알기 쉬운
정보보호개론 3판
흥미로운 암호 기술의 세계
INFORMATION SECURITY and CRYPTOGRAPHY

INFINITYBOOKS
인피니티북스

INFORMATION SECURITY and CRYPTOGRAPHY

CHAPTER 6 공개 키 암호

Section 01
키 배송 문제

1.1 키 배송 문제란?

- **키 배송 문제**(key distribution problem)
  - 대칭 암호를 사용하려면 송신자와 수신자가 대칭키를 사전에 공유해야 하는 문제
  - 대칭 키를 보내지 않으면 받은 복호화할 수 없다
  - 안전하게 키를 보내는 방법은?

키 배송 문제를 해결하기 위한 방법

- 키의 사전 공유에 의한 해결
- 키 배포 센터에 의한 해결
- Diffie-Hellman 키 교환
- 공개 키 암호에 의한 해결

키를 보내 버리면 도청자 이브도 복호화 가능

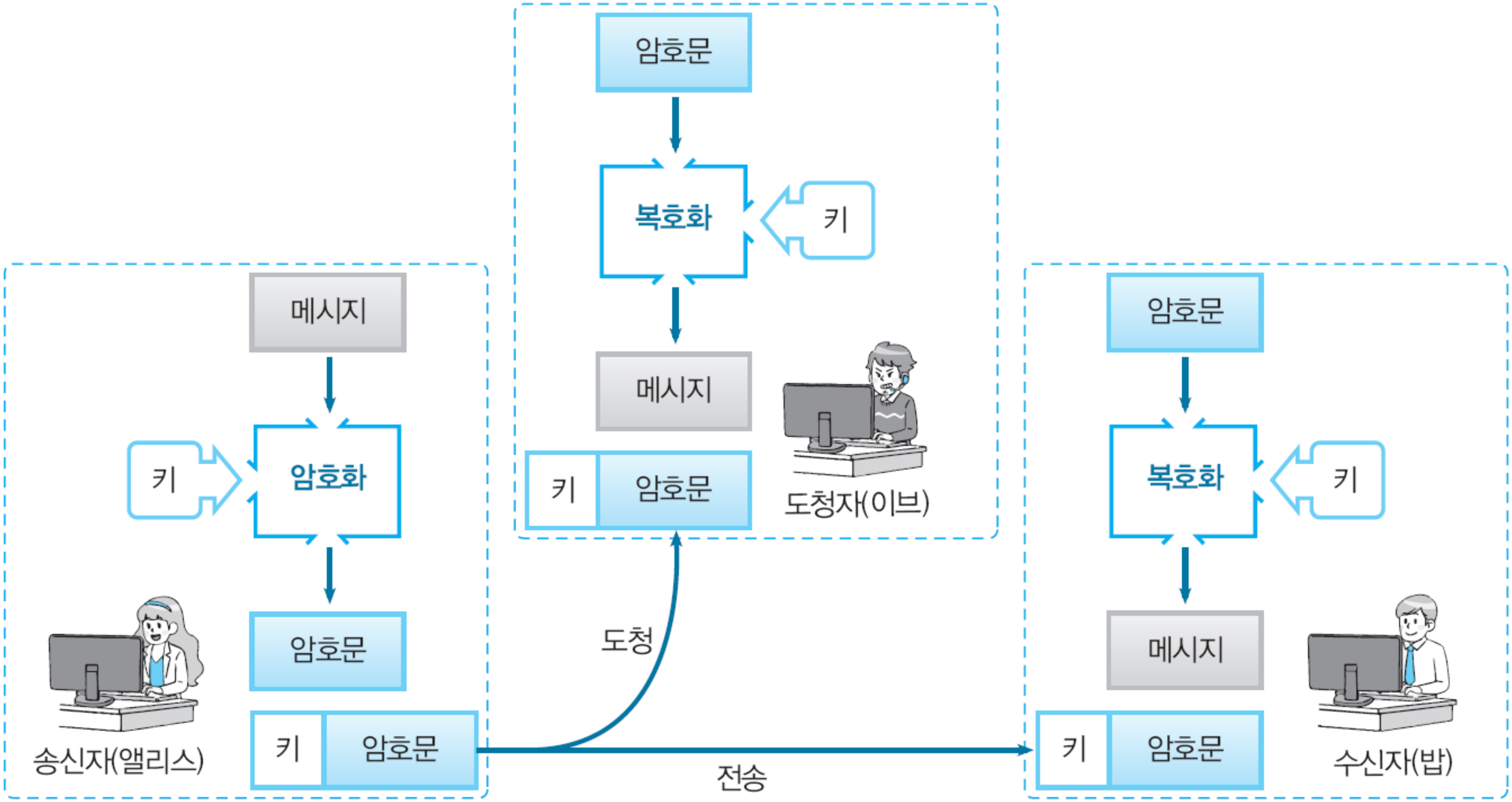

그림 6-1 • 키를 함께 보내면 도청자 이브도 복호화할 수 있다(키 배송 문제)

1.2 키 사전 공유에 의한 키 배송 문제 해결

- 키 사전 공유
  - 「안전한 방법으로 키를 사전에 건네주는」것
  - 직접전달은 안전
  - 이메일/일반메일 등은 위험
  - 인원이 많아지면 관리 해야 할 키 수 증가

사원 1000명 회사

- 1000명의 사원 한 사람 한 사람이 자신 이외의 999명과 통신할 가능성이 있다고 하면, 통신용 키는 1인당 999개가 필요
- 회사 전체로 필요한 키의 수

1000 × 999 ÷ 2 = 49만 9500개

- 현실적이지 못하다

1.3 키 배포 센터에 의한 키 배송 문제 해결

- **키 배포 센터**(key distribution center; KDC)
  - 암호 통신 때마다 통신용의 키를 키 배포 센터에 의뢰해서 개인과 키 배포 센터 사이에서만 키를 사전에 공유
  - 키 배포 센터의 역할을 하는 컴퓨터를 지정
  - 구성원 전원의 키를 보존

앨리스가 밥에게 암호 메일 보내기

1. 앨리스는 키 배포 센터에「밥과 통신하고 싶다」고 신청한다.
2. 키 배포 센터는 의사난수 생성기를 써서 세션 키(K)를 만든다. 이것은 앨리스와 밥이 이번 통신만을 위한 일시적인 키이다.
3. 키 배포 센터는 데이터베이스로부터 앨리스의 키(KA)와 밥의 키(KB)를 꺼낸다.
4. 키 배포 센터는 앨리스의 키를 써서 세션 키를 암호화($C_A = E_{KA}(K)$)해서 앨리스에게 보낸다.
5. 키 배포 센터는 밥의 키를 써서 세션 키를 암호화($C_B = E_{KB}(K)$)해서 밥에게 보낸다.
6. 앨리스는 키 배포 센터로부터 온 세션 키(앨리스의 키로 암호화되어 있음)를 복호화($K=D_{KA}(C_A)$)해서 세션 키를 얻는다.
7. 앨리스는 세션 키를 써서 밥에게 보낼 메일을 암호화($C=E_K(M)$)해서 밥에게 보낸다.
8. 밥은 키 배포 센터로부터 온 세션 키(밥의 키로 암호화되어 있음)를 복호화($K=D_{KB}(C_B)$)해서 세션 키를 얻는다.
9. 밥은 세션 키를 써서 앨리스에게 온 암호문을 복호화($M=D_K(C)$)한다.
10. 앨리스와 밥은 세션 키를 삭제한다.

