알기 쉬운

정보보호개론 3판

흥미로운 암호 기술의 세계

INFORMATION SECURITY and CRYPTOGRAPHY

INFINITYBOOKS
인피니티북스

PART II 인증

INFORMATION SECURITY and CRYPTOGRAPHY

CHAPTER 8 일방향 해시 함수

Section 01
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), 완전성
  - 파일이 변경되지 않았음

파일의 무결성을 조사하고 싶다

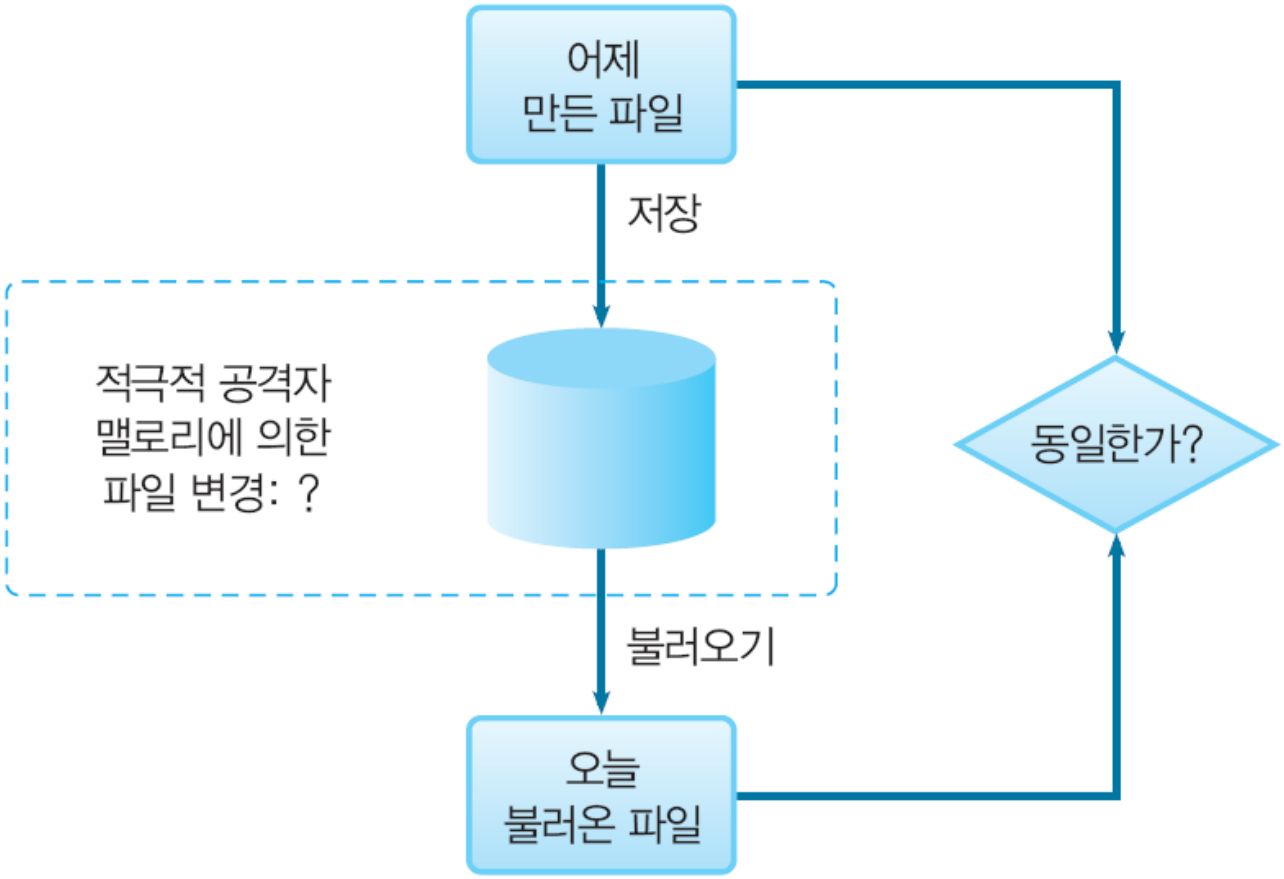


그림 8-1 • 파일의 무결성을 조사하고 싶다

파일 전체를 안전한 장소에 보존해 두고, 나중에 비교하는 방법

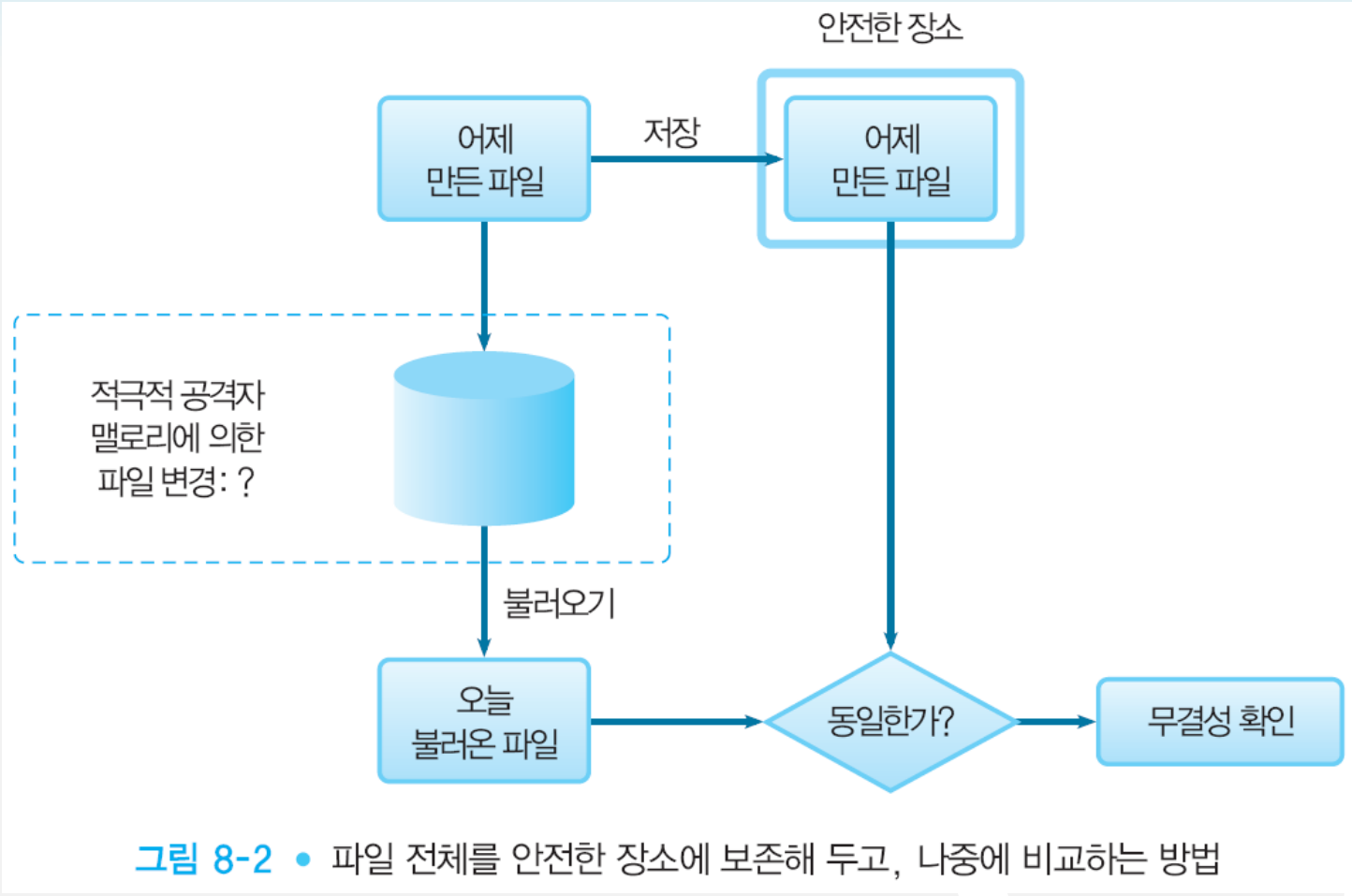


그림 8-2 • 파일 전체를 안전한 장소에 보존해 두고, 나중에 비교하는 방법

- 안전한 장소에 보관이 가능하다면 무결성 확인은 필요 없음
- 파일이 매우 크다면 복사를 하거나 안전한 장소에 두거나 파일을 비교하는 데에도 많은 시간이 소요됨

