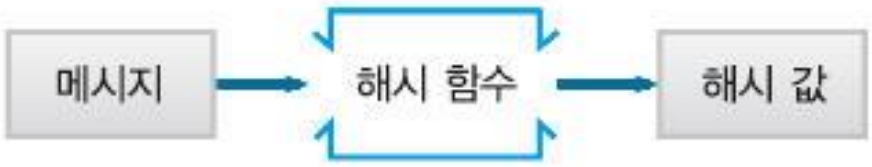
  몫: 15029.9565217391304 3…… 이므로

  - 7자리부터 10자리의 수는 ➔ 7391

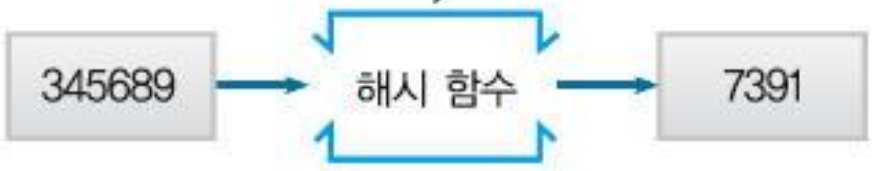
일방향 해시 함수

- 일방향 해시 함수(one-way hash function)
  - 입력과 출력이 각각 1개씩
  - 입력: **메시지**(message)
  - 출력: **해시 값**(hash value)
  - 일방향 해시 함수는 메시지를 기초로 해서 해시 값을 계산

일방향 해시 함수는 메시지를 기초로 해서 해시 값을 계산



입력 메시지를 23으로 나누고
몫의 소수점 이하 7자리부터
10자리까지의 4자리 수를 출력



- 일방향 해시 함수는 메시지를 기초로 해서 해시 값을 계산한다

일정한 크기의 출력

- 해시 값의 길이는 메시지의 길이와는 관계가 없다.
- 메시지가 1비트라도, 1메가 바이트라도, 100기가 바이트라도 일방향 해시 함수는 고정된 길이의 해시 값을 출력
- 예: SHA-1의 출력은 항상 160비트(20바이트)

해시 값은 항상 고정 길이


사용자 패스워드
8바이트
일방향 해시 함수 (SHA-1)
43 B0 4C 54 3B
67 A2 23 3F 7D
36 2B 7A 2B 49
3C D3 AF 27 4A
해시 값 20바이트
스캐너로부터의 영상 데이터
512킬로바이트
일방향 해시 함수 (SHA-1)
73 BF 4C 34 3B
67 A2 45 23 76
3F 76 D2 37 F6
44 47 8F 93 D2
해시 값 20바이트
USB의 모든 파일
4기가바이트
일방향 해시 함수 (SHA-1)
54 3B 4C 34 3B
62 3C D3 AF A2
45 67 A2 23 3F
7D 43 B0 4C 19
해시 값 20바이트
하드 디스크의 모든 파일
1테라바이트
일방향 해시 함수 (SHA-1)
32 2B 23 70 7A
2B 4F 43 B0 4C
54 3B 49 28 67
A2 23 8F 7D 36
해시 값 20바이트
• 해시 값은 항상 고정 길이

1.3 일방향 해시 함수의 성질

- 임의의 길이 메시지로부터 고정 길이의 해시 값을 계산한다
- 해시 값을 고속으로 계산할 수 있다
- 메시지가 다르면 해시 값도 다르다
- 일방향성을 갖는다

고정 길이의 출력

- 어떠한 크기의 메시지라도 크기에 관계없이 입력으로 사용할 수 있어야 한다
- 어떤 길이의 메시지를 입력으로 주더라도 일방향 해시 함수는 짧은 해시 값을 생성

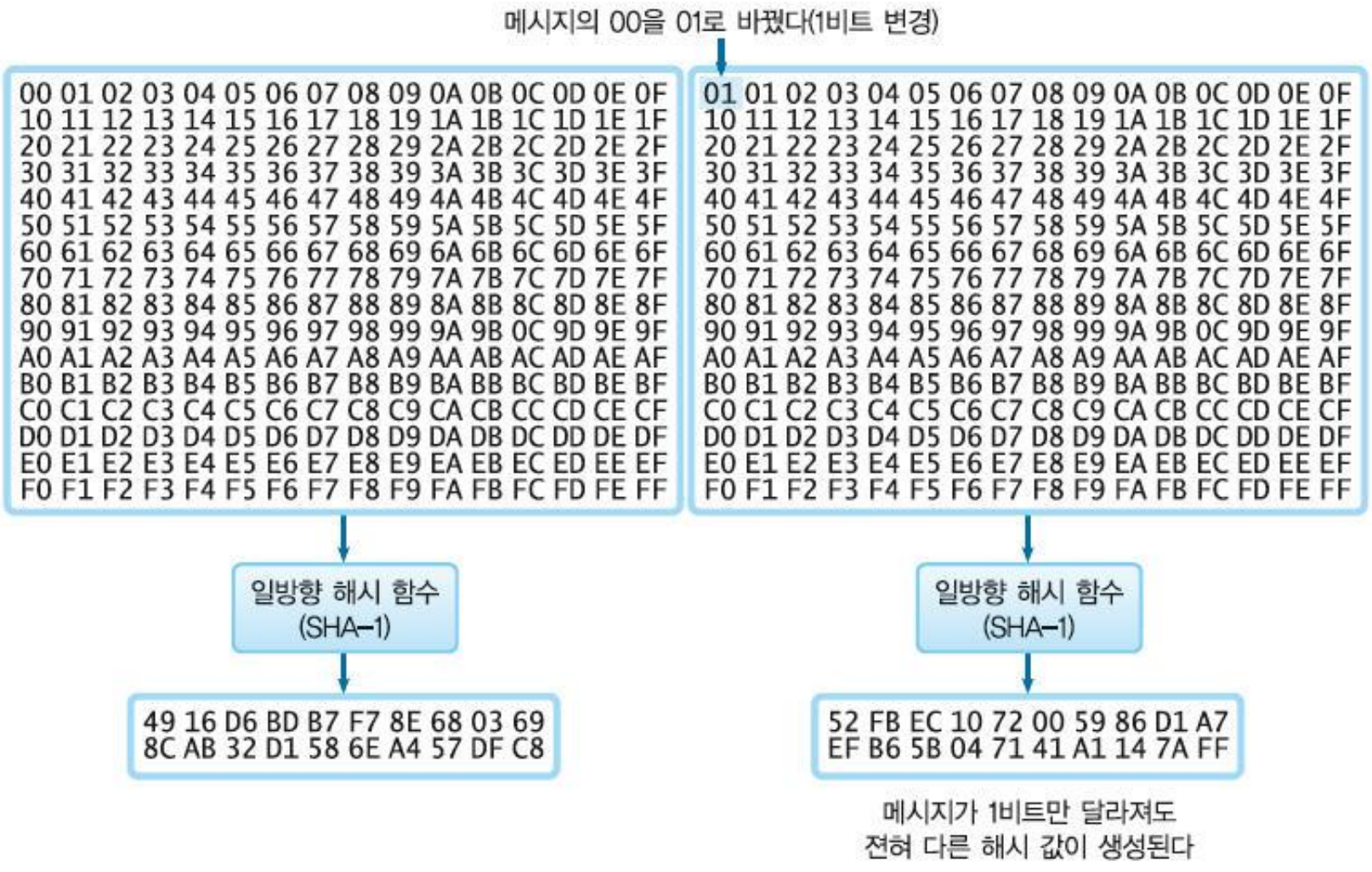
빠른 계산 속도

- 해시 값 계산은 고속이어야 한다
- 메시지가 길어지면 해시 값을 구하는 시간이 길어지는 것은 어쩔 수 없다
- 현실적인 시간 내에 계산할 수 없다면 소용이 없다

