2025/08/07, 2025 보안 기초 세미나

암호학과 네트워크 보안

-11장, 메시지 무결성과 메시지 인증-

송 민 희(23011716@sejong.ac.kr)

세종대학교 프로토콜공학연구실

목 차

- 보충
- 메시지 무결성 (Message Integrity)
- 랜덤 오라클 모델 (Random Oracle Model)
- 메시지 인증 (Message Authentication)

목 차

- 보충
- 메시지 무결성 (Message Integrity)
- 랜덤 오라클 모델 (Random Oracle Model)
- 메시지 인증 (Message Authentication)

보충

- 암호학적 해시함수 (1/5)
  - 기준 - 프리이미지 저항성(Preimage Resistance)
    - 프리이미지 저항성을 충족하지 못하는 예시

```
1.  def simple_hash(s):
2.      return sum(ord(c) for c in s) % 100
3.  M = “PEL”
4.  h_M = simple_hash(M)
```

```
5.  #프리이미지 공격
6.  target_hash = h_M
7.  found = False
8.  for a in range(69, 91):
9.      for b in range(65, 91):
10.         for c in range(65, 91):
11.             s = chr(a) + chr(b) + chr(c)
12.             if simple_hash(s) == target_hash:
13.                 print(f"  → h(M) = {target_hash} 를 만족하는 M = '{s}' 발견")
14.                 found = True
15.                 break
16.         if found:
17.             break
18.     if found:
19.         break
```

```
1. 프리이미지 공격
  → h(M) = 25 를 만족하는 M = 'AFZ' 발견
```

보충

- 암호학적 해시함수 (2/5)
  - 기준 – 제 2 프리이미지 저항성(Second Preimage Resistance) (1/2)
    - 제 2 프리이미지 저항성을 충족하지 못하는 예시 (1/2)

```
1.  def simple_hash(s):
2.      return sum(ord(c) for c in s) % 100
3.  M = “PEL”
4.  h_M = simple_hash(M)
```

```
2. 제2 프리이미지 공격
  → M = 'PEL', M′ = 'AFZ' (같은 해시 25)
```

```
5.  #제 2 프리이미지 공격
6.  original = M
7.  original_hash = h_M
8.  found = False
9.  for a in range(65, 91):
10.     for b in range(65, 91):
11.         for c in range(65, 91):
12.             s = chr(a) + chr(b) + chr(c)
13.             if s != original and simple_hash(s) == original_hash:
14.                 print(f"  → M = '{original}', M′ = '{s}' (같은 해시
    {original_hash})")
15.                 found = True
16.                 break
17.         if found:
18.             break
19.     if found:
20.         break
```

보충

- 암호학적 해시함수 (3/5)
  - 기준 – 제 2 프리이미지 저항성(Second Preimage Resistance) (2/2)
    - 제 2 프리이미지 저항성을 충족하지 못하는 예시 (2/2)
      - MD5
        - 512비트 블록 단위로 데이터를 처리하고 128비트 해시 값을 출력하는 메시지 다이제스트 함수
        - 입력 데이터가 주어졌을 때, 동일한 해시 값을 갖는 다른 입력 값을 생성할 수 있으므로 제 2 프리이미지 저항성을 충족하지 못함
      - RIPEMD-128
        - 512비트 블록으로 입력을 받고 128비트 해시 값을 출력하는 병렬 구조로 설계된 해시 함수
        - 출력 길이가 짧아 기존 해시 값과 같은 값을 만드는 다른 입력 값을 생성할 수 있으므로 제 2 프리이미지 저항성을 충족하지 못함

보충

- 암호학적 해시함수 (4/5)
  - 기준 - 충돌 저항성(Collision Resistance) (1/2)
    - 충돌 저항성을 충족하지 못하는 예시 (1/2)

```
1.  def simple_hash(s):
2.      return sum(ord(c) for c in s) % 100
3.  M = “PEL”
4.  h_M = simple_hash(M)
```

```
5.  #충돌 공격
6.  hash_dict = {}
7.  found = False
8.  for a in range(65, 91):
9.      for b in range(65, 91):
10.         for c in range(65, 91):
11.             s = chr(a) + chr(b) + chr(c)
12.             h = simple_hash(s)
13.             if h in hash_dict:
14.                 print(f"  → 충돌 발견! M = '{hash_dict[h]}', M′ =
    '{s}', 해시값 = {h}")
15.                 found = True
16.                 break
17.             else:
18.                 hash_dict[h] = s
19.         if found:
20.             break
21.     if found:
22.         break
```

```
3. 충돌 공격
  → 충돌 발견! M = 'AAB', M′ = 'ABA', 해시값 = 96
```

보충

- 암호학적 해시함수 (5/5)
  - 기준 - 충돌 저항성(Collision Resistance) (2/2)
    - 충돌 저항성을 충족하지 못하는 예시 (2/2)
      - SHA-1
        - 512비트 블록 단위로 메시지를 처리하고, 160 비트 해시 값을 출력하는 해시 함수
        - 짧은 해시 출력 길이와 구조적 결함으로 인해 서로 다른 두 입력이 같은 해시를 가질 수 있으므로 충돌 저항성을 충족하지 못함
      - HAVAL-128
        - 128 비트 해시 값을 출력하는 메시지 압축 기반 해시 함수
        - 설계 복잡도에 비해 출력 길이가 짧고 일부 라운드 설정에서 충돌 쌍이 계산적으로 쉽게 생성되므로 충돌 저항성을 충족하지 못함

보충

- 랜덤 오라클 모델(Random Oracle Model)
  - 예제 11.2

오라클이 주어진 메시지로부터 다이제스트 생성에 공식 $h(M) = M \ mod \ n$을 사용한다고 해보자.

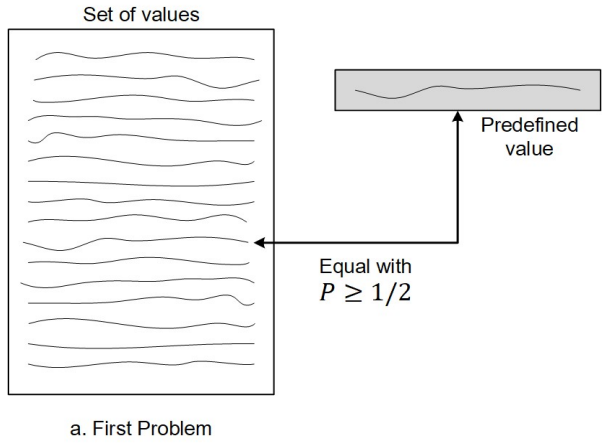
- 상황
  - 오라클에서 이미 $h(M_1)$과 $h(M_2)$를 생성해 둠
  - 새로운 메시지 M_3는 $M_3 = M_1 + M_2$임
- 결과
  - M_3에 대한 새로운 다이제스트는 $[h(M_1) + h(M_2)] \ mod \ n$이 됨

$$h(M_3) = M_3 \ mod \ n = (M_1 + M_2) mod \ n = M_1 \ mod \ n + M_2 \ mod \ n$$
$$= h(M_1) + h(M_2)$$

  - 각 다이제스트는 오라클에 입력되는 메시지에 대해 랜덤하게 선택 되어야 한다는 세 번째 요구조건에 위배됨

보충

- 생일 문제(Birthday Problems)
  - 첫 번째 문제
    - 교실 안에 학생이 k명 있을 때 적어도 한 학생이 특정 생일날일 확률 P가½보다 크거나 같으려면 k의 최소값이 무엇인가?
  - 첫 번째 문제 - 일반화
    - N개의 값(0부터 $N-1$)에 균등하게 분포된 *확률 변수가 있다. 적어도 한 개의 확률 변수가 특정한 값(0과 $N-1$ 사이의 미리 지정된 값)을 가질 확률이 ½ 이상이 되려면 최소한 몇 개의 확률 변수가 필요한가?



a. First Problem

* 확률 변수
확률적인 결과에 따라 결과 값이 바뀌는 변수

보충

- 랜덤 오라클 모델에 대한 공격 (1/4)
  - 제 2 프리이미지 공격
    - 두 번째 생일문제 활용
      - 알고리즘 성공 확률은 $P \approx 1 - e^{-\frac{k-1}{N}}$
      - 50%의 성공을 위한 k의 크기는 $k \approx 0.69 \times N + 1$ 또는 $k \approx 0.69 \times 2^n + 1$
        $\therefore$ 2^n에 비례하는 개수의 다이제스트를 생성해야 함
      - 제 2 프리이미지 공격의 어려움 정도는 2^n에 비례함
    - 예제 11.5

암호학적 해시함수는 64비트 다이제스트를 사용한다.
공격자는 원래의 메시지를 찾을 확률이 0.5 이상이 되도록 하려면 얼마나 많은 다이제스트를 생성해야하는가?

- 생성해야 할 다이제스트의 수는 $k \approx 0.69 \times 2^n + 1 \approx 0.69 \times 2^{64} + 1$
- 공격자가 초당 $2^{30}(\approx 1{,}000{,}000{,}000)$개의 다이제스트를 생성해도 $1 + 0.69 \times 2^{34}$초$(\approx 500$년$)$가 필요함
- 64비트의 다이제스트는 제 2 프리이미지 공격에 대해 안전함

보충

- 랜덤 오라클 모델에 대한 공격 (2/4)
  - 또 다른 충돌 공격
    - 네 번째 생일문제 활용
      - 알고리즘 성공 확률은 $P \approx 1 - e^{-\frac{k^2}{N}}$
      - 50%의 성공을 위한 k의 크기는 $k \approx 0.83 \times N^{\frac{1}{2}}$ 또는 $k \approx 0.83 \times 2^{\frac{n}{2}}$
        $\therefore$ $2^{\frac{n}{2}}$에 비례하는 개수의 다이제스트를 생성해야 함
      - 또 다른 충돌 공격의 어려움 정도는 $2^{\frac{n}{2}}$에 비례함
    - 예제 11.7

한 암호학적 해시함수가 64비트 다이제스트를 사용한다고 하자.
공격자가 동일한 다이제스트를 갖는 서로 다른 두 개의 메시지를 만들 수 있는 확률이 0.5이상 되려면 얼마나 많은 다이제스트가 필요한가?

- 생성해야 할 다이제스트의 수는 $k \approx 0.83 \times 2^{\frac{n}{2}} \approx 0.83 \times 2^{32}$
- 공격자가 초당 2^{20}개의 메시지를 점검할 수 있다면, 0.83×2^{12}초 혹은 1시간 이내로 끝낼 수 있음
- 길이가 64비트인 메시지 다이제스트는 또 다른 충돌 공격에 안전하지 못함

보충

- 랜덤 오라클 모델에 대한 공격 (3/4)
  - 충돌 저항성의 중요성 (1/2)
    - e.g., 256비트 다이제스트를 갖는 SHA-256(Secure Hash Algorithm 256)
      - 프리이미지와 충돌에 대해 모두 저항성이 높음
        - Merkle–Damgård 구조를 기반으로 SHA-1과 다른 개선된 압축함수를 사용함
      - 공격자는 확률 1/2 이상을 가지고 충돌 공격을 수행하려면 $2^{\frac{256}{2}} = 2^{128}$번의 점검을 수행해야 함
      - 공격자가 초당 2^{30}번의 점검을 할 수 있다면, 2^{98}초(수억 년 이상)가 걸림
      - 현실적으로 공격 불가능에 가까운 수준임

보충

- 랜덤 오라클 모델에 대한 공격 (4/4)
  - 충돌 저항성의 중요성 (2/2)
    - e.g., 512비트 다이제스트를 갖는 SHA-512 (Secure Hash Algorithm 512)
      - 프리이미지와 충돌에 대해 모두 저항성이 높음
        - Merkle–Damgård 구조에 Davies-Meyer 구조를 추가하여 설계를 보완함
      - 공격자는 확률 1/2 이상을 가지고 충돌 공격을 수행하려면 $2^{\frac{512}{2}} = 2^{256}$번의 점검을 수행해야 함
      - 공격자가 초당 2^{30}번의 점검을 할 수 있다면, 2^{226}초 (138억년 이상)가 걸림
      - 현실적으로 공격 불가능에 가까운 수준임