파일의 지문

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

파일을 비교하는 대신에 해시 값을 비교하는 방법

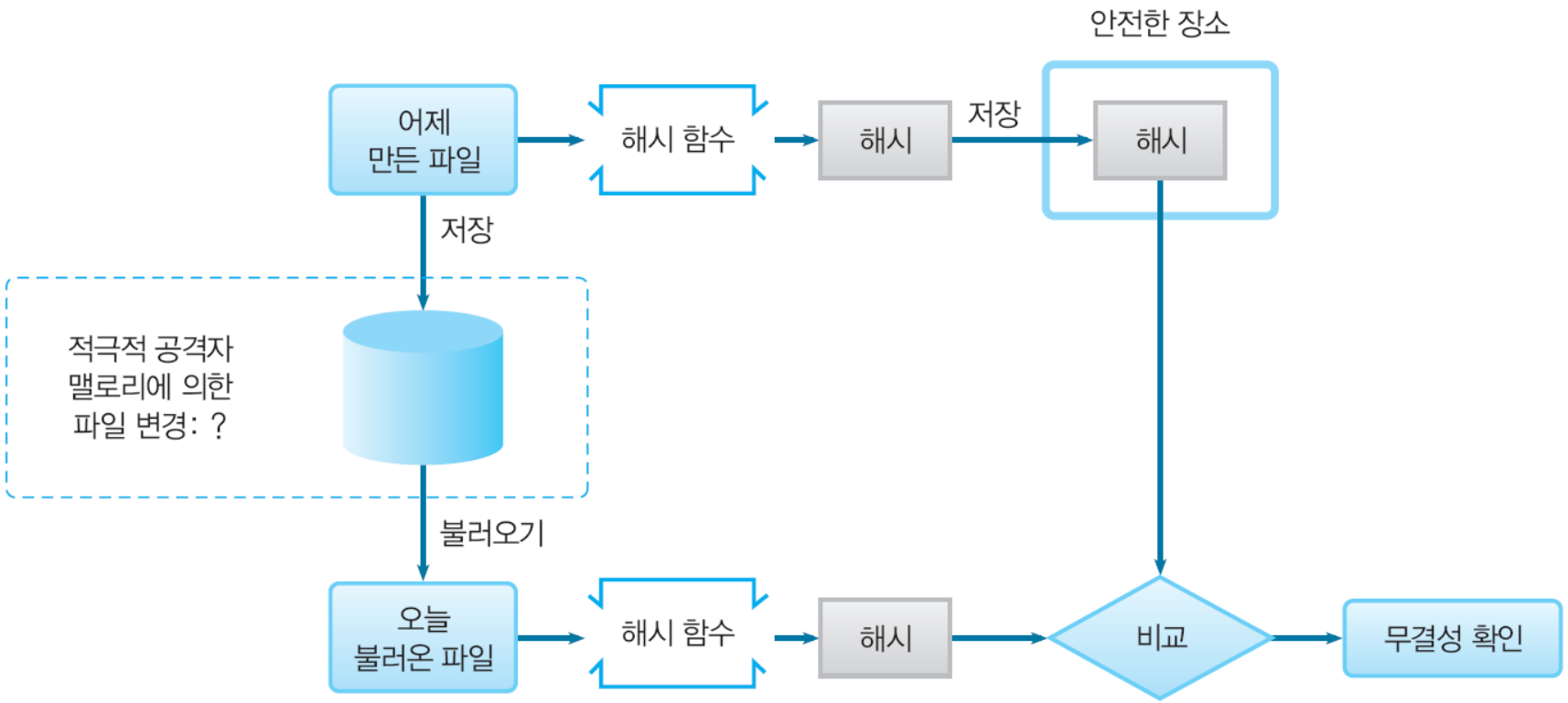


그림 8-3 • 파일을 비교하는 대신에 해시 값을 비교하는 방법

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 나누어보자
- 출력: 7391
  - 몫은 15029.9565217391**7391**3043..... 이므로 7자리부터 10자리의 수는 7391

일방향 해시 함수

- **일방향 해시 함수**(one-way hash function)
  - 입력과 출력이 각각 1개씩 있다.
  - 입력은 **메시지**(message)
  - 출력은 **해시 값**(hash value)
  - 일방향 해시 함수는 메시지를 기초로 해서 해시 값을 계산

일방향 해시 함수는 메시지를 기초로 해서 해시 값을 계산

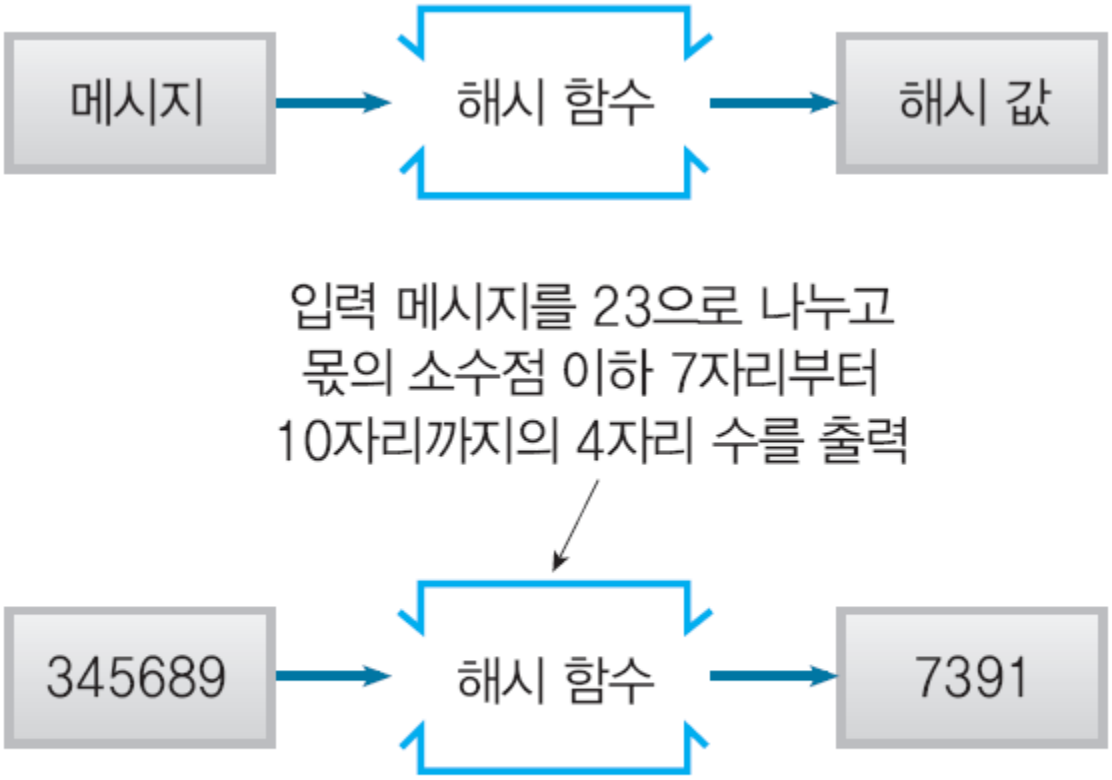


그림 8-4 • 일방향 해시 함수는 메시지를 기초로 해서 해시 값을 계산한다

일정한 크기의 출력

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

해시 값은 항상 고정 길이

그림 8-5 • 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

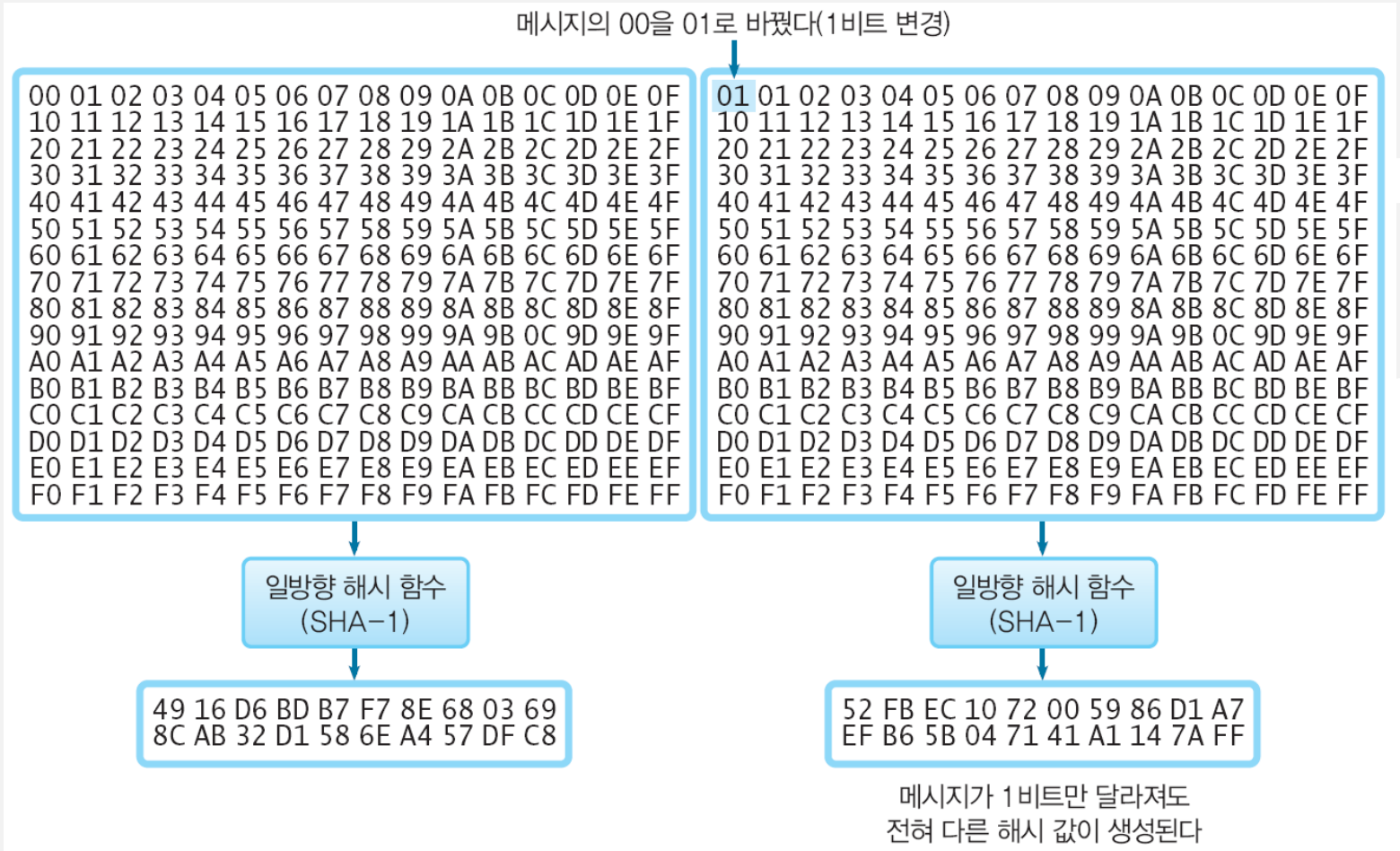


그림 8-6 • 메시지가 1비트만 달라도 다른 해시 값이 된다.

