

# 제 6 장

# 공개 키 암호



**박종혁 교수**

**Tel: 970-6702**

**Email: [jhpark1@seoultech.ac.kr](mailto:jhpark1@seoultech.ac.kr)**

**1절 키 배송 문제**

**2절 공개 키 암호**

**3절 정수론**

**4절 RSA**

**5절 RSA에 대한 공격**

**6절 다른 공개키 암호**

**7절 공개 키 암호에 관한 Q&A**

# 제1절 키 배송 문제

**1.1 키 배송 문제란?**

**1.2 키의 사전 공유에 의한 키 배송 문제의 해결**

**1.3 키 배포 센터에 의한 키 배송 문제의 해결**

**1.4 Diffie-Hellman 키 교환에 의한 키 배송 문제의 해결**

**1.5 공개 키 암호에 의한 키 배송 문제의 해결**

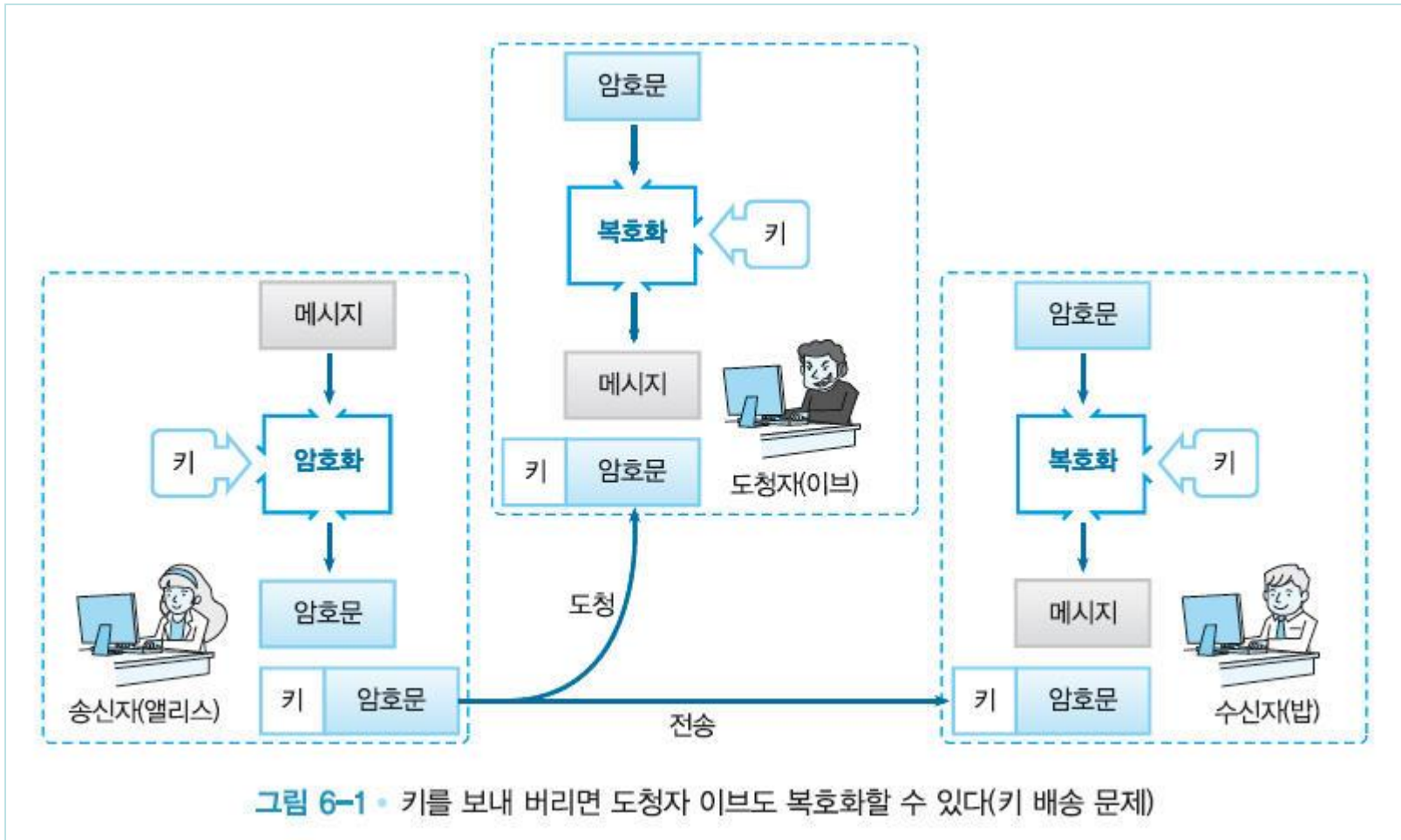
# 1.1 키 배송 문제란?

