제 12 장
키

박 종 혁 교수

Tel: 970-6702

Email: jhpark1@seoultech.ac.kr

1절 키란 무엇인가?

2절 다양한 키

3절 컨텐츠를 암호화하는 키와 키를 암호화 하는 키

4절 키 관리

5절 Diffie-Hellman 키 교환

6절 패스워드를 기초로 한 암호(PBE)

7절 안전한 패스워드를 만들려면

제1절 키란 무엇인가?

1.1 키는 대단히 큰 수

1.2 키는 평문과 동일한 가치를 갖는다

1.3 암호 알고리즘과 키

1.1 키는 대단히 큰 수

- 암호 기술을 사용하려면 반드시 키(key)라 불리는 대단히 큰 수가 필요
- 중요한 것은 수 그 자체의 크기라기보다도 키 공간(Key Space)의 크기
- 키 공간의 크기는 키의 비트 길이로 결정

암호별 키 길이

- DES: 56비트
- 3DES
  - DES-EDE2: 112비트
  - DES-EDE3: 168비트
- AES: 128, 192, 256
- RSA: 1024, 2046

DES 키 예

- 2진수 표현(56비트):

01010001 11101100 01001011 00010010 00111101 01000010
00000011

- 16진수 표현:

51 EC 4B 12 3D 42 03

- 10진수 표현:

23059280286269955

RSA 키 예

- 16진수 표현(1024비트):

00:bc:04:e4:fa:13:39:f0:34:96:20:6b:6c:68:bb:fa:db:77:ff:27:f7:ac:e
c:2f:e7:fd:f0:7f:6d:6f:8c:2a:cd:25:09:5b:24:f4:a1:68:fc:28:ec:c9:25:
e2:ac:ed:de:c8:33:84:f5:b0:a5:09:3a:a7:b1:47:48:c5:cc:4f:8c:79:9c:
f9:06:57:7d:dd:ee:38:f6:cf:14:b2:9c:ea:d3:c0:5d:77:62:f0:47:0d:b9:
1a:40:53:5c:64:70:af:08:5a:c0:f7:cf:75:f9:6c:8d:64:28:1e:20:fe:b7:1
b:19:d3:5a:66:83:72:e2:b0:9b:bd:d3:25:15:0d:32:6f:64:37:94:85:46:
c8:72:be:77:d5:6e:1f:28:2f:c7:69:ed:e7:83:89:33:58:d3:de:a0:bf:40:
e8:43:50:ee:dc:4d:6b:bc:a5:ea:a6:c8:61:8e:f5:c3:64:af:06:15:dc:29:
8b:3f:75:8c:bc:71:44:db:fc:ad:b5:17:1d:6d:89:83:cf:c6:33:bd:bf:45:
a2:fe:0a:9f:a3:11:5f:0f:b9:1f:9c:1a:c2:46:cc:9c:28:66:9f:70:26:3c:2
e:df:aa:80:fe:8c:c5:04:09:25:4f:cd:93:47:3c:37:ea:02:67:92:fe:fc:22
:24:5c:ac:d2:2c:e0:5c:01:33:8a:c1:19:db

1.2 키는 평문과 동일한 가치를 갖는다

- **키는 평문과 같은 가치**
  - 도청자 이브에게 「키가 넘어가는 것」은 「평문이 넘어가는 것」과 같은 것

1.3 암호 알고리즘과 키

- **암호의 기본 상식:**
  - 검증된 암호 알고리즘을 사용
  - 정보의 기밀성은 암호 알고리즘을 비밀로 하는 것이 아님
  - 키를 비밀로 하는 것에 의해 기밀성이 지켜져야 함

제2절 다양한 키

2.1 대칭 암호 키와 공개 키 암호 키

2.2 메시지 인증 코드 키와 디지털 서명 키

2.3 기밀성을 위한 키와 인증을 위한 키

2.4 세션 키와 마스터 키

2.1 대칭 암호 키와 공개 키 암호 키

- **대칭 암호**
  - 키는 송신자와 수신자만 공유
  - 양측이 공유 키를 비밀로 유지
- **공개 키 암호**
  - 암호화와 복호화에서 다른 키 사용
  - 개인 키를 비밀로 유지