해시 함수의 충돌

- **충돌(collision)**
  - 2개의 다른 메시지가 같은 해시 값을 갖는 것
- **충돌 내성(collision resistance)**
  - 충돌을 발견하는 것이 어려운 성질

일방향 해시 함수의 충돌 내성

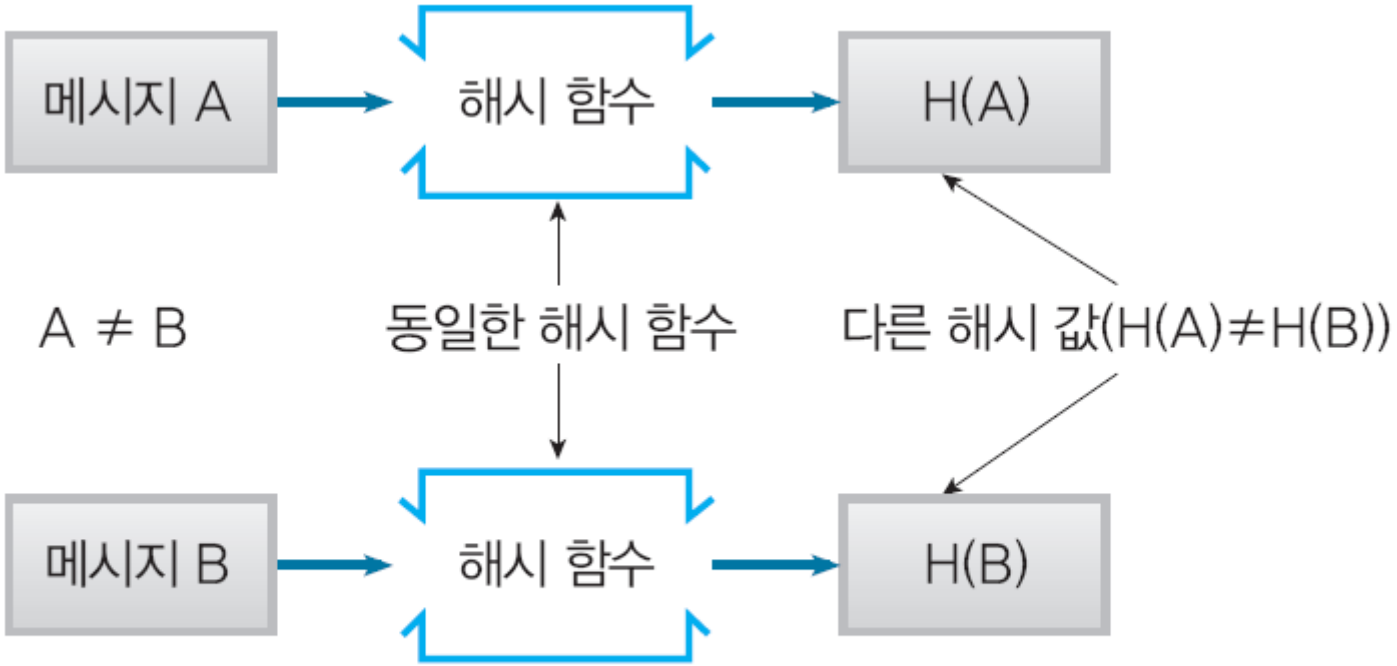


그림 8-7 • 일방향 해시 함수의 충돌 내성

충돌내성

- 약한 충돌 내성
  - 어느 메시지의 해시 값이 주어졌을 때, **그 해시 값과 같은 해시 값을 갖는 다른 메시지를 발견해 내는 것이 매우 곤란**한 성질
- 강한 충돌 내성
  - **해시 값이 일치할 것 같은, 다른 2개의 메시지를 발견해 내는 것이 매우 곤란**한 성질

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
- 유리창을 산산조각으로 깨는 것은 간단하지만 원래 유리창으로 되돌릴 수는 없다

일방향 해시 함수의 일방향성

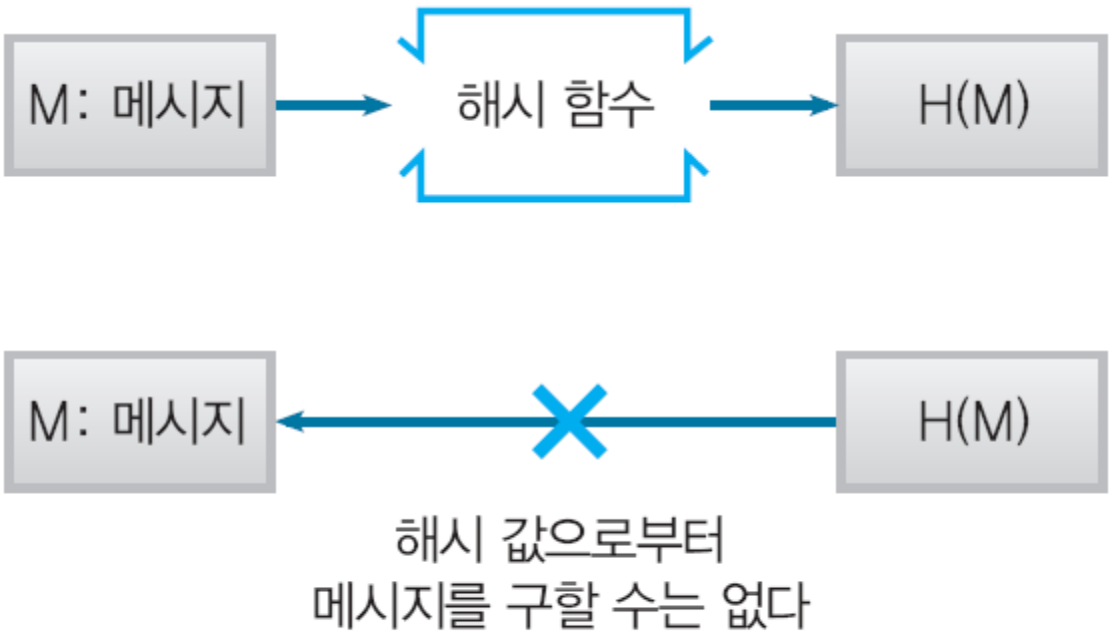


그림 8-8 • 일방향 해시 함수의 일방향성

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

Quiz 1 충돌내성

- 충돌내성의 이야기를 들은 앨리스는 이렇게 생각했다. 어째서『충돌을 발견하는 것이 곤란하다』는 어중간한 성질을 생각하는 것일까. 정확하게『충돌은 존재하지 않는다』고 하는 일방향 해시 함수를 생각하면 될 텐데. 그러나 앨리스의 생각은 잘못된 것이다. 어째서 일까?

임의 길이의 메시지로부터 고정 길이의 해시값을 계산하는 일방향 해시함수는 반드시 충돌하기 때문이다.

- 비둘기집원리(pigeon-hole principle) : N+1 마리의 비둘기를 N개의 비둘기집에 넣는다면 비둘기가 2마리 이상 있는 비둘기 집이 적어도 1개는 존재한다.

Section 02
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
- 해시값을 써서 오리지널 사이트에서 배포하고 있는 파일과 자신이 입수한 파일이 같다는 것을 확인

