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

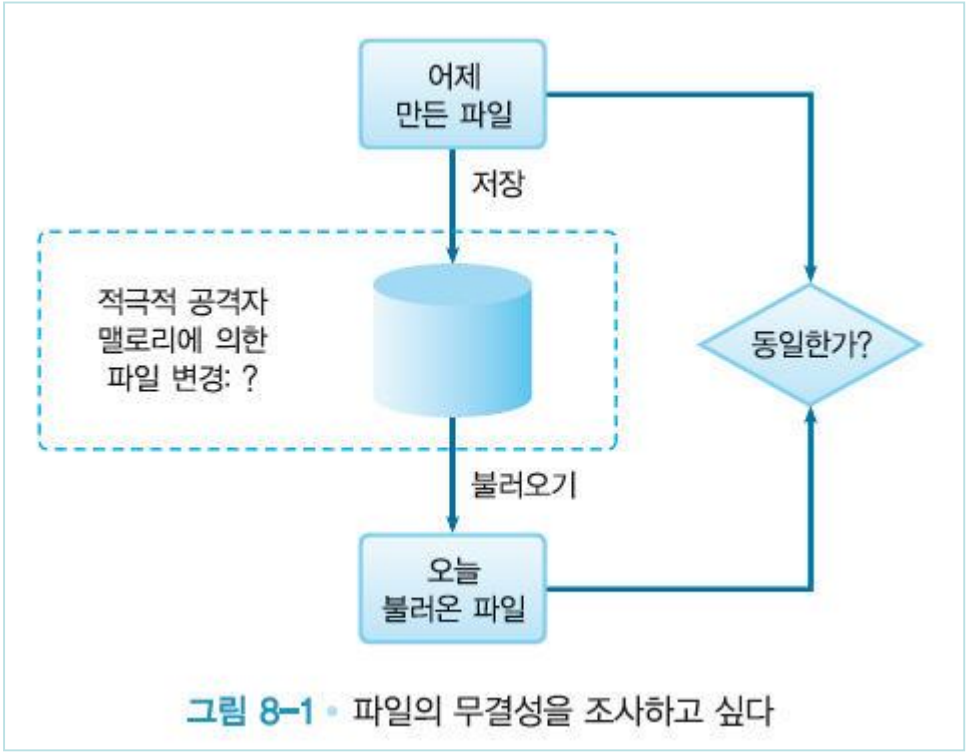

그림 8-1 • 파일의 무결성을 조사하고 싶다

파일 전체를 안전한 장소에 보존해 두고, 나중에 비교하는 방법

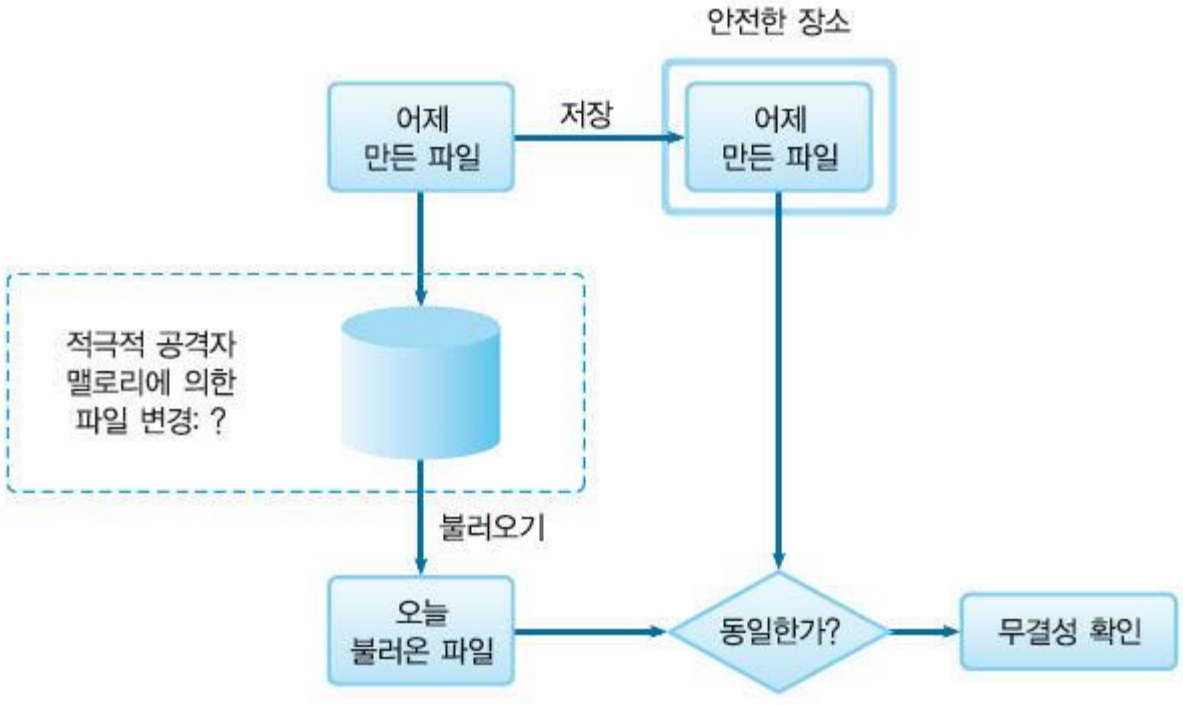

그림 8-2 • 파일 전체를 안전한 장소에 보존해두고 나중에 비교하는 방법

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

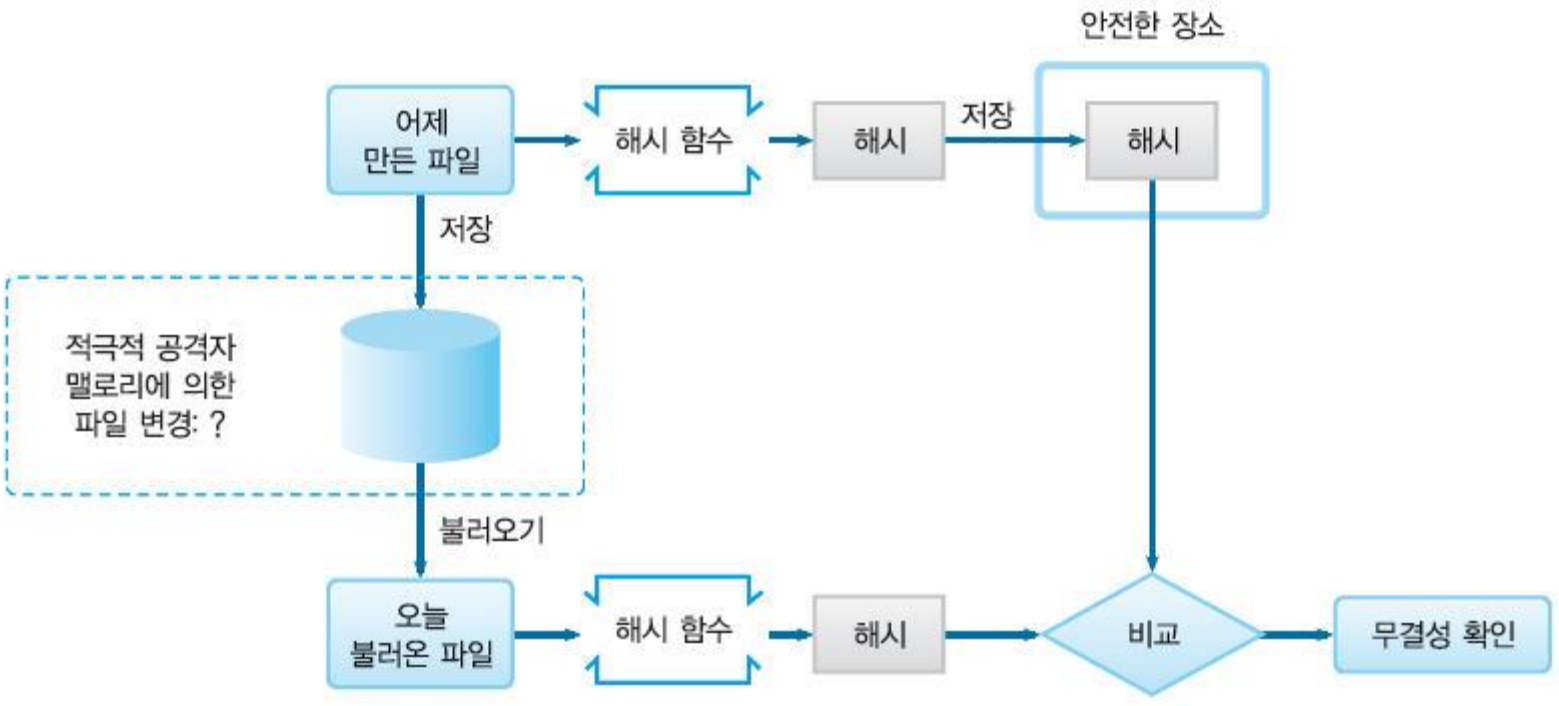

그림 8-3 • 파일을 비교하는 대신에 해시 값을 비교하는 방법

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 15029.956521739130 43…… 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