대칭 암호는 암호화와 복호화에서 공통 키를 사용

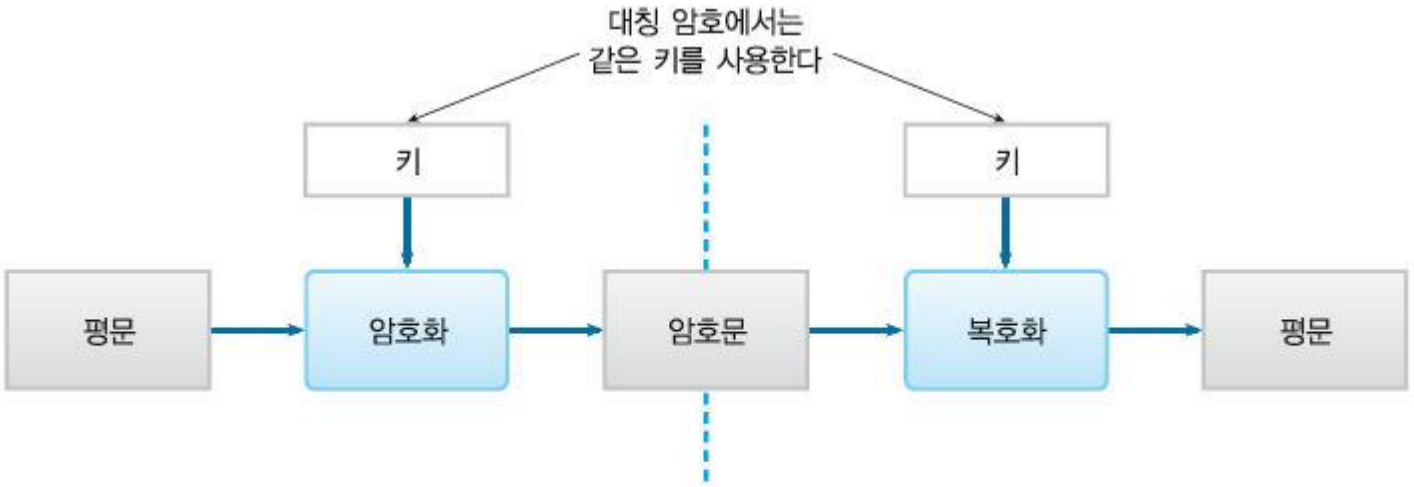


그림 12-1 • 대칭 암호는 암호화와 복호화에서 공통 키를 사용한다

공개 키 암호는 공개 키로 암호화하고, 개인 키로 복호화

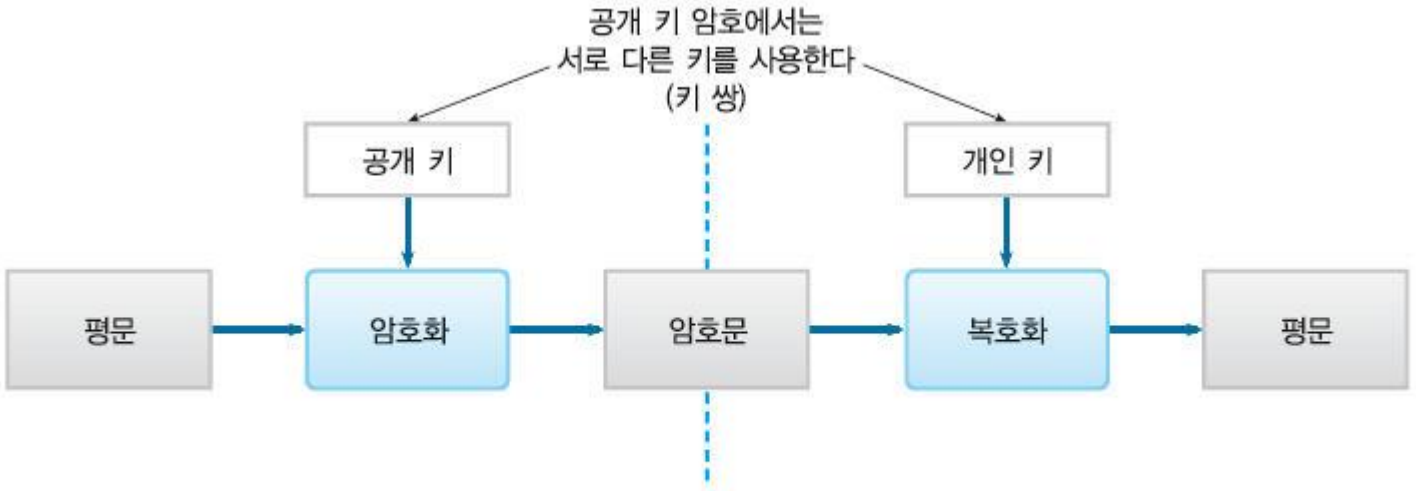


그림 12-2 • 공개 키 암호는 공개 키로 암호화하고 개인 키로 복호화한다

2.2 메시지 인증 코드 키와 디지털 서명 키

- **메시지 인증 코드**
  - 송신자와 수신자가 공통의 키를 사용해서 인증을 수행
- **디지털 서명**
  - 서명 작성과 서명 검증에 서로 다른 키를 사용