소프트웨어 개정 검출을 위해 일방향 해시 함수를 사용

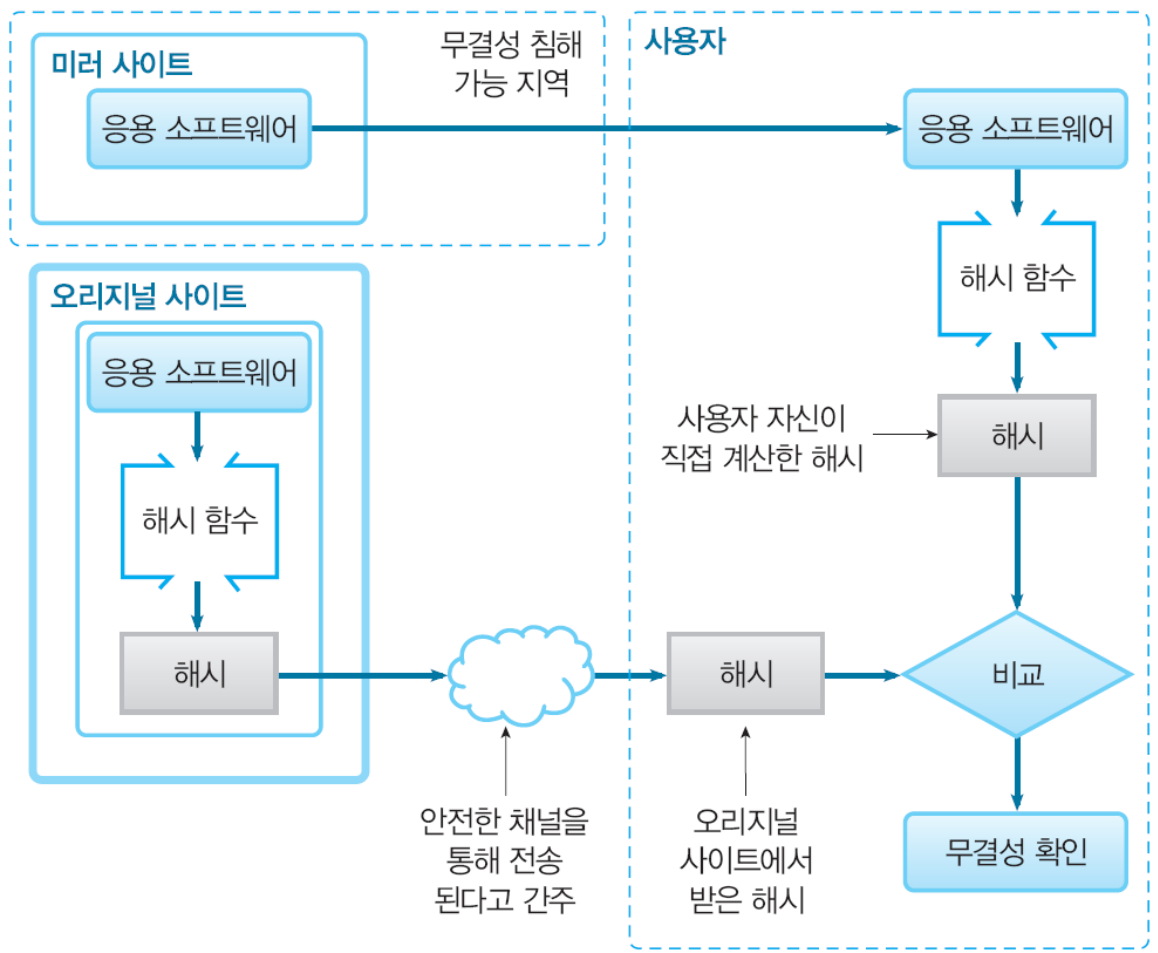


그림 8-9 • 소프트웨어 변경 검출을 위해 일방향 해시 함수를 사용한다

Quiz 2 일방향 해시 함수의 오용

- 앨리스는 자신이 만든 game.exe라는 이름의 파일이 자신이 자고 있는 사이에 변경되지 않았다는 것을 확인하고자 생각했다.

- 그래서 앨리스는 game.exe라는 파일의 해시 값을 계산하여, 그 값을 hashvalue라는 이름의 파일에 기록하고, game.exe와 hashvalue를 같은 하드디스크에 보존해 두었다.

- 이것으로 안심이라고 생각하고 앨리스는 잠을 잤다. 다음 날 아침 앨리스는 일방향 해시 함수를 한 번 더 사용해서 game.exe의 해시 값을 재계산하여, 어제 보존해 둔 파일 hashvalue의 내용과 비교했다. 그러자, 비교 결과 차이가 없다는 것을 알았다.

- 이 결과로부터 앨리스는 game.exe라는 파일이 능동적 공격자 맬로리에 의해 변경되어 있지 않다고 판단했다. 그러나 앨리스의 이 판단은 잘못되어 있다. 어째서 인가?

2.2 패스워드를 기초로 한 암호화

- 패스워드를 기초로 한 암호화(password based encryption; PBE)에서 사용
  - PBE에서는 패스워드와 솔트(의사난수 생성기로 생성된 랜덤값)를 섞은 결과의 해시 값을 구해 그것을 암호화 키로 사용
  - 패스워드 사전 공격(dictionary attack)을 막을 수 있다

2.3 메시지 인증 코드

- 「송신자와 수신자만이 공유하고 있는 키」와 「메시지」를 혼합해서 그 해시 값을 계산한 값
- 통신 중의 오류나 수정 그리고 「가장」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 일단 구하고, 그 해시 값에 대해 디지털 서명을 수행

2.5 의사난수 생성기

- 일방향 해시 함수를 이용한 의사난수 생성
- 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

Section 03
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-3

3.1 MD4와 MD5

- **MD4(Message Digest 4)**
  - Rivest가 1990년에 만든 일방향 해시 함수
  - 128비트의 해시 값
  - RFC 1186, RFC1320
  - Dobbertin에 의해 충돌 발견 방법이 고안
  - 현재는 안전하지 않다

3.1 MD4와 MD5

- **MD5**
  - Rivest가 1991년에 만든 일방향 해시 함수
  - 128비트의 해시 값
  - RFC 1321
  - 암호해독에 취약함을 보여주는 여러 가지 암호해독 방법들이 개발
  - MD5가 완전히 뚫린 것은 아님
    - MD5 내부 구조 일부에 대한 몇 가지 공격 방법 발견
  - 사용을 권장하지 않는다

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

- **SHA-1(Secure Hash Algorithm-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

칼럼

칼럼 6종류의 SHA-2

SHA-2에는 아래에 언급한 6종류가 있다. 이 표를 보면 알 수 있듯이 6종류의 SHA-2를 생성해 내고 있는 것은 실제로는 SHA-256 2종류로서, 이 이외의 종류는 불필요한 비트를 잘라 버리는 방식으로 실현하고 있다. 또한, SHA-224와 SHA-256은 내부 상태를 32 x 8비트로 실현하고 있고, 32비트 CPU에 적용되도록 되어 있다.

표8-C1 • 6종류의 SHA-2

이름	출력	비트 길이	내부 상태
SHA-224	224 비트	32 x 8 = 256	SHA-256로부터 32비트를 버린다.
SHA-256	256 비트	32 x 8 = 256	SHA-256로부터 32비트를 버린다.
SHA-512/224	224 비트	64 x 8 = 512	SHA-512로부터 288비트를 버린다.
SHA-512/256	256 비트	64 x 8 = 512	SHA-512로부터 256비트를 버린다.
SHA-384	384 비트	64 x 8 = 512	SHA-512로부터 128비트를 버린다.
SHA-512	512 비트	64 x 8 = 512	

3.3 RIPEMD-160

- **RIPEMD-160**
  - **RIPE :** Réseaux IP Européens (**RIPE** , French for "European IP Networks")
  - 1996년에 Hans Dobbertin, Antoon Bosselaers, Bart Preneel이 제작
  - 160비트의 해시 값
  - European Union RIPE 프로젝트로 만들어진 RIPEMD 함수의 개정판
  - 비트코인에서 사용
- 별도 버전
  - RIPEMD-128, RIPEMD-256, RIPEMD-320

3.4 SHA-3

- SHA-1의 강한 충돌 내성 침해
- NIST는 SHA-1을 대체하는 차세대 일방향 해시함수로2007년에 “SHA-3” 선정 시작
- AES와 같은 경쟁 방식으로 표준화
- 2012년 선정 완료
  - KECCAK(케챡)을 표준으로 선정
  - 이것이 SHA-3