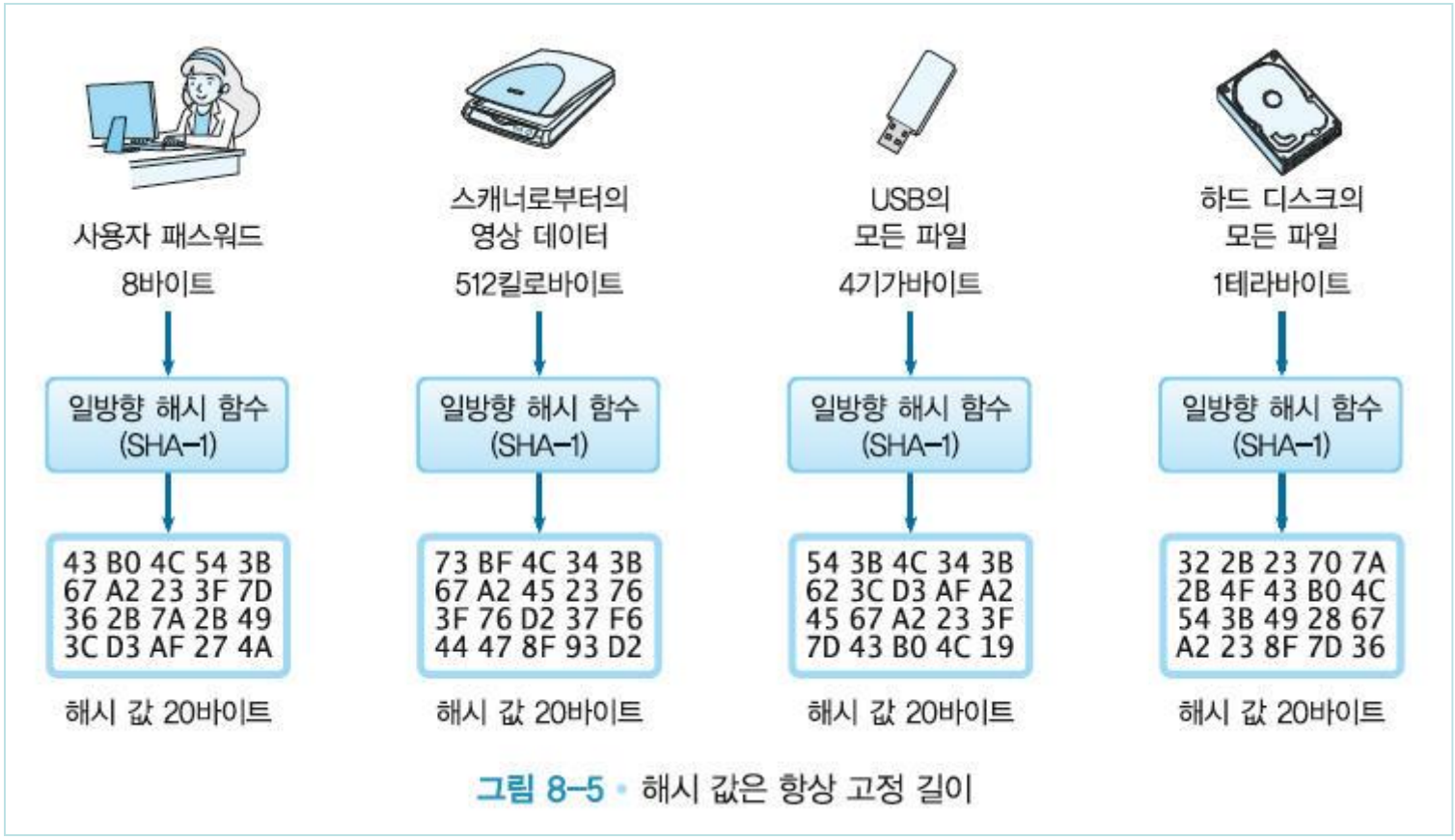

그림 8-5 • 해시 값은 항상 고정 길이

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

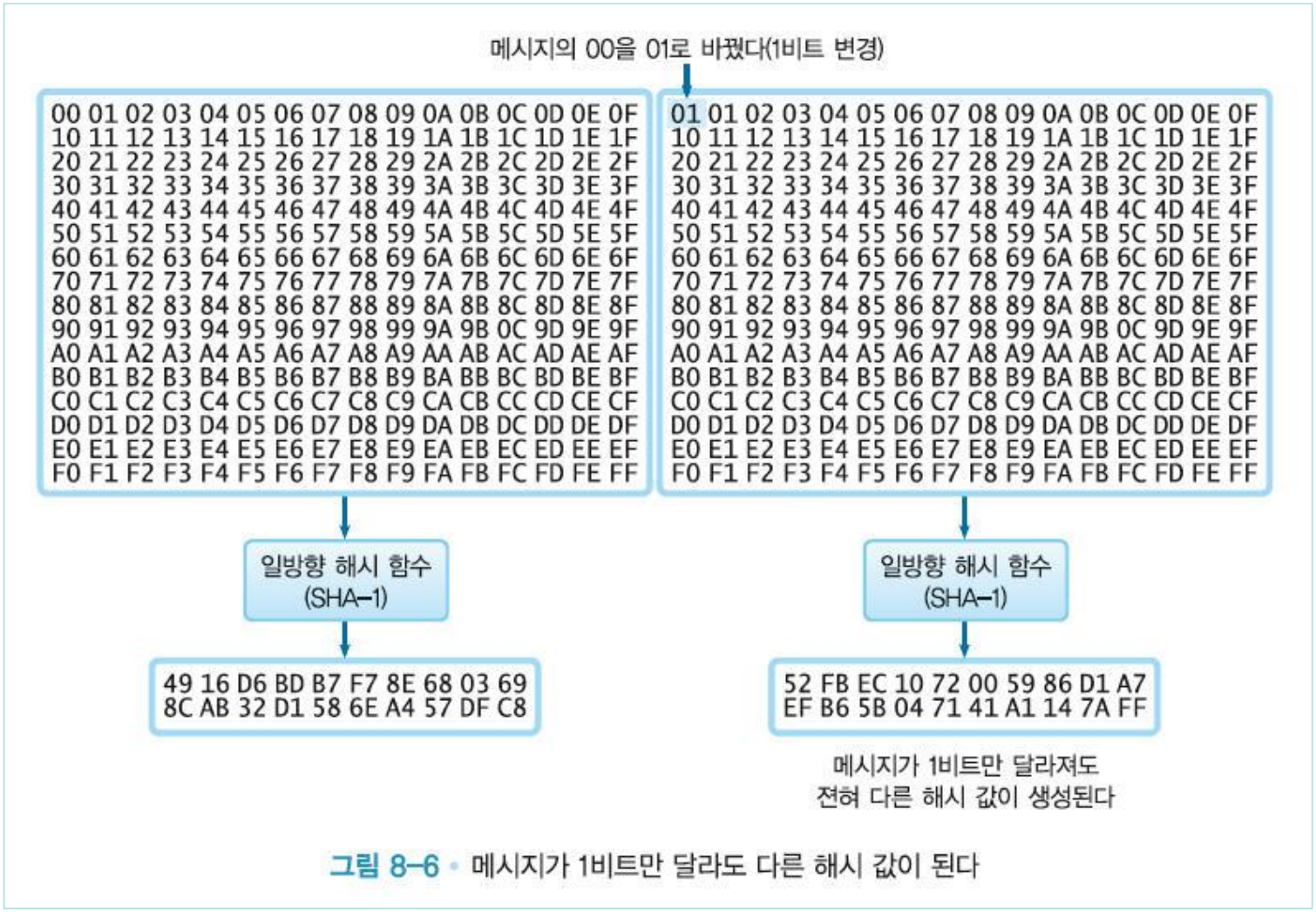

그림 8-6 • 메시지가 1비트만 달라도 다른 해시 값이 된다

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