메시지가 다르면 해시 값도 다르다

- 메시지가 1비트라도 변화하면 해시 값은 매우 높은 확률로 다른 값이 돼야 한다

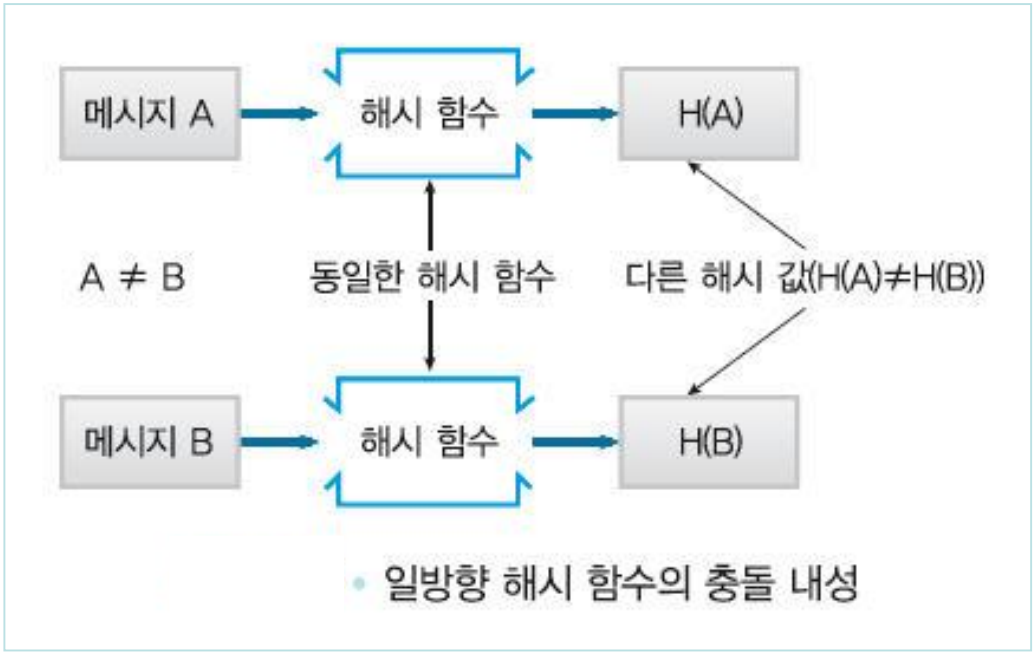
메시지가 1비트만 달라도 다른 해시 값이 된다



- 메시지가 1비트만 달라도 다른 해시 값이 된다

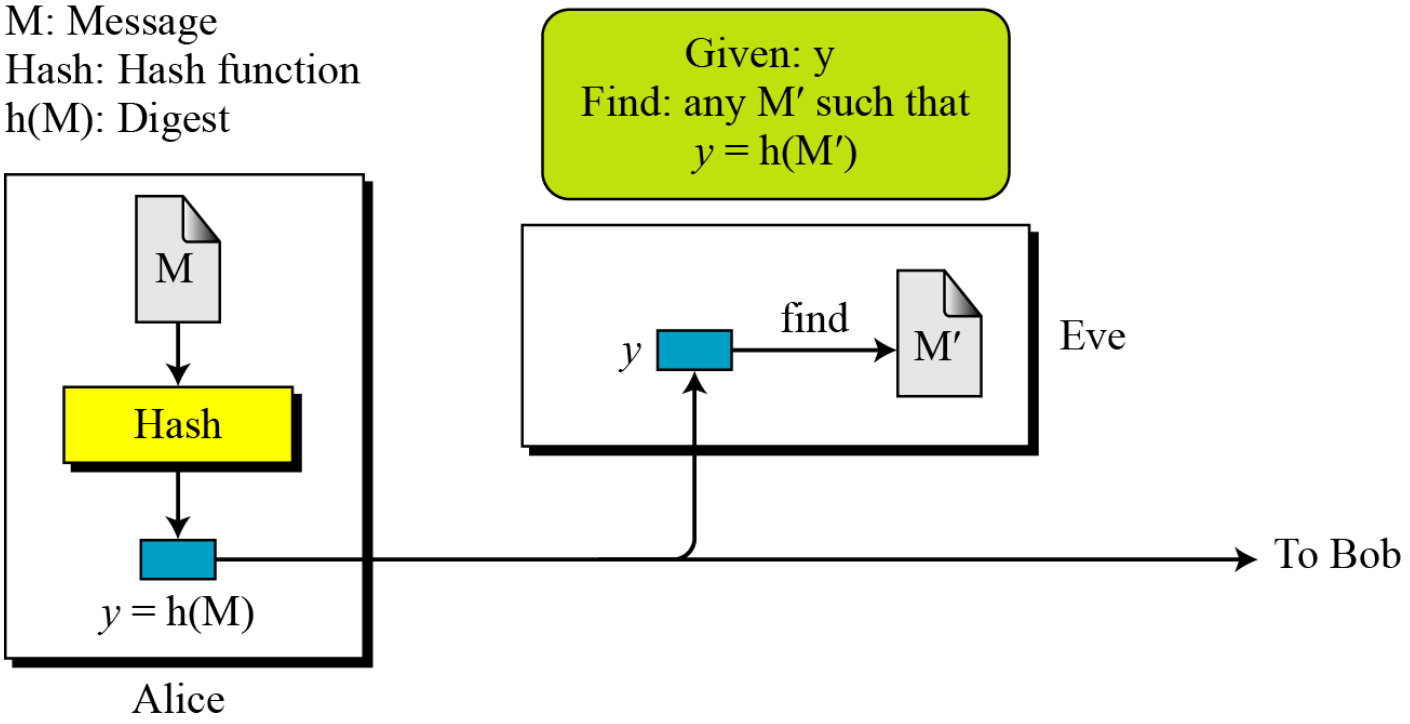
해시 함수의 충돌

- 충돌(collision)
  - 2개의 다른 메시지가 같은 해시 값을 갖는 것

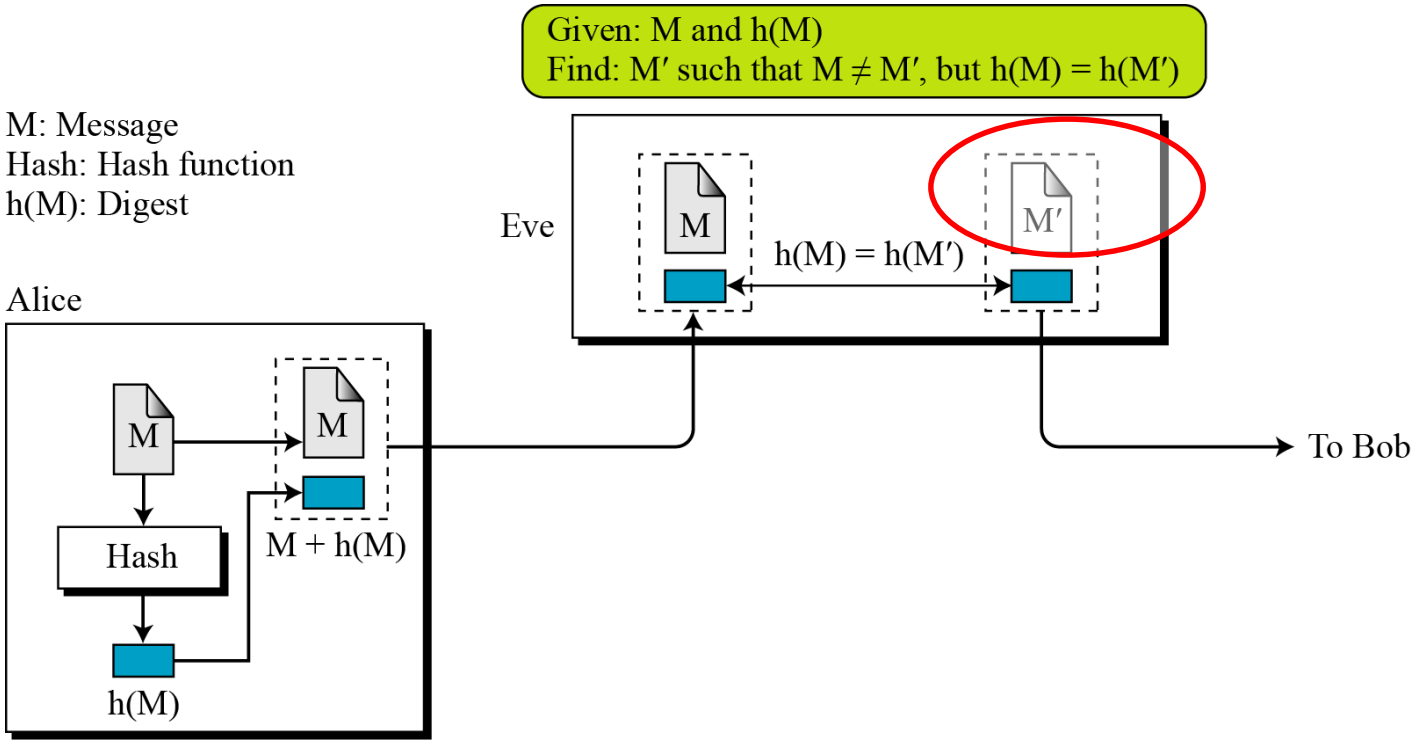


- 충돌 내성(collision resistance)
  - 충돌을 발견하는 것이 어려운 성질

- **역상저항성**
  - 프리이미지 저항성(Preimage Resistance)
  - 약 일방향성 (Weak Onewayness)
  - y=h(M)일때, 해시값(y)을 이용해 입력값 (M)을 구하기가 어려운 성질



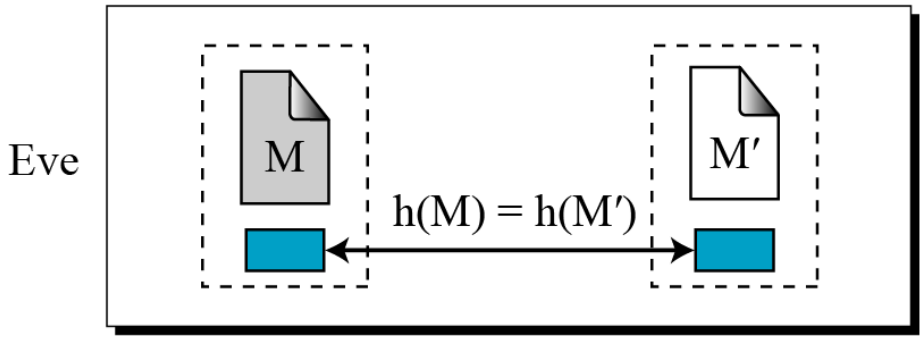
- **약한 충돌 내성**
  - 제2 프리이미지 저항성 : Second Preimage Resistance
  - 어느 메시지의 해시 값이 주어졌을 때,
    그 해시 값과 같은 해시 값을 갖는 다른 메시지를 발견해 내는 것이 매우 곤란한 성질



- **강한 충돌 내성**
  - 충돌 저항성: Collision Resistance
  - **해시 값이 일치할 것 같은, 다른 2개의 메시지를 발견해 내는 것이 매우 곤란**한 성질

M: Message
Hash: Hash function
h(M): Digest

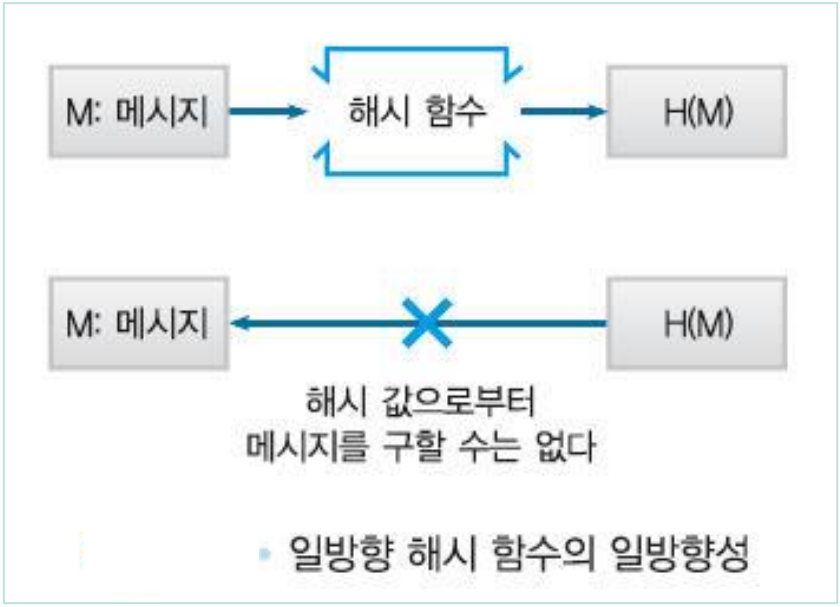
Find: M and M′ such that M ≠ M′, but h(M) = h(M′)



일방향성을 갖는다

- 해시 값으로부터 메시지를 역산할 수 없다는 성질
- 메시지로부터 해시 값을 계산하는 것은 간단히 할 수 있다
- 해시 값으로부터 메시지를 계산하는 것은 불가능해야 한다

일방향 해시 함수의 일방향성



1.4 해시 함수 관련 용어

- **일방향 해시 함수**
  - 메시지 다이제스트 함수(message digest function),
  - 메시지 요약 함수
  - 암호적 해시 함수
- **일방향 해시 함수의 입력이 되는 메시지**
  - 프리 · 이미지(pre-image)
- **해시 값은**
  - 메시지 다이제스트(message digest)
  - 핑거프린트(fingerprint)
- **무결성**
  - 완전성
  - 보전성

- 암호학적 해쉬 함수 H의 요구사항

요구사항	설명
다양한 입력 길이	H는 어떤 크기의 데이터 블록에도 적용될 수 있다.
고정된 출력 길이	H는 고정된 크기의 출력을 만든다.
효율성	$H(x)$는 실질적으로 하드웨어 및 소프트웨어에 적용하기 용이하여야 하며, 어떠한 x에 대해서도 계산이 비교적 수월해야 한다.
선이미지 회피성 (일 방향성)	어떠한 코드 h에 대해서도, $H(x) = h$인 x를 찾는 것은 계산적으로 실행 불가능하다.
2차 선이미지 회피성 (약한 충돌 회피성)	어떠한 블록 x에 대해서도, $H(y) = H(x)$인 $y \neq x$인 것을 찾는 것이 계산적으로 실행 불가능하다.
충돌 회피성 (강한 충돌 회피성)	$H(x) = H(y)$인 어떤 (x, y)짝을 찾는 것이 계산적으로 실행 불가능하다.
의사 난수성	H의 출력이 의사난수에 대한 표준 시험을 만족한다.