키 배포 센터에 의한 키 배송

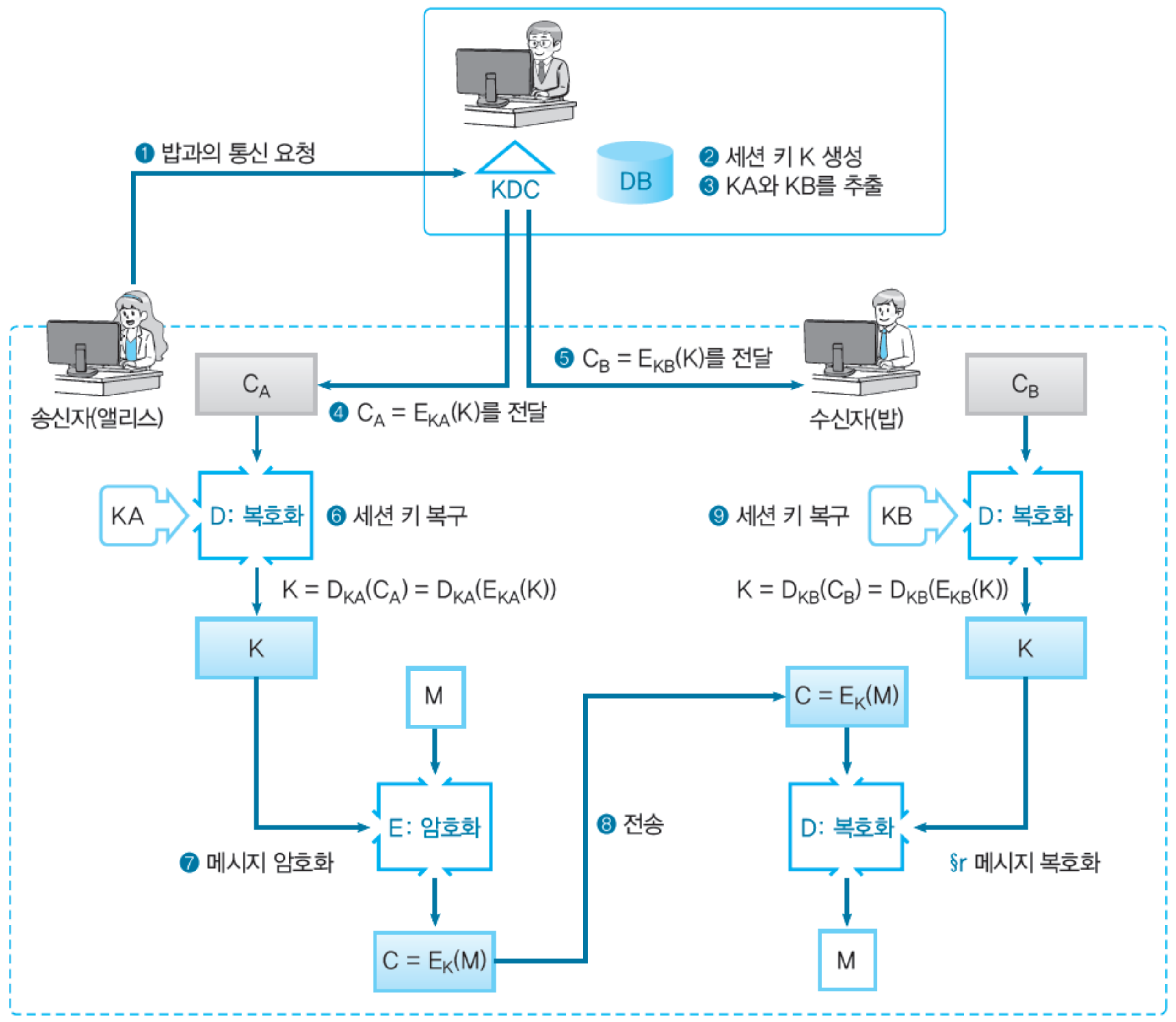

그림 6-2 • 키 배포 센터에 의한 키 배송

키 배포센터의 문제점

- 구성원 수 증가 시 키 배포 센터의 부하
- 키 배포 센터의 컴퓨터가 고장 시 조직 전체의 암호 통신 마비
- 키 배포 센터가 공격의 대상이 될 수 있음
- 키 배포 센터는 신뢰할 수 있는 제 3자이어야 함(트렌트(Trent)로 묘사)

Quiz 1 키 배포 센터의 처리

- 앨리스가 『밥과 통신하고 싶다』고 신청했을 때, 키 배포 센터는 세션키라는 것을 일부러 새로 만들어서, 그것을 암호화해서 앨리스에게 건네 주었다. 어째서 키 배포 센터는 『밥의 키』를 앨리스의 키로 암호화해서 앨리스에게 건네지 않은 것일까?

1.4 Diffie-Hellman 키 교환에 의한 키 배송 문제의 해결

- **Diffie-Hellman 키 교환**
  - 암호 통신을 원하는 두 사람이 있다면 어떤 정보를 교환한다
    - 이 정보는 도청자 이브에게 노출 되어도 무방
  - 두 사람은 교환한 정보를 가지고 동일한 키를 각각 생성할 수 있다
    - 하지만 도청자 이브는 같은 키를 만들 수 없다

1.5 공개 키 암호에 의한 키 배송 문제의 해결

- **공개 키 암호**
  - 대칭 암호
    - 「암호화 키」와 「복호화 키」 동일
  - 공개 키 암호
    - 「암호화의 키」와 「복호화 키」 다르다
    - 「암호화 키」를 가지고 있는 사람이라면 누구든지 암호화할 수 있음
    - 하지만 「암호화 키」를 가지고 있어도 복호화할 수는 없음
    - 복호화 할 수 있는 것은 「복호화 키」를 가지고 있는 사람뿐임

- 수신자는 미리「암호화 키」를 송신자에게 알려 준다.
  - 이「암호화 키」는 도청자에게 알려져도 무방
- 송신자는 그「암호화 키」로 암호화해서 수신자에게 전송
- 암호문을 복호화할 수 있는 자는「복호화 키」를 가지고 있는 사람(수신자)뿐임
- 이렇게 하면「복호화 키」를 수신자에게 배송할 필요가 없음

공개 키 암호를 이용한 키 배송

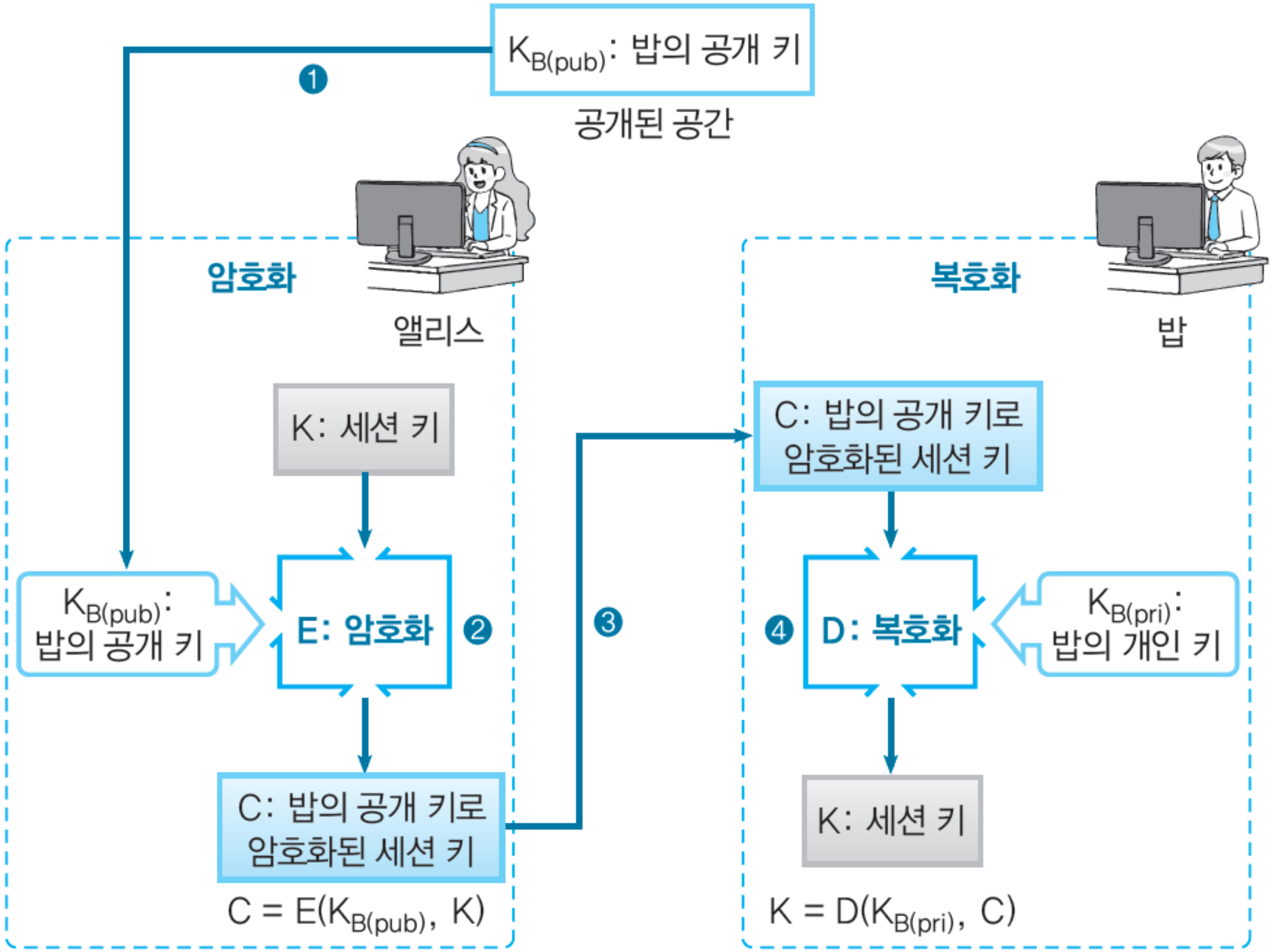

그림 6-3 • 공개 키 암호를 이용한 키 배송

Quiz 2 2개의 암호 알고리즘

- 키 배송 문제 이야기를 들은 앨리스는 이렇게 생각했다.
- 키를 수신자에게 보내면 도청되므로 곤란하다는 것이 키 배송문제라고 생각했다. 키를 그대로 보내니까 문제가 되는 게 아닐까? 먼저 메시지는 AES로 암호화해 둔다. 그리고 나서 암호화에 사용한 AES의 키를 트리플 DES로 암호화해서 보내면 되지 않을까? AES의 키는 트리플 DES로 암호화되어 있으니까 도청되어도 괜찮을 거야.
- 앨리스가 잘못 생각하고 있는 것은 무엇인가?

Section 02
공개 키 암호

2.1 공개 키 암호란?

2.2 공개 키를 사용한 통신의 흐름

2.3 여러 가지 용어

2.4 공개 키 암호로도 해결할 수 없는 문제

2.1 공개 키 암호란?