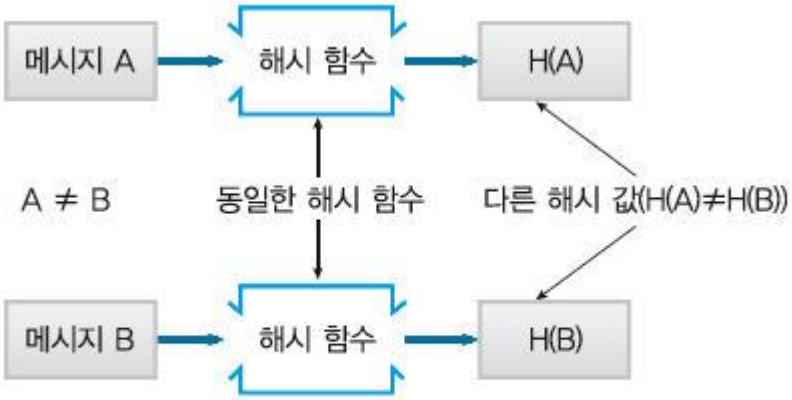

그림 8-7 • 일방향 해시 함수의 충돌 내성

충돌내성

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

- 강한 충돌 내성
  - 해시 값이 일치할 것 같은, 다른 2개의 메시지를 발견해 내는 것이 매우 곤란한 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