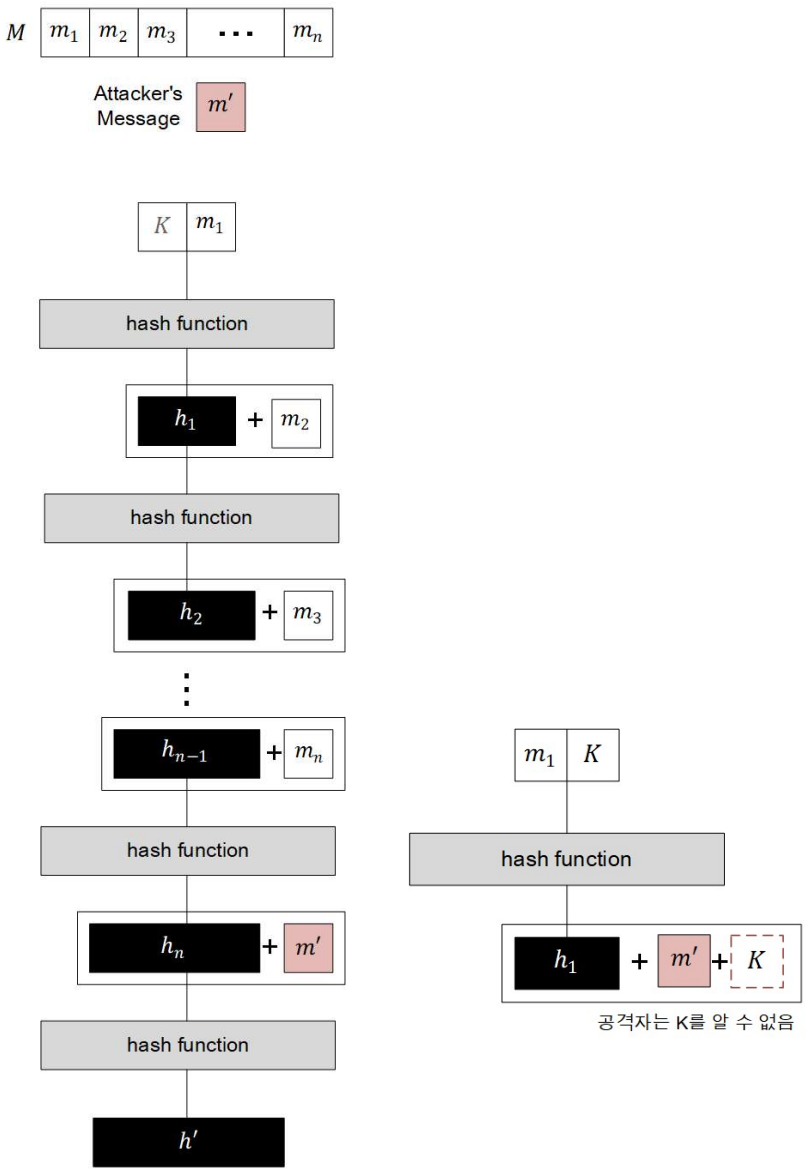
* Davies-Meyer 구조
블록 암호를 이용해 이전 해시 값과 메시지 블록을 결합하여 새로운 해시값을 생성하는 해시 함수의 설계 방식

보충

- 구조에 대한 공격
  - 이상적인 암호학적 해시함수
    - 랜덤 오라클 모델과 같은 해시함수는 내부구조가 전혀 없음
  - 실제 해시 함수
    - 취약한 내부구조를 가지고 있는 경우가 있음
      - e.g., MD5(충돌 생성 가능), SHA-0(논리적 회전 연산 누락)
    - 완벽하게 랜덤한 다이제스트를 생성하는 해시함수를 만드는 것은 현실적으로 불가능 함
    - 공격자가 해시함수를 공격하는 다른 도구를 가지고 있는 경우가 있음
      - e.g., 중간자 공격(MITM, Man-in-Middle Attack)

보충

- 메시지 인증 코드(MAC)
  - 키의 위치
    - 프리픽스 MAC(Prefix-MAC)
      - 구조
        - MAC $= h(K|M)$
      - 특징
        - 구현이 단순함
        - 길이 확장 공격에 취약함
    - 포스트픽스 MAC(Postfix-MAC)
      - 구조
        - MAC $= h(M|K)$
      - 특징
        - 프리픽스 MAC보다 길이 확장 공격에 덜 취약함



보충

- 메시지 인증 코드(MAC)
  - CMAC(Cipher-based MAC)
    - 정의
      - 블록 암호 알고리즘과 대칭 키를 이용하여 메시지의 무결성과 인증을 보장하는 MAC 알고리즘
    - 특징
      - 암호블럭체인(CBC, Cipher Block Chaining) 모드와 유사한 방법으로, CBCMAC이라고도 부름
      - 사용된 암호 알고리즘이 안전할 경우, CMAC도 동일한 수준의 보안성을 가짐
      - 입력 메시지가 블록 단위로 처리되며, 마지막 블록의 패딩 여부에 따라 다른 키가 적용됨
      - 입력 메시지의 길이에 따라 서브키를 적용하므로 메시지 확장으로 동일한 MAC을 구성할 수 없어 길이 확장 공격에 안전함
      - FIPS 113 및 NIST SP 800-38B에 따라 표준화되어 있음

*비밀 키 (Secret Key)
소유자가 비공기로 유지해야 하는 키

*대칭 키 (Symmetric Key)
암호화와 복호화에 동일한 키를 사용하는 방식에서 쓰이는 키

목 차

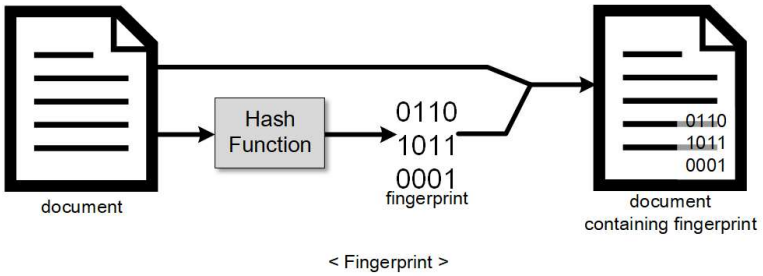
- 보충
- 메시지 무결성 (Message Integrity)
- 랜덤 오라클 모델 (Random Oracle Model)
- 메시지 인증 (Message Authentication)

메시지 무결성

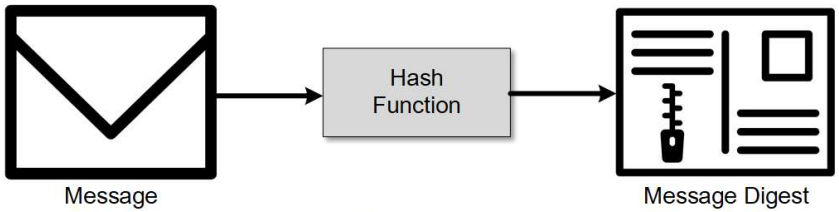
- 무결성(Integrity)
  - 정의
    - 정보가 인가되지 않은 방식으로 변경되거나 손상되지 않도록 보장하는 것
- 메시지 무결성(Message Integrity)
  - 정의
    - 메시지가 전송 중 변경되거나 손상되지 않도록 보장하는 것
  - 메시지 전송에서 무결성이 중요한 경우
    - 메시지의 내용이 변조되어서는 안되는 상황
      - e.g., 유언장, 소프트웨어 업데이트 파일 전송

메시지 무결성

- 무결성 보존 방법 (1/3)
  - 핑거프린트(Fingerprint)
    - 문서의 무결성 검증 방법
    - 문서를 기반으로 생성된 해시 값
  - 메시지 다이제스트(Message Digest)
    - 메시지의 무결성 검증 방법
    - 메시지에 암호학적 해시함수(Cryptographic Hash Function)를 적용하여 생성된 메시지의 압축 이미지



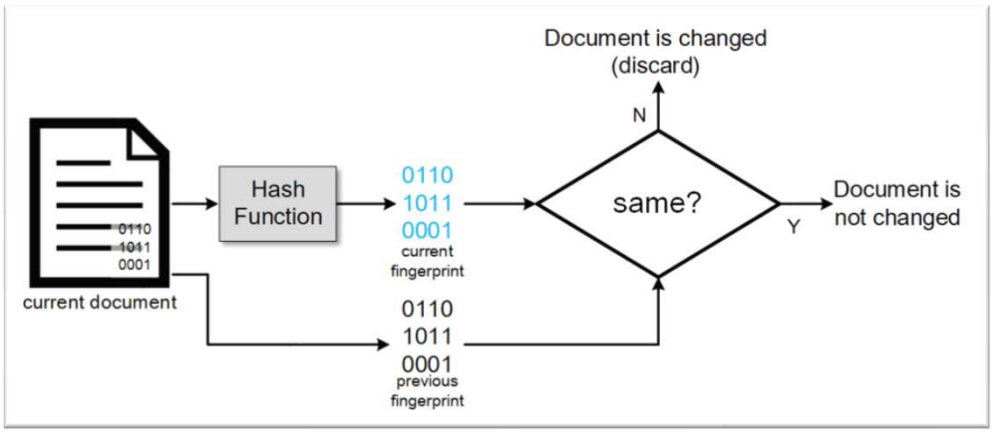
< Fingerprint >



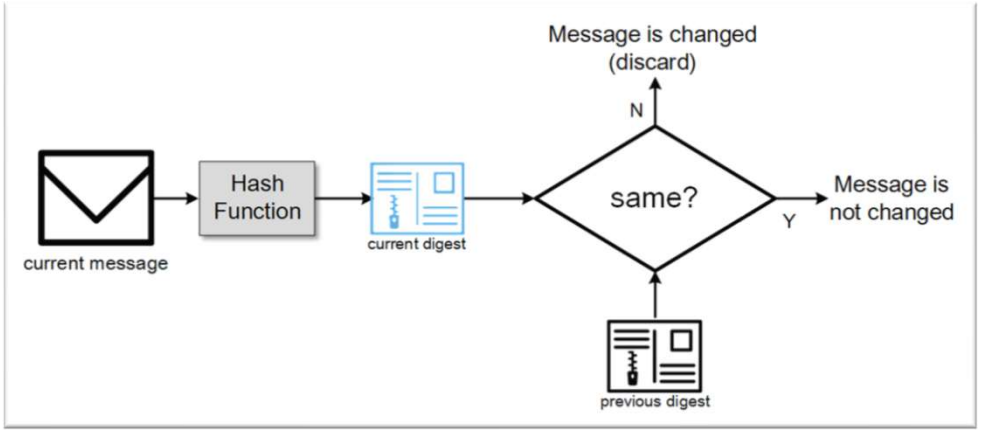
< Message Digest >

메시지 무결성

- 무결성 보존 방법 (2/3)
  - 무결성 검증
    - 기존의 핑거프린트/메시지 다이제스트와 검증을 위해 암호학적 해시함수로 생성된 새로운 핑거프린트/메시지 다이제스트를 비교
      - 두 값이 일치하면 무결성 보존
      - 두 값이 일치하지 않으면 무결성 훼손



< Fingerprint >



< Message Digest >

메시지 무결성

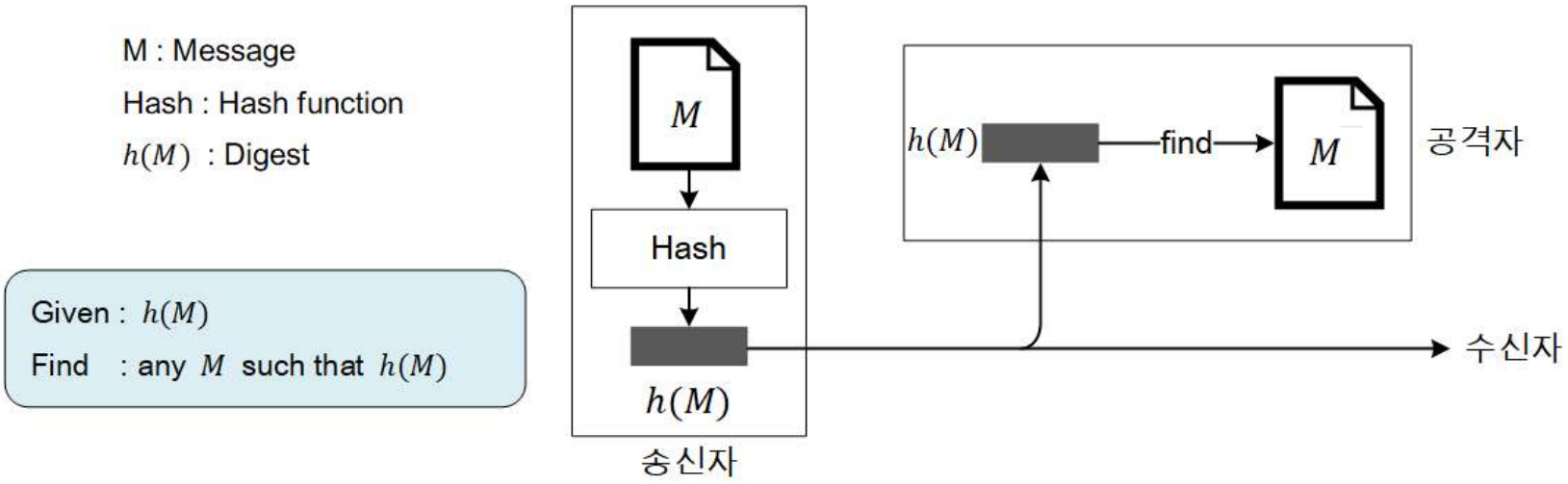
- 무결성 보존 방법 (3/3)
  - 문서와 메시지의 차이
    - 문서(Document)
      - 정보의 기록과 보관을 목적으로 하는 정적인 정보 단위
    - 메시지(Message)
      - 정보의 송수신을 목적으로 하는 동적인 정보 단위
  - 문서의 핑거프린트와 메시지 다이제스트의 차이
    - 문서의 핑거프린트
      - 문서 내부에 삽입되어 있음
    - 메시지 다이제스트
      - 메시지와 물리적으로 분리되어 있음

메시지 무결성

- 암호학적 해시함수(Cryptographic Hash Function) (1/5)
  - 정의
    - 임의 길이의 입력 데이터를 받아 고정 길이의 해시 값으로 변환하며, 프리이미지 저항성, 제 2 프리이미지 저항성, 충돌 저항성의 기준을 충족하는 해시함수
  - 기준
    - 프리이미지 저항성(Preimage Resistance)
    - 제 2 프리이미지 저항성(Second Preimage Resistance)
    - 충돌 저항성(Collision Resistance)

메시지 무결성

- 암호학적 해시함수 (2/5)
  - 기준 - 프리이미지 저항성(Preimage Resistance) (1/2)
    - 정의
      - 어떤 메시지 M에 대한 해시 값 $h(M)$만을 가지고 메시지 M을 찾기 어렵게 하는 성질
    - 프리이미지 공격
      - 주어진 정보 : $h(M)$
      - 구하려고 하는 것 : $h(M)$을 만족하는 M