- **공개 키 암호**(public-key cryptography)
  - 「암호화 키」와 「복호화 키」가 분리
  - 송신자는 「암호화 키」를 써서 메시지를 암호화하고, 수신자는 「복호화 키」를 써서 암호문을 복호화

공개키 암호의 암호화

- 송신자가 필요한 것은 「암호화 키」뿐
- 수신자가 필요한 것은 「복호화 키」뿐
- 도청자에게 알려지면 곤란한 것은 「복호화 키」
- 「암호화 키」는 도청자에게 알려져도 무방
- 수신자가 「복호화 키」를 처음부터 가지고 있고 송신자가 「암호화 키」를 손에 넣을 수 있다면 키 배송문제는 해결됨

공개키의 의미

- **공개 키(public key)- 암호화 키**
  - 「암호화 키」는 일반에게 공개해도 무방
  - 수신자에게 메일로 전달해도 무방
  - 신문의 광고란에 실어도 무방
  - 간판으로 해서 길가에 세워도 무방
  - Web 페이지를 통하여 전 세계에서 읽을 수 있도록 해도 무방
  - 도청자 이브에게 공개 키가 도청되는 것을 신경쓸 필요가 없다

개인키의 의미

- **개인 키(private key)- 복호화 키**
- 「복호화 키」는 미공개
- 이 키는 본인만 사용
- 개인 키는 다른 사람에게 보이거나, 건네 주거나 해서는 안 됨
- 개인 키는 자신의 통신 상대에게도 보여서는 안 됨

공개키-개인키 쌍

- **키 쌍(key pair)**
- 공개 키와 개인 키는 둘이 한 쌍
  - 공개 키로 암호화한 암호문은 그 공개 키와 쌍이 되는 개인 키가 아니면 복호화 할 수 없다
- 수학적인 관계
  - 키 쌍을 이루고 있는 2개의 키는 서로 밀접한 관계
  - 공개 키와 개인 키 쌍은 별개로 만들 수 없음

공개키 암호의 역사

- Whitfield Diffie 와 Martin Hellman(1976)
  - 공개 키 암호의 아이디어를 발표
  - 암호화 키와 복호화 키의 분리성
  - 공개 키가 어떠한 특성을 갖추고 있어야 하는지를 제시
- Ralph Merkle 와 Martin Hellman(1977)
  - 배낭(napsack) 암호
  - 안전하지 않음
- Ron Rivest, Adi Shamir, Leonard Adleman(1978)
  - 공개 키 암호 알고리즘 **RSA** 발표
  - 공개 키 암호의 De Facto Standard
- 영국의 전자통신안전국(CESG, Communication Electronic Security Group) James Ellis 공개키 암호 고안
- CESG Clifford Cocks : RSA와 같은 암호 고안
- CESG Malcolm Williamson : Diffie Hellman과 유사한 알고리즘 고안

2.2 공개 키를 사용한 통신 흐름

- 앨리스가 밥에게 메시지 보내기
  - (1) 밥은 공개 키/개인 키로 이루어진 한 쌍의 키($K_{B(pub)}$/$K_{B(pri)}$)생성
  - (2) 밥은 자신의 공개 키($K_{B(pub)}$)를 앨리스에게 전송
  - (3) 앨리스는 밥의 공개 키를 써서 메시지(P)를 암호화($C=E(K_{B(pub)},P)$)
  - (4) 앨리스는 암호문(C)을 밥에게 전송
  - (5) 밥은 자신의 개인 키($K_{B(pri)}$)를 써서 암호문을 복호화($P=D(K_{B(pri)},C)$)

공개 키를 사용한 메시지 전송

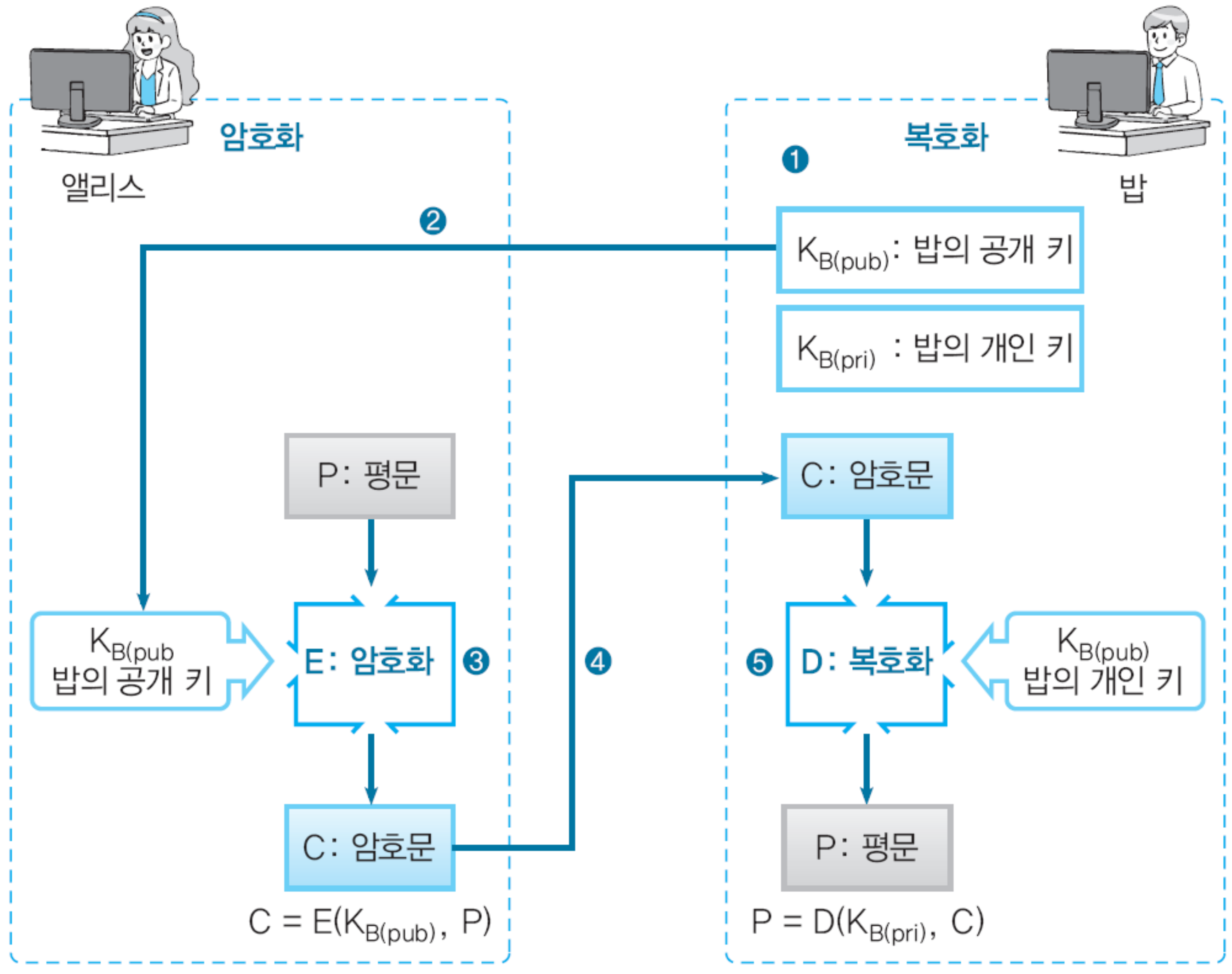

그림 6-4 • 공개 키를 사용해서 앨리스가 밥에게 메시지를 보낸다

2.3 여러 가지 용어

- **대칭 암호**(symmetric cryptography)
  - 동일키 사용해서 암호화와 복호화 수행
  - 암호화와 복호화가 마치 거울처럼 대칭
  - 키: 비밀키(secret key)라고 함
- **비대칭 암호**(asymmetric cryptography)
  - 대칭 암호와의 대비
  - 암호화와 복호화에 다른 키 사용
  - 키: 개인키(private key)와 공개키(public key)

2.4 공개 키 암호로도 해결할 수 없는 문제

- **공개 키의 인증**에 관한 문제
  - 입수한 공개 키의 진위를 판단할 필요
  - 중간자공격(man-in-the-middle attack)
- **공개 키 암호의 속도**
  - 대칭 암호에 비해 처리 속도가 몇 백 배나 늦음