Section 04
일방향 해시 함수 SHA-512

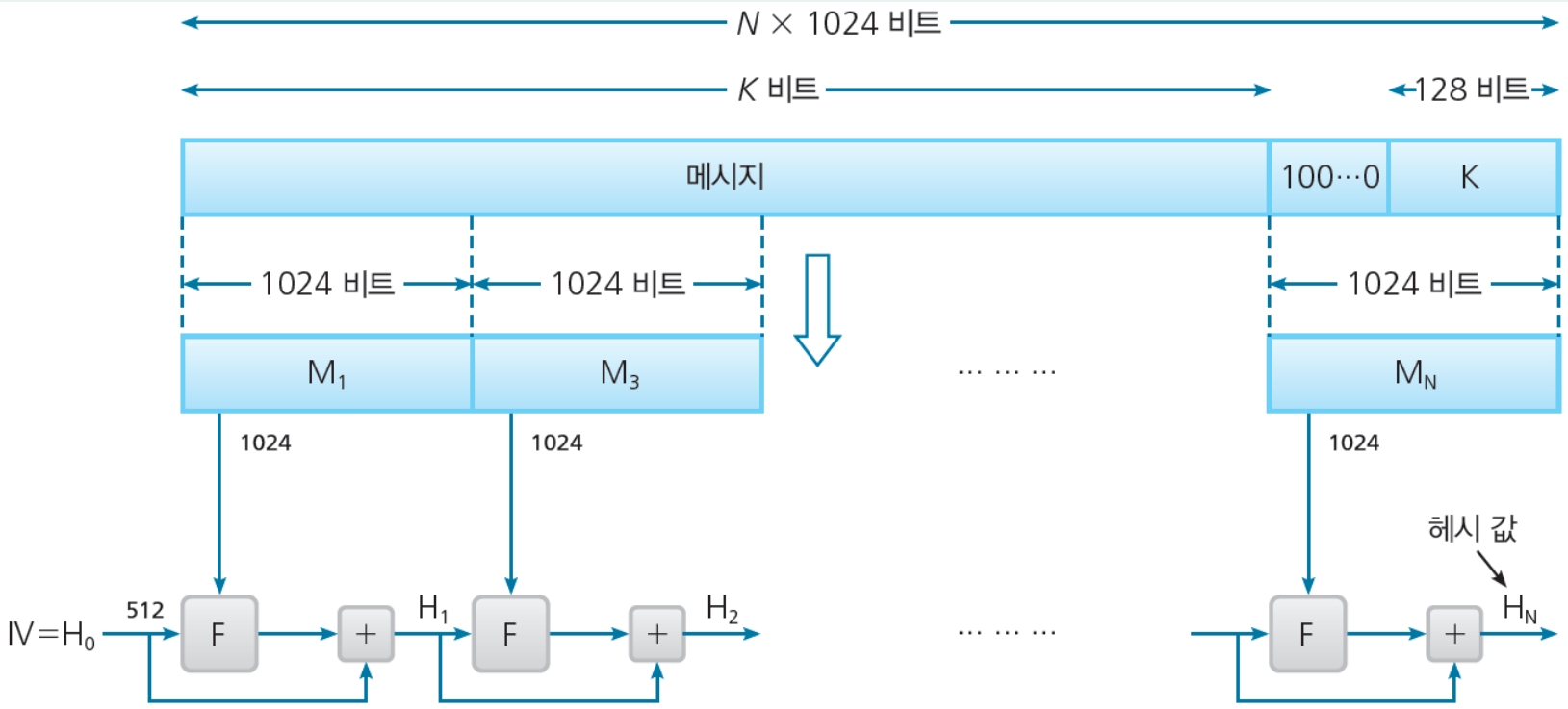
4.1 SHA-512구조

입력: 최대 2^{128}비트 이하 메시지

출력: 512비트 메시지 다이제스트

입력 데이터는 길이 1024비트 블록으로 처리

SHA-512 해시 값 생성



+는 mod 2^{64}로 실행한 합을 의미함

그림 8-10 • SHA-512를 사용한 해시 값 생성

SHA-512 처리 단계

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

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

길이 붙이기

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

MD 버퍼 초기화

- 해시 함수의 중간 값과 최종 값을 저장하기 위해 512-비트 버퍼 사용
- 버퍼를 8개의 64비트 레지스터로 나타냄

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

1024-비트(128-워드) 블록 메시지 처리

- 각 라운드가 80 라운드로 구성
- 각 라운드에서 512 비트 버퍼 abcdefgh를 입력으로 사용
- M_i : 1024비트 블록
- H_{i-1} : 중간 해시값
- W_t : 64비트 값
- K_t : 덧셈 상수 ($0 \le t \le 79$)
- 처음 8개의 소수들의 세제곱근의 소수점 이하 처음 64비트

1024비트 한 개에 대한 SHA-512 처리

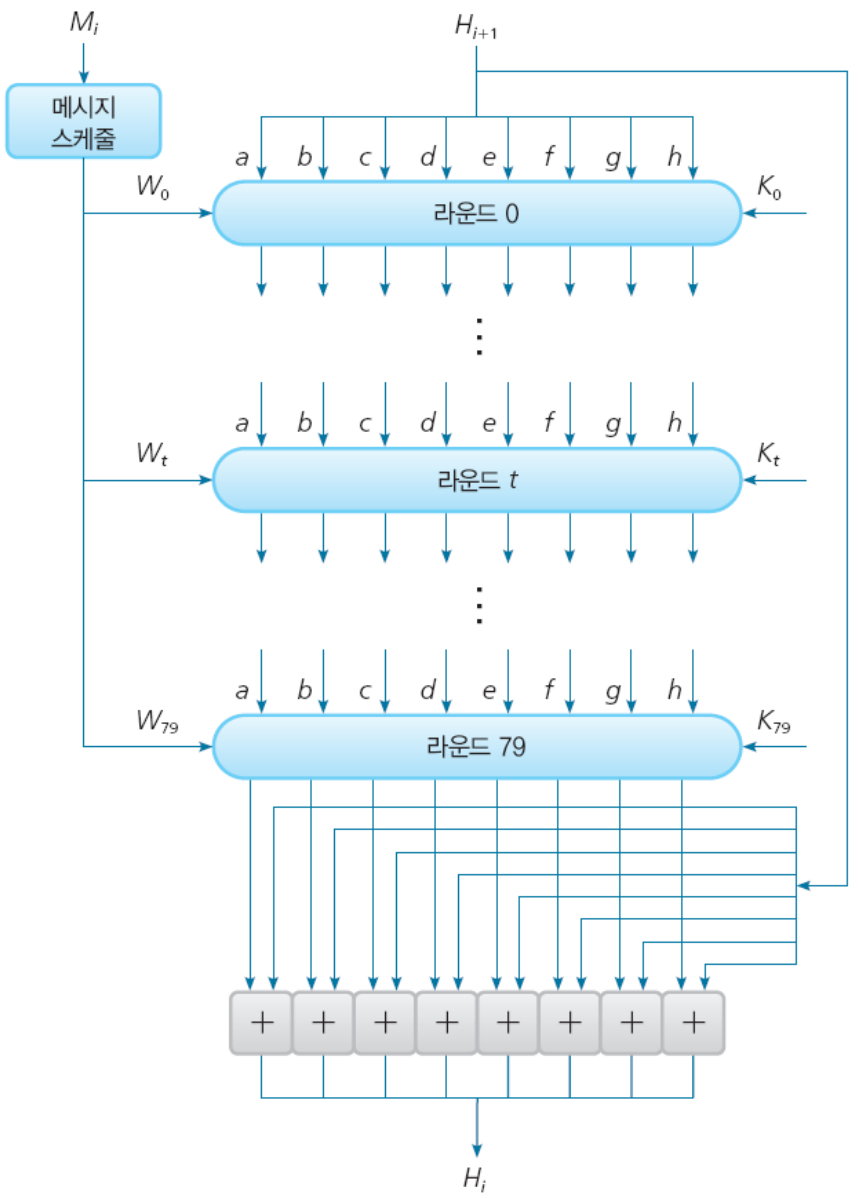


그림 8-11 • 1024비트 한 개에 대한 SHA-512 처리

출력

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

4.2 SHA-512의 안전성

- 2017년까지 약점 발견된 것 없음
- 확률적 안전성
  - 동일한 메시지 다이제스트를 갖는 두 개의 서로 다른 메시지를 찾는 난이도 연산 수행 수는 2^{256}
  - 주어진 다이제스트와 동일한 다이제스트를 갖는 메시지를 찾는 난이도 연산 수행 수는 2^{512}

Section 05
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 **KECCAK**이라는 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

표 8-1 • 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

5.3 SHA-3 최종 후보와 SHA-3 결정

- 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 내장 시 높은 고성능
  - 다른 최종 목록과 비교하여 보안성이 높음

Section 06
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 비트열을 생성할 수 있지만 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)를 규정
  - SHAKE=Secure Hash Algorithm + KECCAK

6.2 스펀지 구조

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

스펀지 구조

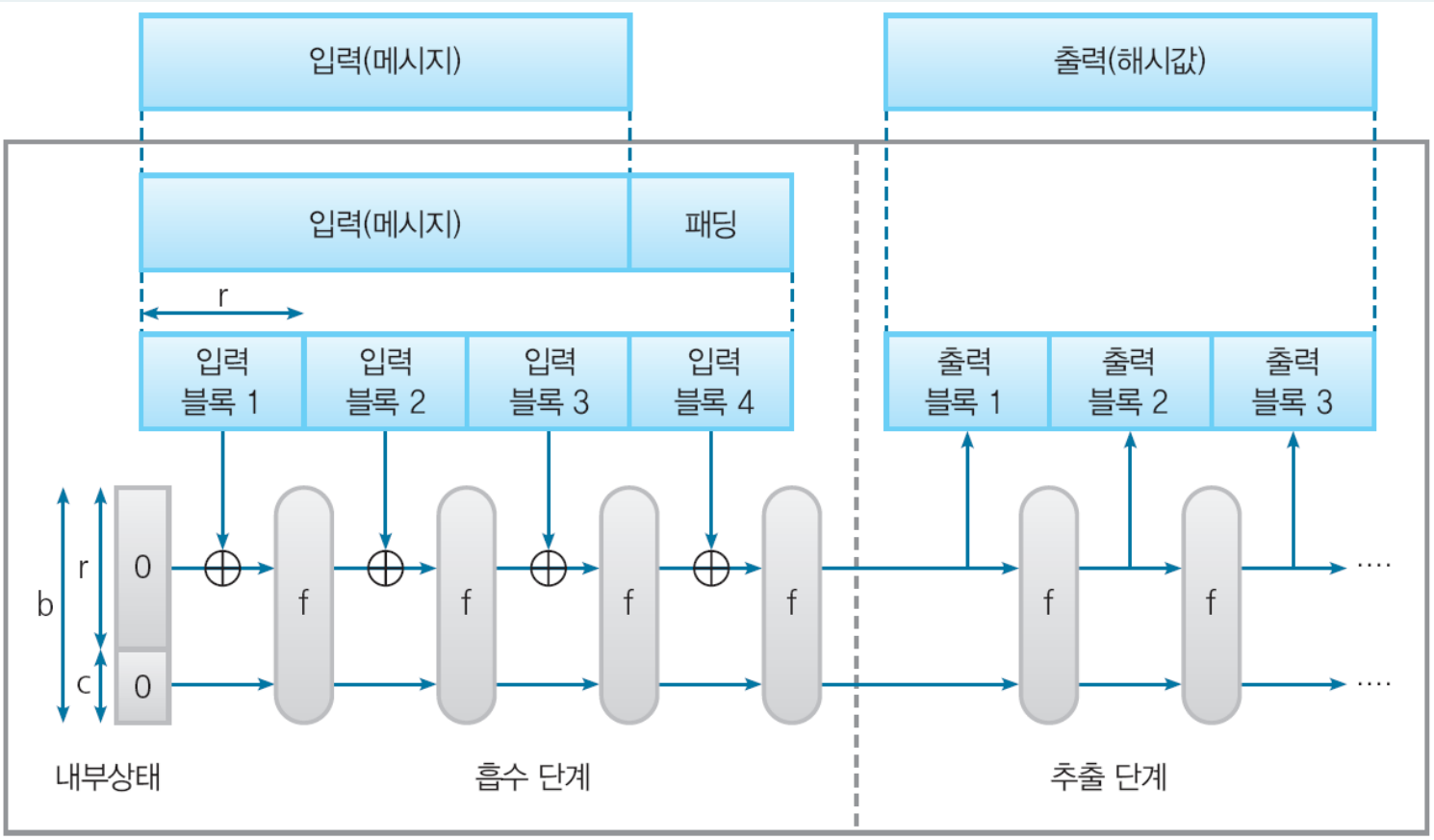


그림 8-12 • 스펀지 구조[6]