메시지 무결성

- 암호학적 해시함수 (3/5)
  - 기준 - 프리이미지 저항성(Preimage Resistance) (2/2)
    - 프리이미지 저항성을 충족하지 못하는 예시

```
1.  def simple_hash(s):
2.      return sum(ord(c) for c in s) % 100
3.  M = “PEL”
4.  h_M = simple_hash(M)
```

```
5.  #프리이미지 공격
6.  target_hash = h_M
7.  found = False
8.  for a in range(69, 91):
9.      for b in range(65, 91):
10.         for c in range(65, 91):
11.             s = chr(a) + chr(b) + chr(c)
12.             if simple_hash(s) == target_hash:
13.                 print(f"  → h(M) = {target_hash} 를 만족하는 M = '{s}' 발견")
14.                 found = True
15.                 break
16.         if found:
17.             break
18.     if found:
19.         break
```

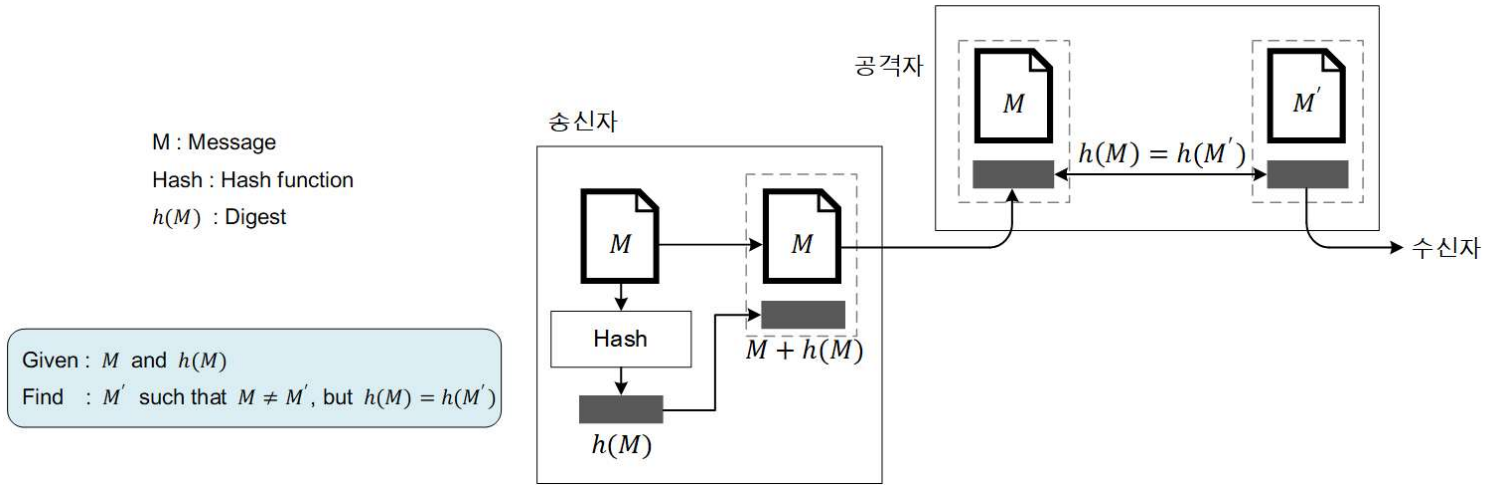
```
1. 프리이미지 공격
  → h(M) = 25 를 만족하는 M = 'AFZ' 발견
```

메시지 무결성

- 암호학적 해시함수 (3/5)
  - 기준 - 프리이미지 저항성(Preimage Resistance) (2/2)
    - 프리이미지 저항성을 충족하지 못하는 예시
      - checksum 함수
        - 데이터를 단순 수학 연산으로 요약한 값
        - 주어진 데이터의 checksum 값과 동일한 checksum 값을 갖는 여러 개의 다른 메시지를 만들 수 있으므로 프리이미지 저항성을 충족하지 못함
      - StuffIt
        - Smith Micro Software에서 개발한 무손실 파일 압축 프로그램
        - 내부 무결성 검사에 checksum, CRC(Cyclic Redundancy Check)기반 방식을 사용하므로 동일한 결과를 만드는 입력을 쉽게 조작하거나 유추 할 수 있어 프리이미지 저항성을 충족하지 못함

메시지 무결성

- 암호학적 해시함수 (4/5)
  - 기존 – 제 2 프리이미지 저항성(Second Preimage Resistance)
    - 정의
      - 어떤 메시지 M과 해시 값$h(M)$이 있을 때, $h(M) = h(M')$을 만족하는 M'을 찾기 어렵게 하는 성질
    - 제 2 프리이미지 공격
      - 주어진 정보 : M과 $h(M)$
      - 구하려고 하는 것 : $h(M) = h(M')$을 만족하는 $M'(\neq M)$



메시지 무결성

- 암호학적 해시함수 (4/5)
  - 기준 – 제 2 프리이미지 저항성(Second Preimage Resistance)
    - 제 2 프리이미지 저항성을 충족하지 못하는 예시

```
1.  def simple_hash(s):
2.      return sum(ord(c) for c in s) % 100
3.  M = “PEL”
4.  h_M = simple_hash(M)
```

```
2. 제2 프리이미지 공격
  → M = 'PEL', M′ = 'AFZ' (같은 해시 25)
```

```
5.  #제 2 프리이미지 공격
6.  original = M
7.  original_hash = h_M
8.  found = False
9.  for a in range(65, 91):
10.     for b in range(65, 91):
11.         for c in range(65, 91):
12.             s = chr(a) + chr(b) + chr(c)
13.             if s != original and simple_hash(s) == original_hash:
14.                 print(f"  → M = '{original}', M′ = '{s}' (같은 해시
    {original_hash})")
15.                 found = True
16.                 break
17.         if found:
18.             break
19.     if found:
20.         break
```

메시지 무결성

- 암호학적 해시함수 (4/5)
  - 기준 – 제 2 프리이미지 저항성(Second Preimage Resistance)
    - 제 2 프리이미지 저항성을 충족하지 못하는 예시
      - MD5
        - 512비트 블록 단위로 데이터를 처리하고 128비트 해시 값을 출력하는 메시지 다이제스트 함수
        - 입력 데이터가 주어졌을 때, 동일한 해시 값을 갖는 다른 입력 값을 생성할 수 있으므로 제 2 프리이미지 저항성을 충족하지 못함
      - RIPEMD-128
        - 512비트 블록으로 입력을 받고 128비트 해시 값을 출력하는 병렬 구조로 설계된 해시 함수
        - 출력 길이가 짧아 기존 해시 값과 같은 값을 만드는 다른 입력 값을 생성할 수 있으므로 제 2 프리이미지 저항성을 충족하지 못함

메시지 무결성

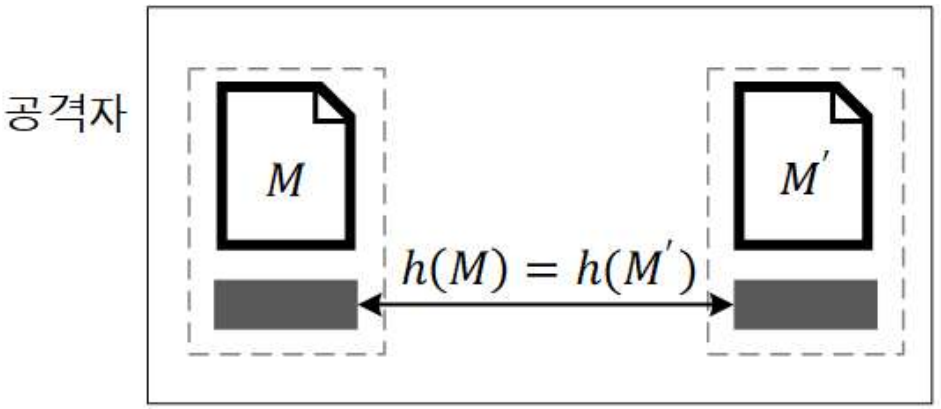
- 암호학적 해시함수 (5/5)
  - 기준 – 충돌 저항성(Collision Resistance)
    - 정의
      - 서로 다른 두 메시지 M, M'에 대해서 $h(M) = h(M')$을 만족하는 임의의 2개의 입력 값(M, M')을 구하지 못하는 성질
    - 충돌 공격
      - 구하려고 하는 것 : $h(M) = h(M')$을 만족하는 쌍(M, M') (단, $M \neq M'$)

M : Message

Hash : Hash function

$h(M)$: Digest

Find : M and M' such that $M \neq M'$, but $h(M) = h(M')$



메시지 무결성

- 암호학적 해시함수 (3/5)
  - 기준 - 충돌 저항성(Collision Resistance)
    - 충돌 저항성을 충족하지 못하는 예시

```
1.  def simple_hash(s):
2.      return sum(ord(c) for c in s) % 100
3.  M = “PEL”
4.  h_M = simple_hash(M)
```

```
5.  #충돌 공격
6.  hash_dict = {}
7.  found = False
8.  for a in range(65, 91):
9.      for b in range(65, 91):
10.         for c in range(65, 91):
11.             s = chr(a) + chr(b) + chr(c)
12.             h = simple_hash(s)
13.             if h in hash_dict:
14.                 print(f"  → 충돌 발견! M = '{hash_dict[h]}', M′ =
    '{s}', 해시값 = {h}")
15.                 found = True
16.                 break
17.             else:
18.                 hash_dict[h] = s
19.         if found:
20.             break
21.     if found:
22.         break
```

```
3. 충돌 공격
  → 충돌 발견! M = 'AAB', M′ = 'ABA', 해시값 = 96
```

메시지 무결성

- 암호학적 해시함수 (3/5)
  - 기준 - 충돌 저항성(Collision Resistance)
    - 충돌 저항성을 충족하지 못하는 예시
      - SHA-1
        - 512비트 블록 단위로 메시지를 처리하고, 160 비트 해시 값을 출력하는 해시 함수
        - 짧은 해시 출력 길이와 구조적 결함으로 인해 서로 다른 두 입력이 같은 해시를 가질 수 있으므로 충돌 저항성을 충족하지 못함
      - HAVAL-128
        - 128 비트 해시 값을 출력하는 메시지 압축 기반 해시 함수
        - 설계 복잡도에 비해 출력 길이가 짧고 일부 라운드 설정에서 충돌 쌍이 계산적으로 쉽게 생성되므로 충돌 저항성을 충족하지 못함

목 차

- 보충
- 메시지 무결성 (Message Integrity)
- 랜덤 오라클 모델 (Random Oracle Model)
- 메시지 인증 (Message Authentication)

랜덤 오라클 모델

- 랜덤 오라클 모델(Random Oracle Model) (1/5)
  - 정의
    - 해시 함수를 완전히 무작위로 작동하는 함수로 가정하는 이상적인 이론적 암호 모델
  - 성질
    1. 임의의 길이를 갖는 메시지에 대해서 오라클은 0과 1로 이루어진 난수 스트링인 고정된 길이의 메시지 다이제스트를 생성해서 제공함
    2. 이미 다이제스트가 존재하는 메시지가 주어지면 오라클은 저장되어 있던 다이제스트를 제공함
    3. 새로운 메시지에 대한 다이제스트는 다른 모든 다이제스트와는 독립적으로 선택될 필요가 있음
       - 즉, 오라클은 다이제스트를 만들기 위해 공식이나 알고리즘을 사용해서는 안됨

랜덤 오라클 모델

- 랜덤 오라클 모델(Random Oracle Model) (2/5)
  - 예제 11.1

오라클이 하나의 표와 동전 한 개로 이루어졌다고 가정해보자.
해당 표는 두 개의 열로 이루어져 있다. 왼쪽 열은 오라클에 의해 다이제스트가 이미 생성된 메시지들을 나타내고, 두 번째 열은 왼쪽의 메시지로부터 생성된 다이제스트 목록이다. 다이제스트의 길이는 메시지의 크기와 무관하게 항상 16비트라고 가정을 하자. 표 11.1 에는 메시지와 이에 대응하는 메시지 다이제스트를 16진수로 나타낸 것이다. 해당 표를 보면 오라클은 이미 3개의 다이제스트를 생성했다. 이 때, 두가지 사건 a와 b가 발생했다고 가정해보자.

표 11.1 최초 3개의 수를 제공하는 Oracle 표

Message	Message Digest
4523AB1352CDEF45126	13AB
723BAE38F2AB3457AC	02CA
AB45CD1048765412AAAB6662BE	A38B

랜덤 오라클 모델

- 랜덤 오라클 모델(Random Oracle Model) (3/5)
  - 예제 11.1 – a

메시지 AB1234CD8765BDAD를 가지고 다이제스트를 생성한다고 하자.

1. 표에는 위의 메시지가 들어있지 않음
2. 오라클은 동전을 16번을 던져 HHTHHHTTHTHHTTTH를 얻음 (H는 앞면, T는 뒷면)
3. 오라클에서는 H를 1, T를 0로 대체하여 이진수 1101 1100 1011 0001을 획득함
   이를 16진수로 변환한 DCB1를 표에 추가함

표 11.1 최초 3개의 수를 제공하는 Oracle 표

Message	Message Digest
4523AB1352CDEF45126	13AB
723BAE38F2AB3457AC	02CA
AB45CD1048765412AAAB6662BE	A38B

→

표 11.2 4개의 수를 제공하는 Oracle 표

Message	Message Digest
4523AB1352CDEF45126	13AB
723BAE38F2AB3457AC	02CA
AB1234CD8765BDAD	DCB1
AB45CD1048765412AAAB6662BE	A38B

랜덤 오라클 모델

- 랜덤 오라클 모델(Random Oracle Model) (4/5)
  - 예제 11.1 – b

메시지 4523AB1352CDEF45126가 다이제스트 생성을 위해 주어졌다고 하자.