Section 03
시계 연산

바늘이 하나밖에 없는 시계

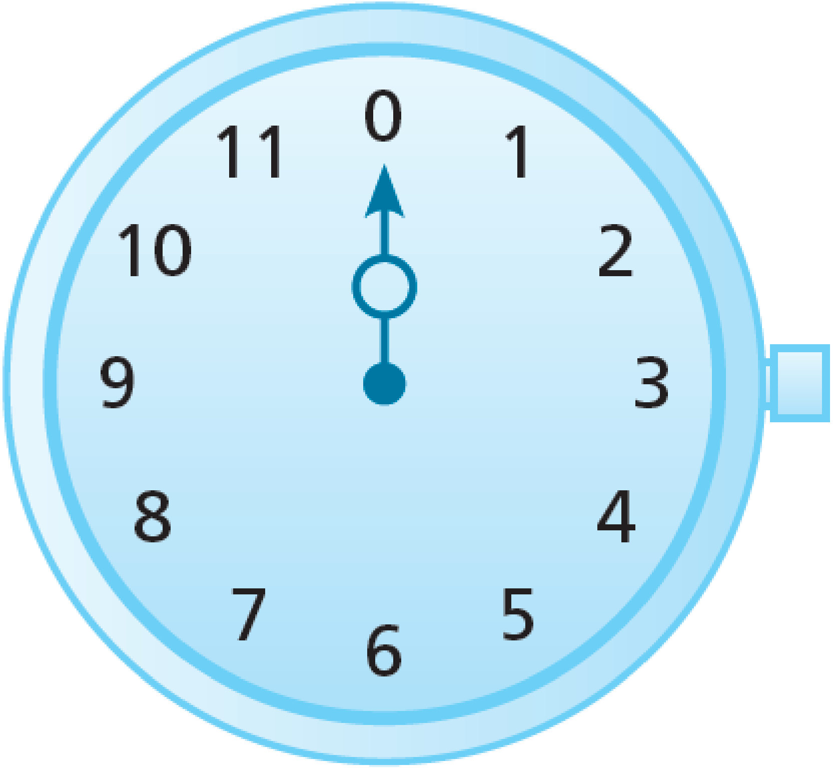

그림 6-5 • 바늘이 하나밖에 없는 시계

3.1 덧셈

- 이 시계를 사용한 덧셈
  - 지금 바늘이 7을 가리키고 있다
    - 오른쪽으로 2 눈금 보내면 바늘은 어디를 가리키는가?
    - 9를 가리킨다.
  - 그럼 바늘이 7을 가리키고 있다
    - 오른쪽으로 6 눈금 보내면 바늘은 어디를 가리키는가?
    - 13일까?

모드 계산

- **mod**
  - 「**나눗셈을 해서 나머지를 구하는 계산**」을 위한 기호(연산자)
- **27 mod 12**
  - 27을 12로 나눈 나머지(27 모드 12)
  - 27 mod 12 = 3
    - 27을 12로 나눈 나머지는 3과 같다
  - 「27과 3은 12를 제수로 해서 **합동**이다」라고 표현

모드 덧셈

- **(7 + 6) mod 12 =?**
- **13 mod 12 = 1**
  - 13을 12로 나눈 나머지는 1
- 시계를 오른쪽으로 돌린다는 것은 덧셈에 해당
- 단, 단순한 덧셈이 아니라「나눗셈의 나머지(mod)」를 생각할 것

3.2 뺄셈

- 뺄셈이라는 것은 덧셈의 역의 연산
- 시계의 바늘을 왼쪽으로 돌리면 되는 것
- 그런데 이 시계는 오른쪽으로만 돌아간다
- 그러면 뺄셈은 어떻게 하면 될까?

7에서 7을 뺀다는 의미는?

- 7에서 7을 빼는 것은 시계 바늘을 왼쪽으로 돌려 0이 되도록 하는 것이다
- 오른쪽으로만 시계 바늘을 돌려서 0에 도착하려면?
- (7 + □) mod 12 = 0
- 7에 무슨 수(□)를 더한 뒤 12로 나누면 나머지가 0이 될까?
- □=5 이다
- 따라서 5가 -7의 역할을 한다

(X + Y) mod 12 = 0 이 되는 X와 Y의 짝

X	Y
0	0
1	11
2	10
3	9
4	8
5	7
6	6
7	5
8	4
9	3
10	2
11	1

3.3 곱셈

- 곱셈은「덧셈을 반복한 것」
- 예:
  - ➢ 7 × 4는 7을 4회 더하기

  $7 \times 4 = 7 + 7 + 7 + 7$
- 시계 연산에서도 마찬가지로 생각
  - – 7 × 4는「7눈금 오른쪽으로 돌리는」조작을 4번 반복
  - – 그 다음 모드를 취함

7 × 4 mod 12

- 7 × 4 mod 12 = 28 mod 12 (7 × 4는 28이므로)
- = 4 (28 ÷ 12는 몫이 2이고 나머지가 4이므로)
- 위의 계산을 통해 4가 구해졌다. 실제로 「「7눈금 오른쪽으로 돌리는」 조작을 4번 반복하면」, 분명히 바늘은 4를 가리킨다.

3.4 나눗셈

- 뺄셈을 생각할 때 덧셈의 역 연산을 생각한 것처럼 곱셈의 역 연산을 생각
- 예:
- 7 × □ mod 12 = 1
  - 7에 □를 곱해서 12의 mod를 취했더니 1이 되었다. 이때의 □는 무엇일까?
  - 바늘의 조작으로 생각하면 「「7눈금 오른쪽으로 돌리는」 조작을 몇 회 반복하면 1이 되는가?」하는 문제
  - 이것은 금방은 알 수 없다.

□에 0, 1, 2, ...을 순서대로 넣어서 7 × □ mod 12를 계산

□의 값	7×□의 값	7×□ mod 12의 값
0	0	0
1	7	7
2	14	2
3	21	9
4	28	4
5	35	11
6	42	6
7	49	1
8	56	8
9	63	3
10	70	10
11	77	5

- □는 7이라는 것을 알 수 있다.
- 7 × 7 mod 12 = 1

모드 나눗셈이란?

- 「mod 12의 세계에서는 7에 무엇을 곱하면 1이 되는가?」라는 문제
  - 바꿔 말하면 「mod 12의 세계에서는 1 ÷ 7의 답은 무엇인가?」라는 문제
  - 즉, 12를 모드로 하는 세계에서 나눗셈을 생각한 것
- 일반적으로 정수의 세계에서는 1을 7로 나누어 보면 나눠지지 않는다. 하지만, 12를 제수로 하는 세계에서는 나머지가 0이 될 수 있다

○ × □ mod 12 = 1

- ○와 □는 모드 12의 세계에서 역수 관계
  - 역수란 서로 곱하면 1이 되는 수
- 보통 산수에서 표현을 한다면

$$\bigcirc \times \frac{1}{\bigcirc} = 1$$

- 즉, ○의 역수 □ 는 $\frac{1}{\bigcirc}=\square$ 이어야 한다
- 하지만 $\frac{1}{\bigcirc}$ 과 같은 표현은 모드 12의 세계에서는 다르게 표현되어야 한다

모드 12에서 곱셈에 대한 역원

- 그러면 0부터 11까지의 모든 수 ○이 역수를 가질까?
- 모드계산에서 「어떤 수의 역수가 존재하는지 어떤지」하는 문제는, 공개 키 알고리즘 RSA에서 「공개 키와 쌍을 이루는 개인 키가 존재하는지 어떤지」하는 문제와 직결

역수 계산

- **0의 역수는 있는가?**
  - 0 눈금의 회전(즉 돌리지 않는 것)을 아무리 반복해도 1에 바늘이 도달 할 수 없으므로,

  0 × □ mod 12 = 1

  을 충족시키는 □는 존재하지 않는다
- **1의 역수는 어떤가?**

  1 × □ mod 12 = 1

  을 충족시키는 □는 분명히 1이다.

역수 계산

- **2의 역수는 있는가?**

  $2 \times \square \bmod 12 = 1$

  을 충족시키는 □는 없다

  – 왜냐 하면 2눈금의 회전을 반복해도 0, 2, 4, 6, 8, 10, 0, 2, 4, 6, 8, …처럼 짝수인 곳만을 가리키기 때문

- 3, 4, … 의 역수가 있는지 계속해서 확인해 보자

역수 계산

- 1 × □ mod 12 = 1 → □ = 1
- 2 × □ mod 12 = 1 → □는 존재하지 않는다
- 3 × □ mod 12 = 1 → □는 존재하지 않는다
- 4 × □ mod 12 = 1 → □는 존재하지 않는다
- 5 × □ mod 12 = 1 → □ = 5
- 6 × □ mod 12 = 1 → □는 존재하지 않는다
- 7 × □ mod 12 = 1 → □ = 7
- 8 × □ mod 12 = 1 → □는 존재하지 않는다
- 9 × □ mod 12 = 1 → □는 존재하지 않는다
- 10 × □ mod 12 = 1 → □는 존재하지 않는다
- 11 × □ mod 12 = 1 → □ = 11

역수를 갖는 수

- 역수를 가지고 있는 수는 1, 5, 7, 11
  - 그러면 이들 수에는 어떤 성질이 있는가?
  - 5, 7, 11이라는 수로부터 「소수인가」라고 생각할지도 모른다
  - 하지만 소수하고는 약간 다르다.
  - 2나 3은 소수이지만, 2나 3은 역수를 갖지 않는다
- mod 12의 세계에서 역수를 갖는 수는 12와 그 수가 1이외의 공통의 약수를 갖지 않는 수이다