제2절 일방향 해시 함수의 응용 예

2.1 소프트웨어의 변경 검출

2.2 패스워드를 기초로 한 암호화

2.3 메시지 인증 코드

2.4 디지털 서명

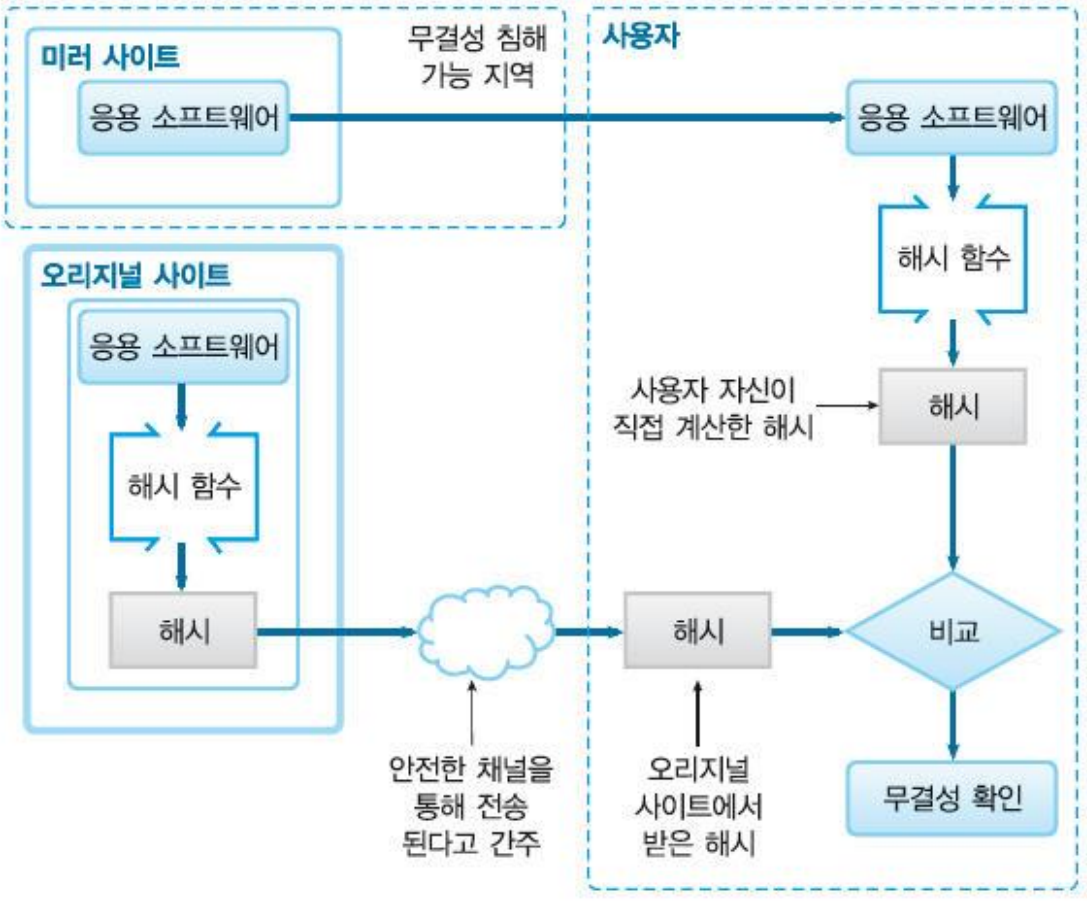
2.5 의사난수 생성기

2.6 일회용 패스워드

2.1 소프트웨어의 변경 검출

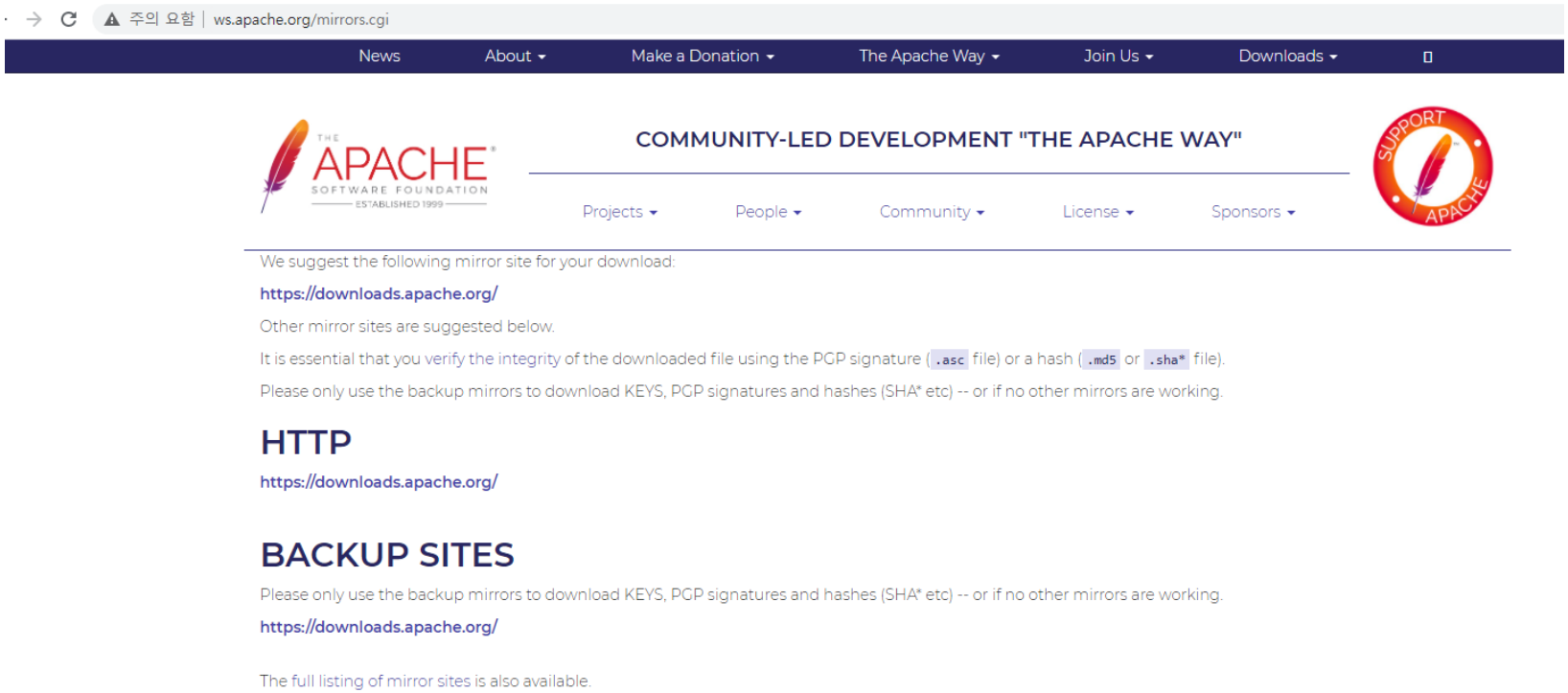
- 자신이 입수한 소프트웨어가 변경 되었는지를 확인하기 위해 일방향 해시 함수를 사용

소프트웨어 개정 검출을 위해 일방향 해시 함수를 사용



- 소프트웨어 변경 검출을 위해 일방향 해시 함수를 사용한다

- APACHE website – SW 다운로드
  - http://ws.apache.org/mirrors.cgi



2.2 패스워드를 기초로 한 암호화

- 패스워드를 기초로 한 암호화(password based encryption; PBE)에서 사용
  - PBE에서는 패스워드와 솔트를 섞은 결과의 해시 값을 구해 그것을 암호화 키로 사용
  - 패스워드 사전 공격(dictionary attack) 방어

2.3 메시지 인증 코드

- 「송신자와 수신자만이 공유하고 있는 키」와 「메시지」를 혼합해서 그 해시 값을 계산한 값
- 통신 중의 오류나 수정 그리고 위장/사칭(Masquerade)을 검출 가능
- SSL/TLS에서 이용

2.4 디지털 서명

- 현실 사회의 서명(사인)이나 날인에 해당하는 온라인 상의 서명

- 처리시간 단축을 위해 일방향 해시 함수를 사용해서
  메시지의 해시 값을 구하고, 그 해시 값에 디지털 서명을 수행

2.5 의사난수 생성기

- **암호 기술에 필요한 난수**
  - 「과거의 난수열로부터 미래의 난수열을 예측하는 것은 사실상 불가능」이라는 성질이 필요
  - 그 예측 불가능성을 보증하기 위해 일방향 해시 함수의 일방향성을 이용

2.6 일회용 패스워드

- 원타임 패스워드(one-time password)
  - 정당한 클라이언트인지 아닌지를 서버가 인증할 때에 사용
  - 일방향 해시 함수를 써서 통신 경로 상에 흐르는 패스워드를 1회(one-time)만 사용하도록 고안
  - 패스워드가 도청되어도 악용될 위험성이 없다

제 3절 일방향 해시 함수의 예

3.1 MD4와 MD5

3.2 SHA-1, SHA-256, SHA-384, SHA-512

3.3 RIPEMD-160

3.4 SHA(Advanced Hash Standard)와 SHA-3

3.1 MD4와 MD5

- MD4
  - Rivest가 1990년에 만든 일방향 해시 함수
  - 128비트의 해시 값
  - Dobbertin에 의해 충돌 발견 방법이 고안
  - 현재는 안전하지 않다

3.1 MD4와 MD5

- MD5
  - Rivest가 1991년에 만든 일방향 해시 함수
  - 128비트의 해시 값
  - MD5가 완전히 뚫린 것은 아니지만, MD5 내부 구조의 일부에 대한 공격 방법이 몇가지 발견
  - 사용을 권장하지 않는다

3.2 SHA-1, SHA-256, SHA-384, SHA-512

- SHA-1
  - NIST(National Institute of Standards and Technology)에서 제작
  - 160비트의 해시 값
  - SHA
    - 1993년에 미국의 연방정보처리표준
  - SHA-1
    - 1995년에 발표된 개정판
    - 메시지 길이 상한: 2^{64}비트 미만
    - 큰 값이므로 현실적인 적용에는 문제가 없음

SHA-2

- SHA-256:
  - 256 비트의 해시 값
  - 메시지의 길이 상한 2^{64}비트 미만
- SHA-384:
  - 384 비트의 해시 값
  - 메시지의 길이 상한 2^{128}비트 미만
- SHA-512:
  - 512 비트의 해시 값
  - 메시지의 길이 상한 2^{128}비트 미만

3.3 RIPEMD-160

- RIPEMD-160
  - 1996년에 Hans Dobbertin, Antoon Bosselaers, Bart Preneel이 제작
  - 160비트의 해시 값
  - European Union RIPE 프로젝트로 만들어진 RIPEMD 함수의 개정판

3.4 SHA-3

- SHA-1의 강한 충돌 내성 침해
- NIST는 SHA-1을 대체하는 차세대 일방향 해시함수로 2007년에 "SHA-3" 선정 시작
- AES와 같은 경쟁 방식으로 표준화
- 2012년 선정 완료
  - KECCAK(케챡)을 표준으로 선정
  - 이것이 SHA-3