일방향 해시 함수의 일방향성

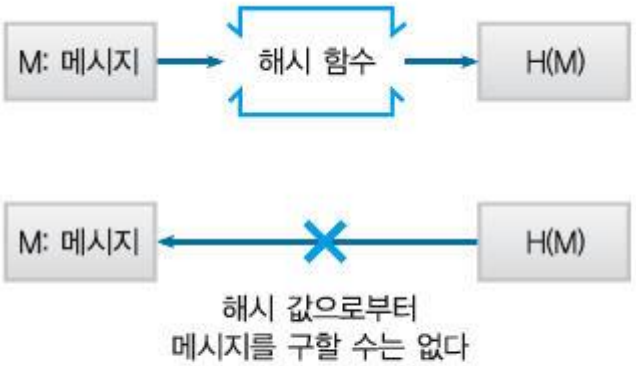

그림 8-8 • 일방향 해시 함수의 일방향성

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

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

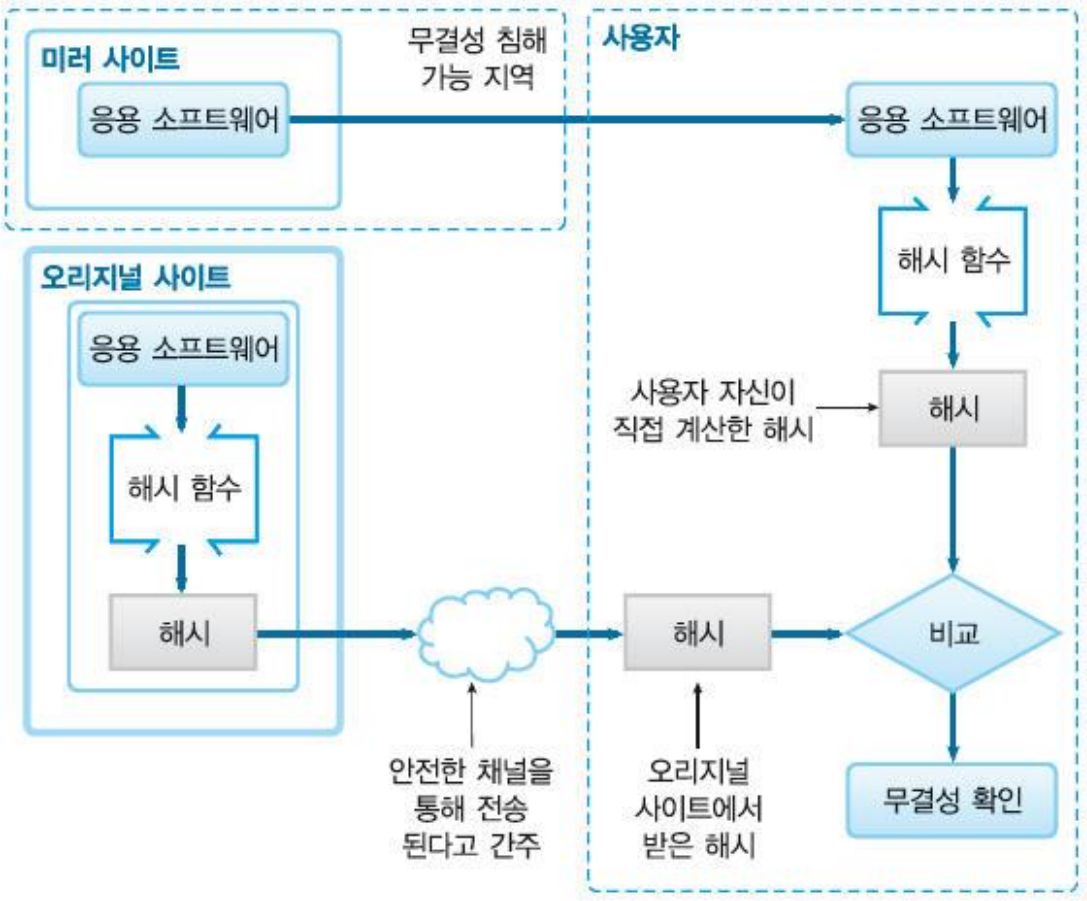

그림 8-9 • 소프트웨어 변경 검출을 위해 일방향 해시 함수를 사용한다

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 「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

- **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
  - 암호해독에 취약함을 보여주는 여러 가지 암호 해독 방법들이 개발
  - MD5가 완전히 뚫린 것은 아니지만, MD5 내부 구조의 일부에 대한 공격 방법이 몇 개 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(Advanced Hash Standard)와 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