역수를 갖는 수

- 1 × □ mod 12 = 1 → □ = 1
- 2 × □ mod 12 = 1 → □는 존재하지 않는다
  … 2와 12는 모두 2로 나뉘어 떨어진다
- 3 × □ mod 12 = 1 → □는 존재하지 않는다
  … 3과 12는 모두 3으로 나뉘어 떨어진다
- 4 × □ mod 12 = 1 → □는 존재하지 않는다
  … 4와 12는 모두 4로 나뉘어 떨어진다
- 5 × □ mod 12 = 1 → □ = 5
- 6 × □ mod 12 = 1 → □는 존재하지 않는다
  … 6과 12는 모두 6으로 나뉘어 떨어진다
- 7 × □ mod 12 = 1 → □ = 7
- 8 × □ mod 12 = 1 → □는 존재하지 않는다
  … 8과 12는 모두 4로 나뉘어 떨어진다
- 9 × □ mod 12 = 1 → □는 존재하지 않는다
  … 9와 12는 모두 3으로 나뉘어 떨어진다
- 10 × □ mod 12 = 1 → □는 존재하지 않는다
  … 10과 12는 모두 2로 나뉘어 떨어진다
- 11 × □ mod 12 = 1 → □ = 11

최대 공약수를 이용한 표현

- 1 × □ mod 12 = 1 → □ = 1 ... 1과 12의 최대공약수는 1
- 5 × □ mod 12 = 1 → □ = 5 ... 5와 12의 최대공약수는 1
- 7 × □ mod 12 = 1 → □ = 7 ... 7과 12의 최대공약수는 1
- 11 × □ mod 12 = 1 → □ =11 ... 11과 12의 최대공약수는 1

- 2 × □ mod 12 = 1 → □는 존재하지 않는다 ... 2와 12의 최대공약수는 1이 아니다
- 3 × □ mod 12 = 1 → □는 존재하지 않는다 ... 3과 12의 최대공약수는 1이 아니다
- 4 × □ mod 12 = 1 → □는 존재하지 않는다 ... 4와 12의 최대공약수는 1이 아니다
- 6 × □ mod 12 = 1 → □는 존재하지 않는다 ... 6과 12의 최대공약수는 1이 아니다
- 8 × □ mod 12 = 1 → □는 존재하지 않는다 ... 8과 12의 최대공약수는 1이 아니다
- 9 × □ mod 12 = 1 → □는 존재하지 않는다 ... 9와 12의 최대공약수는 1이 아니다
- 10 × □ mod 12 = 1 → □는 존재하지 않는다 ... 10과 12의 최대공약수는 1이 아니다

서로 소(relatively prime)

- 12와의 최대공약수가 1인 수(5, 7, 11)는 수학에서는 12와 **서로 소**라고 한다
- 「12와 서로 소인 수」는 말하자면 「12에 있어서의 소수」이다

3.5 거듭제곱

- 곱셈이 덧셈을 반복한 것인 것처럼 거듭제곱은 곱셈을 반복한 것
  - 예를 들면 7^4, 즉 「7의 4제곱」은 7을 4회 곱한 것이다.

$7^4 = 7 \times 7 \times 7 \times 7$

- 「시계에서의 거듭제곱」
  - 이번에는 곱셈의 반복
- 풀어쓰면 다음과 같은 표현
  - 「「「「7눈금 오른쪽으로 돌린다」를 7회 반복한다」를 7회 반복한다」를 7회 반복한다」

7^4 mod 12=?

$$7^4 \bmod 12 = 7 \times 7 \times 7 \times 7 \bmod 12$$

$$= 2401 \bmod 12$$

($\because$ $7 \times 7 \times 7 \times 7$은 2401)

$$= 1$$

($\because$ 2401÷ 12는 몫이 200이고 나머지가 1이다)

7^4 mod 12 의 간편식

- 7^4을 전부 계산하고 나서 mod를 취하는 대신에 계산의 도중에 mod를 취해도 같은 결과

$$
\begin{aligned}
&7 \times 7 \times 7 \times 7 \bmod 12 \\
&=[(7 \times 7 \bmod 12) \times (7 \times 7 \bmod 12)] \bmod 12 \\
&=[(49 \bmod 12) \times (49 \bmod 12)] \bmod 12 \\
&=[(1 \bmod 12) \times (1 \bmod 12)] \bmod 12 \\
&=1 \times 1 \bmod 12 \\
&=1 \bmod 12
\end{aligned}
$$

3.6 대수

- 거듭제곱의 역 연산은 대수라 불리고 있다. 보통의 수학에서는 대수를 구하는 계산은 그다지 어렵지 않다. 예를 들면,

  $$7^X = 49$$

  에서 X는 2라는 것은 금방 알 수 있다
- 비록 숫자가 커져도 대수를 구하는 계산은 그다지 어렵지 않다
- 사실 수학적으로 $X = \log_7 49 = \log_7 (7)^2 = 2$

이산 대수(discrete logarithm)

- 시계 계산에 있어서의 대수는 **이산 대수**라고 한다. 예를 들면,

$$7^X \bmod 13 = 8$$

이 되는 X는 무엇이며, 어떻게 구할까?

- 다음과 같이 조사해 가면 X는 9라는 것을 알 수 있다

7^X mod 13 = 8 이 되는 X 구하기

- $7^0 \bmod 13 = 1$
- $7^1 \bmod 13 = 7$
- $7^2 \bmod 13 = 10$
- $7^3 \bmod 13 = 5$
- **$7^4 \bmod 13 = 9$**
- $7^5 \bmod 13 = 11$
- $7^6 \bmod 13 = 12$
- $7^7 \bmod 13 = 6$
- $7^8 \bmod 13 = 3$
- $7^9 \bmod 13 = 8$

이산대수의 활용

- 모드로 사용되는 숫자가 매우 크면 이산 대수 계산이 매우 어렵고 시간이 대단히 많이 걸린다
- 이산 대수 구하기 고속 알고리즘이 없다
  - 이 두 가지 사실이 비대칭 암호 RSA의 안전성을 보장해준다
- 응용
  - Diffie-Hellman 키 교환
  - ElGamal 방식의 공개키 암호

3.7 시계의 바늘에서 RSA로

이제

$$7^4 \bmod 12$$

를 보고 당황하지 않을 수 있는가?

침착하게 7을 4제곱해서 12로 나눈 나머지라고 읽을 수 있는가?

이제 RSA를 이해할 준비가 되었다

Quiz 3 거듭제곱의 mod

- 아래의 식의 값을 구하시오.

$$7^{16} \bmod 12$$

Section 04

RSA

4.1 RSA란 무엇인가?

- **RSA**는 공개 키 암호 알고리즘의 하나
  - RSA 이름
    - 개발자 3명의 이름
    - Ron Rivest, Adi Shamir, Leonard Adleman의 이니셜 (**R**ivest-**S**hamir-**A**dleman)
  - 응용
    - 공개 키 암호
    - 디지털 서명
    - 키 교환

4.2 RSA에 의한 암호화

- RSA에서 평문도 키도 암호문도 숫자로 변환한 뒤 실행
- RSA의 암호화는 다음 식으로 표현

암호문 = (평문)^E mod N

(RSA에 의한 암호화)

E와 N은 무엇일까?

- **(E, N): 공개 키**
  - E(Encryption)와 N(Number)이라는 한 쌍의 수를 알면 누구라도 암호화를 행할 수 있다
  - E와 N이 RSA 암호화에 사용되는 키
  - E와 N은 면밀한 계산을 통해 생성

4.3 RSA에 의한 복호화

- 복호화도 간단하다

평문 = (암호문)^D mod N

(RSA의 복호화)

D와 N은 무엇일까?

- **(D, N): 개인 키**
  - D(Decryption)와 N(Number)이라는 한 쌍의 수를 알면 누구라도 복호화를 행할 수 있다
  - D와 N이 RSA 복호화에 사용되는 키
  - D와 N도 면밀한 계산을 통해 생성
  - E와 D는 밀접한 연관관계

RSA의 암호화 · 복호화

키 쌍	공개 키	수 E와 수 N
	개인 키	수 D와 수 N
암호화		암호문 = (평 문)^E mod N (평문을 E제곱해서 N으로 나눈 나머지)
복호화		평 문 = (암호문)^D mod N (암호문을 D제곱해서 N으로 나눈 나머지)