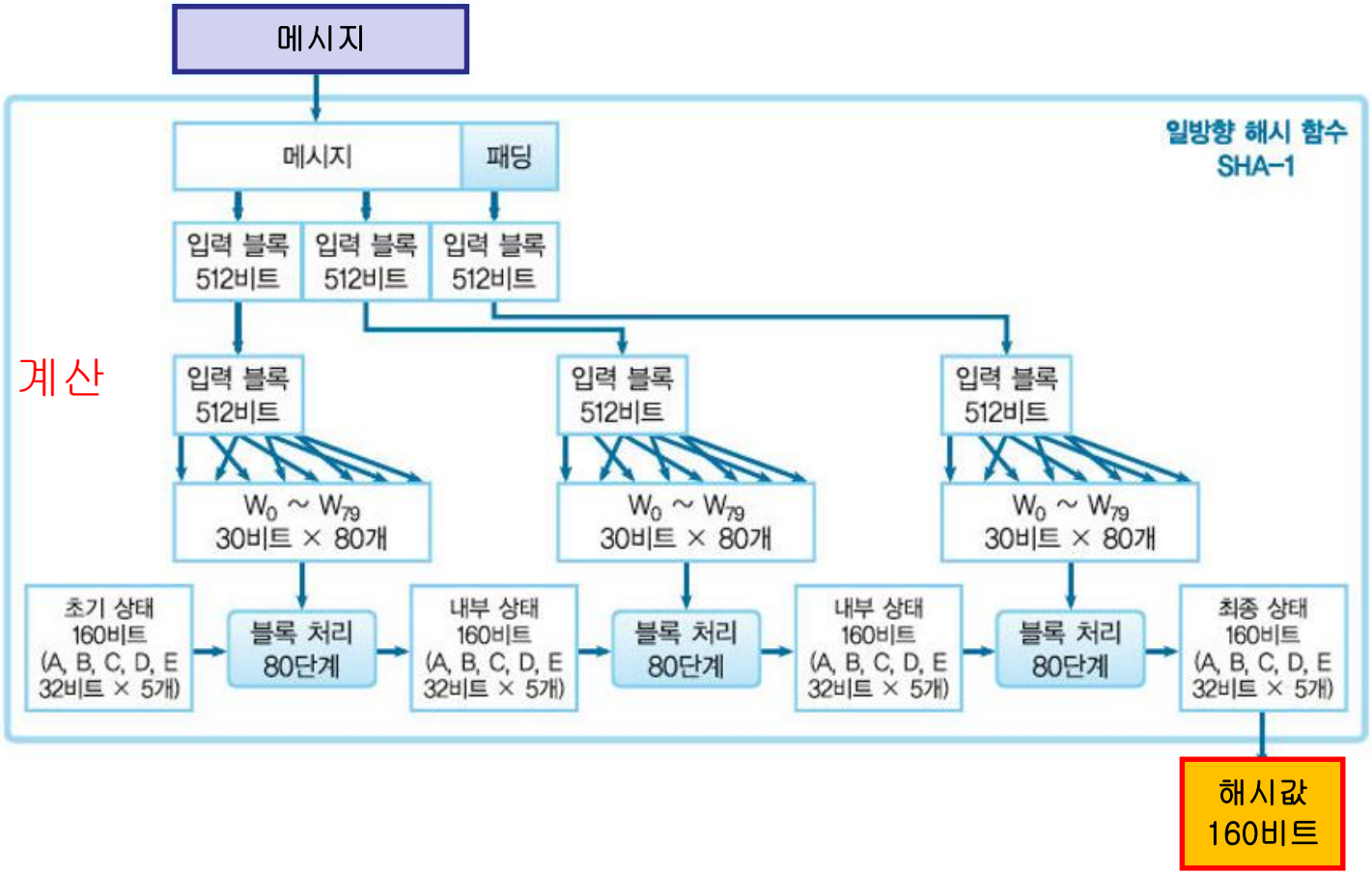
제 4절 일방향 해시 함수 SHA-512

- 일방향 해시 함수 SHA-1의 개요

1.패딩

2.$W_0 \sim W_{79}$의 계산

3.블록처리

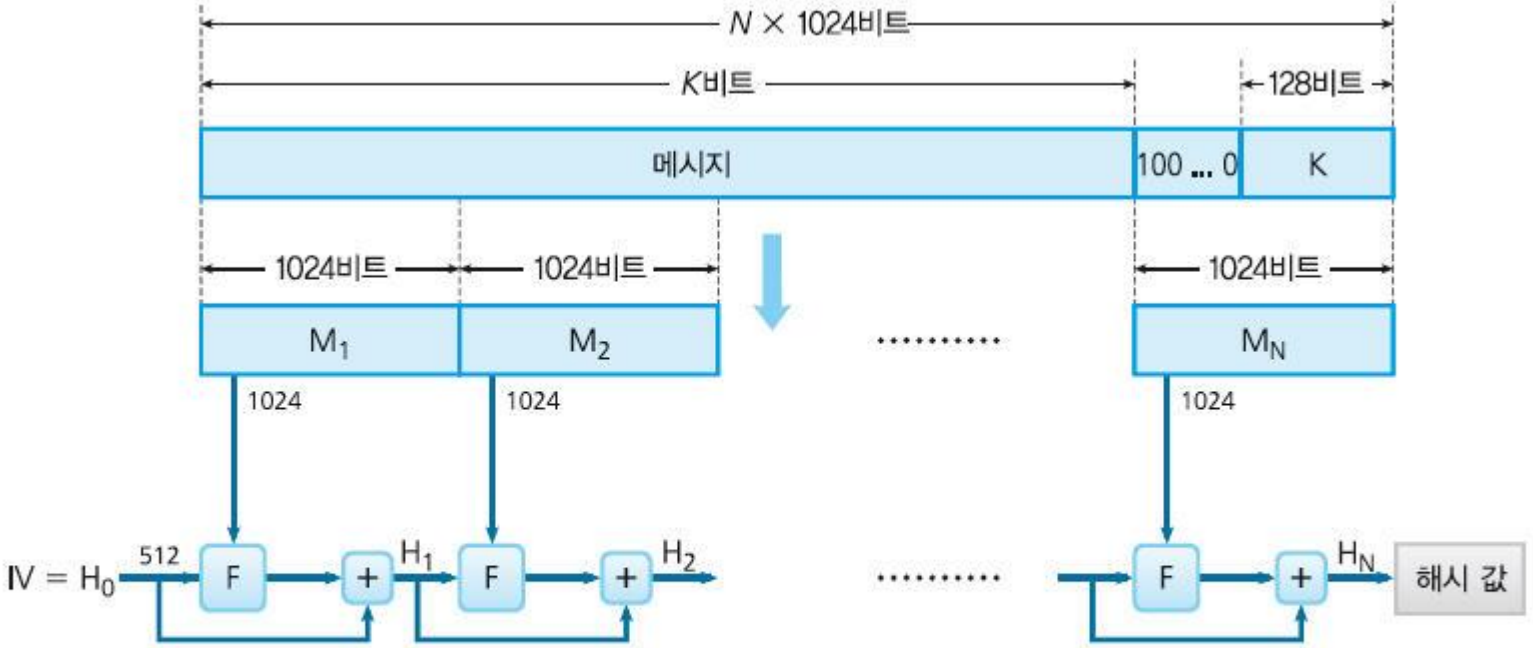


제 4절 일방향 해시 함수 SHA-512

- SHA-512구조
  - 입력: 최대 2^{128}비트 이하 메시지
  - 출력: 512비트 메시지 다이제스트

- 입력 데이터는 길이 1024비트 블록으로 처리

SHA-512 해시값의 생성



+는 mod 2^{64}로 실행한 합을 의미함

- SHA-512를 사용하는 메시지 다이제스트 생성

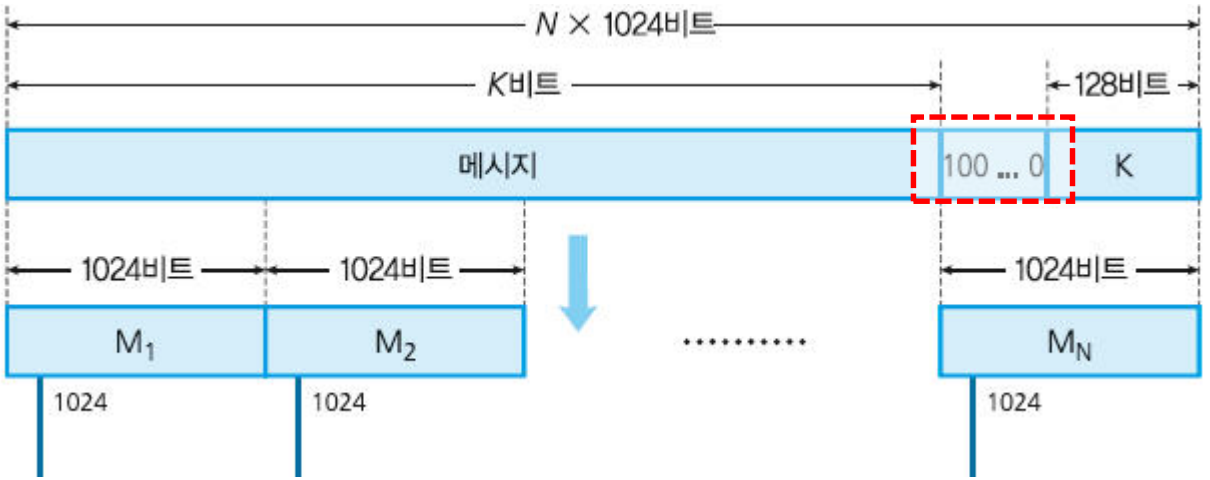
SHA-512 처리 단계

- 패딩 비트 붙이기
- 길이 붙이기
- MD 버퍼 초기화
- 1024-비트(128-워드)블록 메시지 처리
- 출력

SHA-512 처리 단계

1. 패딩 비트 붙이기

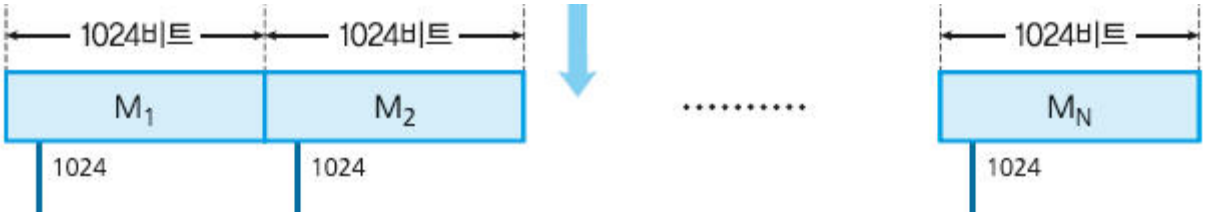
- 메시지 뒤에 여분의 데이터를 부가하여 메시지의 길이가 896 (mod 1024)가 되도록 만듬
- 메시지가 1024비트의 배수여도 패딩 추가
- 패딩의 첫 번째 비트는 1
- 나머지 비트는 모두 0



SHA-512 처리 단계

2. 길이 붙이기

- 128 비트 블록 K를 메시지에 추가
- 1024 비트 블록들을 M_1, M_2, …, M_N으로 표현
- 총 길이=Nx1024 비트



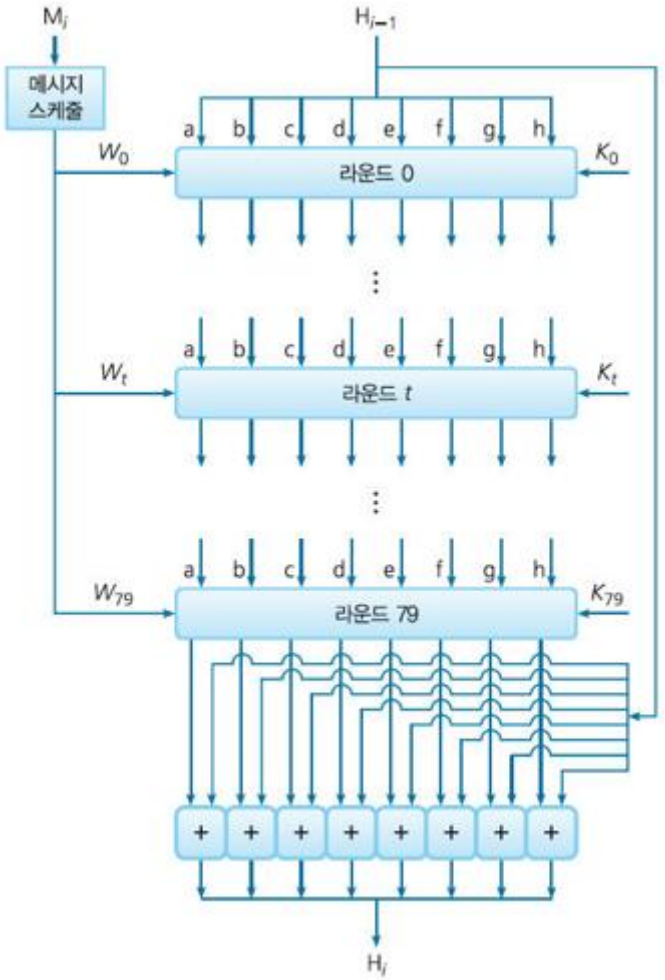
3. MD 버퍼 초기화

- 해시 함수의 중간 값과 최종 값을 저장하기 위해 512비트 버퍼를 사용
- 이 버퍼를 8개의 64비트 레지스터(a, b, c, d, e, f, g, h)로 표현