1. 오라클은 표를 점검하고, 위 메시지에 대한 다이제스트가 이미 표에 존재한다는 것을 알아챔
2. 오라클은 단순히 대응되는 다이제스트 값인 13AB를 제공함

표 11.1 최초 3개의 수를 제공하는 Oracle 표

Message	Message Digest
4523AB1352CDEF45126	13AB
723BAE38F2AB3457AC	02CA
AB45CD1048765412AAAB6662BE	A38B

랜덤 오라클 모델

- 랜덤 오라클 모델(Random Oracle Model) (5/5)
  - 예제 11.2

오라클이 주어진 메시지로부터 다이제스트 생성에 공식 $h(M) = M \ mod \ n$을 사용한다고 해보자.

- 상황
  - 오라클에서 이미 $h(M_1)$과 $h(M_2)$를 생성해 둠
  - 새로운 메시지 M_3는 $M_3 = M_1 + M_2$임
- 결과
  - M_3에 대한 새로운 다이제스트는 $[h(M_1) + h(M_2)] \ mod \ n$이 됨

$$h(M_3) = M_3 \ mod \ n = (M_1 + M_2) mod \ n = M_1 \ mod \ n + M_2 \ mod \ n$$
$$= h(M_1) + h(M_2)$$

  - 각 다이제스트는 오라클에 입력되는 메시지에 대해 랜덤하게 선택 되어야 한다는 세 번째 요구조건에 위배됨

랜덤 오라클 모델

- 비둘기 집 원리(Pigeonhole Principle) (1/2)
  - 의미
    - 만약 $n+1$마리의 비둘기가 n개의 비둘기 집에 들어가 있다면 적어도 한 비둘기 집에는 두 마리의 비둘기가 들어있음
  - 일반화된 정의
    - 만약 $kn+1$마리의 비둘기가 n개의 비둘기 집에 들어가 있다면 적어도 한 비둘기 집에는 $k+1$마리의 비둘기가 들어가 있어야 함
  - 해시에서의 비둘기 집 원리
    - 어떤 다이제스트는 한 개 이상의 메시지에 대응됨
      - 다이제스트의 길이가 메시지 길이보다 짧아야 함
    - 가능한 메시지들의 집합과 다이제스트들의 집합 관계는 다수-대-일 $(n:1)$ 관계임

랜덤 오라클 모델

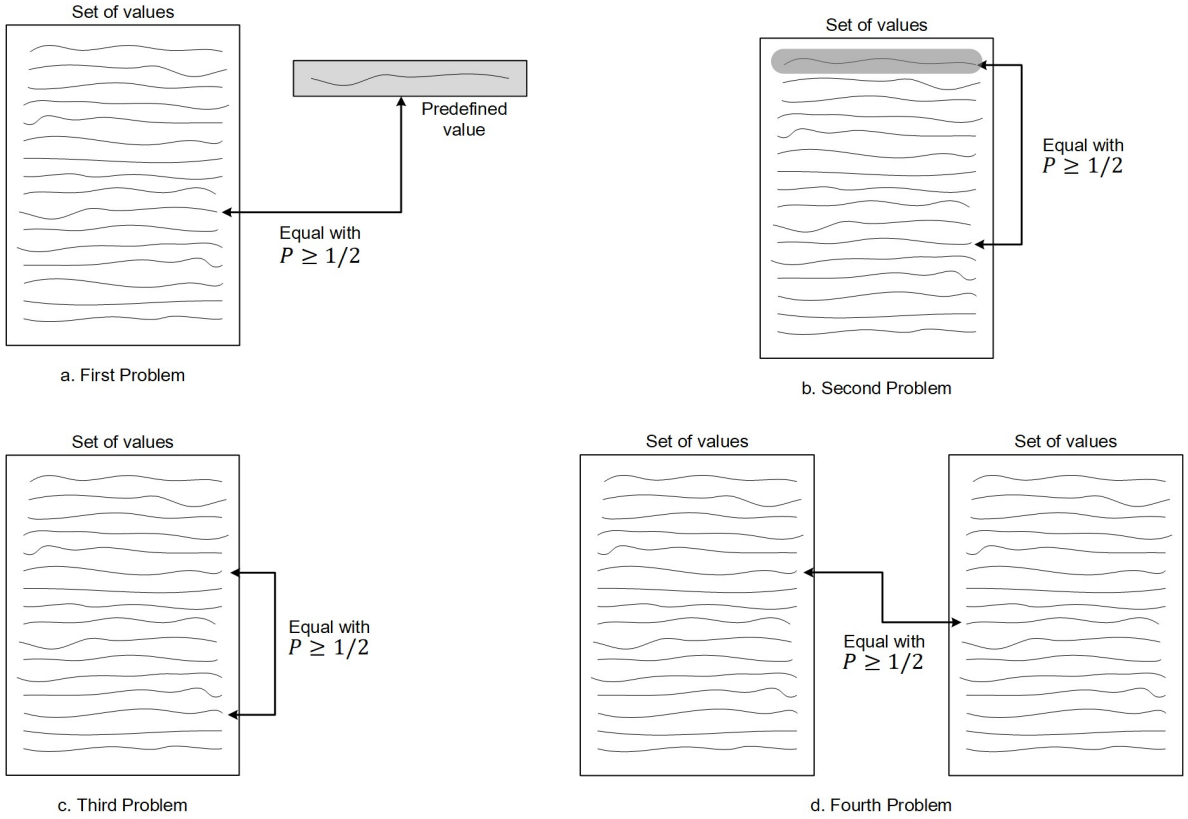
- 비둘기 집 원리(Pigeonhole Principle) (2/2)
  - 예제 11.3

해시함수에 적용되는 메시지 길이가 6비트이고, 다이제스트 길이는 오직 4비트이다. 그러면 가능한 다이제스트(비둘기 집)의 총 개수는 $2^4 = 16$이고, 가능한 메시지(비둘기)의 총 개수는 $2^6 = 64$개다.

- $n = 16$이고, $kn + 1 = 64 \quad \therefore \ k \geq 3$
- 적어도 하나의 다이제스트는 $k + 1 = 4$개의 메시지와 대응됨

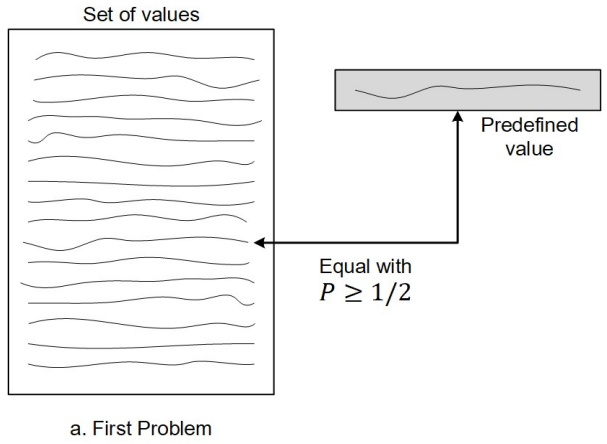
랜덤 오라클 모델

- 생일 문제(Birthday Problems) (1/14)
  - 정의
    - 충돌이 발생할 확률이 직관보다 훨씬 빠르게 증가한다는 점을 보여주는 확률 이론의 예시



랜덤 오라클 모델

- 생일 문제(Birthday Problems) (2/14)
  - 첫 번째 문제
    - 교실 안에 학생이 k명 있을 때 적어도 한 학생이 특정 생일날일 확률 P가½보다 크거나 같으려면 k의 최소값이 무엇인가?
  - 첫 번째 문제 - 일반화
    - N개의 값(0부터 $N-1$)에 균등하게 분포된 *확률 변수가 있다. 적어도 한 개의 확률 변수가 특정한 값(0과 $N-1$ 사이의 미리 지정된 값)을 가질 확률이 ½ 이상이 되려면 최소한 몇 개의 확률 변수가 필요한가?



a. First Problem

* 확률 변수
확률적인 결과에 따라 결과 값이 바뀌는 변수

랜덤 오라클 모델

- 생일 문제(Birthday Problems) (3/14)
  - 첫 번째 문제 – 해답 (1/2)
    - 확률 P
      1. P_{sel}을 선택된 표본이 사전에 지정된 값과 같을 확률이라고 하면, $P_{sel} = \frac{1}{N}$
      2. Q_{sel}을 선택된 표본이 사전에 지정된 값과 같지 않을 확률이라고 하면, $Q_{sel} = 1 - P_{sel} = 1 - \frac{1}{N}$
      3. 각 표본이 독립이고, Q는 어느 표본도 사전에 주어진 값과 같지 않을 확률이라고 한다면, $Q = {Q_{sel}}^k = \left(1 - \frac{1}{N}\right)^k$
      4. P가 적어도 하나의 표본이 사전에 지정된 값과 같을 확률이라면,

$$P = 1 - Q = 1 - \left(1 - \frac{1}{N}\right)^k = 1 - e^{-\frac{k}{N}}$$

$$\therefore\ P = 1 - e^{-\frac{k}{N}}$$

- 근사식
  $\lim_{x \to 0}(1 - x) \approx e^{-x}$

랜덤 오라클 모델

- 생일 문제(Birthday Problems) (4/14)
  - 첫 번째 문제 – 해답 (2/2)
    - 최소 표본 크기 k ($P \geq \frac{1}{2}$)

$$P \geq \frac{1}{2}$$

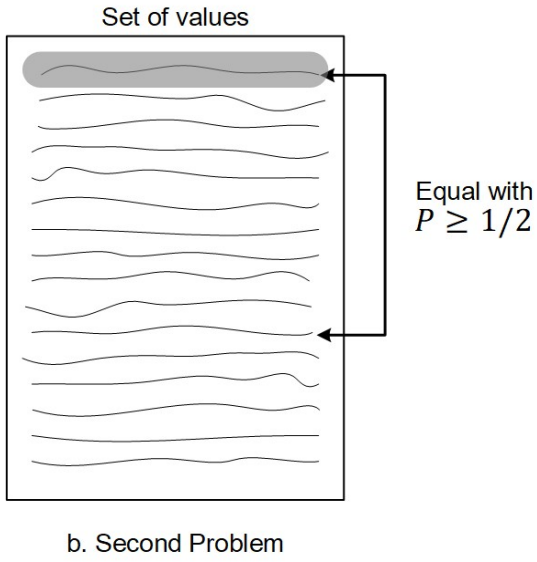
$$1 - e^{-\frac{k}{N}} \geq \frac{1}{2}$$

$$k \geq (\ln 2) \times N$$

$$\therefore k \geq (\ln 2) \times N$$

랜덤 오라클 모델

- 생일 문제(Birthday Problems) (5/14)
  - 두 번째 문제
    - 교실 안에 학생이 k명 있을 때 적어도 한 학생이 교사가 미리 선정한 학생과 같은 생일날일 확률 P가½보다 크거나 같으려면 k의 최소값이 무엇인가?
  - 두 번째 문제 - 일반화
    - N개의 값(0부터 $N-1$)에 균등하게 분포된 확률 변수가 있다. 적어도 한 개의 확률 변수가 미리 선택 된 확률 변수의 값 (0과 $N-1$ 사이의 미리 지정된 값)을 가질 확률이 ½이상이 되려면 최소한 몇 개의 확률 변수가 필요한가?



b. Second Problem

랜덤 오라클 모델

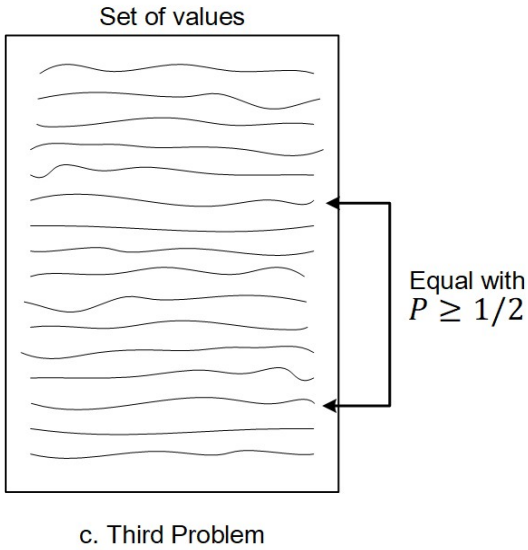
- 생일 문제(Birthday Problems) (6/14)
  - 두 번째 문제 – 해답
    - 확률 P
      1. 표본 집합에서 하나의 표본을 선택한 뒤에 남아있는 표본의 수가 $k-1$개 임
      2. 사전에 지정된 값이 표본 값 중 하나라는 사실을 제외하고 첫 번째 문제와 동일하므로 첫 번째 문제의 확률에 k대신 $k-1$을 대입함

      $\therefore P = 1 - e^{-\frac{k-1}{N}}$
    - 최소 표본 크기 k
      - 확률과 동일하게 첫 번째 문제의 표본 크기에 k 대신 $k-1$을 대입