제 4절 일방향 해시 함수 SHA-1

- 패딩
- $W_0 \sim W_{79}$ 계산
- 블록 처리
- 단계 1 처리

일방향 해시 함수 SHA-1의 개요

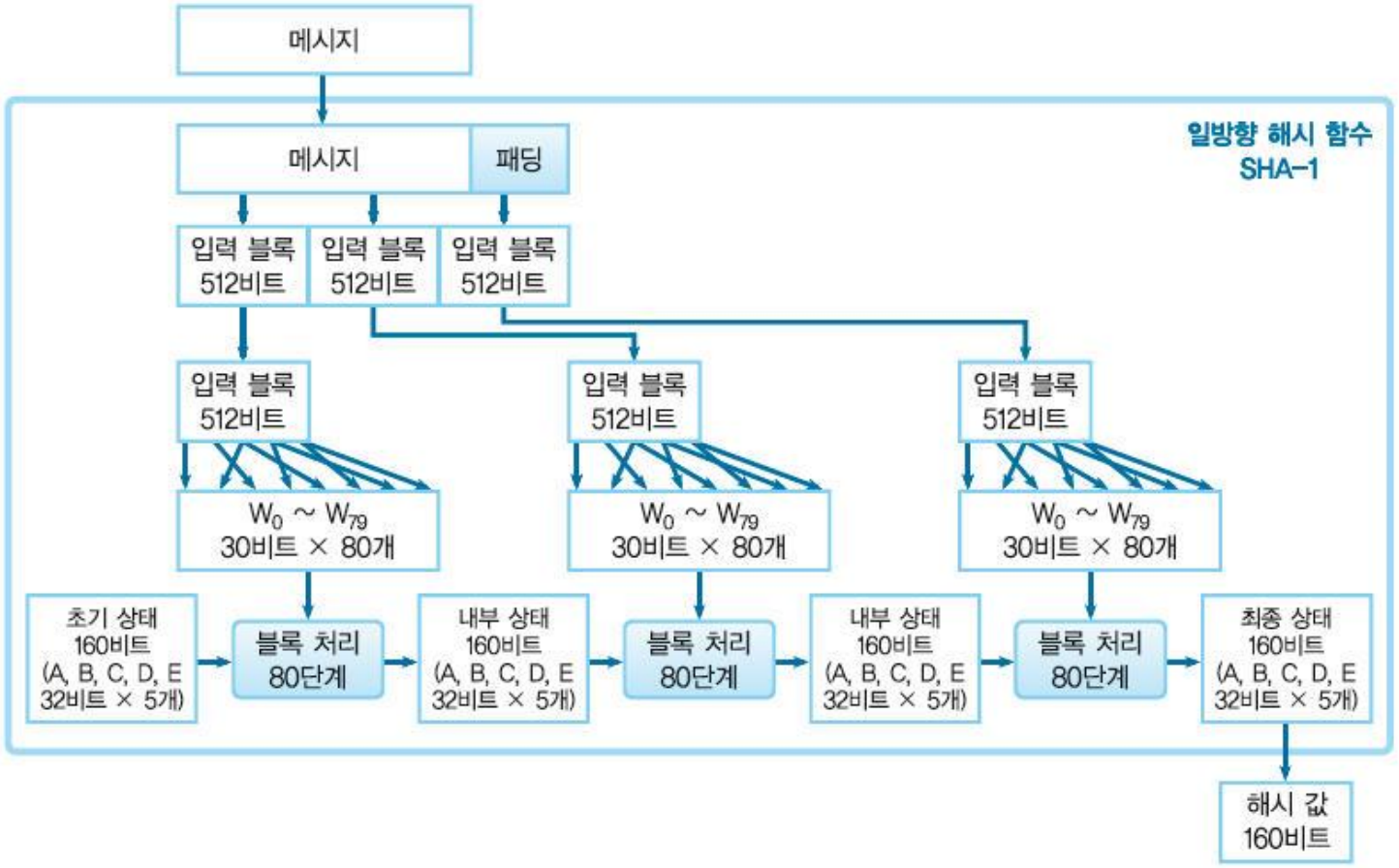

그림 8-10 • 일방향 해시 함수 SHA-1의 개요

패딩

- 메시지 뒤에 여분의 데이터를 부가하여 메시지의 길이가 512비트의 정수배가 되도록 하는 것

패딩의 예

- 입력: Hello. (6바이트(48비트))의 메시지
  - ASCII 코드로 부호화하여 2진수로 표현하면

H	e	l	l	o	.
01001000	01100101	01101100	01101100	01101111	0011110

  - 여기에 1을 붙인다

- 01001000 01100101 01101100 01101100 01101111 00101110 1

패딩의 예

- 메시지의 길이가 512비트의 정수배가 될 때까지 0이라는 1비트의 값을 부가
- 단, 마지막 블록의 마지막 64비트는 메시지 길이정보 영역으로서 비워 둔다
- “Hello.”의 경우
  - 원래 메시지 길이(48비트) 정보 저장을 위해 끝의 64비트를 뺀 나머지를 0으로 채워넣는다
    - 512–64 = 448 비트가 될 때까지 0을 부가

0의 추가

01001000 01100101 01101100 01101100 01101111 00101110 10000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

길이 정보 추가

- 데이터 길이: 48비트를 2진수로 표현하면

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00110000

- 이것을 마지막에 추가

01001000 01100101 01101100 01101100 01101111 00101110 10000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00110000

W_0 ~ W_{79}의 계산

- 입력 블록 512비트마다 32비트× 80개의 값(W_0 ~ W_{79})을 계산
- 입력 블록 512비트를 32비트×16개로 분할하여 W_0 ~ W_{15} 로 이름을 붙인다
- W_{16}부터 W_{79}는 아래와 같이 계산
  - $W_{16} = (W_0 \oplus W_2 \oplus W_8 \oplus W_{13})$을 1비트 회전
  - $W_t = (W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3})$을 1비트 회전, t=17~79