- 키 배송 문제(key distribution problem)
  - 대칭 암호를 사용하려면 송신자와 수신자가 대칭키를 사전에 공유해야 하는 문제
  - 대칭 키를 보내지 않으면 밥은 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



## 1.2 키의 사전 공유에 의한 키 배송 문제의 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  
$$1000 \times 999 \div 2 = 49\text{만 } 9500\text{개}$$
- 현실적이지 못하다



## 1.3 키 배포 센터에 의한 키 배송 문제의 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

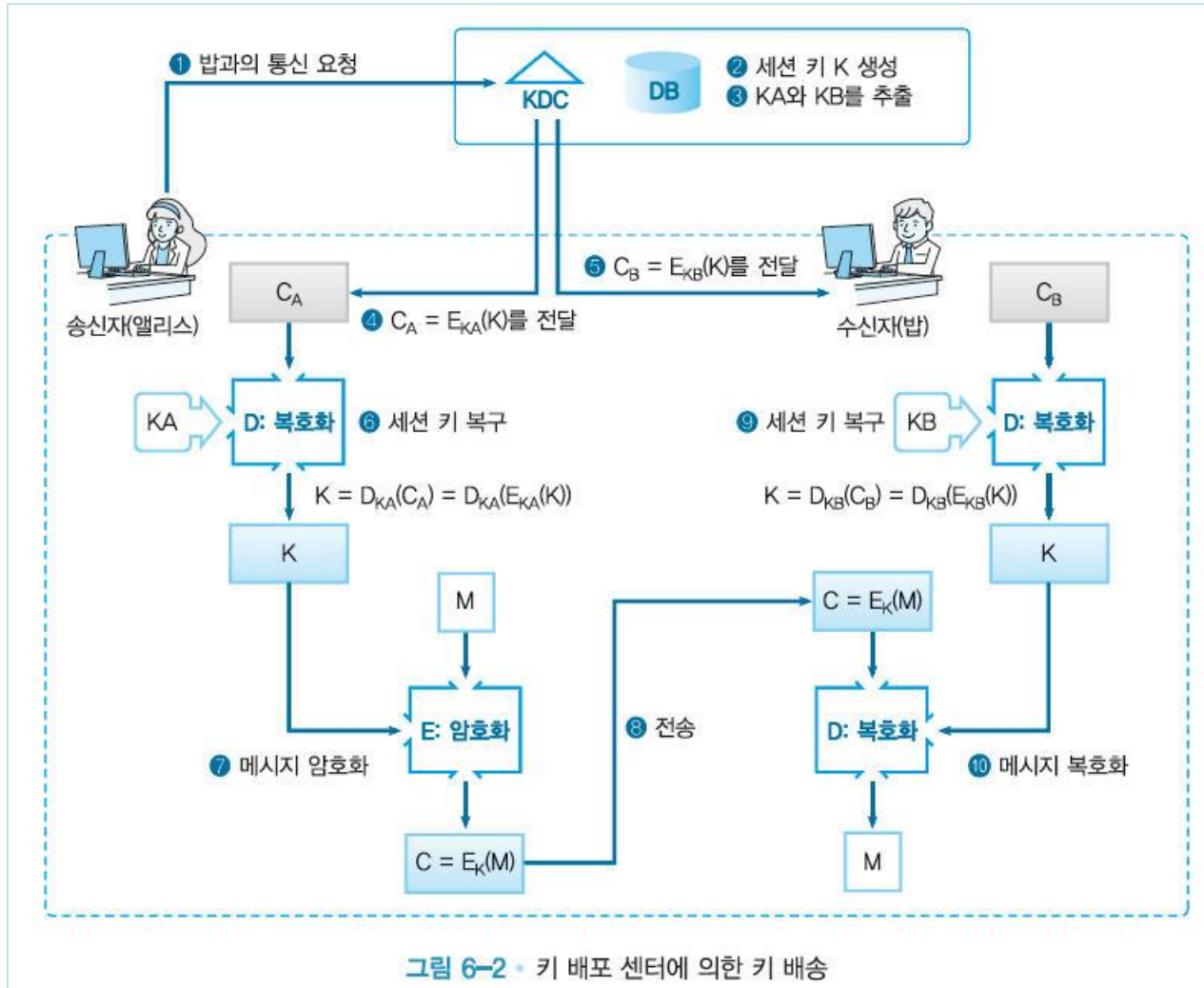
# 앨리스가 밥에게 암호 메일 보내기 (1/2)

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( $CA = E_{KA}(K)$ )해서 앨리스에게 보낸다.
5. 키 배포 센터는 밥의 키를 써서 세션 키를 암호화( $CB = E_{KB}(K)$ )해서 밥에게 보낸다.