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
- 입력 블록을 다 처리하면 흡수 단계를 종료하고 추출 단계를 진행

추출 단계(I)

- 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(II)

- 「내부 상태를 혼합해서 출력 블록(r비트)를 추출」하는 처리
- 비트율(r 비트) 단위로 스펀지구조의 내부 상태로 부터 비트열이 「추출」되어 가는 것을 이미지화 할 수 있음
- 수분이 스펀지로 부터 추출되는 것과 같은 모양

6.3 듀플렉스 구조

- 입력과 출력이 같은 속도로 진행
- 해시 값을 구하는 용도만이 아니라 의사난수 생성기, 스트림 암호, 인증이 포함된 암호, 메시지 인증 코드 등 암호학자의 도구 상자에 들어있는 대부분의 기능을 활용할 수 있음

듀플렉스 구조

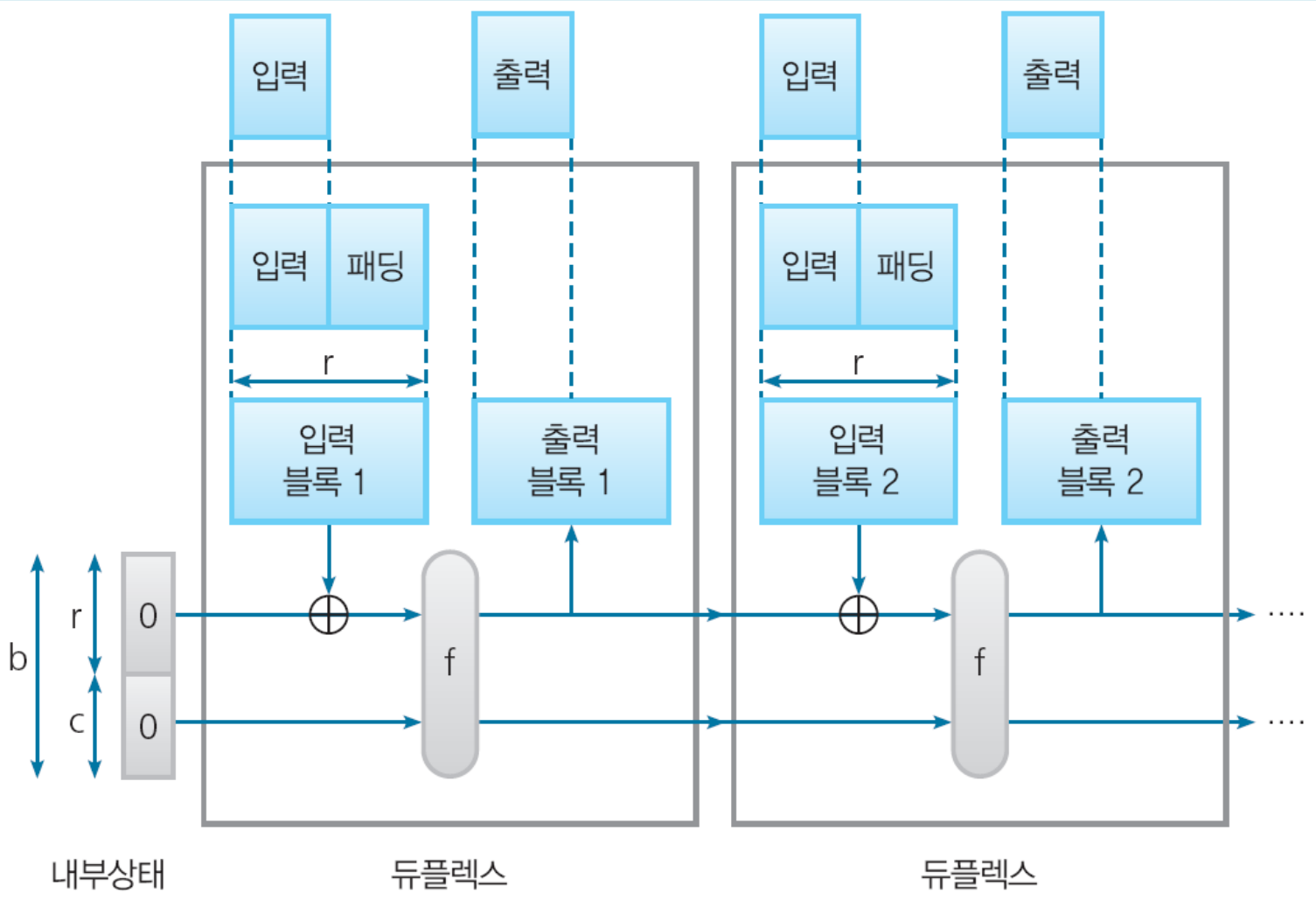


그림 8-13 • 듀플렉스 구조

6.4 KECCAK의 내부 상태

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

내부 상태(b=5x5xz 비트)

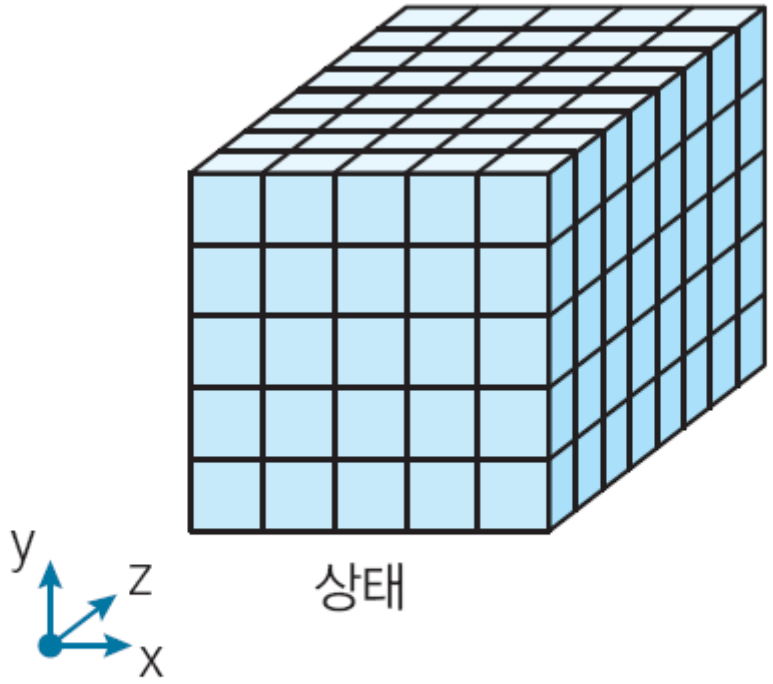


그림 8-14 • KECCAK의 내부 상태(state)[7]