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

4. 1024-비트(128-워드)블록 메시지 처리

- 각 라운드가 80 라운드로 구성
- 각 라운드에서 512 비트버퍼 abcdefgh를 입력으로 사용



* 1024비트 한 개에 대한 SHA-512 처리

5. 출력

- N개의 1024-비트 블록을 모두 처리하면 N번째 단계에서 512-비트 메시지 다이제스트가 출력

SHA-512의 안전성

- 2017년까지 약점 발견된 것 없음

- **확률적 안전성**
  - **약 충돌성**: 동일한 메시지 다이제스트를 갖는 두 개의 서로 다른 메시지를 찾는 난이도 연산 수행 수는 2^{256}
  - **강 충돌성**: 주어진 다이제스트와 동일한 다이제스트를 갖는 메시지를 찾는 난이도 연산 수행 수는 2^{512}

제 5절 SHA-3 선정 과정

5.1 SHA-3란 무엇인가?

5.2 SHA-3 선정 과정

5.3 SHA-3 최종 후보와 SHA-3 결정

5.1 SHA-3란 무엇인가?

- SHA-3(Secure hash Algorithm)
  - 이론적 공격방법이 알려져 버린 SHA-1을 대신하는 새로운 표준의 일방향 해시 알고리즘
  - 2012년에 KECCAK이라는 알고리즘을 SHA-3으로 선정

5.2 SHA-3 선정 과정

- 미국 표준화 기관 NIST에서 공모
- 실질적으로는 세계적인 표준
- 경쟁방식에 의한 표준화(Standardization by competition) 방식으로 선정

5.3 SHA-3 최종 후보와 SHA-3 결정

- NIST에서 2007년에 SHA-3 공모
- 2008년까지 응모한 알고리즘 수는 64개
- 2010년에 SHA-3 최종후보로서 5개의 알고리즘이 선정
  - SHA-3 최종 후보(알파벳 순)

명칭	응모자
BLAKE	Jean-Philippe Aumasson, Luca Henzen, Willi Meier, Raphael C.-W. Pyhan
Grostl	Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schlaffer, Soren S. Thomsoen
JH	Hongjun Wu
Keccak	Guido Bertoni, Joan Daemen, Gilles Van Assche, Michael Peeters
Skein	Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, Jesse Walker

- 현재 SHA-2와 SHA-3 공존해서 사용

- KECCAK이 SHA-3로 선정된 이유
  - SHA-2와 전혀 다른 구조임
  - 구성이 투명하여 구조를 해석하기 쉬움
  - 다양한 디바이스에서 구동되고, 조합 형태로도 양호
  - 하드웨어 상에 내장시 높은 고성능
  - 다른 최종 목록과 비교하여 보안성이 높음

제 6절 KECCAK

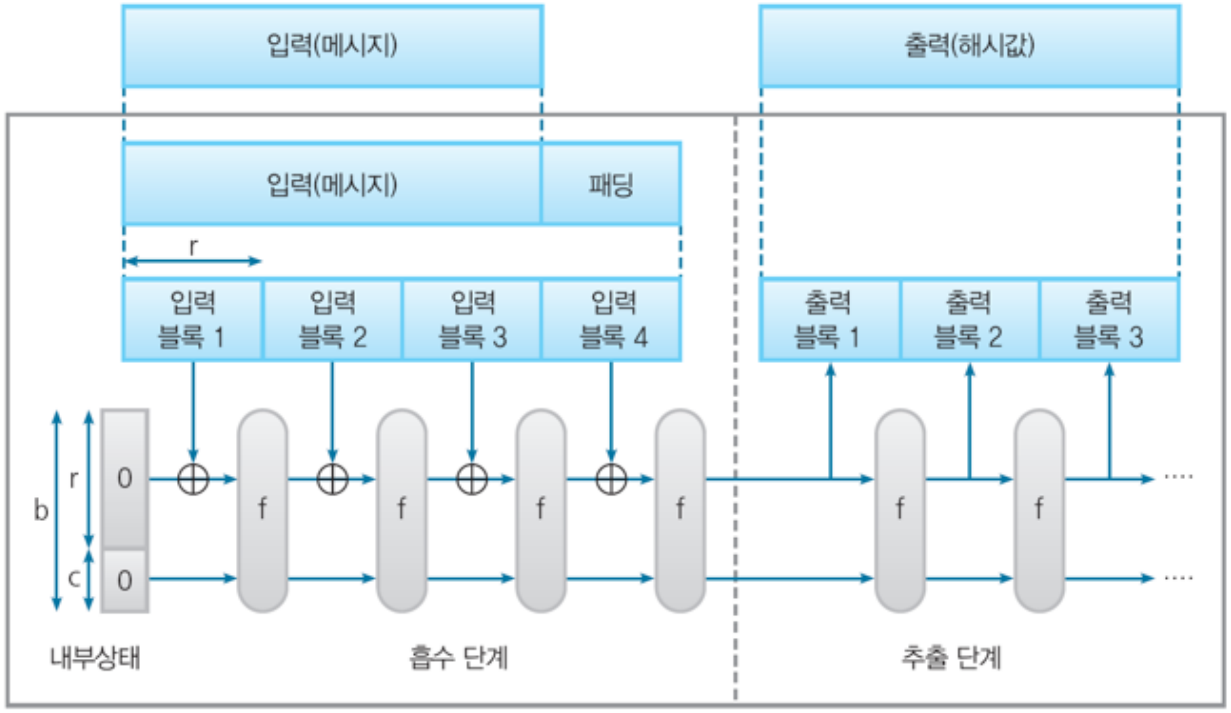
6.1 KECCAK 이란 무엇인가?
6.2 스펀지 구조
6.3 듀플렉스 구조
6.4 KECCAK의 내부 상태
6.5 함수 KECCAK-f[b]
6.6 θ 단계
6.7 ρ 단계
6.8 π 단계
6.9 χ 단계
6.10 ι 단계
6.11 KECCAK에 대한 공격
6.12 약한 KECCAK에 대한 공격 테스트

6.1 KECCAK란 무엇인가?

- SHA-3으로 선정된 일방향 해시 함수 알고리즘
- 해시 값으로서 임의 길이의 비트열 생성
- SHA-2 비트 길이에 맞춰져 있음
- SHA3-224, SHA3-256, SHA3-384, SHA3-512 4종류가 SHA-3으로 규정
- 입력 데이터 크기에 제한 없음
- SHAKE128과 SHAKE256이라는 임의 길이의 출력을 가진 함수 (extendable-output function; XOF)가 규정
  - SHAKE: Secure Hash Algorithm + KECCAK

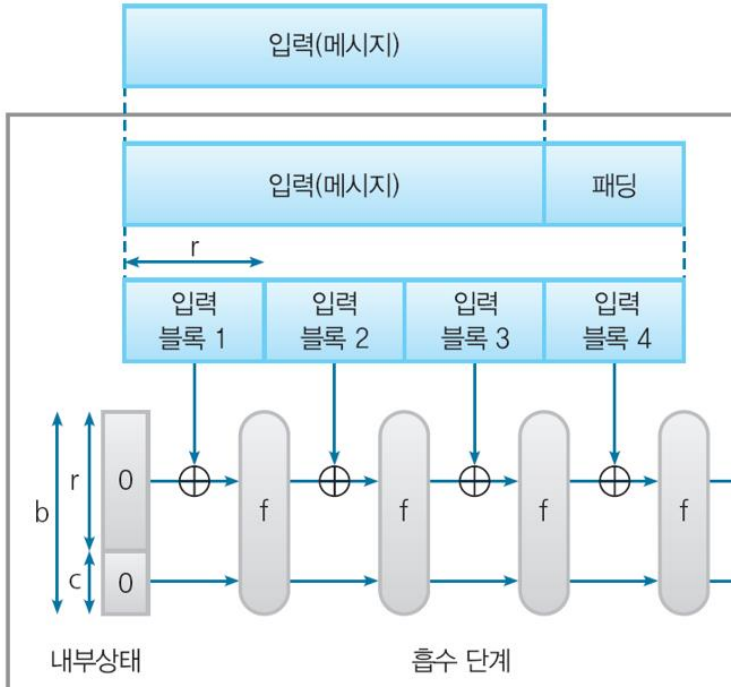
6.2 스펀지 구조

- SHA-1과 SHA-2와는 전혀 다른 스펀지 구조가 사용
- 입력되는 메시지에 패딩을 시행한 후
  - 두 개의 상태를 통해서 해시 값을 출력
    . 흡수 단계(absorbing phase)
    . 추출 단계(squeezing phase)



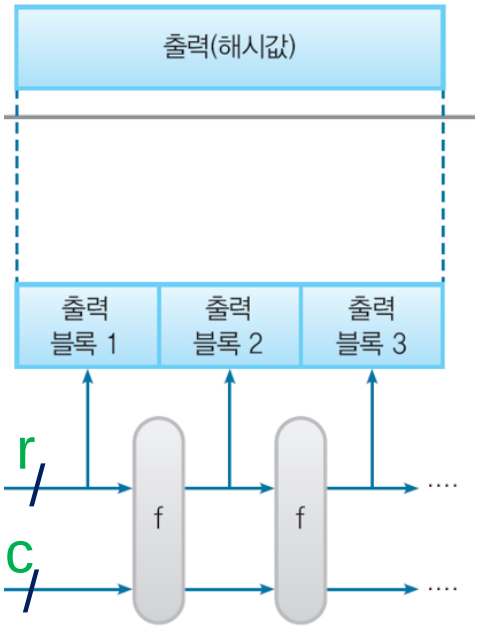
흡수 단계

- 패딩된 입력 메시지를 r 비트 단위의 입력 블록으로 분할
- 우선 「내부상태의 r 비트」와 「입력 블록 1」의 XOR을 구하고, 그것을 「함수 f에 입력」
- 다음에 「함수 f의 출력 r 비트」와 입력 블록 2」을 XOR 하고, 그것을 다시 한 번 「함수 f에 입력」
- 이 과정을 입력 블록이 다 할 때까지 계속 진행
- 입력 블록을 다 처리하면 흡수 단계를 종료하고 추출 단계를 진행



추출 단계

- 우선 「함수 f의 출력 중 r 비트」를 「출력 블록 1」로 기술하고, 출력전체(r + c 비트)는 함수 f로 다시 입력
- 다음에 「함수 f 출력 중 r 비트」를 「출력블록2」로 기술하고, 출력전체(r + c 비트)는 함수 f로 다시 입력
- 이것을 필요한 비트 수의 출력을 얻을 때까지 계속 반복