      $\therefore k \geq (\ln 2) \times N + 1$

랜덤 오라클 모델

- 생일 문제(Birthday Problems) (7/14)
  - 세 번째 문제 (생일 패러독스)
    - 교실 안에 학생이 k명 있을 때 적어도 두 학생이 같은 생일날일 확률 P가 ½ 보다 크거나 같으려면 k의 최소값이 무엇인가?
  - 세 번째 문제 - 일반화
    - N개의 값(0부터 $N-1$)에 균등하게 분포된 확률 변수가 있다. 적어도 두 개의 확률 변수 값이 동일할 확률이 ½이상이 되려면 최소한 몇 개의 확률 변수가 필요한가?



c. Third Problem

랜덤 오라클 모델

- 생일 문제(Birthday Problems) (8/14)
  - 세 번째 문제 – 해답 (1/3)
    - 확률 P
      1. 한 번에 한 개씩 표본에 확률을 지정함
         표본 i가 사전 표본들 중 한 값과 같을 확률을 P_i라고 하고,
         표본 i가 사전 표본들과 다른 값을 가질 확률을 Q_i라고 지정함
         a. 첫 번째 표본 이전에는 사전 표본이 없으므로 $P_1 = 0, Q_1 = 1 - 0 = 1$
         b. 두 번째 표본 이전에는 사전 표본이 한 개이고, 첫 번째 표본은 N개의 값 중 하나이므로, $P_2 = \frac{1}{N}$이고, $Q_2 = (1 - \frac{1}{N})$
         c. 세 번째 표본 이전에는 사전 표본에 두 개이고, 이 두 개의 표본 각각은 N개의 값 중 하나이므로, $P_3 = \frac{2}{N}$이고, $Q_3 = (1 - \frac{2}{N})$
         d. 동일한 논리를 계속 적용하면, $P_k = \frac{k-1}{N}$, $Q_k = [1 - \frac{k-1}{N}]$

랜덤 오라클 모델

- 생일 문제(Birthday Problems) (9/14)
  - 세 번째 문제 – 해답 (2/3)
    - 확률 P

      2. 모든 표본들이 독립이라면, 모든 표본이 서로 다른 값을 가질 확률은 아래와 같음

$$Q = Q_1 \times Q_2 \times \cdots \times Q_k = 1 \times \left(1 - \frac{1}{N}\right) \times \left(1 - \frac{2}{N}\right) \times \cdots \times \left(1 - \frac{k-1}{N}\right)$$

$$Q = \left(e^{-\frac{1}{N}}\right) \times \left(e^{-\frac{2}{N}}\right) \times \cdots \times \left(e^{-\frac{k-1}{N}}\right)$$

$$Q = e^{-\frac{k(k-1)}{2N}} = e^{-\frac{k^2}{2N}}$$

      3. 적어도 두 개의 표본이 동일한 값을 가질 확률을 P라고 한다면, P는 아래와 같음

$$P = 1 - Q = 1 - e^{-\frac{k^2}{2N}}$$

$$\therefore P = 1 - e^{-\frac{k^2}{2N}}$$

- 등차공식
  $1 + 2 + \cdots + (k-1) = \frac{k(k-1)}{2}$
- 근사식
  $\lim_{x \to 0}(1 - x) \approx e^{-x}$
  $\lim_{k \to \infty} k(k-1) \approx k^2$

랜덤 오라클 모델

- 생일 문제(Birthday Problems) (10/14)
  - 세 번째 문제 – 해답 (3/3)
    - 최소 표본 크기 k ($P \geq \frac{1}{2}$)

$$P \geq \frac{1}{2}$$

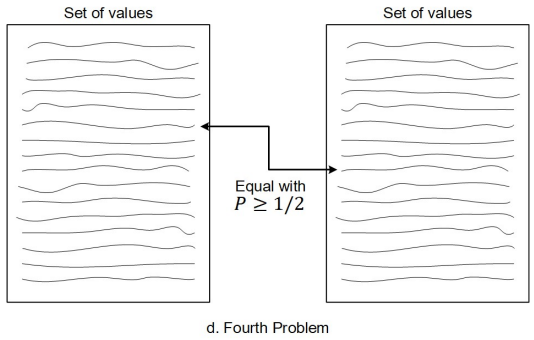
$$1 - e^{-\frac{k^2}{2N}} \geq \frac{1}{2}$$

$$k \geq (2 \ln 2)^{\frac{1}{2}} \times N^{\frac{1}{2}}$$

$$\therefore k \geq (2 \ln 2)^{\frac{1}{2}} \times N^{\frac{1}{2}}$$

랜덤 오라클 모델

- 생일 문제(Birthday Problems) (11/14)
  - 네 번째 문제
    - 두 개의 반이 있는데, 각 반에는 k명의 학생이 있다. 적어도 첫번째 반에서 선택된 한 학생과 다른 반에서 선택된 한 학생의 생일이 같을 확률 P가 ½이상이 되기 위해 필요한 k의 최소값이 무엇인가?
  - 네 번째 문제 - 일반화
    - N개의 값(0부터 $N-1$)에 균등하게 분포된 확률 변수가 있다. 여기서 각각 k개의 확률 변수들로 이루어진 두 집단을 생각해보자. 이때 적어도 첫 번째 집단에서 선택한 한 확률 변수의 값이 다른 집단에서 선택한 한 확률 변수의 값과 동일할 확률이 ½이상이 되려면 최소한 몇 개의 확률 변수가 필요한가? (k의 최소값은 무엇인가?)



랜덤 오라클 모델

- 생일 문제(Birthday Problems) (12/14)
  - 네 번째 문제 – 해답 (1/2)
    - 확률 P

> • 근사식
> $\lim_{x\to 0}(1-x) \approx e^{-x}$

1. 첫 번째 문제에서 첫 번째 표본 집합의 모든 표본들이 두 번째 표본 집합의 첫 번째 표본 값과 다르게 될 확률은 $Q_1 = \left(1-\frac{1}{N}\right)^k$
2. 첫 번째 표본 집합의 모든 표본들이 두 번째 표본 집합의 첫 번째와 두 번째 표본과 다른 값을 가지게 될 확률은
$$Q_2 = \left(1-\frac{1}{N}\right)^k \times \left(1-\frac{1}{N}\right)^k$$
3. 해당 논리를 계속 적용하면, 첫 번째 표본 집합의 모든 표본들이 두 번째 표본 집합의 어떤 표본과 같지 않게 될 확률은
$$Q_k = \left(1-\frac{1}{N}\right)^k \times \left(1-\frac{1}{N}\right)^k \times \cdots \times \left(1-\frac{1}{N}\right)^k \rightarrow Q_k = \left(1-\frac{1}{N}\right)^{k^2}$$
$$Q_k = e^{-\frac{k^2}{N}}$$

랜덤 오라클 모델

- 생일 문제(Birthday Problems) (13/14)
  - 네 번째 문제 – 해답 (2/2)
    - 확률 P

      4. 첫 번째 표본 집합의 적어도 한 표본이 두 번째 표본 집합의 한 표본과 동일한 값을 가지게 될 확률을 P라고 하면,

         $P = 1 - Q_k = 1 - e^{-\frac{k^2}{N}}$

      $\therefore P = 1 - e^{-\frac{k^2}{N}}$

    - 최소 표본 크기 k $(P \geq \frac{1}{2})$

      $P \geq \frac{1}{2}$

      $1 - e^{-\frac{k^2}{N}} \geq \frac{1}{2}$

      $k \geq (\ln 2)^{\frac{1}{2}} \times N^{\frac{1}{2}}$

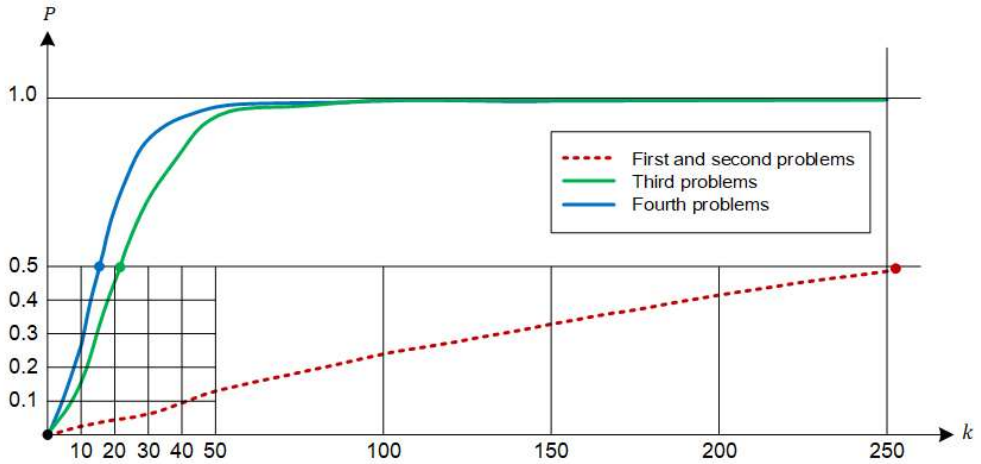
      $\therefore k \geq (\ln 2)^{\frac{1}{2}} \times N^{\frac{1}{2}}$

랜덤 오라클 모델

- 생일 문제(Birthday Problems) (14/14)
  - 문제 해답 요약 표

Problem	Probability	General value for k	Value of k with $P = 1/2$	Number of students ($N = 365$)
1	$P \approx 1 - e^{-k/N}$	$k \approx \ln\left[\frac{1}{1-P}\right] \times N$	$k \approx 0.69 \times N$	253
2	$P \approx 1 - e^{-(k-1)/N}$	$k \approx \ln\left[\frac{1}{1-P}\right] \times N + 1$	$k \approx 0.69 \times N + 1$	254
3	$P \approx 1 - e^{-k(k-1)/2N}$	$k \approx \{2 \ln\left[\frac{1}{1-P}\right]\}^{\frac{1}{2}} \times N^{\frac{1}{2}}$	$k \approx 1.18 \times N^{\frac{1}{2}}$	23 → 생일 패러독스에 대한 해답
4	$P \approx 1 - e^{-k^2/2N}$	$k \approx \{\ln\left[\frac{1}{1-P}\right]\}^{\frac{1}{2}} \times N^{\frac{1}{2}}$	$k \approx 0.83 \times N^{\frac{1}{2}}$	16

  - 생일문제 그래프



랜덤 오라클 모델

- 랜덤 오라클 모델에 대한 공격 (1/14)
  - 가정
    - 해시함수가 n비트 다이제스트를 생성함
    - 다이제스트는 0부터 $N-1$ 사이에 균등하게 분포되는 확률 변수임
    - 가능한 다이제스트의 개수는 2^n개임 ($N = 2^n$)
    - 오라클은 N개의 값들 중 하나를 랜덤하게 선택함
      - 단, 모든 경우가 선택되는 것은 아님
        - 어떤 값은 전혀 선택되지 않을 수도, 어떤 값은 여러 차례 선택될 수도 있음
    - 해시함수 알고리즘이 공개되어 있음
    - 공격자는 다이제스트의 크기인 n값을 알고 있음

랜덤 오라클 모델

- 랜덤 오라클 모델에 대한 공격 (2/14)
  - 프리이미지 공격 (1/2)
    - 공격 내용
      - 공격자는 다이제스트 $D = h(M)$을 가로채고, $D = h(M')$가 되는 임의의 메시지 M'을 찾으려고 함
      - 공격자는 D를 찾기 위한 k개의 메시지를 생성하고 알고리즘을 구동함

```
Preimage_Attack (D)
{
   for (i = 1 to k)
   {
      create(M[i])
      T ← h(M[i])                    // T 는 임시 다이제스트
      if (T = D)  return M[i]
   }
   return  failure
}
```

랜덤 오라클 모델

- 랜덤 오라클 모델에 대한 공격 (3/14)
  - 프리이미지 공격 (2/2)
    - 첫 번째 생일문제 활용
      - 알고리즘 성공 확률은 $P \approx 1 - e^{-\frac{k}{N}}$
      - 50%의 성공을 위한 k의 크기는 $k \approx 0.69 \times N$ 또는 $k \approx 0.69 \times 2^n$
        $\therefore$ 2^n에 비례하는 개수의 다이제스트를 생성해야 함
      - 프리이미지 공격의 어려움 정도는 2^n에 비례함
    - 예제 11.4

암호학적 해시함수는 64비트 다이제스트를 사용한다.
공격자는 원래의 메시지를 찾을 확률이 0.5 이상이 되도록 하려면 얼마나 많은 다이제스트를 생성해야하는가?

      - 생성해야 할 다이제스트의 수는 $k \approx 0.69 \times 2^n \approx 0.69 \times 2^{64}$
      - 공격자가 초당 $2^{30}(\approx 1{,}000{,}000{,}000)$개의 다이제스트를 생성해도 0.69×2^{34}초$(\approx 500$년$)$가 필요함
      - 64비트의 다이제스트는 프리이미지 공격에 대해 안전함

랜덤 오라클 모델

- 랜덤 오라클 모델에 대한 공격 (4/14)
  - 제 2 프리이미지 공격 (1/2)
    - 공격 내용
      - 공격자는 다이제스트 $D = h(M)$과 이에 대응하는 메시지 M을 가로챘고, $h(M') = D$를 만족하는 다른 메시지 M'을 구하려고 함
      - 공격자는 M'을 찾기 위한 $k-1$개의 메시지를 생성하고 알고리즘을 구동함

```
Second_Preimage_Attack (D, M)
{
   for (i = 1 to k − 1)
   {
      create(M[i])
      T ← h(M[i])                    // T 는 임시 다이제스트
      if (T = D) return M[i]
   }
   return failure
}
```

랜덤 오라클 모델

- 랜덤 오라클 모델에 대한 공격 (5/14)
  - 제 2 프리이미지 공격 (2/2)
    - 두 번째 생일문제 활용
      - 알고리즘 성공 확률은 $P \approx 1 - e^{-\frac{k-1}{N}}$
      - 50%의 성공을 위한 k의 크기는 $k \approx 0.69 \times N + 1$ 또는 $k \approx 0.69 \times 2^n + 1$
        $\therefore$ 2^n에 비례하는 개수의 다이제스트를 생성해야 함
      - 제 2 프리이미지 공격의 어려움 정도는 2^n에 비례함
    - 예제 11.5

암호학적 해시함수는 64비트 다이제스트를 사용한다.
공격자는 원래의 메시지를 찾을 확률이 0.5 이상이 되도록 하려면 얼마나 많은 다이제스트를 생성해야하는가?