RSA의 암호화와 복호화

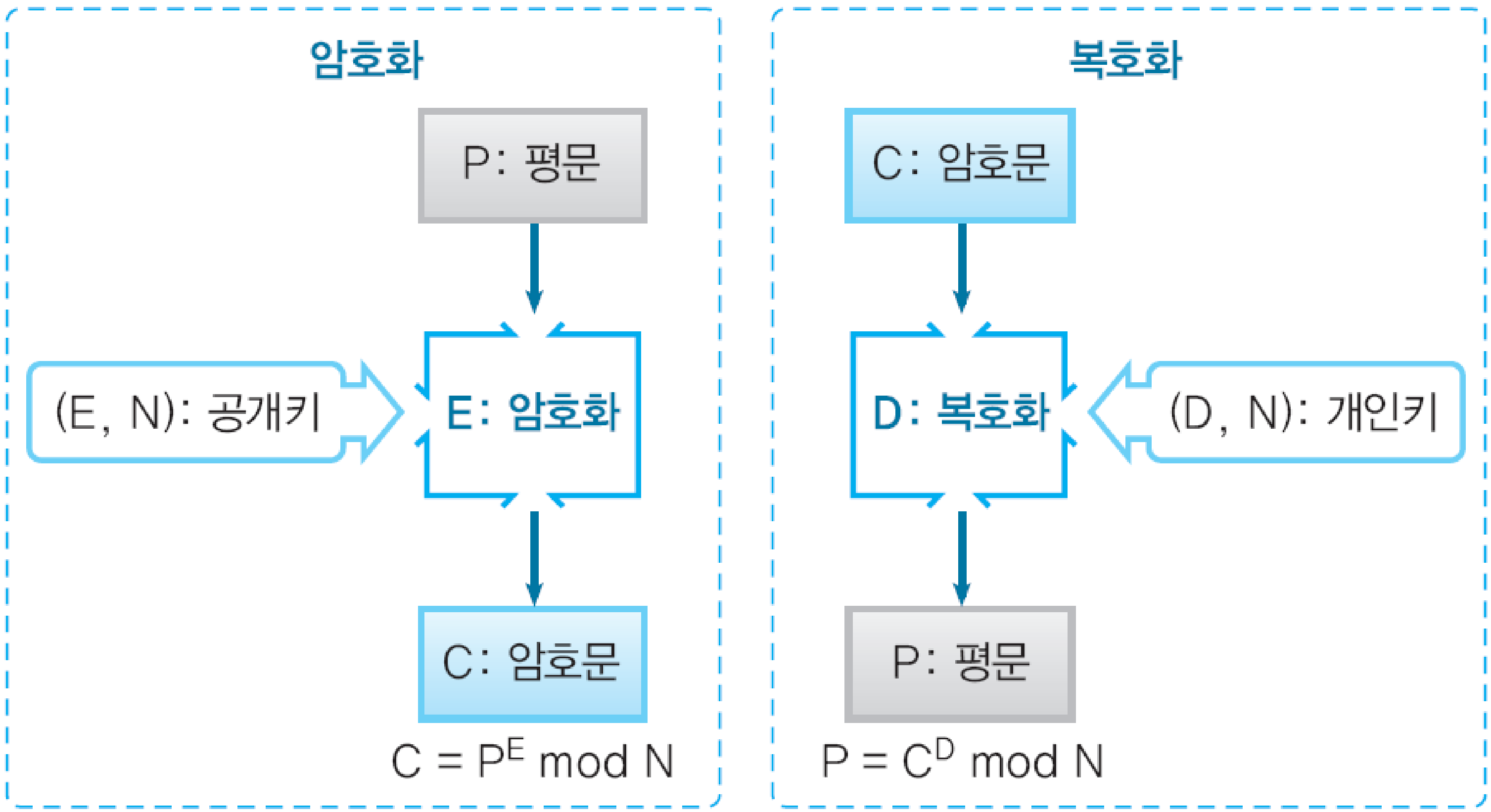

그림 6-6 • RSA 암호화와 복호화

4.4 키 쌍의 생성

1) N을 구한다
2) L을 구한다
(L은 키 쌍을 생성할 때만 등장하는 수이다)
3) E를 구한다
4) D를 구한다

N 구하기

- 큰 소수를 2개 준비(p와 q)
- N = p × q (p, q는 소수)
- p,q의 크기가 작으면 암호해독이 용이하지만, 크면 처리 시간이 길어짐

L 구하기

- L 은 RSA의 암호화나 복호화에 사용안 함
- 키 쌍을 만들 때 임시로 사용
- L = lcm(p-1, q-1)

(L은 p-1 과 q-1의 최소공배수)

(lcm;least common multiple)

E 구하기

- 다음 두 식을 만족하는 수 E를 하나 찾아 낸다
- 1 < E < L
- gcd(E, L) = 1 (E와 L은 서로 소)

(gcd ; greatest common divisor)

D 구하기

- 다음 두 식을 만족하는 수 D를 하나 찾아 낸다
- $1 < D < L$
- $E \times D \bmod L = 1$

RSA 키 쌍 생성

(1) N을 구한다 의사난수 생성기로 p와 q를 구한다 p와 q는 소수 $N = p \times q$
(2) L을 구한다 $L = \mathrm{lcm}(p-1, q-1)$ L은 p-1과 q-1의 최소공배수
(3) E를 구한다 $1 < E < L$ $\gcd(E, L) = 1$ E와 L과의 최대공약수는 1(E와 L은 서로 소)
(4) D를 구한다 $1 < D < L$ $E \times D \bmod L = 1$

RSA 키 쌍

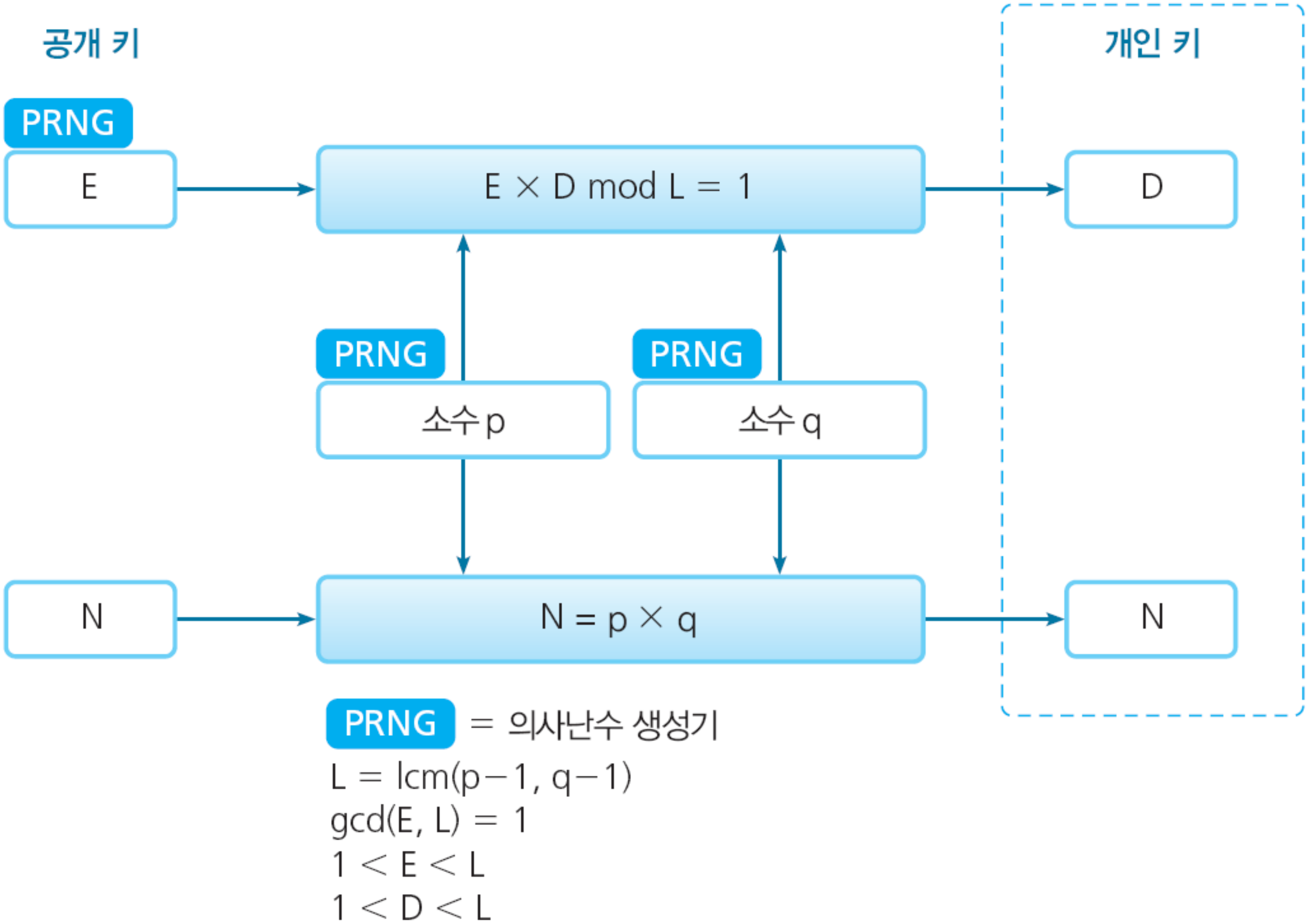

그림 6-7 • RSA의 키 쌍

4.5 구체적 계산

- 구체적인 수를 써서 RSA의 키 쌍 생성 · 암호화 · 복호화를 실제로 구현
- 너무 큰 수(p와 q)를 사용하면 계산이 힘들기 때문에 작은 수를 이용하여 계산

RSA 예

- p 와 q 선택하기
  - 2개의 소수 p=17, q=19 선택
- N 구하기
  - N = p × q = 17 × 19 = 323
- L 구하기
  - L = lcm(p-1, q-1) = lcm(16, 18) = 144 (16과 18의 최소공배수)
- E 구하기(선택하기)
  - gcd(E, L) = 1 이 되는 수 E 를 선택하자.
  - E가 될 수 있는 수는 다음과 같은 수이다.
  - 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, …
  - 우리는 E=5를 선택(다른 수를 선택해도 무방)
- D 구하기
  - E × D mod L = 5 × 29 mod 144 = 145 mod 144 = 1 이므로 D=29

RSA 예

- 공개 키: (E, N) = (5, 323)
- 개인 키: (D, N) = (29, 323)