내부 상태의 일부

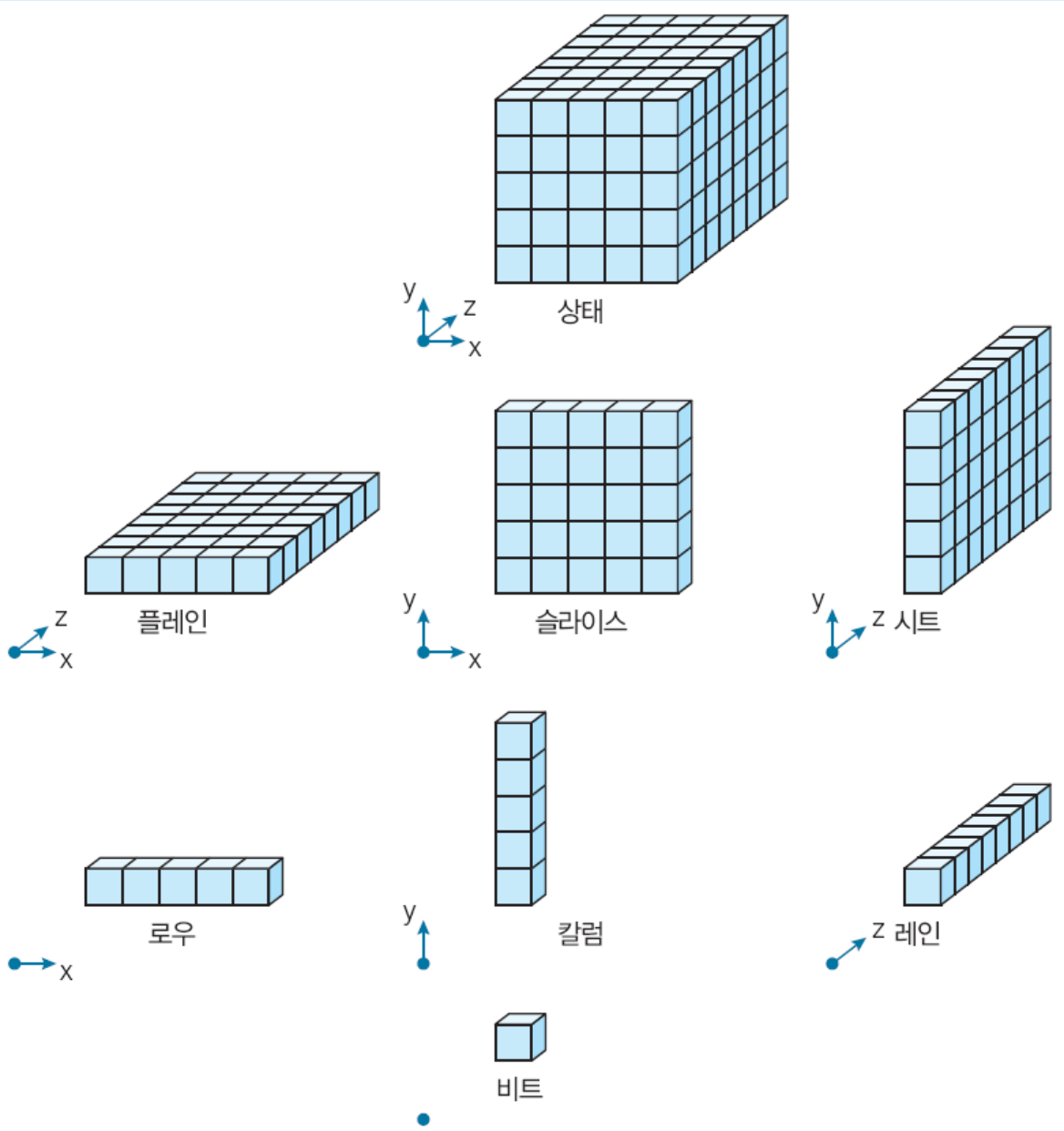


그림 8-15 • KECCAK 내부 상태의 일부

6.5 함수 KECCAK-f[b](l)

- 내부상태를 혼합하는 함수
- b를 폭이라고 함
  - b= 25, 50, 100, 200, 400, 800, 1600 7종류
  - SHA-3에서는 b=1600 사용
  - SHA-3의 내부상태 b=5 x 5 x 64=1600 비트
- 폭 b를 변경하여 내부 상태의 비트 폭을 변경할 수 있음
- 폭이 변해도 기본적인 형태는 변하지 않는다. 마트료시카 구조

6.5 함수 KECCAK-f[b](II)

- 아래에 설명하게 될 θ, ρ, π, χ, ι라는 5개 단계를 1라운드라고 할 때 총 $12 + 2i$ 라운드를 반복하는 함수
- 폭 b= 25 X 2^i
- SHA-3으로 사용되는 KECCAK-f[1600]의 경우 라운드 수는 24

$$2^i = \frac{b}{25} = \frac{1600}{25} = 64$$

$$i = 6$$

12+2x6=24라운드

θ 단계

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

θ 단계

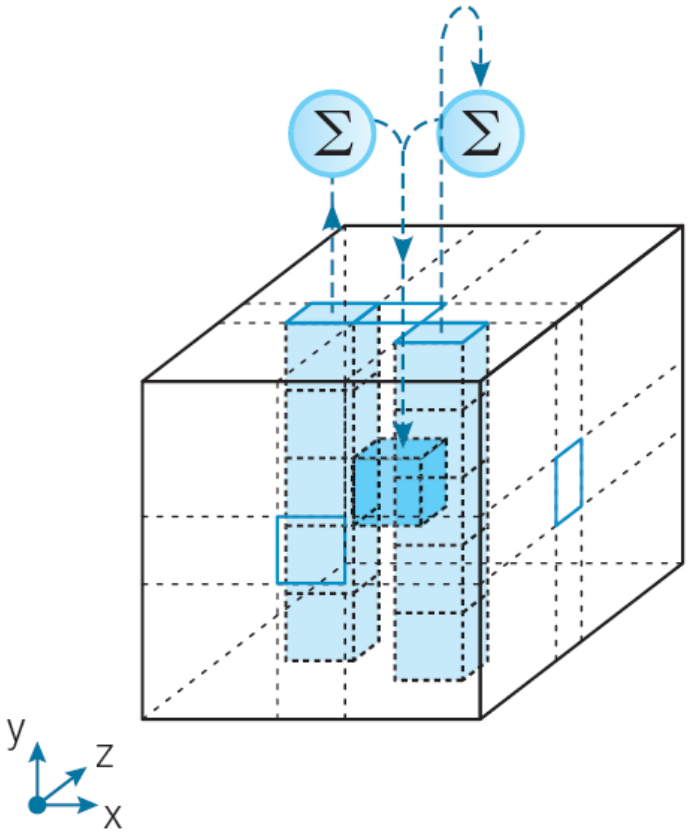


그림 8-16 • θ 단계

ρ 단계

- z 방향(레인 방향) 비트 이동

ρ 단계

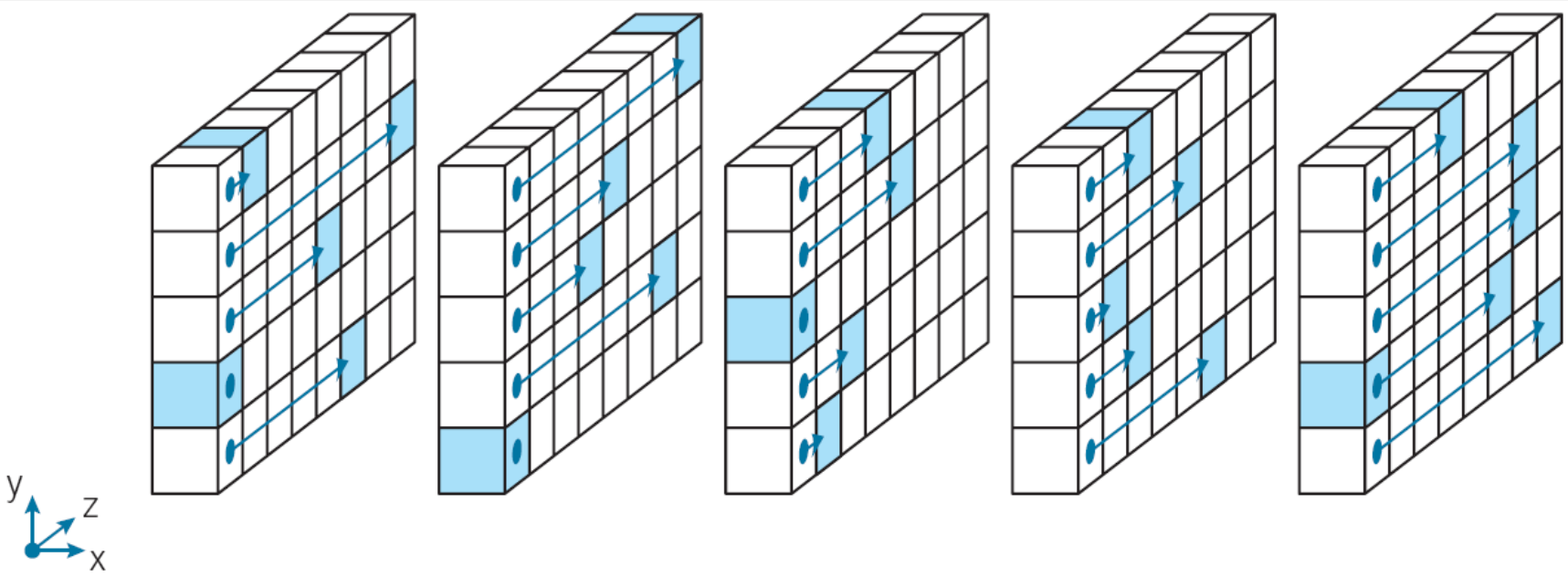


그림 8-17 • ρ 단계

π 단계

- 비트 이동을 레인 전체에 대해 수행

π 단계(특정 1개의 슬라이스에 적용한 경우)