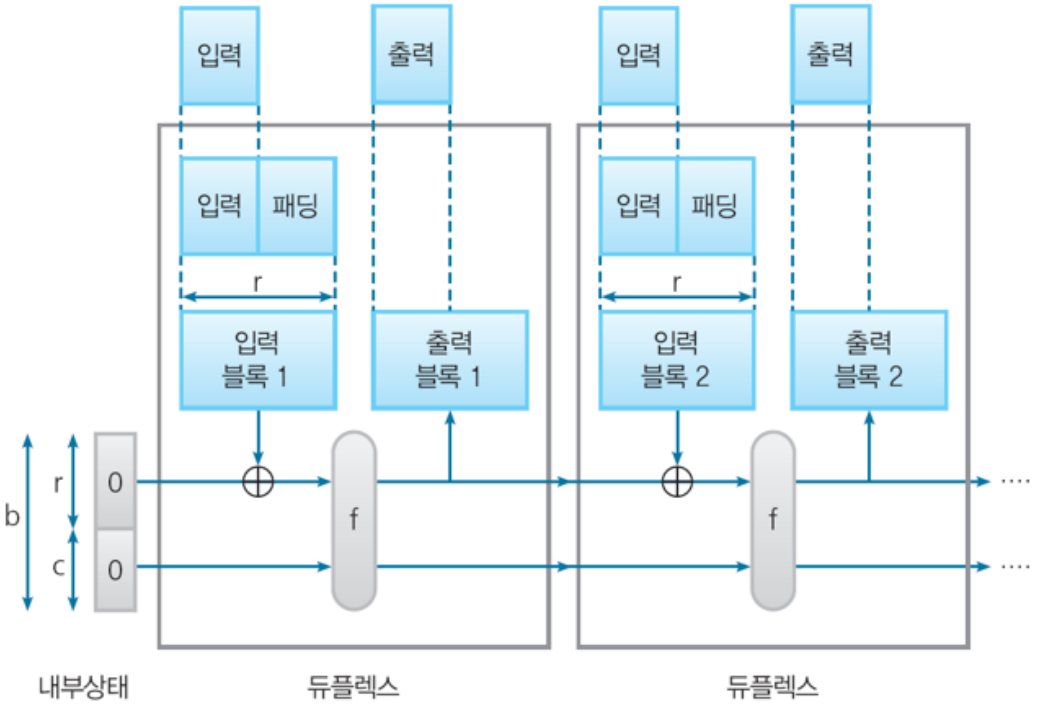


추출 단계

6.3 듀플렉스 구조

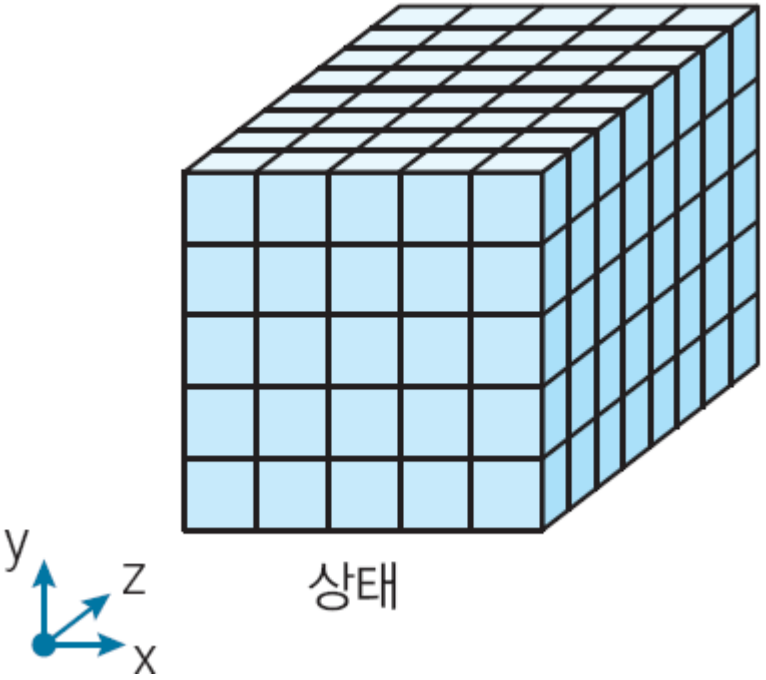
- 스펀지구조의 변형으로 듀플렉스(duplex) 구조도 제안
  - 스펀지구조: 입력을 모두 흡수하는 과정이 끝난 뒤 출력 시작
  - 듀플렉스구조: 입력과 출력이 같은 속도로 진행

- 해시 값을 구하는 용도 + 암호학자의 도구상자에 들어있는 대부분의 기능을 활용 가능
  - 의사난수 생성기
  - 스트림 암호
  - 인증이 포함된 암호
  - 메시지 인증 코드 등



6.4 KECCAK의 내부 상태

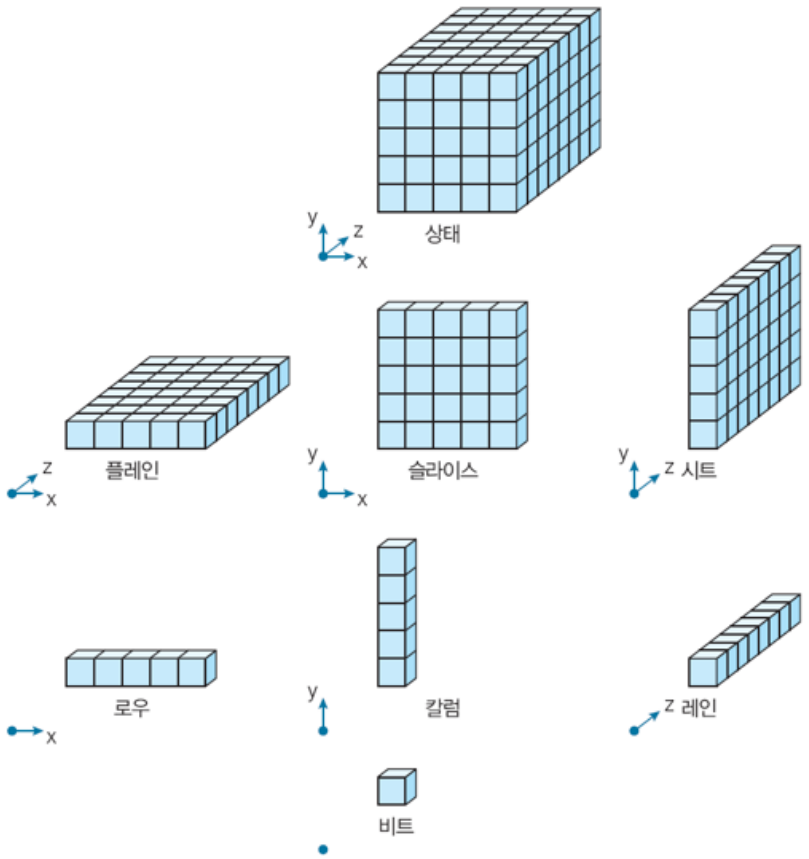
- 3차원 비트 배열
- 1비트를 나타내는 작은 입방체가 b개,
  - 5 x 5 x z 방향수만큼 큰 육면체
- 상태(state)
  - x, y, z의 모든 방향을 고려한 3차원 내부상태 전체



* 내부 상태(b=5x5xz 비트)

KECCAK의 내부 상태의 일부

- **내부 상태 중 2차원을 고려할 때,**
  - Xz평면: 플레인
  - yx평면: 슬라이스
  - yz평면: 시트
- **1차원만을 고려할 때,**
  - x방향: 로우
  - y방향: 칼럼
  - z방향: 레인
- **예)**
  - 레인 5 x 5 = 25개를 모아 상태를 만들어 가고있다
  - 슬라이스가 레인길이만큼 모여 상태를 만들어 가고있다



- **내부상태 b비트를 효과적으로 혼합한 것이 함수f를 실현한 것이 KECCAK 임**

6.5 함수 KECCAK-f[b]

- 내부상태를 혼합하는 함수 f
- b: 폭
  - 내부상태의 비트 길이를 나타내는 척도
  - B의 7종류: 25, 50, 100, 200, 400, 800, 1600

    $(25, 25x2^1, 25x2^2, \cdots.)$

    .SHA-3에서는 b=1600 사용

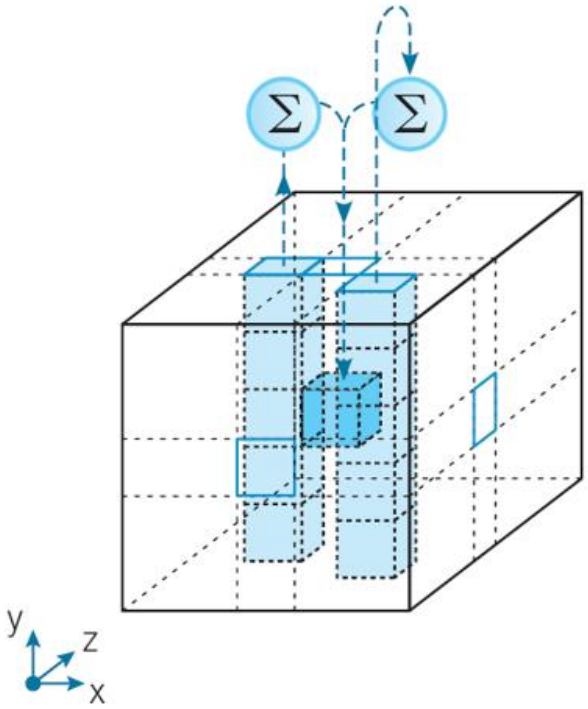
    .SHA-3의 내부상태 b=5 x 5 x 64=1600 비트

- KECCAK-f[b]는 아래에 설명하게 될 θ, ρ, π, χ, ι라는 5개 단계를 1 라운드라고 할 때
  - 총 12 + 2ι 라운드를 반복하는 함수