암호화

- 평문은 N=323 보다 작은수
- 예로 평문=123이라 하고 암호화를 해보자

$$평문^{E} \bmod N = 123^{5} \bmod 323$$
$$= 225$$

복호화

$$\text{암호문}^{D} \bmod N = 225^{29} \bmod 323$$

$$= 123$$

칼럼 225^{29} mod 323의 계산

- $29=10+10+9$
- $225^{29} = 225^{10+10+9} = 225^{10} \times 225^{10} \times 225^{9}$
  - $225^{10} = 332525673007965087890625$
  - $225^{9} = 1477891880035400390625$
  - $225^{10} \bmod 323 = 332525673007965087890625 \bmod 323 = 16$ ...... (1)
  - $225^{9} \bmod 323 = 1477891880035400390625 \bmod 323 = 191$ ...... (2)

$$\begin{aligned}
225^{29} \bmod 323 &= 225^{10} \times 225^{10} \times 225^{9} \bmod 323 \\
&= \underline{(225^{10} \bmod 323)} \times \underline{(225^{10} \bmod 323)} \times \underline{(225^{9} \bmod 323)} \bmod 323 \\
&= 16 \times 16 \times 191 \bmod 323 \\
&= 48896 \bmod 323 \\
&= 123
\end{aligned}$$

- 따라서 $225^{29} \bmod 323 = 123$

Section 05
RSA에 대한 공격

5.1 암호문으로부터 평문 구하기

5.2 전사 공격

5.3 E와 N으로부터 D 구하기

5.4 중간자 공격

해독자(공격자)가 가진 정보

- 암호 해독자가 알고 있는 것
  - 암호문 : 도청해서 구한다
  - E와 N : 공개 키로서 공개
- 암호 해독자가 모르는 것
  - 평문 : 지금부터 해독하려고 하는 내용
  - D : 개인 키 중 적어도 D는 모름
  - 기타 : 키 쌍을 만든 p, q, L을 모름

5.1 암호문으로부터 평문 구하기

$$암호문 = (평문)^{E} \bmod N$$

에서 평문을 구하려면 이산 대수 문제를 풀어야 함

- 이산 대수 문제는 매우 곤란
- 현재까지 아직 이산 대수를 구하는 빠른 방법을 알지 못함

5.2 전사 공격

- 전사공격(brute-force attack)
  - D의 후보가 되는 수를 순서대로 모두 시도해서 복호화 해본다
  - D의 비트 수가 크면 클수록 어려워진다
  - 비트 수가 충분히 크면 전사공격으로 수 D를 찾아내는 것은 현실적으로는 불가능
  - RSA에서는 p와 q의 비트 수로서 512 비트 이상을 사용
  - N은 1024 비트 이상을 이용
  - E나 D는 N과 같은 정도의 크기로 할 수 있으므로 D를 찾으려면 1024 비트 이상의 전사공격이 필요
  - 현실적으로 불가능

5.3 E와 N으로부터 D 구하기

$$E \times D \bmod L = 1$$

- L은 lcm(p-1, q-1)이므로 E로부터 D를 계산할 때는 p와 q를 사용
- 암호해독자는 p와 q를 전혀 모름
- 해독자는 D를 구할 수 없음
- RSA의 안전성을 위해 소수 p와 q를 암호해독자가 모르게 해야 함

N의 소인수 분해하는 공격

- $N = p \times q$라는 관계식을 공격자는 알고 있고 N은 공개되어 있다
- N으로부터 p와 q를 구할 수는 없는 것일까?
- p와 q는 소수이기 때문에 N으로부터 p와 q를 구한다는 것은 자연수 **N을 소인수분해하는 것**

소인수 분해

- 큰 수를 고속으로 소인수분해 할 수 있는 방법이 발견되면 RSA를 깰 수 있다
- 그러나 현재 큰 수의 소인수분해를 고속으로 행하는 방법은 아직 발견되지 않았다
- 소인수분해를 간단히 수행하는 방법이 존재하는지의 여부도 아직 모른다
- 학생들도 한 번 시도해보기 바란다

p 와 q 추측하는 공격

- 소인수분해를 하지 않아도 p와 q가 암호 해독자에게 알려질 가능성은 있다
- p와 q는 의사난수 생성기로 생성하기 때문에 의사난수 생성기의 품질이 나쁘면 p와 q를 암호 해독자가 추측할 수 있다
- 난수 생성기가 강력해서 암호 해독자가 추측할 수 없어야 한다

기타 공격

- N을 소인수분해 해서 p와 q를 구할 수 있으면 D를 구할 수 있다
- 「D를 구하는 것」이 「N을 소인수분해 하는 것」과 수학적 같은지 아닌지가 증명되어 있지 않다
- 「D를 구하는 것」이 「N을 소인수분해 하는 것」이 결정적 다항식 시간으로 같다는 것을 2004년 알렉산더 메이(Alexander May)가 증명함

5.4 중간자 공격

- **중간자(man-in-the-middle) 공격**
- RSA를 해독하는 것은 아니다
- 기밀성을 침해하는 공격
- 공격자 맬로리가 송신자와 수신자 사이에서 송신자에 대해서는 수신자처럼, 수신자에 대해서는 송신자처럼 행세하는 공격

중간자공격 절차

1) 앨리스는 밥의 공개 키 요청
2) 맬로리는, 앨리스의 요청을 도청
3) 밥은 자신의 공개 키($K_{B(pub)}$)를 앨리스에게 전송
4) 맬로리는 밥의 이 메일이 앨리스에게 도달하지 못하도록 하고, 밥의 공개 키를 보존
5) 맬로리는 자신의 공개 키($K_{M(pub)}$)를 밥의 공개 키라고 속여서 앨리스에게 전송
6) 앨리스는 자신의 메시지(P)를 밥의 공개 키(실은 맬로리의 공개 키)로 암호화($C=E(K_{M(pub)}, P)$)

중간자공격 절차

7) 앨리스는 암호화한 메시지(C)를 밥에게 전송
8) 맬로리는 앨리스의 암호 메일을 갈취해서 자신의 개인키($K_{M(pri)}$)로 복호화($P=D(K_{M(pri)},C)$) 하고 평문(P)을 확보
9) 맬로리는 앨리스 행세를 하며 위조 메일(P′)을 만들고 위의 단계 (4)에서 보존해 둔 밥의 공개키($K_{B(pub)}$)를 써서 이 위조 메일을 암호화($C'= E(K_{B(pub)}, P')$)하여 밥에게 전송
10) 밥은 받은 암호 메일(C′)을 자신의 개인 키로 복호화하고 메일(P′)을 읽게 된다