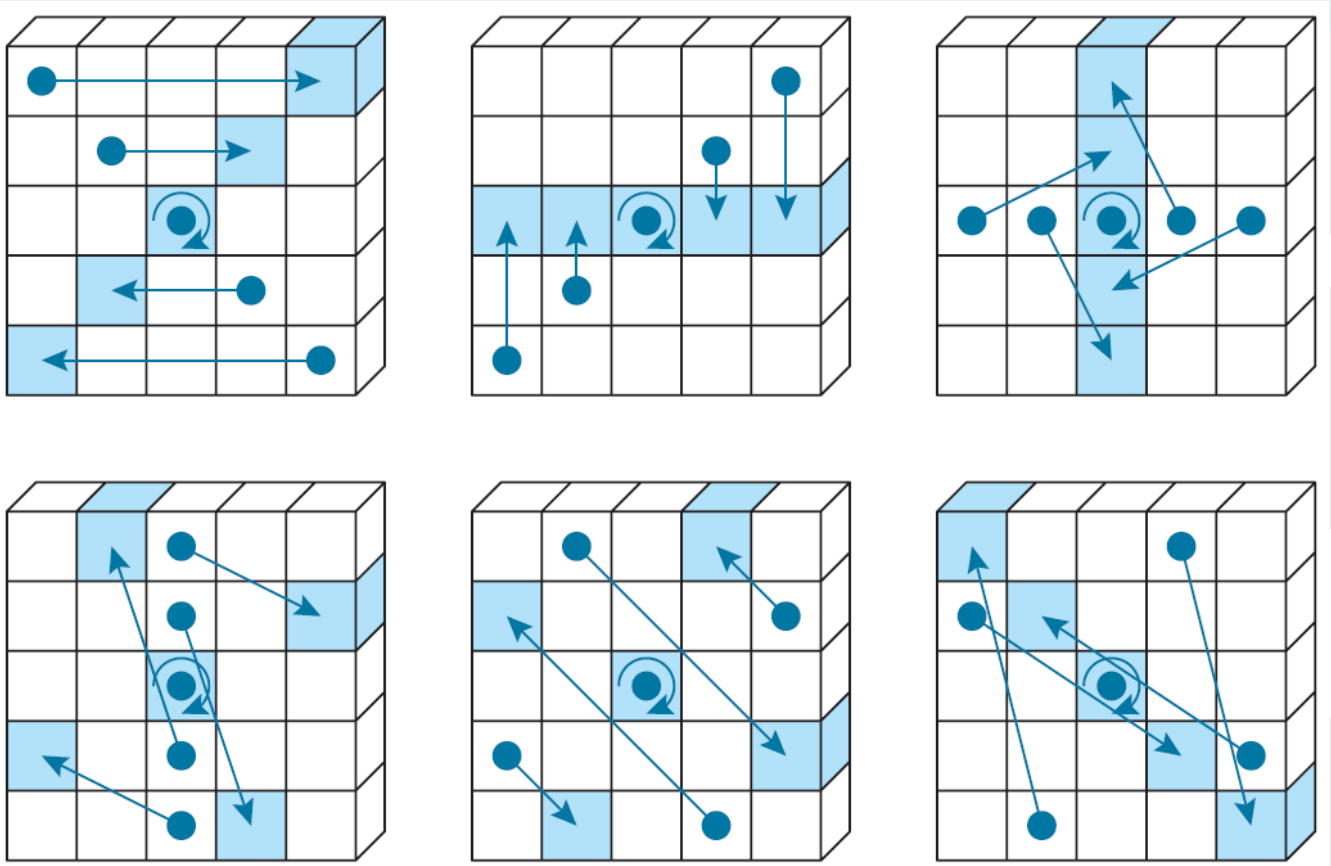


그림 8-18 • π 단계

χ 단계

- 논리회로 NOT와 AND 연산을 활용하여 수행

χ 단계

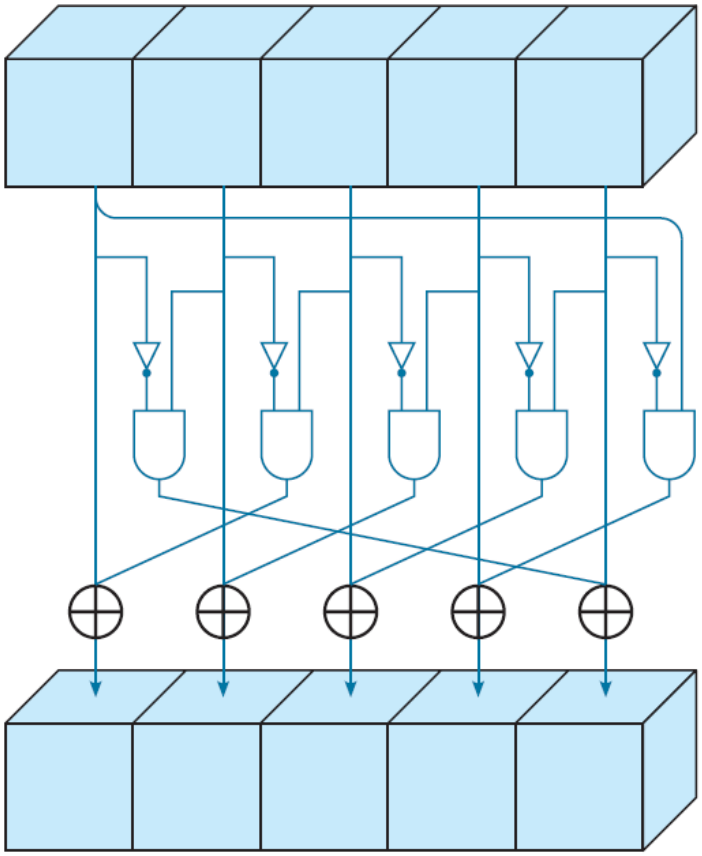


그림 8-19 • χ 단계

ι 단계

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

6.11 KECCAK에 대한 공격

- MD변환
  - MD4, MD5, RIPEMD, RIPEMD-160, SHA-1, SHA-2 등, 지금까지 거의 일방향 해시 함수 알고리즘에서 이용
- 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

6.12 약한 KECCAK에 대한 공격 테스트

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

Section 07
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
- 예: 맬로리는 앨리스의 컴퓨터에서 계약서 파일을 발견하고, 그 안의

  「앨리스의 지불 금액은 백만 원으로 한다.」

  부분을,

  「앨리스의 지불 금액은 일억 원으로 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 일억 원(一億원)으로 한다.
  - 앨리스의 지불 금액은 일억 원(壹億원)으로 한다.
  - 앨리스의 지불 금액은 100000000원으로 한다.
  - 앨리스의 지불 금액은 ₩100000000으로 한다.
  - 앨리스의 지불 금액은, 일억 원(一億원)으로 한다.
  - 앨리스가 지불하는 금액은 일억 원으로 한다.
  - 앨리스는 일억 원을 지불하는 것으로 한다.
  - 대금으로서, 앨리스는 일억 원을 지불한다.

공격 스토리 1

- 「일억 원 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
- 같은 해시값을 갖는 다른 파일을 만들어 내는것이 일방향 해시함수에 대한 공격으로 일종의 전사공격
- 일방향 해시함수의『약한 충돌내성』을 깨고자 하는 공격

공격 스토리 1

- SHA3-512 전사공격의 시행 횟수의 기대치

  $n/2^{512}$ (n개의 메시지)

- 특정 해시값을 가진 메시지를 발견하는 공격
  - 프리이미지 공격 : 하나의 해시 값이 부여될 경우 그 해시 값과 동일한 해시 값을 가진 메시지를 어떠한 것이라도 좋으니까 찾아내는 공격
  - 제2프리이미지공격 : 어떤 메시지1이 부여되고 그 메시지1의 해시 값과 같은 해시 값을 가진 별개의 메시지2를 찾아내는 공격

7.2 생일 공격(공격 스토리2)

- **「강한 충돌 내성」을 깨고자 하는 공격**
  - 맬로리는 해시 값이 같은 값을 갖는 「백만 원 계약서」와 「일억 원 계약서」를 미리 만들어 둔다
  - 맬로리는 시치미를 떼고 「백만 원 계약서」를 앨리스에게 건네주고, 해시 값을 계산시킨다
  - 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-1

- 「어느 특정일을 정하고 2명이 그 날 태어날」 가능성은 확실히 높지 않다. 그러나 「1년의 어느 날이라도 상관없으므로 2명이 같은 날 태어날」 가능성은 의외로 높다.
- N명 중 적어도 2명의 생일이 일치할 확률을 계산하는 것이 아니라 먼저 N명 전원의 생일이 일치 하지 않을 확률을 구하고 그것을 1에서 빼는 것
- 생일이 일치 하지 않을 조합수: 365x364x……x(365-N+1)
- 모든 경우의 수 : 365^N
- 확률 : $1-\frac{365X364X\ \ldots(365-N+1)}{365^N}$
- N=23 일 때 확률은 0.507297
- 『1년 일수가 Y일 있다고 하고, N명으로 구성된 그룹 안에서 적어도 2명의 생일이 일치할 확률이 2분의 1이상이 되기 위한 최저의 N은 얼마인가?』 N≒$\sqrt{Y}$ (1년의 일수의 제곱근)

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

이유-2

- 맬로리는 계약서 N 개를 작성한다.
- 「앨리스가 사용하는 일방향 해시 함수가 M 비트 길이의 해시 값을 갖는다고 하자. M 비트 길이의 해시 값이 취할 수 있는 모든 수는 2^M 개이다.

- N≒$\sqrt{Y}$=$\sqrt{2^M}$ = $2^{M/2}$(1년의 일수의 제곱근)
- N=$2^{M/2}$ 이라면 맬로리는 2분의 1의 확률로 생일 공격에 성공하게 된다.

스토리 비교

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

스토리 비교

- 스토리2
  - 맬로리가 2 개의 계약서를 만드는 것이므로 해시 값은 뭐라도 상관없다
  - 백만 원 계약서와 일억 원 계약서의 해시 값이 같기만 하면 된다.

Section 08

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

Quiz 3 앨리스의 알고리즘

- 일방향 해시 함수의 이야기를 들은 앨리스는『뭐야! 그러니까 일방향 해시함수라는 건 검사합(checksum) 같은 거구나』라고 생각하고 다음과 같은 알고리즘을 만들었다.

(1) 메시지를 처음부터 256비트씩 잘라낸다. 잘라낸 개개의 데이터를 256비트로 표현된 수라고 간주하고 모두를 더한다. 더한 결과, 256비트를 넘는 초과 자릿수는 무시한다.

(2) 모두 더한 결과를 해시값(256비트)으로 삼는다.

이것을 일방향 해시 함수로서 사용할 수 있을까?

Section 09
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

Quiz 3 일방향 해시 함수의 기초 지식

- 일방향 해시 함수에 관한 다음 문장 중 바른 것에는 O, 틀린 것에는 X를 하시오.

(1) SHA3-512는 임의 길이의 데이터를 512 비트로 변환하는 대칭 암호 알고리즘이 일종이다.

(2) 어떤 메시지의 해시 값과 같은 해시 값을 갖는 다른 메시지를 발견해 내는 것은 지극히 곤란하다.

(3) 같은 해시 값을 갖는 다른 2개의 메시지를 발견해 내는 것은 지극히 곤란하다.

(4) SHA3-512의 해시 값은 64 바이트이다.

(5) 메시지를 1비트만 변경하면 해시 값도 1비트만 변화한다.