- SHA-3으로 사용되는 KECCAK-f[1600]의 경우
  - 라운드 수는 24

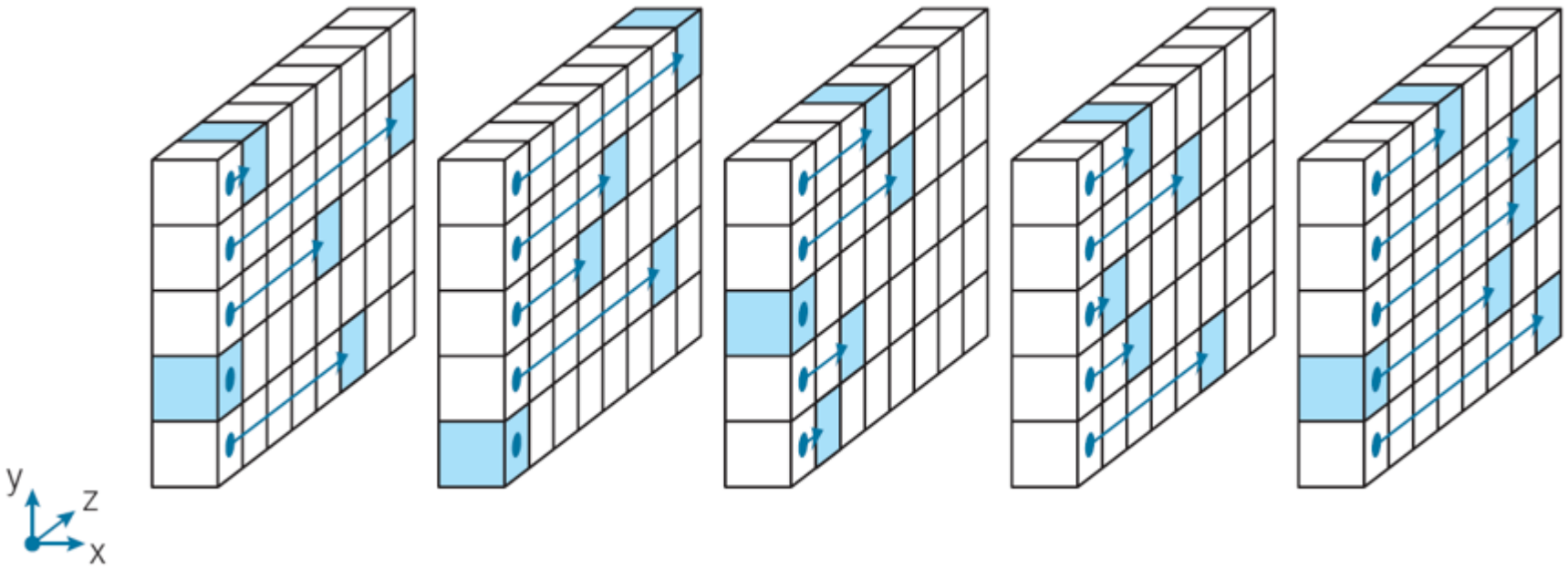
θ 단계

- 위치를 이동시킨 칼럼 2개의 5 비트를 XOR하여(Σ표시)
- 그 결과를 목적 비트와 XOR 처리 후 목적 비트를 대체



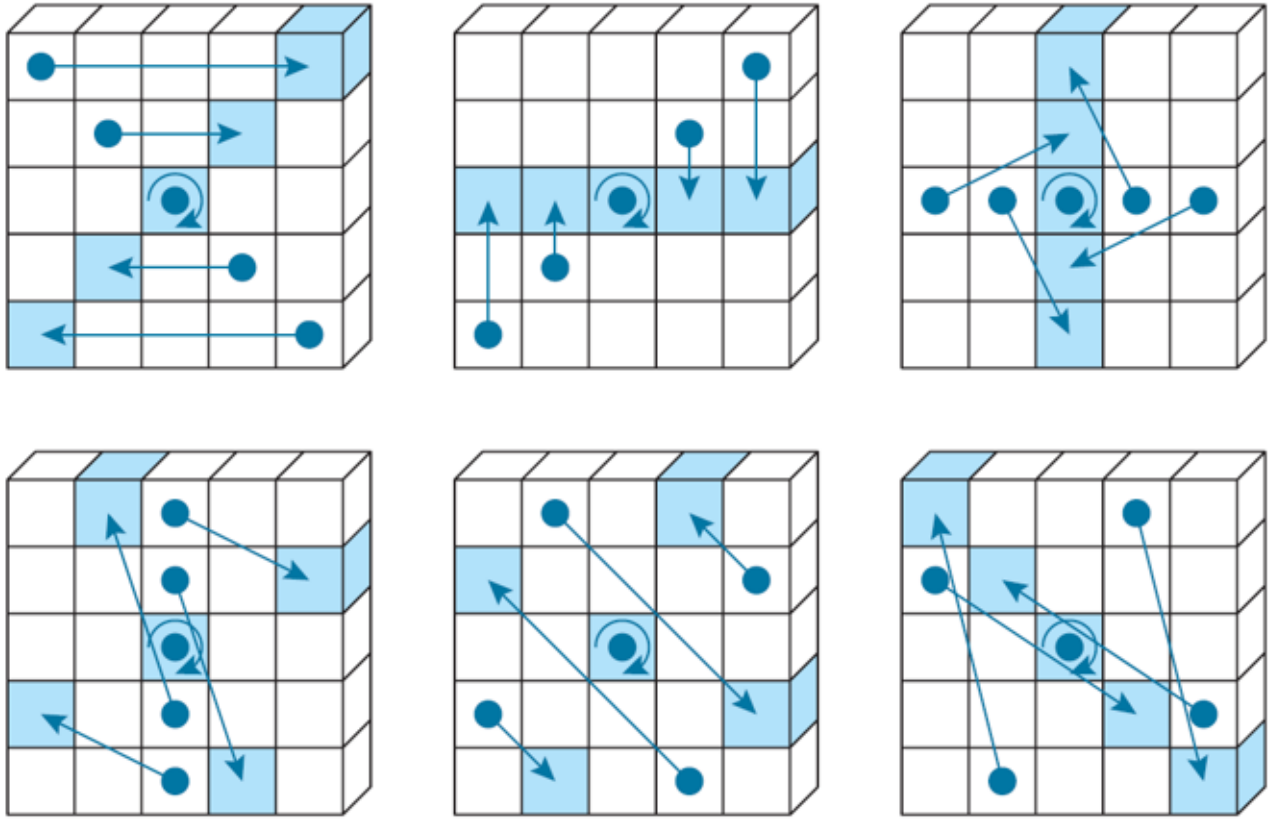
ρ 단계

- z 방향(레인 방향) 비트 이동



π 단계

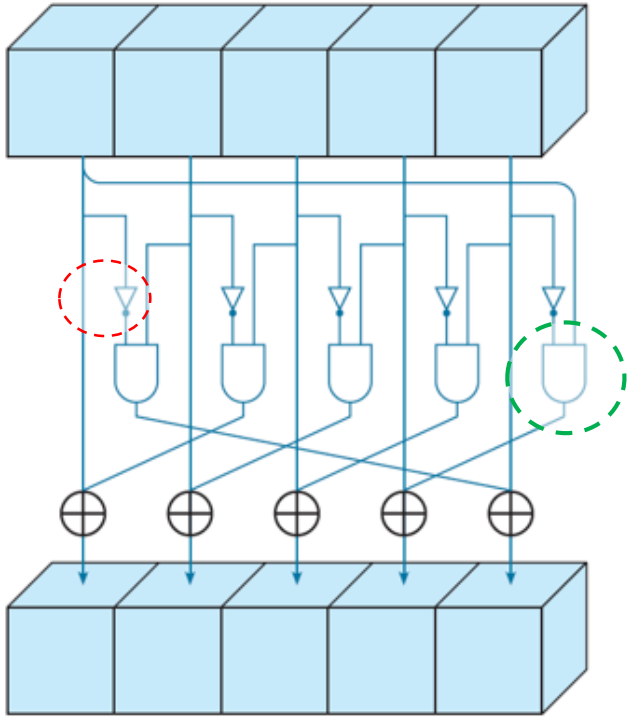
- 특정 1개의 슬라이스에 π단계를 적용하고 있는 모습
- 비트 이동을 레인 전체에 대해 수행



χ 단계

- 특정 1개의 로우에 χ 단계를 적용하고 있는 모습
- 논리회로 NOT연산과 AND연산을 활용하여 수행

비트반전　　2개의 입력 모두1일 때만 1 출력



ι 단계

- 라운드 정수라고 불리는 정수와 단계 전체의 비트를 XOR 처리
- 내부상태의 비대칭성을 위해 수행

6.6 KECCAK에 대한 공격

- MD(Merkle-Damgard)변환
  - MD4, MD5, RIPEMD, RIPEMD-160, SHA-1, SHA-2 등 지금까지 거의 일방향 해시 함수 알고리즘에서 압축 함수를 반복해서 사용
- SHA-1에 대한 논리적인 공격방법이 발견
- SHA-2는 SHA-1과 같은 MD구조를 사용했기 때문에 근본적으로 문제를 해결할 필요가 있음

- KECCAK은
  - MD구조와 다른 스펀지 구조로 만들어져 있기 때문에 SHA-1을 공격하는 데 사용된 방법이 KECCAK에는 통용되지 않음
  - 현재까지 공격 방법이 알려진 것이 없음

6.7 약한 KECCAK에 대한 공격 테스트

- KECCAK 은 반복 되는 구조로 되어 있고 여러 라운드로 구성됨
- 약한 KECCAK 을 구성하는 것이 가능
- KECCAK 설계자에 의한 컨테스트
  (KECCAK Crunchy Crypto Collision and Pre-image Contest)
  - 구조를 명확히 해 해석이 쉽도록 간략형 KECCAK 을 만들고 이를 공격해보는 콘테스트
  - SHA-3의 안전도 측정 방법

제 7절 일방향 해시 함수에 대한 공격

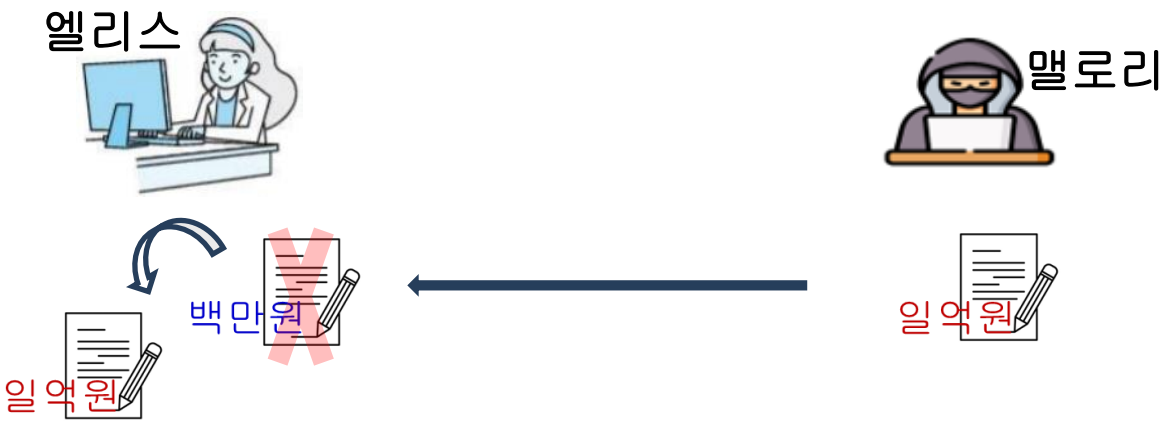
7.1 전사 공격(공격 스토리1)
7.2 생일 공격(공격 스토리2)

7.1 전사 공격(공격 스토리1)

- 일방향 해시 함수의 「약한 충돌내성」을 깨고자 하는 공격

> ❖ 약한 충돌 내성
> 어느 메시지의 해시 값이 주어졌을 때, 그 해시 값과 같은 해시 값을 갖는 다른 메시지를 발견해 내는 것이 매우 곤란한 성질

- 예: 맬로리는 앨리스의 컴퓨터에서 계약서 파일을 발견하고, 그 안의 「앨리스의 지불 금액은 백만원으로 한다.」 부분을, 「앨리스의 지불 금액은 일억원으로 한다.」로 바꾸고 싶다



공격 스토리 1

- 「문서의 의미를 바꾸지 않고 얼마만큼 파일을 수정할 수 있을까」를 생각
- 동일한 내용을 다른 형태로 표현
  - 앨리스의 지불 금액은 일억원(一億원)으로 한다.
  - 앨리스의 지불 금액은 일억원(壹億원)으로 한다.
  - 앨리스의 지불 금액은 100000000원으로 한다.
  - 앨리스의 지불 금액은 ₩100000000으로 한다.
  - 앨리스의 지불 금액은, 일억원(一億원)으로 한다.
  - 앨리스가 지불하는 금액은 일억 원으로 한다.
  - 앨리스는 **일억원을 지불하는** 것으로 한다.
  - 대금으로서, 앨리스는 일억 원을 지불한다.