메시지 인증 코드 키

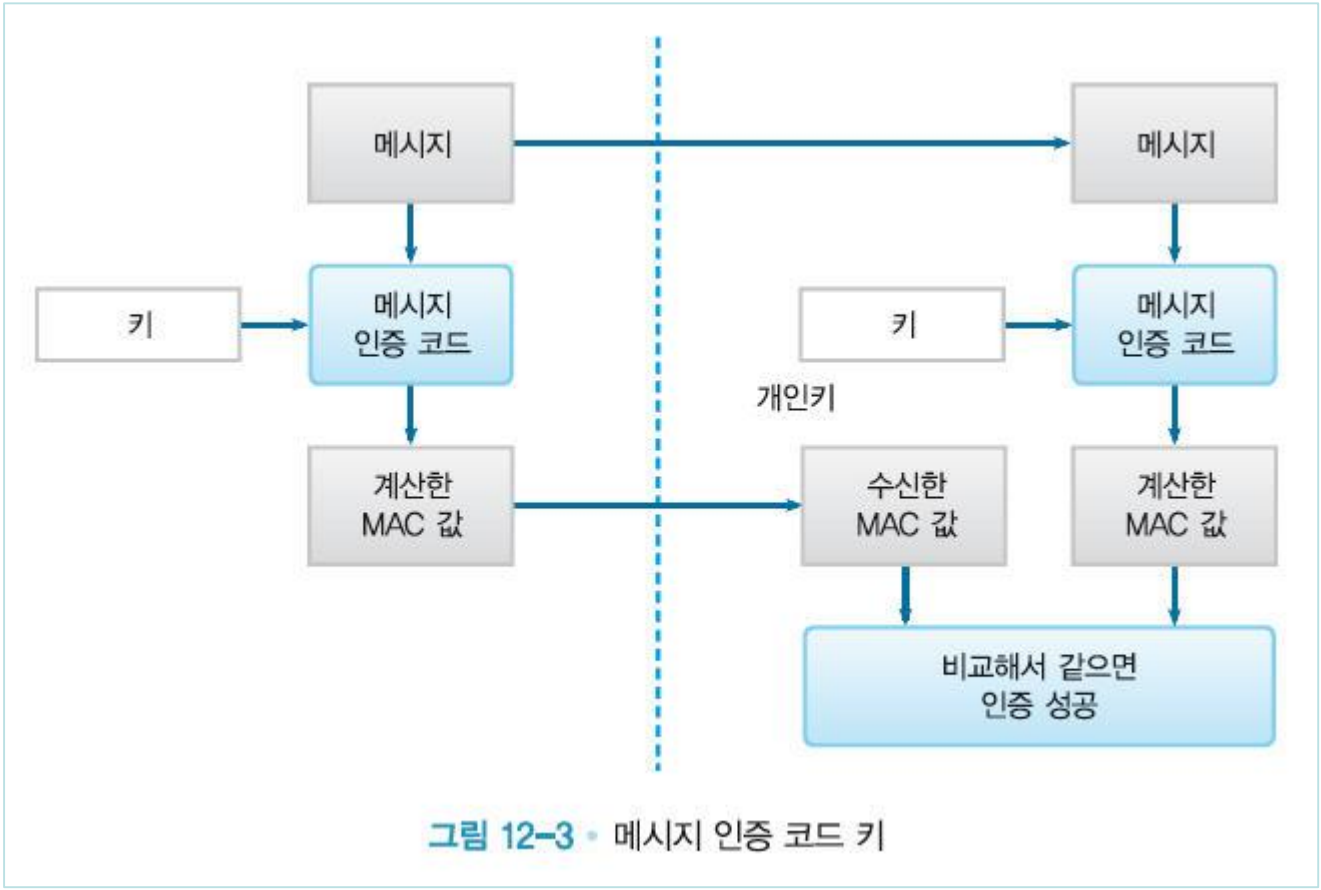


그림 12-3 • 메시지 인증 코드 키

디지털 서명 키

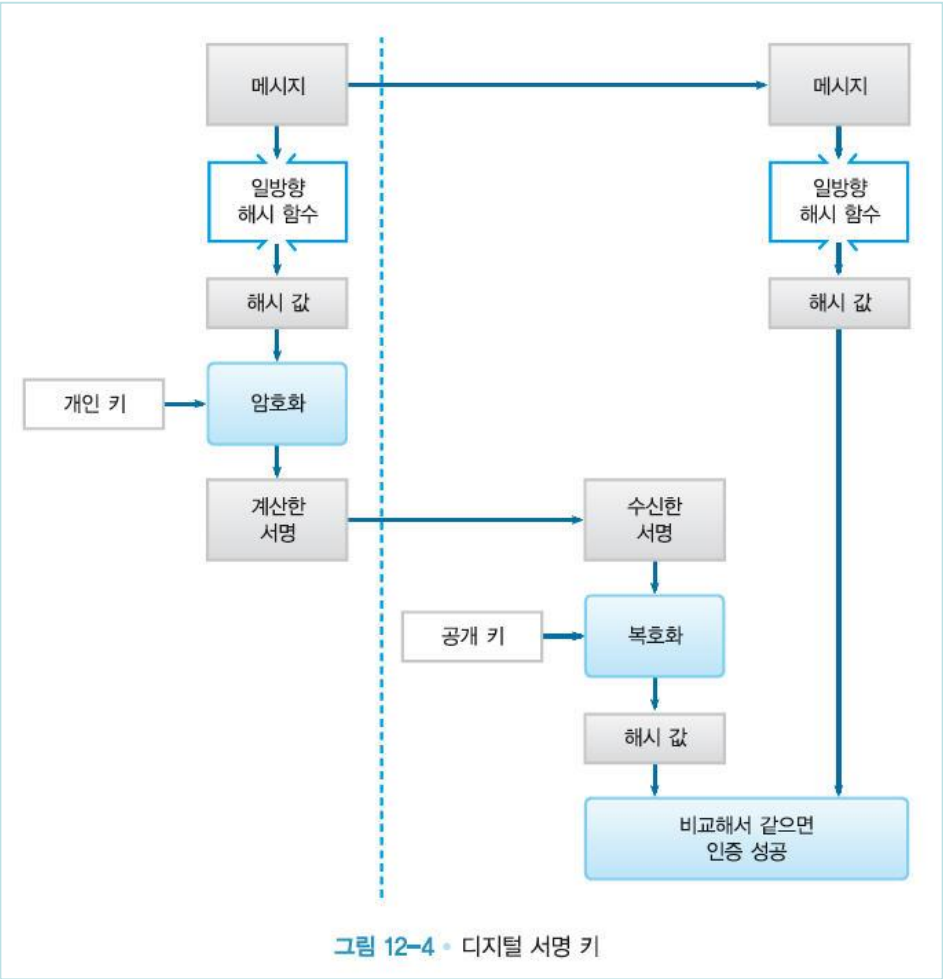


그림 12-4 • 디지털 서명 키

2.3 기밀성을 위한 키와 인증을 위한 키

- **보안 속성에 따른 분류**
  - 기밀성을 유지하기 위한 키:
    - 대칭 암호나 공개 키 암호에서 사용하는 키
    - 복호화 키를 모르면 복호 불가
  - 인증을 수행하기 위한 키:
    - 메시지 인증 코드나 디지털 서명에서 사용하는 키
    - 키를 모르면 데이터 변경이나 위장 불가

2.4 세션 키와 마스터 키

- 키 사용 횟수에 따른 분류
  - 세션 키(session key):
    - 통신 때마다 한 번만 사용되는 키
  - 마스터 키(master key)
    - 반복적으로 사용되는 키

제3절 컨텐츠를 암호화하는 키와 키를 암호화 하는 키

- 키를 사용할 때 암호화 대상에 따른 분류
  - CEK(contents encrypting key):
    - 정보(콘텐츠)가 암호화의 대상
  - KEK(key encrypting key):
    - – 키가 암호화의 대상

콘텐츠를 암호화하는 키(CEK)와 키를 암호화하는 키(KEK)

그림 12-5 • 컨텐츠를 암호화하는 키(CEK)와 키를 암호화하는 키(KEK)

제4절 키 관리

4.1 키 생성

4.2 키 배송

4.3 키 갱신

4.4 키 보존

4.5 키 폐기

4.1 키 생성

- 난수를 이용한 키 생성
- 패스워드를 이용한 키 생성

난수를 이용한 키 생성

- **난수 사용**
  - **이유:** 키 성질로 「다른 사람이 추측하기 어려워야 한다」를 가져야 하기 때문
  - 난수는 추측하기 어렵기 때문에 키로 적합
- **난수 생성**
  - 하드웨어를 사용하는 것이 좋지만
  - 통상적으로 암호용으로 설계된 의사난수 생성기 소프트웨어를 사용