1비트 회전한 모양

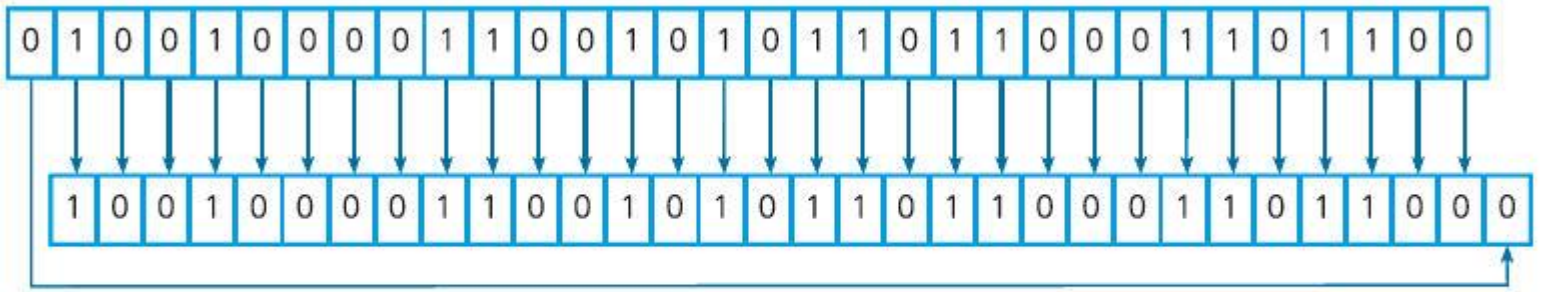

그림 8-11 • 01001000 01100101 01101100 01101100를 1비트 회전한 모양

입력 블록 512비트로 부터 80개의 32비트값(W_0 ~ W_{79})생성

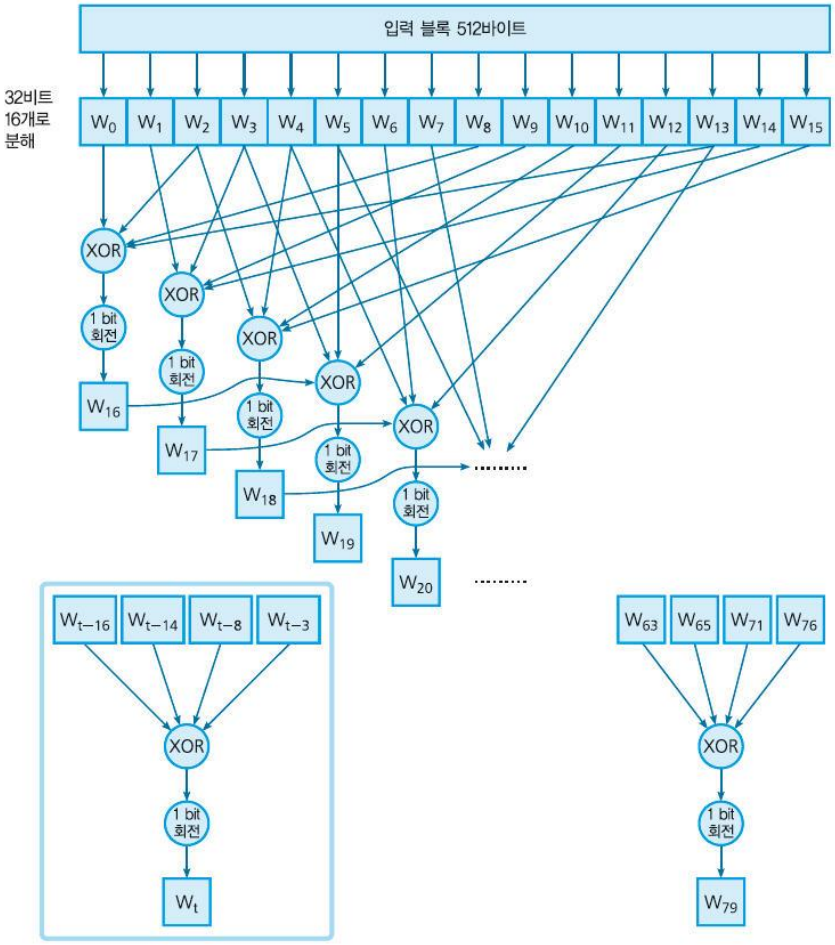

그림 8-12 • 입력 블록 512비트로부터 80개의 32비트 값(W_0 ~ W_{79})을 생성한다

블록 처리

- 입력 블록에 대해 80 단계씩의 처리를 행한다
- 입력 블록의 정보를 기초로 내부 상태(160비트)를 변화

입력 블록 512비트를 160비트의 내부 상태에 섞기(80 단계)

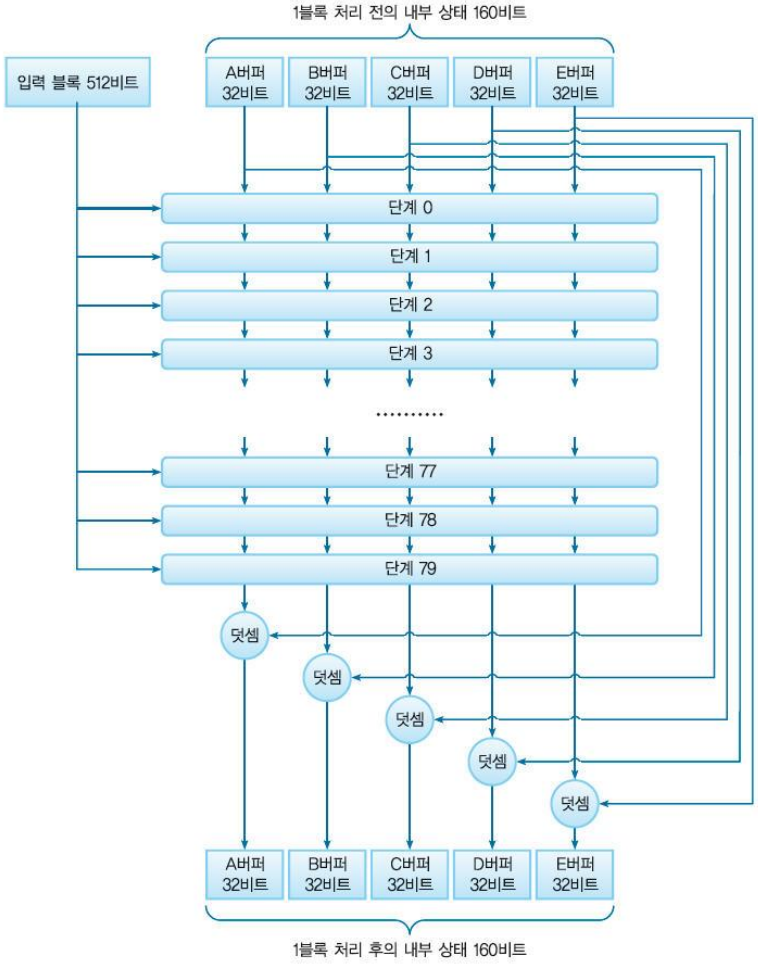

그림 8-13 • 입력 블록 512비트를 160비트의 내부 상태에 섞어 넣는다(80 단계)