공격 스토리 1

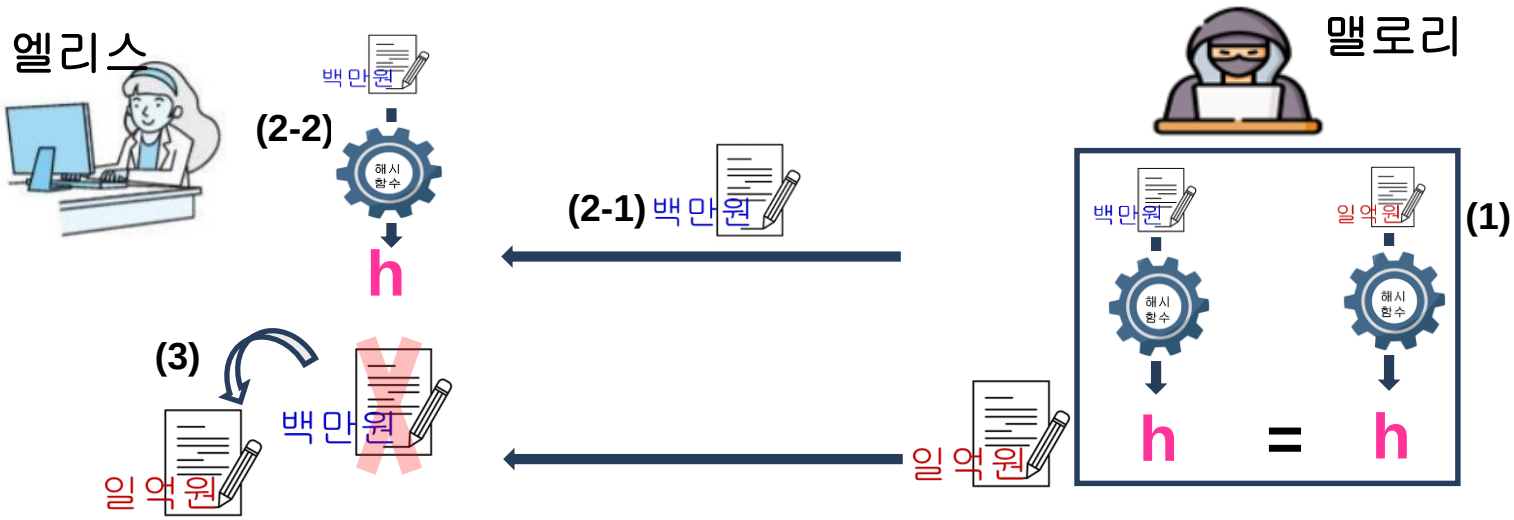
- 「일억원 지불의 계약서」를 기계적으로 대량 작성
  - 그 중에 앨리스가 만든 오리지널 「백만 원 계약서」와 같은 해시 값을 생성하는 것을 발견할 때까지 작성한다
  - 발견한 새로운 계약서로 교환

7.2 생일 공격 (공격 스토리2)

- 「강한 충돌 내성」을 깨고자 하는 공격

> ❖ 강한 충돌 내성
> 어느 해시 값이 일치할 것 같은, 다른 2개의 메시지를 발견해 내는 것이 매우 곤란한 성질

- 예)
  1. 맬로리는 해시 값이 같은 값을 갖는 「백만원 계약서」와 「일억원 계약서」를 미리 만들어 둔다
  2. 맬로리는 시치미를 떼고 「백만 원 계약서」를 앨리스에게 건네주고, 해시 값을 계산시킨다
  3. 맬로리는 스토리1과 마찬가지로 「백만 원 계약서」와 「일억 원 계약서」를 살짝 바꾼다.



생일 공격(birthday attack)

- N명 중 적어도 2명의 생일이 일치할 확률이 「2분의 1」 이상이 되도록 하기 위해서는 N은 최저 몇 명이면 될까?
- 답:
  - 놀랍게도 N = 23이다
  - 단 23명만 있으면 「2분의 1」 이상의 확률로 적어도 2명의 생일이 일치 한다

이유

- 「어느 특정일을 정하고 2명이 그 날 태어날」 가능성은 확실히 높지 않다.
- 그러나 「1년의 어느 날이라도 상관없으므로 2명이 같은 날 태어날」 가능성은 의외로 높다.
  ➔ 생일 패러독스 (Birthday Paradox)

참고: 생일 문제(Birthday problem)

- 사람이 임의로 모였을 때 그 중에 생일이 같은 두 명이 존재할 확률을 구하는 문제
- 생일이 같은 두 명이 반드시 존재하며, 23명 이상이 모인다면 그 중 두 명이 생일이 같은 확률은 1/2를 넘는다.
- n명의 사람이 있을 때 그 중 생일이 같은 사람이 둘 이상 있을 확률 $p(n)$
- 모든 사람이 생일이 다를 확률 $\bar{p}(n)$은 $1 - p(n)$
  - 두 번째 사람의 생일은 첫 번째 사람과 다르고, 세 번째 사람의 생일은 첫 번째와 두 번째 모두와 달라야 하므로 다음과 같은 식을 얻을 수 있다.

$$\begin{aligned}\bar{p}(n) &= 1 \times \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right) \times \cdots \times \left(1 - \frac{n-1}{365}\right) \\ &= \frac{365 \times 364 \times 363 \times \cdots \times (365 - n + 1)}{365^n} \\ &= \frac{365!}{365^n (365 - n)!}\end{aligned}$$

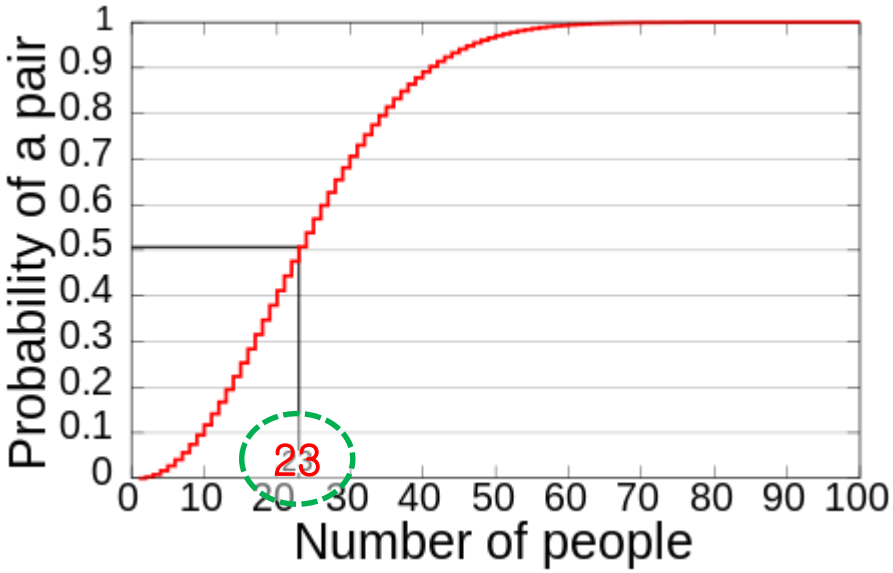
여기서, n≤365 인 자연수이고, !는 계승

  - 최종적으로,

$$p(n) = 1 - \frac{365!}{365^n (365 - n)!}$$

- p(n)을 계산

n	$p(n)$
1	0.0%
5	2.7%
10	11.7%
20	41.1%
23	50.7%
30	70.6%
40	89.1%
50	97.0%
60	99.4%
70	99.9%
100	99.99997%
200	99.99999999999999999999999998%
300	$(100 - (6\times10^{-80}))\%$
350	$(100 - (3\times10^{-129}))\%$
365	$(100 - (1.45\times10^{-155}))\%$
366	100%
367	100%



스토리2의 「생일 공격」

1) 맬로리는 백만 원 계약서를 N개 작성
2) 맬로리는 일억 원 계약서를 N개 작성
3) 맬로리는 (1)의 해시 값 N개와 (2)의 해시 값 N개를 비교해서 일치하는 것이 있는지 찾는다
4) 일치하는 것이 발견되면 그 백만 원 계약서와 일억 원 계약서를 가지고 앨리스를 속이러 간다

N의 크기

- 문제가 되는 것은 N의 크기이다
- N이 작으면 맬로리는 감쪽같이 생일 공격에 성공할 수 있다
- N이 크면 시간도 메모리양도 많이 필요해지기 때문에 생일 공격은 어려워진다
- N은 해시 값의 비트 길이에 의존

스토리 비교

- 스토리1
  - 앨리스가 백만원 계약서를 만들었기 때문에 해시 값은 고정되어 있다.
  - 맬로리는 그 해시 값과 같은 메시지를 발견해 내는 약한 충돌 내성을 공격하는 것

스토리 비교

- 스토리2
  - 맬로리가 2개의 계약서를 만드는 것이므로 해시 값은 뭐라도 상관 없다
  - 백만원 계약서와 일억원 계약서의 해시 값이 같기만 하면 된다.

제 8절 어떤 일방향 해시 함수를 사용하면 좋은가

- MD5는 안전하지는 않으므로 사용해서는 안 됨
- SHA-2는 SHA-1에 대한 공격방법에 대한 대처를 하여 고안했기 때문에 안전
- SHA-3은 안전
- 자작 알고리즘은 사용해서는 안 됨

제 9절 일방향 해시 함수로 해결할 수 없는 문제

- 일방향 해시 함수는 「조작 또는 변경」을 검출할 수 있지만, 「거짓 행세」검출은 못 한다

- 인증:
  - 이 파일이 정말로 앨리스가 작성한 것인지를 확인하는 것
  - 인증을 수행하기 위한 기술
    - 메시지 인증 코드
    - 디지털 서명

해시함수 보안 강도 비교

- 해시함수의 보안 강도는 출력비트 길이가 아님
  (해시함수의 보안 강도는 PPT 92 권고사항 참고)
- 해시함수의 안정성을 위한 세 가지 조건(충돌 저항, 역상 저항, 제2역상 저항성) 중 하나라도 만족 하지 않는 값을 찾는 것

보안 강도	대칭키 암호 알고리즘 (최소 키 길이)	공개키 암호 알고리즘				해시 함수[11] (보안 강도)
		인수분해 (최소 키 길이)	이산대수		타원곡선 암호 (최소 키 길이)	
			공개키 (최소 키 길이)	개인키 (최소 키 길이)		
80	80	1024	1024	160	160	80
112	112	2048	2048	224	224	112
128	128	3072	3072	256	256	128
192	192	7680	7680	384	384	192
256	256	15360	15360	512	512	256

- 키 길이에 따른 안전성 유지 기간에 대한 NIST 권고

(단위: 비트)

암호 기술 / 기간	대칭키 암호 알고리즘 (키 길이)	공개키 암호 알고리즘(키 길이)			해시함수[14] (보안 강도)
		인수분해 기반	이산대수 기반	타원곡선 암호기반	
2030년 까지	112 이상	2048 이상	2048(공개키), 224(개인키) 이상	224 이상	112 이상
2031년 이후	128 이상	3072 이상	3072(공개키), 256(개인키) 이상	256 이상	128 이상

- 대표적인 권고 해시 알고리즘

종류		출력값 길이 (비트)	보안 강도 (비트)	참조규격
SHA-2	224	224	112	NIST FIPS 180-4
	256	256	128	
	384	384	192	
	512	512	256	
SHA-3	224	224	112	NIST FIPS 202
	256	256	128	
	384	384	192	
	512	512	256	

참고문헌

- 암호학과 네트워크 보안, Behrouz A. Forouzan 지음, 이재광외 3인 역, 한티미디어
- 컴퓨터 보안과 암호, WILLIAM STALLINGS 지음, 최용락외 2인 역, 그린출판사
- 생일문제, 위키백과, https://ko.wikipedia.org/wiki/생일_문제
- 금융부문 암호기술 활용 가이드, 금융보안원

Q & A

Thanks!