의사난수 만들기

- **자신이 적당한 바이트 열을 만들면 안 됨**
  - 이유: 스스로는 랜덤한 값이라고 생각하고 생성해도, 거기에는 아무래도 인위적인 편중이 있기 때문에 랜덤한 값이 되지 못함

- **암호용으로 이용하는 의사난수 생성기**
  - 반드시 암호용으로 설계되어 있는 것을 선택
    - 이유: 암호용으로 설계되어 있지 않은 의사난수 생성기는 「예측 불가능」 성질을 갖지 않기 때문

패스워드

- 패스워드(password) 혹은 패스 프레이즈(passphrase)로부터 키를 만드는 경우도 있다
- 패스 프레이즈라는 것은 복수의 단어로 이루어지는 긴 패스워드
- 패스워드를 키로 직접 이용하지 않고, 패스워드를 일방향 해시 함수에 입력해서 얻어진 해시 값을 키로 이용

PBE와 솔트(salt)

- 「패스워드를 기초로 한 암호」(password based encryption; PBE)
  - 패스워드에 **솔트**(salt)라 불리는 난수를 부가해서 일방향 해시 함수에 입력하고 그 출력을 키로 사용
  - 사전 공격(dictionary attack)을 막기 위한 조치

4.2 키 배송

- 키 배송 문제
  - 키를 사전에 공유하는 방법
  - 키 배포 센터를 이용하는 방법
  - 공개 키 암호를 사용하는 방법
- 또 하나의 방법:
  - Diffie-Hellman 키 교환

4.3 키 갱신

- 키 갱신(key updating)
  - 공통 키를 사용하여 통신을 하고 있는 중에 정기적으로(예를 들면 1000문자 통신할 때마다) 키를 교환해 가는 방법
  - 송신자와 수신자가 동시에 같은 방법으로 키를 교환해야만 함
  - **현재 키의 해시 값을 다음 키로 사용**

키 갱신의 장점

- 키 노출시 과거 통신의 복호화를 막을 수 있다
- 이를 백워드 시큐리티(backward security)라 한다

4.4 키 보존

- **키를 반복해서 사용할 경우 키 보존 문제를 고려**
  - 키 기억
    - 보통 실용적 키의 크기나 비트화된 표현 등으로 기억할 수 없다
  - 키 암호화
    - 키를 암호문과 동일한 컴퓨터 내에 두는 것은 어리석은 짓
    - 파일 형태로 보존된 키를 금고 등의 안전한 장소에 보관한다(공간적 제약)
    - **키를 암호화해서 보존하는** 기술을 사용

키를 암호화하는 키

- KEK(Key Encryption Key)
  - 키를 암호화하는 키
  - 다수의 키를 한 개의 키(KEK)로 암호화 하여 보관한다

인증 기관의 계층화와 키 계층의 비교

그림 12-6 • 인증 기관의 계층화와 키 계층의 비교

4.5 키 폐기

- **왜 키를 버리지 않으면 안 될까?**
  - 불필요해 진 키는 확실히 삭제
- **어떻게 버리는 것인가?**
  - 암호 소프트웨어뿐만 아니라 컴퓨터 전체가 보안을 염두에 두고 설계
- **키를 잃어버리면 어떻게 될까?**
  - 대칭 암호의 공유 키 분실
  - 메시지 인증 코드 키
  - 공개키 암호의 개인키 분실

제5절 Diffie-Hellman 키 교환

5.1 Diffie-Hellman 키 교환

5.2 Diffie-Hellman 키 교환의 수순

5.3 이브는 키를 계산 할 수 없는 것일까?

5.4 원시근의 의미

5.5 구체적 키 교환의 예

5.1 Diffie-Hellman 키 교환

- Diffie-Hellman 키 교환(Diffie-Hellman key exchange)
  - 1976년에 휘트필드 디피(Whitfield Diffie)와 마틴 헬먼(Martin Hellman)이 발명한 알고리즘
  - 타인에게 알려져도 상관없는 정보를 두 사람이 교환하는 것만으로 공통의 비밀 값을 만들어내는 방법
  - IPsec에서는 Diffie-Hellman 키 교환을 개량한 방법을 사용

5.2 Diffie-Hellman 키 교환의 수순

1) 앨리스는 밥에게 2개의 소수 P와 한 원시근 G를 송신
2) 앨리스는 난수 A를 준비
3) 밥은 난수 B를 준비
4) 앨리스는 밥에게 G^A mod P라는 수를 송신
5) 밥은 앨리스에게 G^B mod P라는 수를 송신
6) 앨리스는 밥이 보낸 수를 A제곱해서 mod P를 계산
   - 앨리스가 계산한 키 = $(G^B \bmod P)^A \bmod P = G^{B\times A} \bmod P$
7) 밥은 앨리스가 보낸 수를 B제곱해서 mod P를 계산
   - 밥이 계산한 키 = $(G^A \bmod P)^B \bmod P = G^{A\times B} \bmod P$