- 생성해야 할 다이제스트의 수는 $k \approx 0.69 \times 2^n + 1 \approx 0.69 \times 2^{64} + 1$
- 공격자가 초당 $2^{30}(\approx 1{,}000{,}000{,}000)$개의 다이제스트를 생성해도 $1 + 0.69 \times 2^{34}$초$(\approx 500$년$)$가 필요함
- 64비트의 다이제스트는 제 2 프리이미지 공격에 대해 안전함

랜덤 오라클 모델

- 랜덤 오라클 모델에 대한 공격 (6/14)
  - 충돌 공격 (1/2)
    - 공격 내용
      - 공격자는 $h(M) = h(M')$을 만족하는 2개의 다른 메시지 M과 M'를 구하려고 함
      - 공격자는 k개의 메시지를 생성하고 알고리즘을 구동함

```
Collision_Attack ()
{
  for (i = 1 to k)
  {
    create(M[i])
   D[i]←h(M[i])      // D[i]는 생성된 다이제스트 목록
   for (j = 1 to i − 1)
   {
     if (T = D)  return M[i]
   }
  }
  return  failure
}
```

랜덤 오라클 모델

- 랜덤 오라클 모델에 대한 공격 (7/14)
  - 충돌 공격 (2/2)
    - 세 번째 생일문제 활용
      - 알고리즘 성공 확률은 $P \approx 1 - e^{-\frac{k^2}{2N}}$
      - 50%의 성공을 위한 k의 크기는 $k \approx 1.18 \times N^{\frac{1}{2}}$ 또는 $k \approx 1.18 \times 2^{\frac{n}{2}}$
        $\therefore$ $2^{\frac{n}{2}}$에 비례하는 개수의 다이제스트를 생성해야 함
      - 충돌 공격의 어려움 정도는 $2^{\frac{n}{2}}$에 비례함
    - 예제 11.6

한 암호학적 해시함수가 64비트 다이제스트를 사용한다고 하자.
공격자가 동일한 다이제스트를 갖는 서로 다른 두 개의 메시지를 만들 수 있는 확률이 0.5이상 되려면 얼마나 많은 다이제스트가 필요한가?

- 생성해야 할 다이제스트의 수는 $k \approx 1.18 \times 2^{\frac{n}{2}} \approx 1.18 \times 2^{32}$
- 공격자가 초당 2^{20}개의 메시지를 점검할 수 있다면, 1.18×2^{12}초 혹은 2시간 이내로 끝낼 수 있음
- 길이가 64비트인 메시지 다이제스트는 충돌 공격에 안전하지 못함

랜덤 오라클 모델

- 랜덤 오라클 모델에 대한 공격 (8/14)
  - 또 다른 충돌 공격 (1/2)
    - 공격 내용
      - 공격자는 원본 메시지 M으로부터 k개의 서로 다른 메시지 $M_1, M_2, \ldots, M_k$를 생성하고, 위조된 메시지 M'로부터 k개의 서로 다른 메시지 $M'_1, M'_2, \ldots, M'_k$ 를 생성함
        - 메시지의 일부를 중복시키거나 수정하되, 원본 메시지 본래의 뜻은 훼손시키지 않아야 함
          - e.g., 공백 추가, 단어 변경, 단어 추가 등

```
Alternate_Collision_Attack ()
{
   for (i = 1 to k)
   {
    D[i]←h(M[i])
    D'[i]←h(M'[i])
   if (D[i] = D'[j])  return (M[i], M'[j])
   }
   return  failure
}
```

랜덤 오라클 모델

- 랜덤 오라클 모델에 대한 공격 (9/14)
  - 또 다른 충돌 공격 (2/2)
    - 네 번째 생일문제 활용
      - 알고리즘 성공 확률은 $P \approx 1 - e^{-\frac{k^2}{N}}$
      - 50%의 성공을 위한 k의 크기는 $k \approx 0.83 \times N^{\frac{1}{2}}$ 또는 $k \approx 0.83 \times 2^{\frac{n}{2}}$
        $\therefore$ $2^{\frac{n}{2}}$에 비례하는 개수의 다이제스트를 생성해야 함
      - 또 다른 충돌 공격의 어려움 정도는 $2^{\frac{n}{2}}$에 비례함
    - 예제 11.7

한 암호학적 해시함수가 64비트 다이제스트를 사용한다고 하자.
공격자가 동일한 다이제스트를 갖는 서로 다른 두 개의 메시지를 만들 수 있는 확률이 0.5이상 되려면 얼마나 많은 다이제스트가 필요한가?

- 생성해야 할 다이제스트의 수는 $k \approx 0.83 \times 2^{\frac{n}{2}} \approx 0.83 \times 2^{32}$
- 공격자가 초당 2^{20}개의 메시지를 점검할 수 있다면, 0.83×2^{12}초 혹은 1시간 이내로 끝낼 수 있음
- 길이가 64비트인 메시지 다이제스트는 또 다른 충돌 공격에 안전하지 못함

랜덤 오라클 모델

- 랜덤 오라클 모델에 대한 공격 (10/14)
  - 충돌 저항성의 중요성
    - 각 공격 유형별 난이도
      - 공격의 어려움 정도(order)가 프리이미지 공격이나 제 2 프리이미지 공격보다 충돌 공격의 경우 더 낮다는 것을 알 수 있음
      - 해시 알고리즘이 충돌 공격에 저항성을 가지고 있다면 프리이미지 공격이나 제 2 프리이미지 공격에 대해서는 안전함

Attack	Value of k with $P = 1/2$	Order
Preimage	$k \approx 0.69 \times 2^n$	2^n
Second preimage	$k \approx 0.69 \times 2^n + 1$	2^n
Collision	$k \approx 1.18 \times 2^{n/2}$	$2^{n/2}$
Alternate collision	$k \approx 0.83 \times 2^{n/2}$	$2^{n/2}$

랜덤 오라클 모델

- 랜덤 오라클 모델에 대한 공격 (11/14)
  - 충돌 저항성의 중요성 (1/4)
    - e.g., 64비트 다이제스트를 가지는 원래의 해시함수
      - 프리이미지와 충돌에 대해 모두 저항성이 낮음
      - 공격자는 확률 1/2 이상을 가지고 공격을 시행할 경우 오직 $2^{\frac{64}{2}} = 2^{32}$번의 점검만 수행하면 됨
      - 공격자가 초당 2^{20}번의 점검을 수행한다고 하면, $\frac{2^{32}}{2^{20}} = 2^{12}$초로 약 1시간 만에 공격을 할 수 있음

랜덤 오라클 모델

- 랜덤 오라클 모델에 대한 공격 (12/14)
  - 충돌 저항성의 중요성 (2/4)
    - e.g., 128비트 다이제스트를 갖는 MD5(Message-Digest algorithm 5)
      - 프리이미지와 충돌에 대해 모두 저항성이 낮음
      - 공격자는 확률 1/2 이상을 가지고 충돌 공격을 수행하려면 $2^{\frac{128}{2}} = 2^{64}$번의 점검만 수행하면 됨
      - 공격자가 초당 2^{30}번의 점검을 할 수 있다면, 2^{34}초(500년 이상)가 걸림
      - 현실적으로는 쉽지 않은 횟수이지만, 이론상으로는 충돌 저항성이 비교적 낮음

랜덤 오라클 모델

- 랜덤 오라클 모델에 대한 공격 (13/14)
  - 충돌 저항성의 중요성 (3/4)
    - e.g., 160비트 다이제스트를 갖는 SHA-1(Secure Hash Algorithm 1)
      - 이론상으로는 충돌 저항성이 높지만, 수학적 약점이 발견되어 실제 충돌 공격이 가능함
        - Merkle–Damgård 구조의 자체의 수학적 약점과 취약한 압축 함수를 사용함
      - 공격자는 확률 1/2 이상을 가지고 충돌 공격을 수행하려면 $2^{\frac{160}{2}} = 2^{80}$번의 점검을 수행해야 함
      - 공격자가 초당 2^{30}번의 점검을 할 수 있다면, 2^{50}초(1만년 이상)가 걸림
      - 이론상으로는 안전하나, 실제로는 더 적은 시간 내 공격이 가능함

* Merkle–Damgård 구조
가변 길이 입력 메시지를
고정 길이 해시값으로 압축하는
해시 함수의 설계 방식

랜덤 오라클 모델

- 랜덤 오라클 모델에 대한 공격 (13/14)
  - 충돌 저항성의 중요성 (3/4)
    - e.g., 256비트 다이제스트를 갖는 SHA-256(Secure Hash Algorithm 256)
      - 프리이미지와 충돌에 대해 모두 저항성이 높음
        - Merkle–Damgård 구조를 기반으로 SHA-1과 다른 개선된 압축함수를 사용함
      - 공격자는 확률 1/2 이상을 가지고 충돌 공격을 수행하려면 $2^{\frac{256}{2}} = 2^{128}$번의 점검을 수행해야 함
      - 공격자가 초당 2^{30}번의 점검을 할 수 있다면, 2^{98}초(수억 년 이상)가 걸림
      - 현실적으로 공격 불가능에 가까운 수준임

랜덤 오라클 모델

- 랜덤 오라클 모델에 대한 공격 (14/14)
  - 충돌 저항성의 중요성 (4/4)
    - e.g., 512비트 다이제스트를 갖는 SHA-512 (Secure Hash Algorithm 512)
      - 프리이미지와 충돌에 대해 모두 저항성이 높음
        - Merkle–Damgård 구조에 Davies-Meyer 구조를 추가하여 설계를 보완함
      - 공격자는 확률 1/2 이상을 가지고 충돌 공격을 수행하려면 $2^{\frac{512}{2}} = 2^{256}$번의 점검을 수행해야 함
      - 공격자가 초당 2^{30}번의 점검을 할 수 있다면, 2^{226}초 (138억년 이상)가 걸림
      - 현실적으로 공격 불가능에 가까운 수준임

* Davies-Meyer 구조
블록 암호를 이용해 이전 해시 값과 메시지 블록을 결합하여 새로운 해시값을 생성하는 해시 함수의 설계 방식

랜덤 오라클 모델

- 구조에 대한 공격
  - 이상적인 암호학적 해시함수
    - 랜덤 오라클 모델과 같은 해시함수는 내부구조가 전혀 없음
  - 실제 해시 함수
    - 취약한 내부구조를 가지고 있는 경우가 있음
      - e.g., MD5(충돌 생성 가능), SHA-0(논리적 회전 연산 누락)
    - 완벽하게 랜덤한 다이제스트를 생성하는 해시함수를 만드는 것은 현실적으로 불가능 함
    - 공격자가 해시함수를 공격하는 다른 도구를 가지고 있는 경우가 있음
      - e.g., 중간자 공격(MITM, Man-in-Middle Attack)

목 차

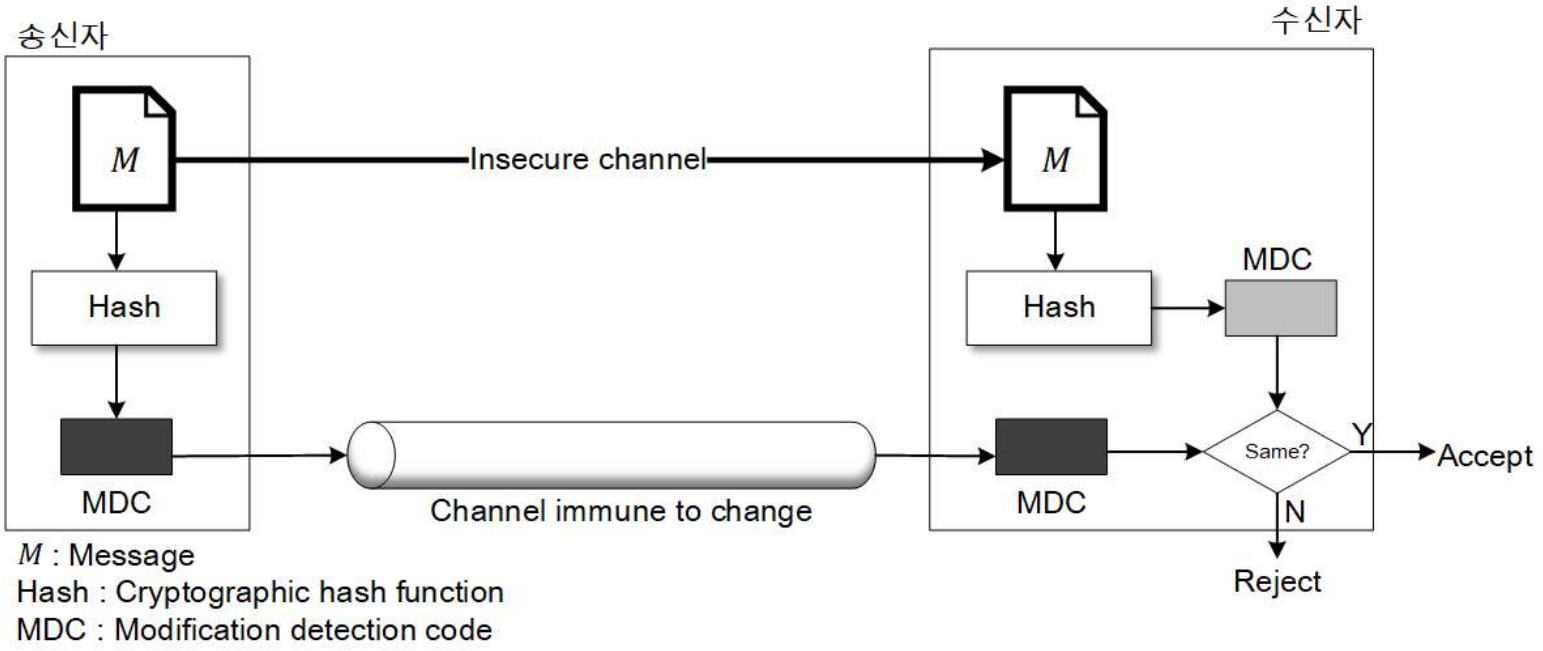
- 보충
- 메시지 무결성 (Message Integrity)
- 랜덤 오라클 모델 (Random Oracle Model)
- 메시지 인증 (Message Authentication)