1 단계 처리

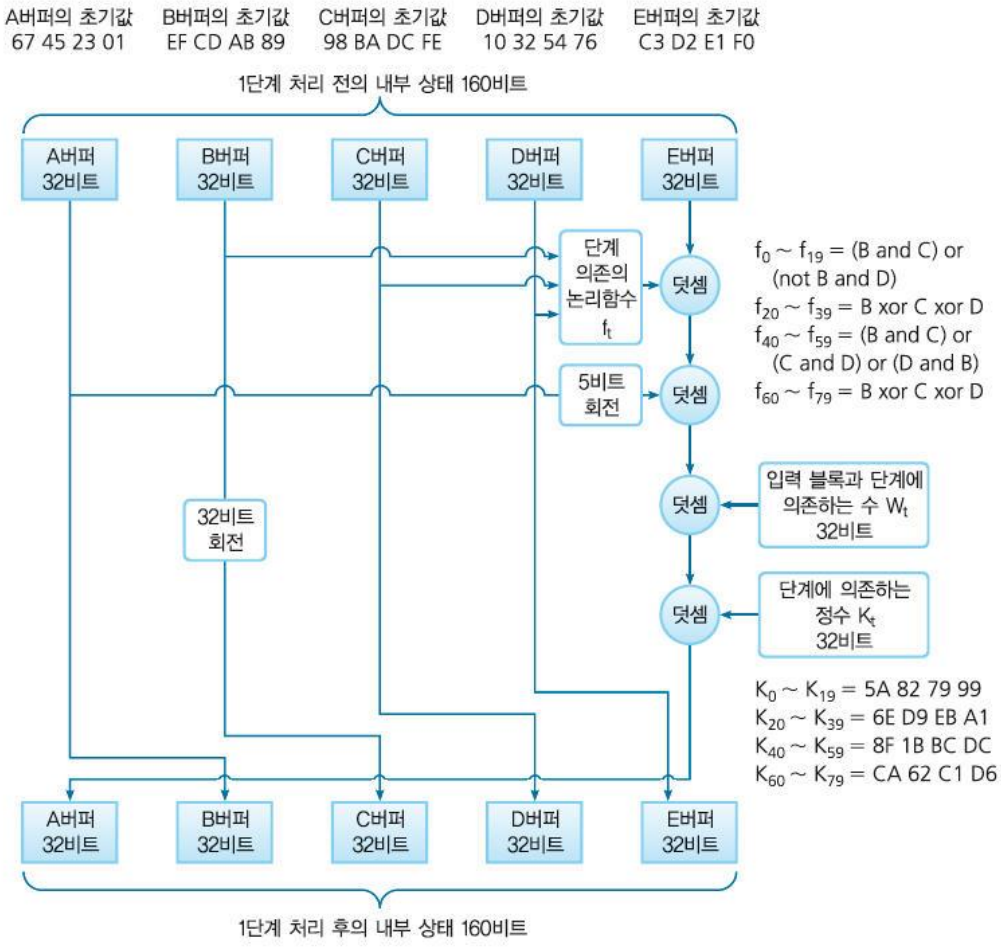

그림 8-14 • 1단계 처리

제 5절 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 처리 단계

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

SHA-512를 이용한 메시지 다이제스트 생성

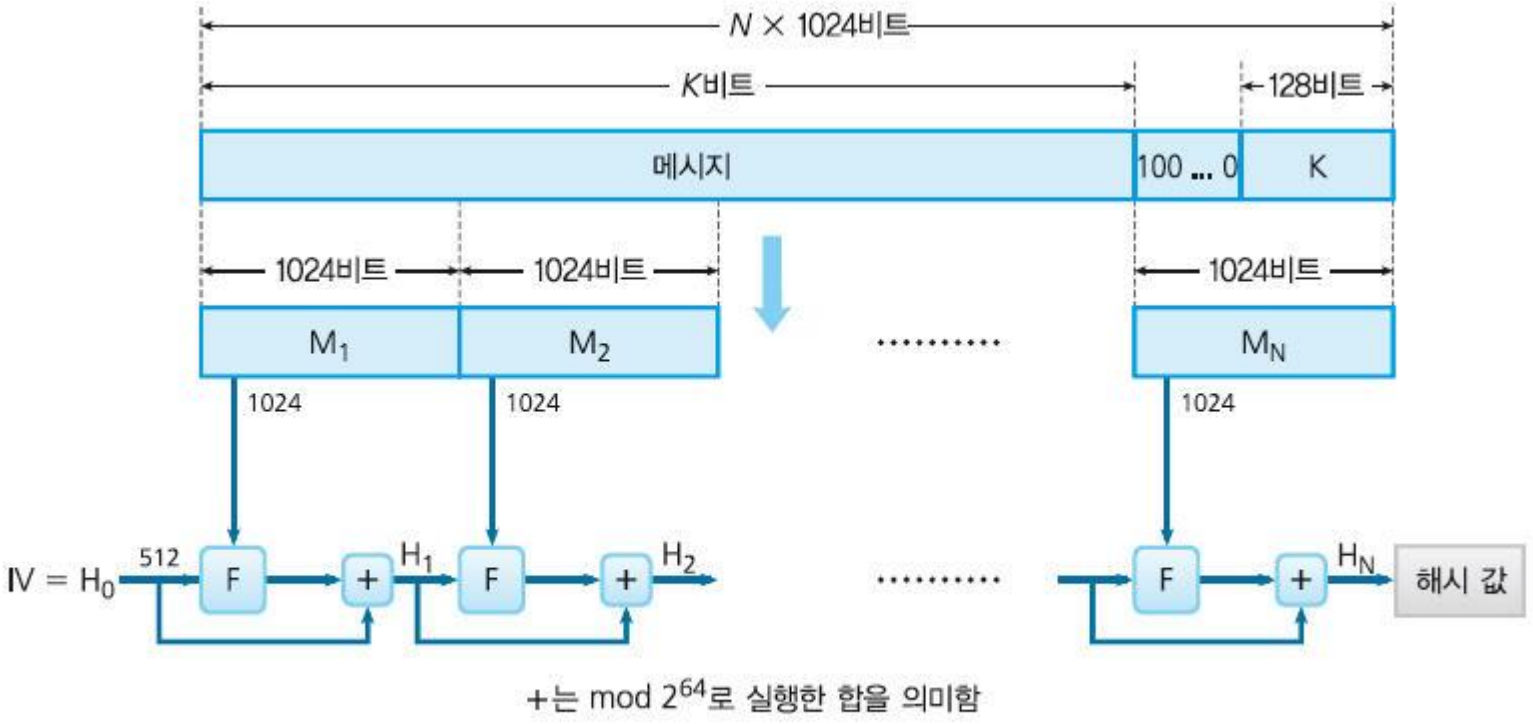

그림 8-15 • SHA-512를 사용하는 메시지 다이제스트 생성

1024비트 한 개에 대한 SHA-512 처리

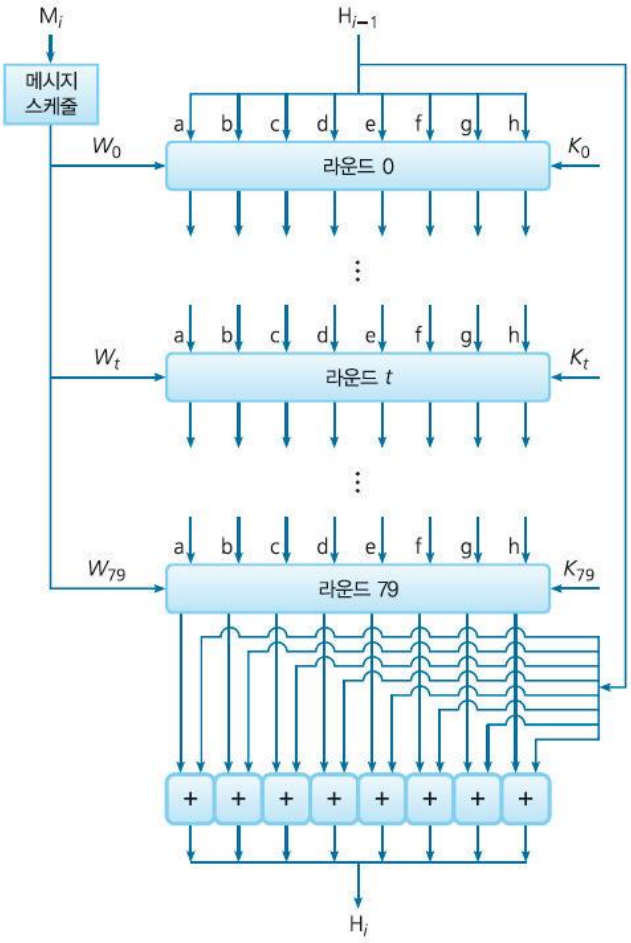

그림 8-16 • 1024비트 한 개에 대한 SHA-512 처리

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}

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

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 내장시 높은 고성능
  - 다른 최종 목록과 비교하여 보안성이 높음

KECCAK란 무엇인가?

- SHA-3으로 선정된 일방향 해시 함수 알고리즘
- 해시 값으로서 임의 길이의 비트열 생성
- SHA-2 비트 길이에 맞춰져 있음
- SHA3-224, SHA3256, 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