앨리스가 계산한 키 = 밥이 계산한 키

Diffie-Hellman 키 교환

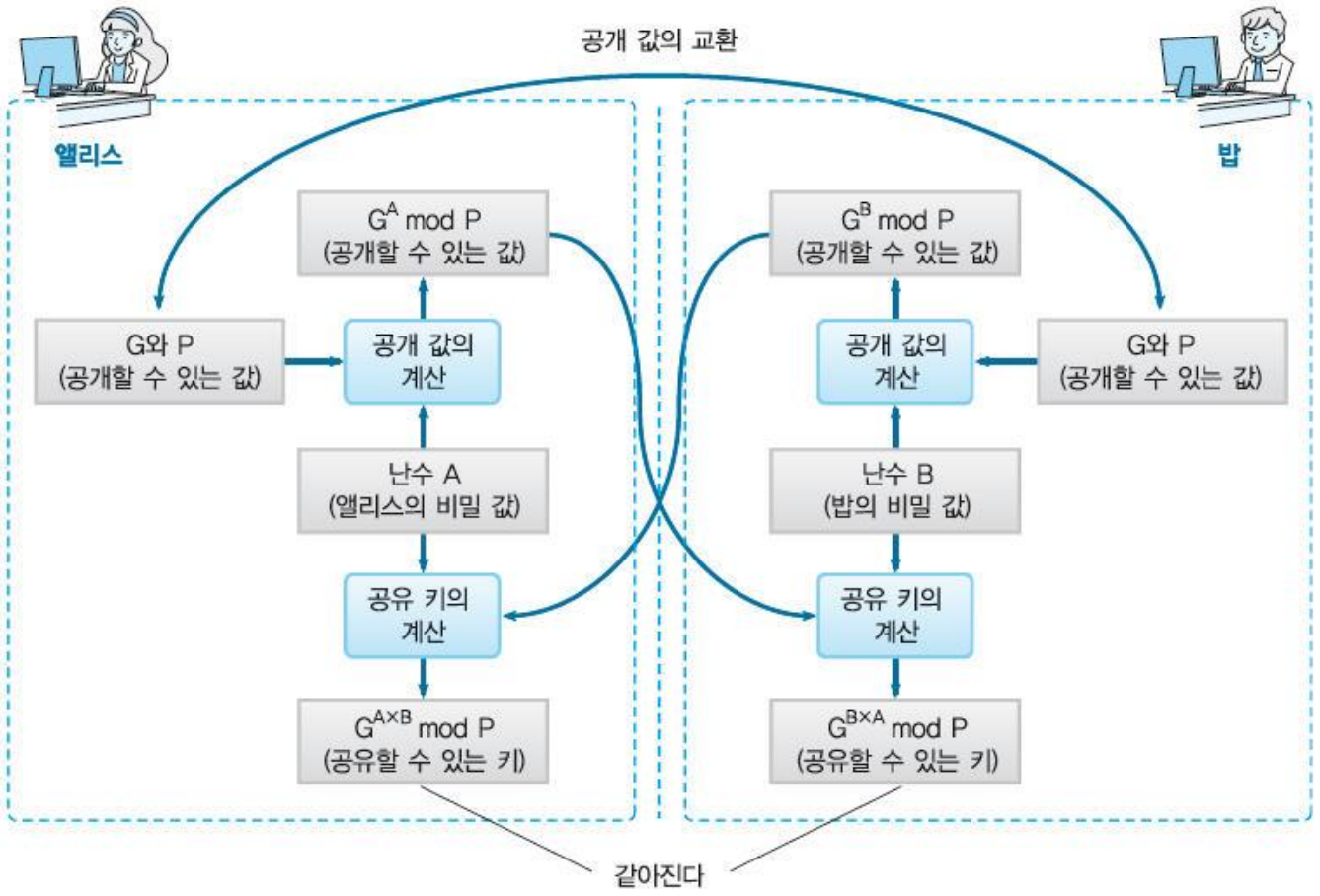


그림 12-7 • Diffie-Hellman 키 교환

5.3 이브는 키를 계산 할 수 없는 것일까?

- 공격자 이브가 알 수 있는 것
  - P, G, G^A mod P, G^B mod P라는 4개의 수
- 이 4개의 수로부터 앨리스와 밥이 공유한 키 ($G^{A\times B}$ mod P)를 계산하는 것은 수학적으로 난해
- 유한체상의 이산대수문제:
  - G^A mod P로부터 수 A를 효율적으로 계산하는 알고리즘은 아직 없음

Diffie-Hellman 키교환의 안전성

- 유한체상의 이산대수문제를 풀기 어렵기 때문에 Diffie-Hellman 키 교환의 안전성이 보장
- 단 소수 p가 적당히 커야 하고, 양측이 선택하는 수도 랜덤해야 한다.

5.4 원시근의 의미

$G^A \bmod P$의 표(P = 13인 경우)

G \ A	1	2	3	4	5	6	7	8	9	10	11	12	원시근 여부
0	0	0	0	0	0	0	0	0	0	0	0	0	
1	1	1	1	1	1	1	1	1	1	1	1	1	
2	2	4	8	3	6	12	11	9	5	10	7	1	원시근
3	3	9	1	3	9	1	3	9	1	3	9	1	
4	4	3	12	9	10	1	4	3	12	9	10	1	
5	5	12	8	1	5	12	8	1	5	12	8	1	
6	6	10	8	9	2	12	7	3	5	4	11	1	원시근
7	7	10	5	9	11	12	6	3	8	4	2	1	원시근
8	8	12	5	1	8	12	5	1	8	12	5	1	
9	9	3	1	9	3	1	9	3	1	9	3	1	
10	10	9	12	3	4	1	10	9	12	3	4	1	
11	11	4	5	3	7	12	2	9	8	10	6	1	원시근
12	12	1	12	1	12	1	12	1	12	1	12	1	

$2^i \bmod 13$ 계산

$2^1 \bmod 13 = 2$
$2^2 \bmod 13 = 4$
$2^3 \bmod 13 = 8$
$2^4 \bmod 13 = 3$
$2^5 \bmod 13 = 6$
⋮
$2^{11} \bmod 13 = 7$
$2^{12} \bmod 13 = 1$

- 따라서

  $\{2^i \bmod 13 \mid i = 1,2, \dots, 12\} = \{1,2,3,4,5,6,7,8,9,10,11,12\}$
- 그러므로 2는 13의 원시근

5.5 구체적 키 교환의 예

1) 앨리스는 밥에게 2개의 수 P=13 과 G=2를 송신
2) 앨리스는 랜덤한 수 A=9를 준비
3) 밥은 랜덤한 수 B=7을 준비
4) 앨리스는 밥에게 $G^A \bmod P = 2^9 \bmod 13 = 5$를 송신
5) 밥은 앨리스에게 $G^B \bmod P = 2^7 \bmod 13 = 11$를 송신

키 교환 예

6) 앨리스는 밥이 보내 온 수 11을 A 제곱해서 P로 mod를 계산

앨리스가 계산한 키 = $(G^B \bmod P)^A \bmod P$

$= 11^A \bmod P$

$= 11^9 \bmod 13$

$= 8$

7) 밥은 앨리스가 보내 온 수 5를 B 제곱해서 P로 mod를 계산

밥이 계산한 키 = $(G^A \bmod P)^B \bmod P$

$= 5^B \bmod P$

$= 5^7 \bmod 13$

$= 8$

제6절 패스워드를 기초로 한 암호(PBE)

6.1 패스워드를 기초로 한 암호란 무엇인가?