메시지 인증

- 변경 감지 코드(MDC; Modification Detection Code) (1/2)
  - 정의
    - 메시지의 무결성을 보장하는 메시지 다이제스트
  - 무결성 검증
    - 송신자는 MDC를 생성하여 메시지와 MDC를 모두 수신자에게 보냄
    - 수신자는 수신한 메시지로부터 새로운 MDC를 생성하여 수신된 MDC와 비교함
      - 두 값이 일치하면 무결성 보장
      - 두 값이 일치하지 않으면 무결성 훼손

메시지 인증

- 변경 감지 코드(MDC) (2/2)
  - 전달 채널
    - 메시지는 안전하지 않은 채널을 통해 전달 가능함
    - MDC는 안전한 채널을 통해 전달해야 함
      - 공격자가 메시지를 위조했더라도, MDC를 통해 위조된 것임을 증명할 수 있음



메시지 인증

- 메시지 인증
  - 정의
    - 메시지가 변경되지 않았으며, 신뢰할 수 있는 송신자로부터 전송되었음을 증명하는 것

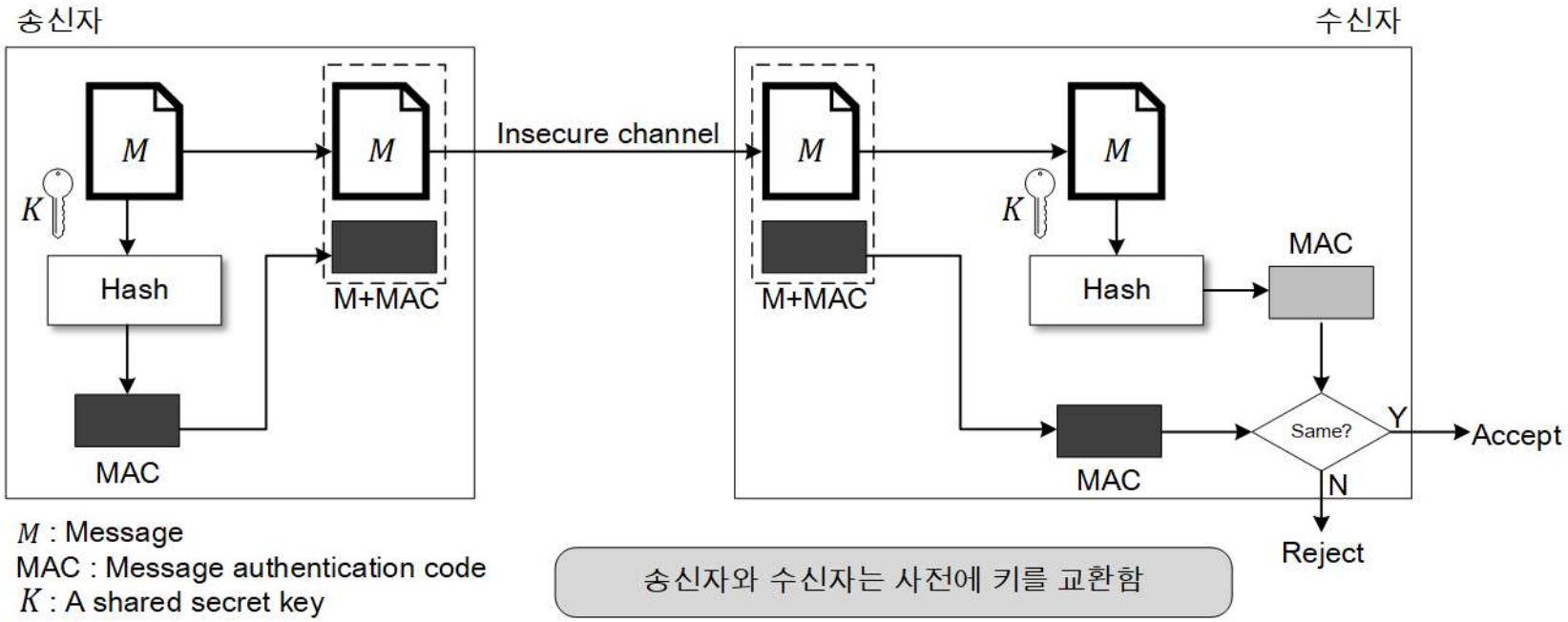
- 메시지 인증 코드(MAC; Message Authentication Code) (1/13)
  - 정의
    - 메시지와 비밀키를 함께 입력하여 생성된 값

메시지 인증

- 메시지 인증 코드(MAC) (2/13)
  - 무결성과 출원지 검증
    - 수신자와 송신자 사이의 비밀 키를 포함시킴
    - 송신자는 키 K와 메시지 M를 이어 붙인 $K|M$에 해시함수를 적용하여 MAC인 $h(K|M)$을 생성하여 메시지와 함께 전달함
    - 수신자는 $K|M$으로부터 새로운 MAC를 생성하여 수신된 MAC와 비교함
      - 두 값이 일치하면 무결성 보장
      - 두 값이 일치하지 않으면 무결성 훼손
      - 비밀 키를 이용하므로 송신자 인증 가능

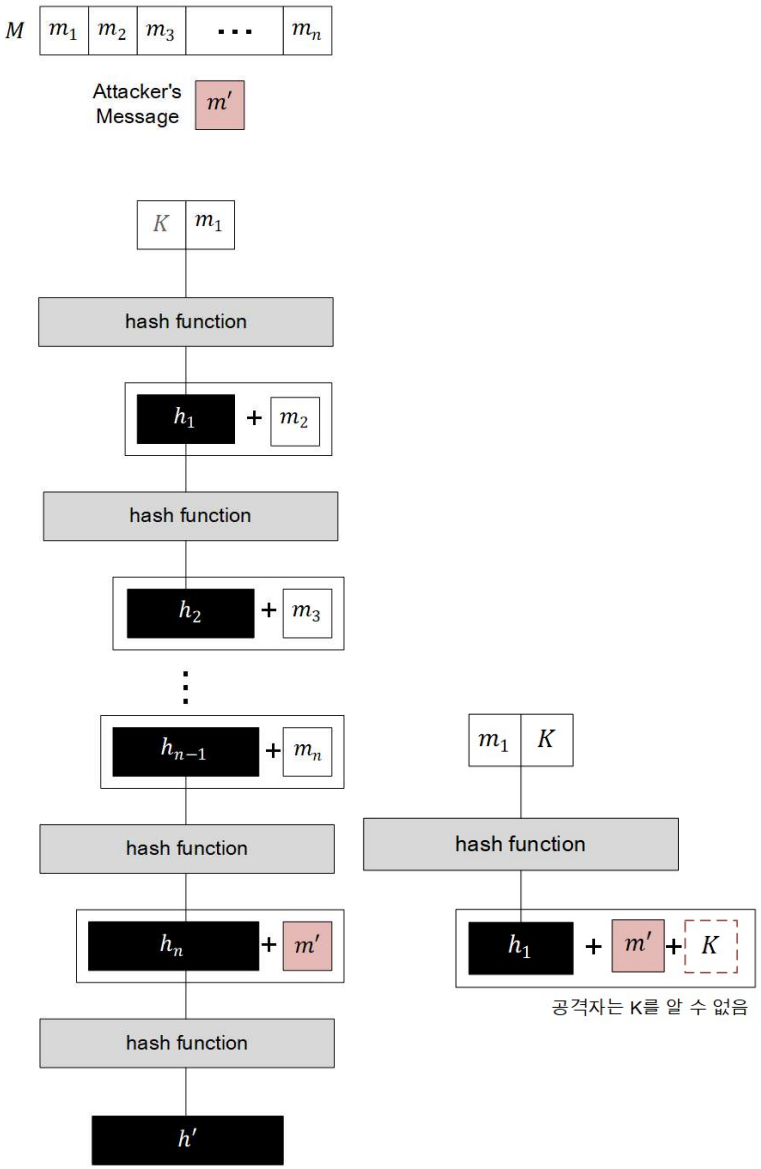
메시지 인증

- 메시지 인증 코드(MAC) (3/13)
  - 전달 채널
    - 메시지와 MAC는 하나의 안전하지 않은 채널로 보낼 수 있음
      - 공격자는 메시지를 볼 수는 있으나, 비밀 키를 알 수 없으므로 메시지를 위조할 수 없음



메시지 인증

- 메시지 인증 코드(MAC) (4/13)
  - 키의 위치
    - 프리픽스 MAC(Prefix-MAC)
      - 구조
        - MAC $= h(K|M)$
      - 특징
        - 구현이 단순함
        - 길이 확장 공격에 취약함
    - 포스트픽스 MAC(Postfix-MAC)
      - 구조
        - MAC $= h(M|K)$
      - 특징
        - 프리픽스 MAC보다 길이 확장 공격에 덜 취약함



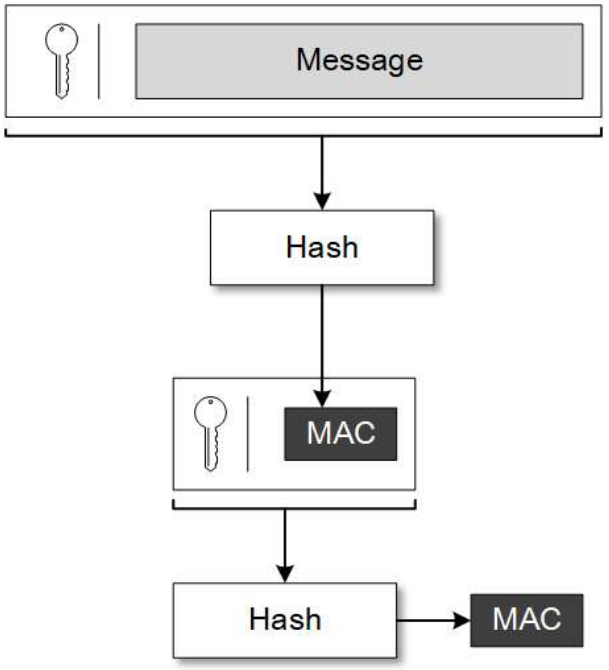
메시지 인증

- 메시지 인증 코드(MAC) (5/13)
  - MAC의 안전성
    - 공격자가 M과 $h(K|M)$을 가로챈 경우 가능한 공격
      - 전수 조사 공격(Brute Force Attack)
        - 키의 길이가 짧은 경우
        - 모든 가능한 키를 메시지의 앞부분에 붙여 $h(K|M)$과 동일한 값이 나올 때 까지 시도 할 수 있음
      - 프리이미지 공격(Preimage Attack)
        - 키의 길이가 충분히 커서 전수 조사 공격이 불가능한 경우
        - 알고리즘을 이용하여 $h(X)$가 공격자가 가로챈 MAC와 같게 되는 X를 찾아냄
      - 선택 메시지 공격(Chosen Message Attack)
        - 공격자가 여러 쌍의 메시지와 MAC을 확보한 경우
        - 메시지 쌍들과 MAC들을 조합하거나 분석하여 새로운 메시지에 대한 유효한 MAC를 찾아낼 수 있음

MAC의 안전성은 사용하는 해시 알고리즘의 안전성에 종속됨

메시지 인증

- 메시지 인증 코드(MAC) (6/13)
  - 중첩된 MAC(Nested MAC)
    - 개요
      - MAC의 안전성을 높이기 위해서 설계됨
    - 구조
      1. 키를 메시지와 이어 붙이고 해시하여 중간 단계의 다이제스트를 생성함
      2. 키를 중간 단계 다이제스트에 이어 붙이고 해시하여 최종적인 다이제스트를 생성함