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

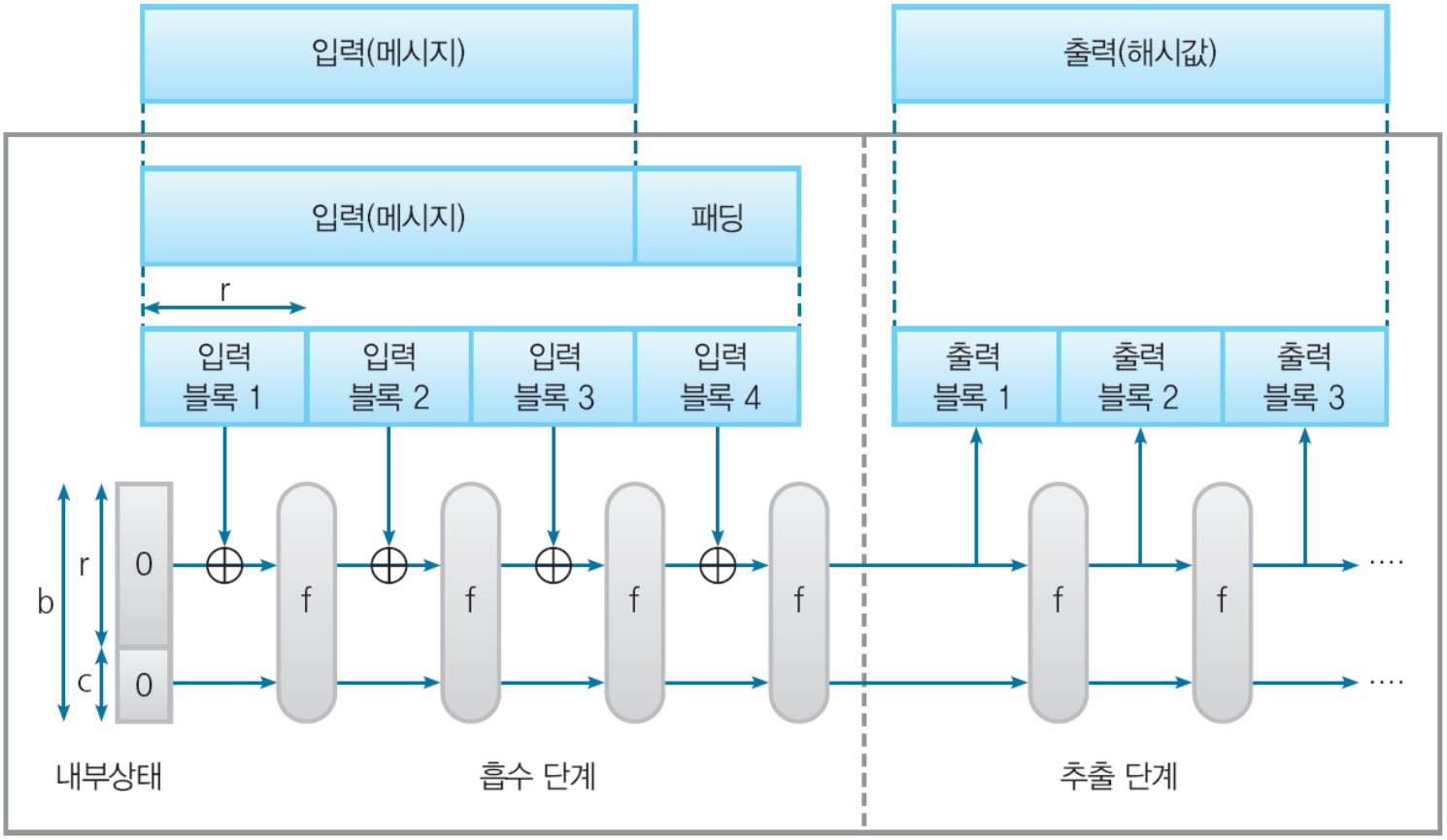

그림 8-12 • 스펀지 구조[6]

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

제 6절 일방향 해시 함수에 대한 공격

6.1 전사 공격(공격 스토리1)
6.2 전사 공격(공격 스토리2)

6.1 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

6.2 전사 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

- 「어느 특정일을 정하고 2명이 그 날 태어날」 가능성은 확실히 높지 않다. 그러나 「1년의 어느 날이라도 상관없으므로 2명이 같은 날 태어날」 가능성은 의외로 높다.

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 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 상관 없다
  - 백만 원 계약서와 일억 원 계약서의 해시 값이 같기만 하면 된다.

제 7절 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 8절 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

Q & A

Thank You!