6.2 PBE의 암호화

6.3 PBE의 복호화

6.4 솔트의 역할

6.5 패스워드의 역할

6.6 PBE의 개선

6.1 패스워드를 기초로 한 암호란 무엇인가?

- **패스워드를 기초로 한 암호(password based encryption; PBE)**
  - 패스워드를 기초로 해서 만든 키로 암호화를 수행하는 방법
  - RSA사의 PKCS #5 규격으로 규정되어 있는 PBE는 Java의 java.crypto 패키지 등에 내장
  - 암호 소프트웨어 PGP에서 키를 보존

PBE 절차

중요한 메시지의 기밀성을 유지하고 싶다.

↓

메시지를 그대로 디스크에 보존하면 누군가에게 읽혀질 수도 있다.

↓

키(CEK)를 사용해서 메시지를 암호화하자.

↓

하지만 이번에는 키(CEK)의 기밀성을 유지해야 한다.

↓

키(CEK)를 그 대로 디스크에 보존하는 것은 위험하다.

↓

다른 키(KEK)를 사용해서 키(CEK)를 암호화하자.

↓

그렇지만 이번에는 키(KEK)의 기밀성을 유지해야 한다. 이래 가지고는 빙빙 맴도는 것에 지나지 않는다.

↓

그럼 키(KEK)는 패스워드로부터 만들기로 하자.

↓

패스워드만으로 만들면 사전 공격을 받을 위험이 있다.

↓

그렇다면 키(KEK)는 솔트와 패스워드로부터 만들기로 하자.

↓

「솔트」는 암호화한 키(CEK)와 함께 보존해 두고, 키(KEK)는 버리기로 하자.

↓

「패스워드」는 자신의 머릿속에 보존해 두기로 하자.

6.2 PBE의 암호화

1) KEK 생성
2) 세션 키 생성과 암호화
3) 메시지 암호화

KEK 생성

- 의사난수 생성기로 솔트(salt)라는 난수를 생성
- 솔트와, 앨리스가 입력한 패스워드를 순서대로 일방향 해수 함수에 입력
- 얻어진 해시 값이 키의 암호화를 위한 키(KEK)

세션 키 생성과 암호화

- 의사난수 생성기를 사용해서 세션 키를 생성
- KEK를 사용해서 암호화하고, 솔트와 함께 안전한 장소에 보존
- 세션 키의 암호화가 끝나면 KEK는 폐기
  - 솔트와 패스워드만 있으면 KEK는 복원 가능

메시지 암호화

- 세션 키를 사용해서 메시지를 암호화
- PBE의 암호화에서 하는 것
  - 솔트
  - KEK로 암호화된 세션 키
  - 세션 키로 암호화된 메시지
- 「솔트」와 「KEK로 암호화된 세션 키」는 안전한 장소에 보관