맬로리에 의한 중간자 공격

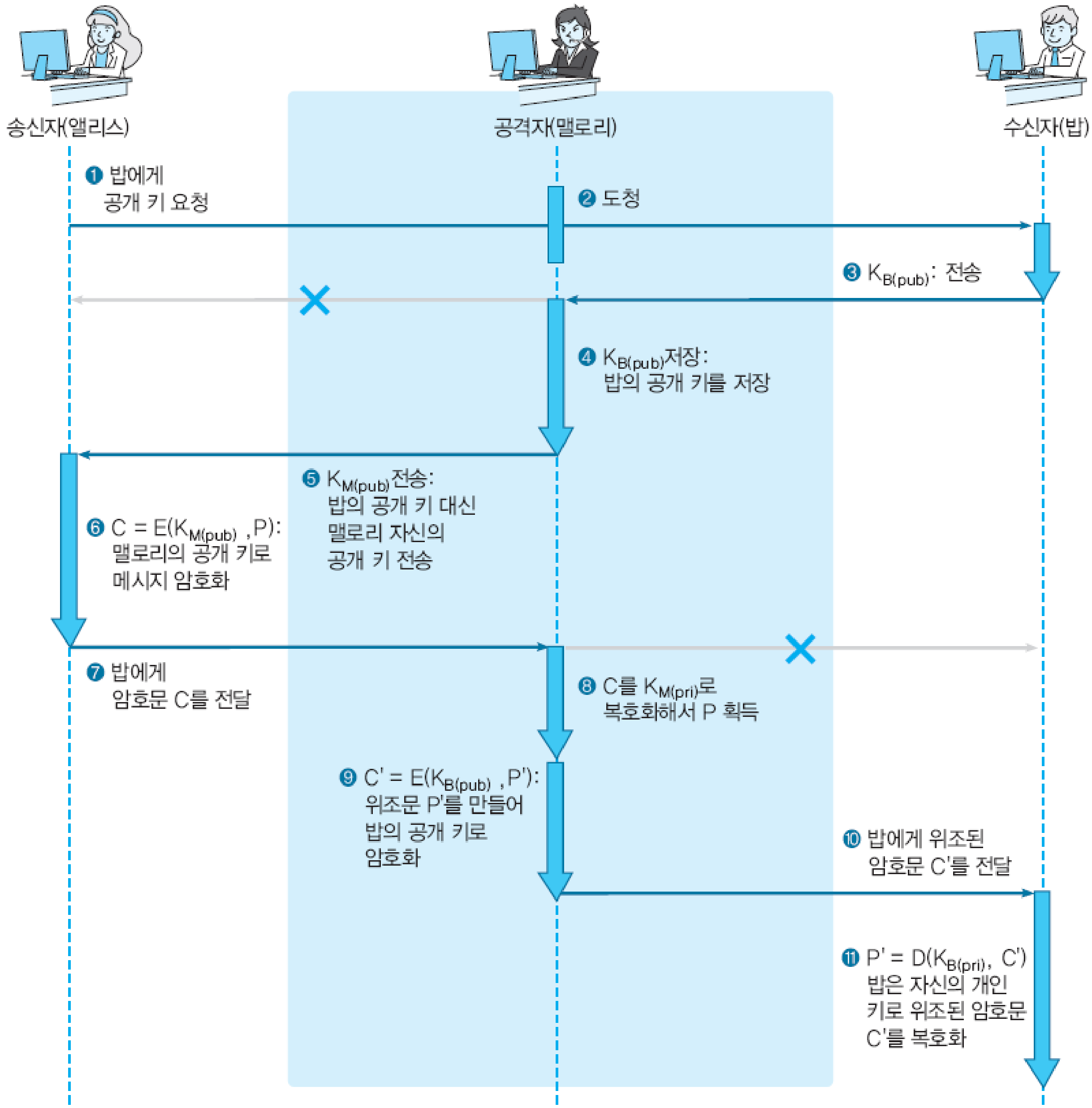

그림 6-8 • 맬로리에 의한 man-in-the-middle 공격

Section 06
선택 암호문 공격

- 복호 오라클(Decryption Oracle)
  - 임의의 데이터를 송신하면 그것을 암호문으로 간주하고 회신 해주는 서비스
- 선택암호문 공격(Chosen Ciphertext Attack)
  - 복호 오라클 공격을 공격자가 이용할 수 있다고 가정한 공격
  - 공격 대상인 암호문은 제외

복호 오라클 서비스의 의미

- 넌센스처럼 보이지만 실제 네트워크에서 오류메시지 반환을 악용하는 공격
- 위조 암호문을 여러 차례 전송하여 반환된 오류 메시지나 타이밍 정보를 활용해 평문을 추측
- RSA의 경우 선택암호문 공격으로 약간의 정보 취득 가능

RSA-OAEP
(Optimal Asymmetric Encryption Padding)

- RSA를 개량해서 선택암호문공격으로부터 안전하게 만든 것
- 암호문에 인증 과정을 추가한 방법
- 평문 해시 값과 정해진 개수의 0 등으로 만들어진 인증정보를 평문 앞에 추가한 뒤 그 후에 RSA로 암호화한다.
- 복호화시 RSA로 복호화한 후 선두에 올바른 인증 정보가 나타나지 않으면 오류로 판정

Section 07
기타 공개키 암호

7.1 ElGamal 방식

7.2 Rabin 방식

7.3 타원곡선 암호

7.1 ElGamal 방식

- **ElGamal 방식**은 Taher ElGamal에 의한 공개 키 알고리즘
- RSA는 소인수분해의 어려움을 이용
- ElGamal 방식은 이산 대수를 구하는 것이 어렵다는 것을 이용
- ElGamal 방식 암호화에서는 암호문의 길이가 평문의 2배가 되어 버린다는 결점
- GnuPG에서 사용

7.2 Rabin 방식

- **Rabin 방식**은 M. O. Rabin에 의한 공개 키 알고리즘
- Rabin 방식은 mod N으로 평방근을 구하는 것이 어렵다는 사실을 이용
- Rabin 방식 공개 키 암호의 해독은 소인수분해 정도로 어렵다는 것이 증명

7.3 타원곡선 암호

- **타원 곡선 암호**(elliptic curve cryptosystems; ECC)는 최근 주목받고 있는 공개 키 암호 알고리즘
- RSA에 비해 키의 비트 수가 적다
- 타원 곡선 위에 곱셈을 정의하고, 이 곱셈의 역연산이 어렵다는 것을 이용

Section 08
공개 키 암호에 관한 Q&A

8.1 공개 키 암호의 기밀성

8.2 공개 키 암호와 대칭 암호의 키 길이

8.3 대칭 암호의 미래

8.4 RSA와 소수

8.5 RSA와 소인수 분해

8.6 RSA의 비트 길이

8.1 공개 키 암호의 기밀성

- **의문:** 공개 키 암호는 대칭 암호보다도 기밀성이 높은가?
- **답:** 이것만으로는 답할 수 없다. 왜냐 하면 키의 비트 길이에 따라 기밀성의 정도는 변화하기 때문

8.2 공개 키 암호와 대칭 암호의 키 길이

- **의문:** 1024비트 길이의 키를 갖는 공개 키 암호와, 128비트 길이의 키를 갖는 대칭 암호에서는 비트 길이가 긴 공개 키 암호 쪽이 안전한가?
- **답:** 아니다. 공개 키 암호의 키 길이와, 대칭 암호의 키 길이는 직접 비교할 수 없다.

전사공격에 대한 같은 강도를 갖는 키 길이 비교

대칭 암호의 키 길이	공개 키 암호의 키 길이
128비트	2304비트
112비트	1792비트
80비트	768비트
64비트	512비트
56비트	384비트

8.3 대칭 암호의 미래

- **의문:** 공개 키 암호가 생겼기 때문에 앞으로 대칭 암호는 사용할 필요가 없는가?
- **답:** 아니다.
  - 일반적으로 같은 정도의 기밀성을 갖는 키 길이의 경우, 공개 키 암호는 대칭 암호보다도 몇 백배나 느리다
  - 공개 키 암호는 긴 메시지를 암호화하기에는 적합하지 않다
  - 목적에 따라 대칭 암호와 공개키 암호 두 가지 모두 사용

8.4 RSA와 소수(I)

- **의문:** RSA의 키 쌍을 모두가 자꾸 만들어 가면 그 사이 소수가 없어져 버리는 것은 아닐까?
- **답:** 그럴 염려는 없다. 512비트로 표현할 수 있는 소수의 수는 대략 10^{150}으로 전 우주에 존재하는 원자의 개수보다도 많은 수이다

8.4 RSA와 소수(II)

- 세계 인구를 100억 명 이라고 하고 한 사람이 1초에 100억개의 키 쌍을 만든다고 할 때 100억년 걸리면 몇 개의 키 쌍이 만들어 질까?
- 1년~366 x 24 x 60 x 60 = 31,622,400초
- 100억명 x 100억개 x 31,622,400초 x 100억년

  = 31,622,400,000,000,000,000,000,000,000,000,000,000,000개

  ➡ 10^{39} 보다 적은 수
- 512비트로 표현할 수 있는 소수 =10^{150}

8.5 RSA와 소인수 분해

- **의문:** RSA로 암호화할 때 큰 수를 소인수 분해 할 필요가 있는 것일까?
- **답:** 아니다. RSA의 암호화에서도, 복호화에서도, 그리고 키 쌍의 생성에서도 큰 수를의 소인수분해를 할 필요는 없다.

8.5 RSA와 소인수 분해

- **의문:** RSA로 암호화할 때 큰 수를 소인수 분해 하는 것과 같은 것인가?
- **답:** 같은 것인지 아닌지 아직 모름
  - RSA 개인키를 구하는 것이 N의 소인수 분해와 같다는 것을 2004년 알렉산더 메이(Alexander May)가 증명
  - 분명히 소인수분해를 고속으로 할 수 있다면 RSA는 해독됨
  - RSA를 해독하려면 소인수분해를 꼭 해야 한다는 것이 증명된 것은 아님
  - 어쩌면 소인수분해를 하지 않아도 해독할 수 있는 방법이 발견될지도 모름

8.6 RSA의 비트 길이

- **의문:** 소인수분해 되지 않기 위해서 N은 몇 비트 길이가 필요한가?
- **답:** 아무리 비트 수가 커도 언젠가는 소인수분해 된다

512비트 수 하나 인수분해하기

- 512비트로 주어진 한 수는 1999년 8월에 소인수분해
- 9주간의 사전 계산과 5.2개월간에 걸친 292대의 컴퓨터에 의한 계산이 필요
- 이만큼의 컴퓨터 자원과 시간을 들여서 겨우 1개의 수를 소인수분해 할 수 있었다

640비트 N

- RSA사가 제시한 640비트 N(193자리 10진수)

  31074182404900437213507500358885679300373460228427275457201619488232064405180815045563468296717232867824379162728380334154710731085019195485290073377248227835257423864540146917366024776523466O9

  는 2005.11.2일에 인수분해 되었다

704비트 N

- RSA사가 제시한 704비트 N(212자리 10진수)

  74037563479561712828046796097429573142593188889231289084936232638972765034028266276891996419625117843995894330502127585370118968098286733173273108930900552505116877063299072396380786710086096962537934650563796359

는 아직 인수분해 되지 않았다

- 상금은 3만불. 한 번 시도해보길…

Quiz 4 공개키 암호의 기초 지식

- 다음 문장 중 바른 것에는 O, 틀린 것에는 X를 표시하시오.

1. 공개키 암호로 암호화할 때 수신자의 공개키가 필요하다.
2. 공개키 암호로 암호화된 암호문을 복호화하기 위해서는 공개키 암호화 쌍을 이루고 있는 개인키가 필요하다.
3. 공개키 암호의 개인키는 암호화한 메시지와 함께 수신자에게 송신할 필요가 있다.
4. 일반적으로 공개키 암호보다도 대칭키 암호 쪽이 빠르다.
5. 소인수 분해를 고속으로 푸는 방법이 발견되면 RSA도 고속으로 풀 수 있다.