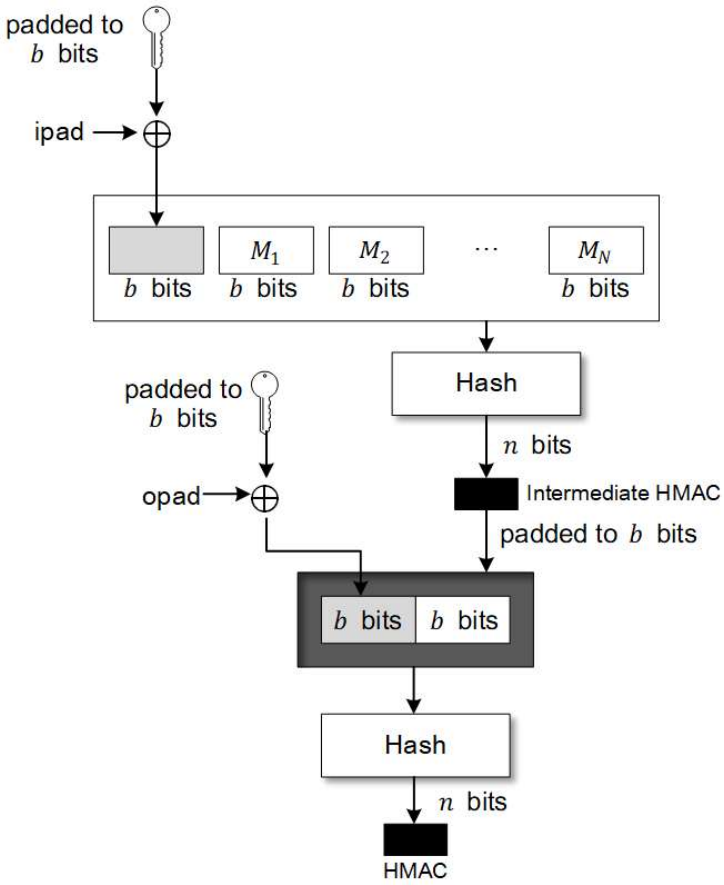


메시지 인증

- 메시지 인증 코드(MAC) (7/13)
  - HMAC(Hashed MAC) (1/3)
    - 정의
      - 비밀 키와 해시 함수를 함께 사용하여 메시지의 무결성과 인증을 보장하는 MAC 알고리즘
    - 특징
      - 사용된 해시 함수가 충돌 저항성을 가진다면 HMAC도 동일한 수준의 보안성을 가짐
      - 입력 메시지가 해시 함수 내부 상태에 직접 영향을 주지 않는 중첩된 해시 구조로 인해 길이 확장 공격에 안전함
      - NIST FIPS 198-1에 의해 표준화 되어있음

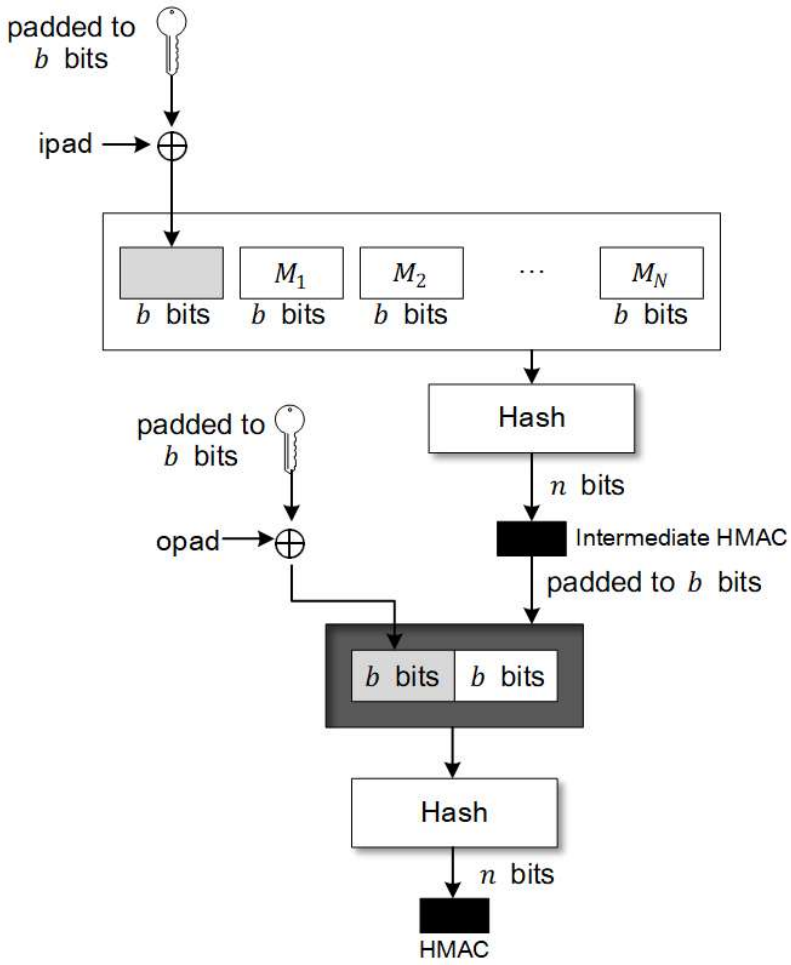
메시지 인증

- 메시지 인증 코드(MAC) (8/13)
  - HMAC(Hashed MAC) (2/3)
    - 구조 (1/2)
      1. 메시지를 b비트의 길이를 가진 N개의 블록으로 분리함
      2. 비밀 키의 왼쪽에 0으로 된 열을 추가하여 b비트가 되도록 함
         - 패딩 이전 비밀 키의 길이는 n비트 이상을 권장 (n은 HMAC의 크기)
      3. 2단계의 결과를 ipad(Input Pad)라고 부르는 상수와 XOR 함
         - ipad는 $b/8$번의 연속된 2진열 00110110 (=0x36)로 이루어짐
      4. 3단계로 얻은 블록을 N개로 이루어진 $N+1$개의 블록을 얻음



메시지 인증

- 메시지 인증 코드(MAC) (9/13)
  - HMAC(Hashed MAC) (3/3)
    5. 4단계에서 얻은 $N+1$개의 블록을 해시함수에 적용하여 n비트 다이제스트를 생성함
       - 이 때 생성된 다이제스트가 중간 HMAC
    6. n비트로 된 중간 HMAC의 왼쪽에 0으로 이루어진 열을 패딩하여 b비트 블록으로 만듦
    7. 2단계와 3단계를 다른 상수인 opad(Output Pad)로 반복함
       - opad는 $b/8$번의 연속된 2진열 01011100(=0x5C)로 이루어짐
    8. 7단계에서 얻은 결과를 6단계에서 얻은 블록의 앞에 붙임
    9. 9단계에서 얻은 결과에 동일한 해시함수를 적용하여 최종 n비트 HMAC을 생성함



메시지 인증

- 메시지 인증 코드(MAC) (10/13)
  - CMAC(Cipher-based MAC) (1/4)
    - 정의
      - 블록 암호 알고리즘과 대칭 키를 이용하여 메시지의 무결성과 인증을 보장하는 MAC 알고리즘
    - 특징
      - 암호블럭체인(CBC, Cipher Block Chaining) 모드와 유사한 방법으로, CBCMAC이라고도 부름
      - 사용된 암호 알고리즘이 안전할 경우, CMAC도 동일한 수준의 보안성을 가짐
      - 입력 메시지가 블록 단위로 처리되며, 마지막 블록의 패딩 여부에 따라 다른 키가 적용됨
      - 입력 메시지의 길이에 따라 서브키를 적용하므로 메시지 확장으로 동일한 MAC을 구성할 수 없어 길이 확장 공격에 안전함
      - FIPS 113 및 NIST SP 800-38B에 따라 표준화되어 있음

*비밀 키 (Secret Key)
소유자가 비공기로 유지해야 하는 키

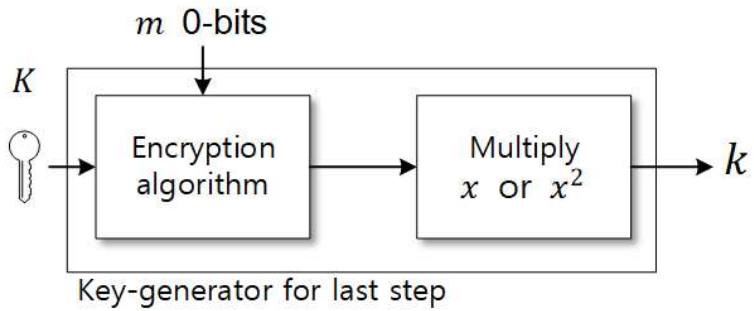
*대칭 키 (Symmetric Key)
암호화와 복호화에 동일한 키를 사용하는 방식에서 쓰이는 키

메시지 인증

- 메시지 인증 코드(MAC) (11/13)
  - CMAC(Cipher-based MAC) (2/4)
    - 구조 (1/3)
      - 서브 키 k 생성
        1. m개의 0비트로 이루어진 평문에 암호 키 K를 사용하여 암호 알고리즘을 적용함
        2. 1단계의 결과에 대해 곱연산을 수행함
           - m은 특정 프로토콜에 의해 선택되는 차수가 m인 기약 다항식을 갖는 체 $GF(2^m)$안에서의 곱연산임

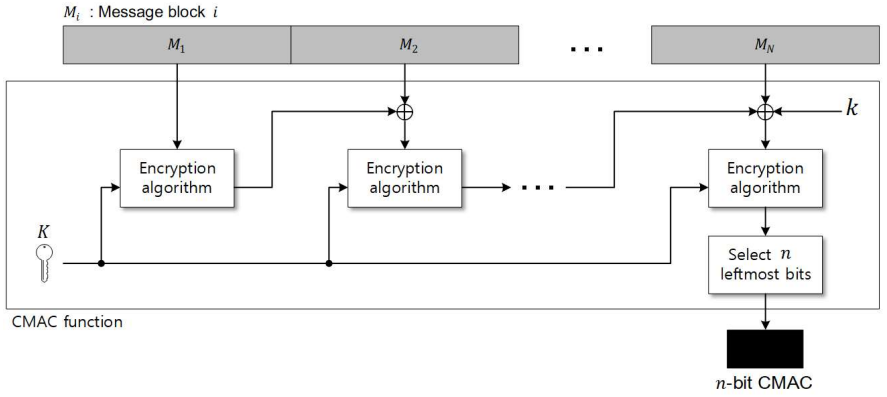
           a. 패딩이 적용 되지 않은 경우에는 알고리즘 연산 결과에 x를 곱함

           b. 패딩이 적용 된 경우에는 알고리즘 연산 결과에 x^2을 곱함



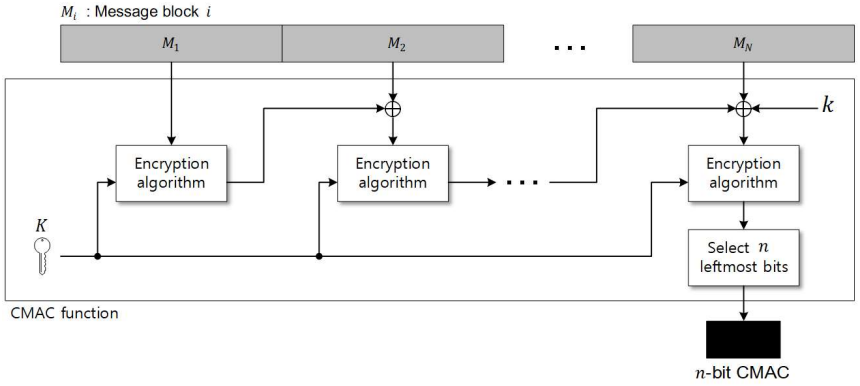
메시지 인증

- 메시지 인증 코드(MAC) (12/13)
  - CMAC(Cipher-based MAC) (3/4)
    - 구조 (2/3)
      - CMAC 생성
        1. 메시지를 N개의 길이가 m비트인 블록으로 분리함
           - CMAC의 크기는 n비트임
        2. 마지막 블록이 m비트에 모자라면 첫 번째 비트를 1로, 나머지 비트를 0으로 채워서 m비트로 패딩함
        3. 메시지의 첫 번째 블록을 대칭 키 K로 암호화 하여 m비트 블록을 생성함
        4. 3단계의 결과를 다음 블록과 XOR하고 그 결과를 다시 암호화하여 새로운 m비트 블록을 생성함

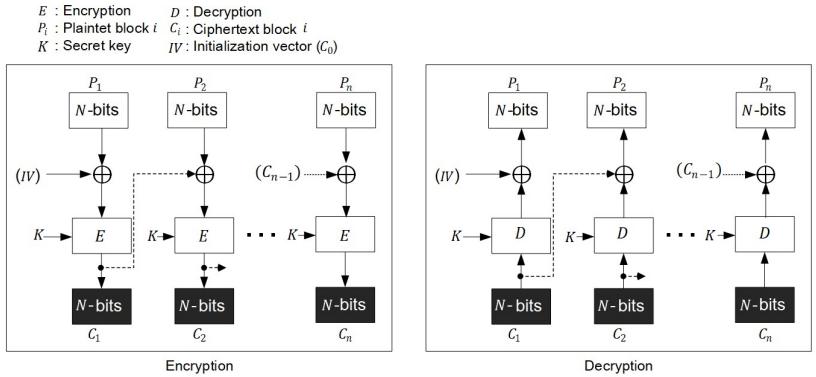


메시지 인증

- 메시지 인증 코드(MAC) (13/13)
  - CMAC(Cipher-based MAC) (4/4)
    - 구조 (3/3)
      - CMAC 생성
        5. 3단계의 첫 번째 블록을 4단계의 결과로 변경하며 3, 4단계를 마지막 블록이 암호화 될 때까지 반복함
           - 마지막 블록에는 별도로 생성된 서브 키 k를 사용함
        6. 모든 과정 이후 마지막 블록의 가장 왼쪽에 있는 n개의 비트가 CMAC임



    - CBC모드와 차이
      - CBC는 각 출력된 암호문으로 보내지고 동시에 다음 평문 블록과 XOR을 수행한 중간 암호 블록들이 전달되지 않음



Thanks!

송 민 희(23011716@sejong.ac.kr)