PBE의 암호화

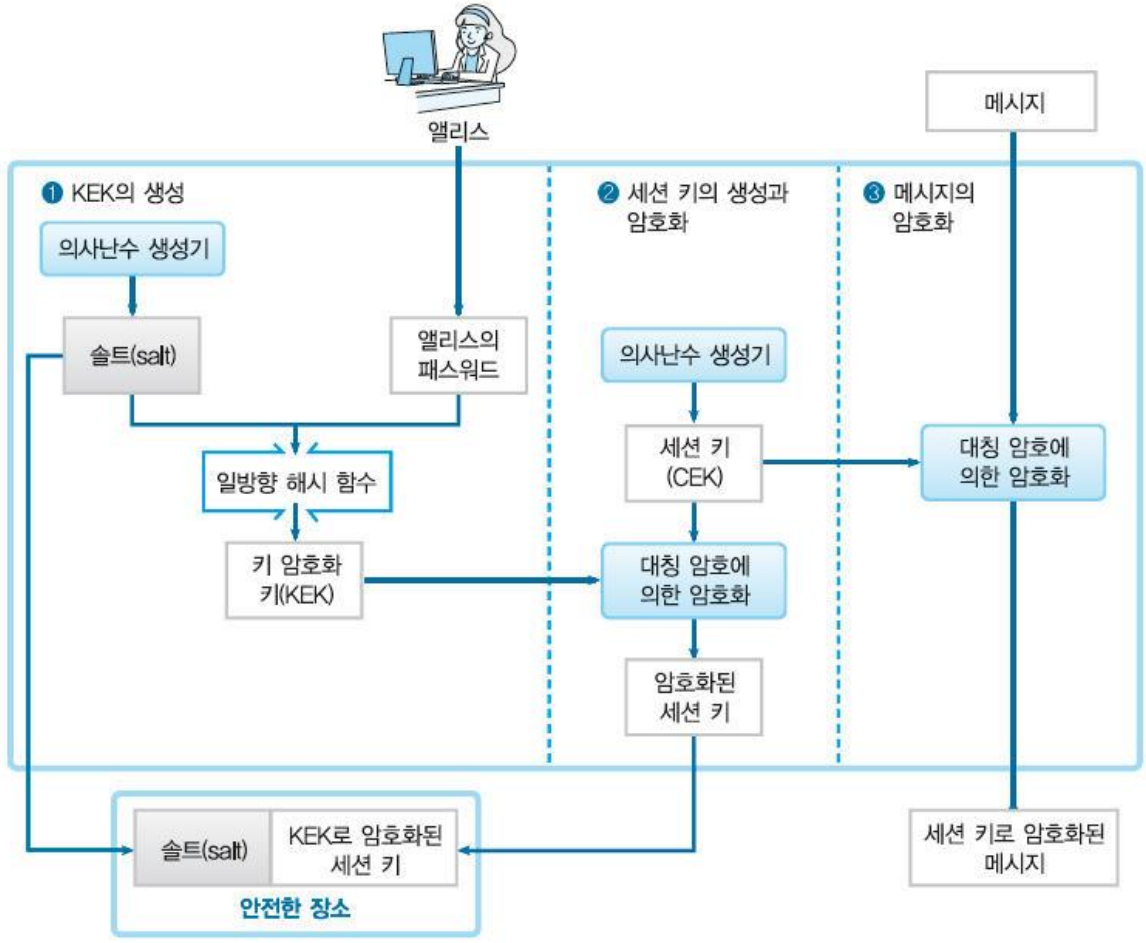


그림 12-8 • PBE의 암호화

6.3 PBE의 복호화

1) KEK 복원
2) 세션 키 복호화
3) 메시지 복호화

KEK 복원

- 보존해 둔 솔트와, 앨리스가 입력한 패스워드를 일방향 해시 함수에 순서대로 입력
- 이것은 KEK를 생성했을 때와 같은 계산이므로 얻어진 해시 값은 KEK

세션 키 복호화

- 세션키 구하기
- 보존해 둔 「KEK로 암호화된 세션 키」를 가지고 와서, (1)에서 복원시킨 KEK를 사용해서 복호화

메시지 복호화

- 복호화한 세션 키를 사용해서 암호화된 메시지를 복호화

PBE의 복호화

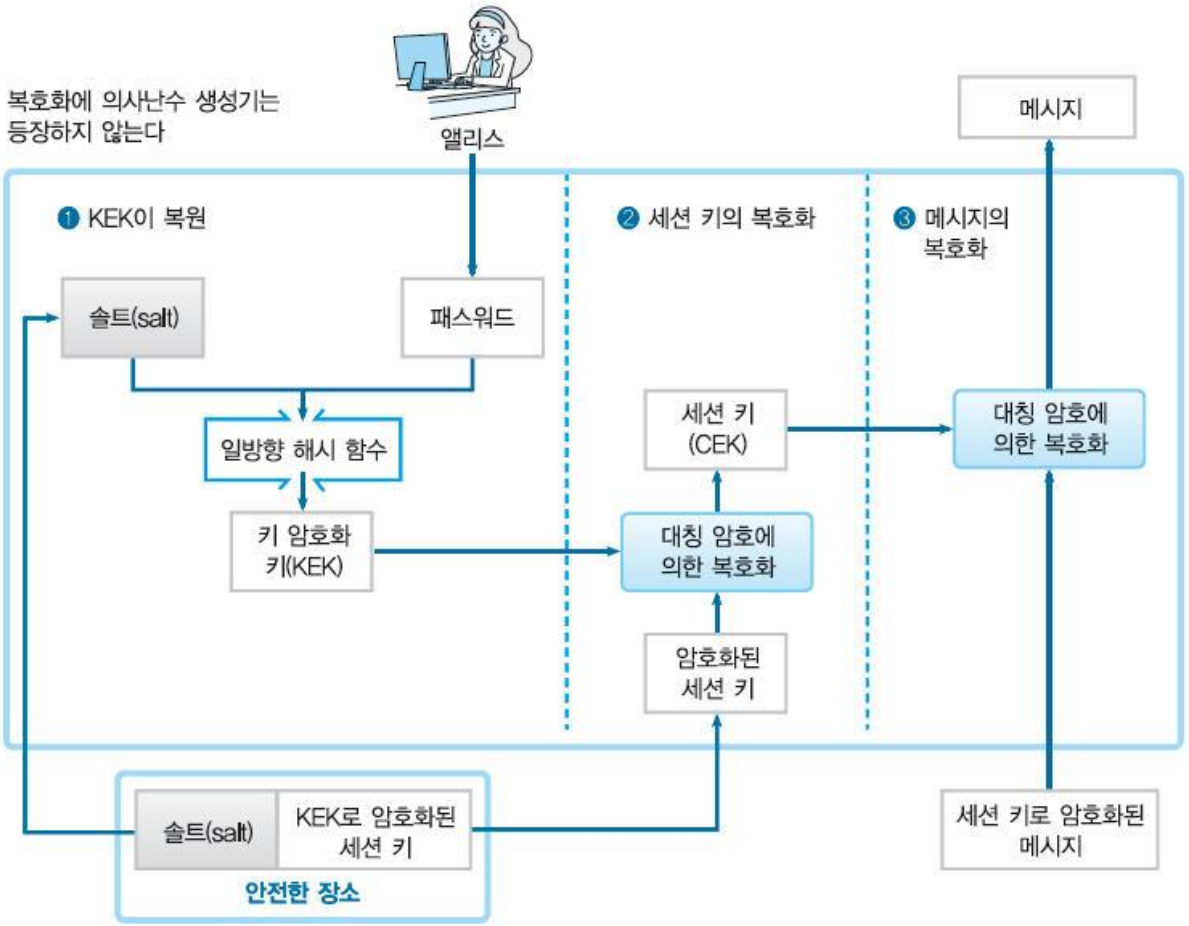


그림 12-9 • PBE의 복호화

6.4 솔트의 역할

- 의사난수 생성기로 만들어지는 랜덤한 수로 키(KEK)를 만들 때에 패스워드와 함께 일방향 해시 함수에 입력
- 사전 공격을 막기 위해 필요

솔트 미사용의 경우

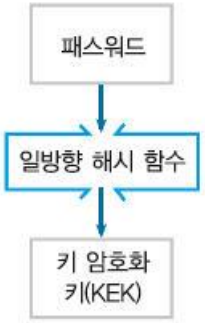
- 적극적 공격자 맬로리는 사전 데이터 등을 기초로 해서 KEK의 후보를 미리 대량으로 만들어 두는 것이 가능
- 암호화된 세션 키를 훔친 다음 복호화를 시도하는데, KEK의 후보를 미리 만들어 둠으로써 시행시간을 대폭 단축할 수가 있다

솔트를 사용할 경우

- KEK 후보의 종류 수가 솔트의 비트 길이만큼 늘어나기 때문에, KEK의 후보를 미리 만들어 놓는다는 것이 매우 어렵다
- 솔트가 확보되지 않으면 KEK의 후보 생성 불가
- 솔트에 의해 KEK의 후보수가 대폭 증가되기 때문

사전 공격과 솔트의 역할

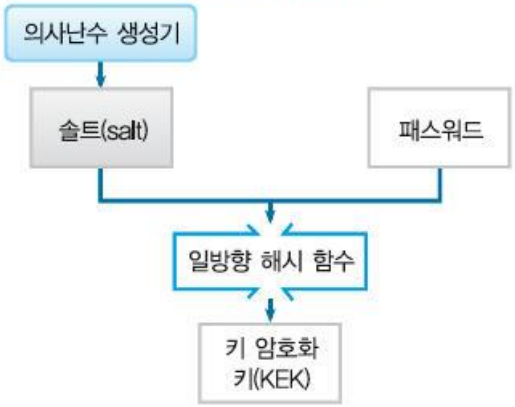
솔트를 사용하지 않은 경우



공격자는 패스워드에 대응하는 KEK값을 미리 계산할 수 있다
(사전 공격이 가능하다)

패스워드	대응하는 KEK의 값
abc	02 E3 29 13 2A D0
abcd	F5 21 62 FE 72 77
abcxyz	81 75 8E B2 9F 66
hello	3E F3 C7 06 DF B7
pass	18 1C 48 22 E6 EF
...	...

솔트를 사용한 경우



솔트가 다르면 비록 패스워드가 같아도
KEK값이 다르므로 사전 공격이 불가능하다

솔트	패스워드	대응하는 KEK의 값
5B94E7	abc	4D 85 FD 69 87 38
E5AB9D	abc	EB 4D CB A9 C3 A4
F8DC3B	abc	09 70 F0 7D AC 20
C6541B	abc	44 40 32 6F AB 16
F6C109	abc	1F C5 3C 14 DF D8
...	...	...

그림 12-10 • 사전 공격과 솔트의 역할

6.5 패스워드의 역할

- 충분한 비트 수를 갖는 패스워드를 기억할 수는 없다
- PBE에서는 패스워드로 만든 키(KEK)로 세션 키(CEK)를 암호화
- 패스워드로 만든 키(KEK)는 의사난수 생성기로 만든 세션 키(CEK)보다도 약하다
- 말하자면 튼튼한 금고의 키를 약한 금고에 보관하고 있는 것과 같은 것
- PBE를 이용하려면 솔트와 암호화한 CEK를 물리적으로 지키는 방법을 병용해야 함
  - 예: CEK를 항상 휴대하고 있는 IC 카드에 보관

6.6 PBE의 개선

- **여러 번의 일방향 해시 사용**
  - KEK를 만들 때 일방향 해시 함수를 여러 번 통과하도록 하면 안전
  - 사용자 입장에서 해시 함수를 1000회 반복
    하는 것은 용이
  - 공격자 맬로리에게는 작은 차이가 큰 부담
    - 바른 KEK를 찾을 때까지 대량의 패스워드를 시도해야만 함

제7절 안전한 패스워드를 만들려면

7.1 자신만이 알 수 있는 정보를 사용할 것

7.2 복수 패스워드를 사용할 것

7.3 메모를 유효하게 사용할 것

7.4 패스워드의 한계

안전한 패스워드 만들기

- 자신만이 알 수 있는 정보를 사용할 것
- 복수의 패스워드를 나누어 쓸 것
- 메모를 유효하게 사용할 것
- 패스워드의 한계를 알 것

7.1 자신만이 알 수 있는 정보를 사용할 것

- **중요한 것의 이름을 사용해서는 안 된다**
  - 배우자 이름, 애인 이름, 아이 이름, 애완동물 이름, 유명인 이름, 자동차 이름, 브랜드명 등
- **자신에 관한 정보를 사용해서는 안 된다**
  - 자신의 이름, 자신의 로그인 명, 주소, 사원 번호 등
- **타인이 보기 쉬운 정보를 사용해서는 안 된다**
  - 명언, 유명한 인용구, 사전의 예문, 웹에서 찾은 말, 키보드의 배열을 이용한 문자열(qwert, asdfghjkl 등), 무지개 색, 혹성의 이름, 성좌, 달 이름, 요일 이름 등

7.2 복수 패스워드를 사용할 것

- 하나의 패스워드를 다양한 용도에 사용해서는 안됨
- 정보의 가치에 따라 패스워드를 구별해서 사용
- 패스워드의 일부만을 바꾸어 복수의 패스워드로 나누어 사용해서는 안됨
  - 예:
    - 회사 컴퓨터의 로그인용: tUniJw1
    - 집 컴퓨터의 로그인용 : tUniJw2
    - 메일의 디지털 서명용 : tUniJw3
    - 온라인 쇼핑용 : tUniJw4

7.3 메모를 유효하게 사용할 것

- 패스워드를 메모에 써 넣고, 컴퓨터 모니터에 붙여 놓아서는 안 됨
- 메모를 유용하게 사용하는 것은 결코 나쁘지 않다
- 메모를 물리적인 키와 동일하게 취급
- 패스워드의 일부분만을 메모해 두는 것은 특히 유효

7.4 패스워드의 한계

- 가정: 영어 알파벳과 숫자열중의 8문자로 한정

- 62개 문자

  ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmn
  opqrstuvwxyz0123456789

- 영어 알파벳과 숫자 8문자로 된 문자열의 가능성

$$62\times 62\times 62\times 62\times 62\times 62\times 62\times 62$$
$$= 62^8$$
$$= 218340105584896$$

- 약 218조 종류

패스워드의 한계

- 키의 비트 수로 말하면 48비트 정도에 지나지 않음
- 이 정도의 길이는 전사 공격이 가능한 길이
- 만약 적극적 공격자의 컴퓨터가 1초간에 1억 개의 패스워드를 만들어서 시험할 수 있다면, 약 25일에 모든 패스워드를 체크할 수 있음

Q & A

Thank You!