## 앨리스가 밥에게 암호 메일 보내기 (2/2)

6. 앨리스는 키 배포 센터로부터 온 세션 키(앨리스의 키로 암호화되어 있음)를 복호화( $K = D_{K_A}(C_A)$ )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( $K = D_{K_B}(C_B)$ )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



# 키 배포센터의 문제점

- 구성원 수 증가시 키 배포 센터의 부하
- 키 배포 센터의 컴퓨터가 고장시 조직 전체의 암호 통신 마비
- 키 배포센터가 공격의 대상이 될 수 있다

## 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 (1/2)

- 공개 키 암호

- 대칭 암호

- 「암호화 키」와 「복호화 키」 동일

- 공개 키 암호

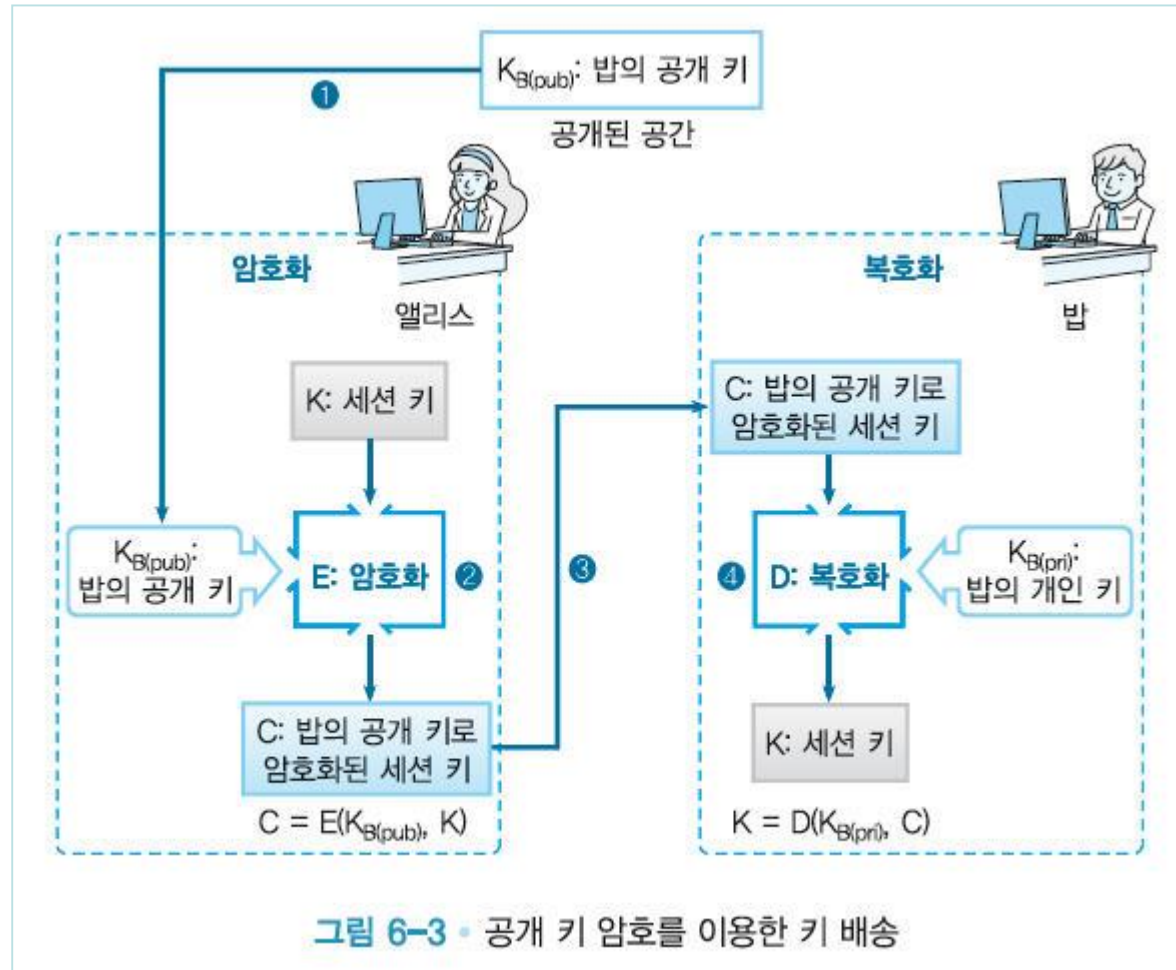
- 「암호화의 키」와 「복호화 키」 다르다
    - 「암호화 키」를 가지고 있는 사람이라면 누구든지 암호화 할 수 있음
    - 하지만 「암호화 키」를 가지고 있어도 복호화할 수는 없다
    - 복호화 할 수 있는 것은 「복호화 키」를 가지고 있는 사람 뿐임

## 1.5 공개 키 암호에 의한 키 배송 문제의 해결 (2/2)

- 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)뿐
- 이렇게 하면 「복호화 키」를 수신자에게 배송할 필요가 없음



# 공개 키 암호를 이용한 키 배송



# 제2절 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

# 공개키의 의미

- 공개 키(public key)
  - 「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 신경 쓸 필요가 없다

# 개인키의 의미

- 개인 키(private key)
- 「복호화 키」는 미공개
- 이 키는 본인만 사용
- 개인 키는 다른 사람에게 보이거나, 건네주거나 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 복호화할 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
- Ron Rivest, Adi Shamir, Leonard Adleman(1978)
  - 공개 키 암호 알고리즘 RSA 발표



## 2.2 공개 키를 사용한 통신의 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

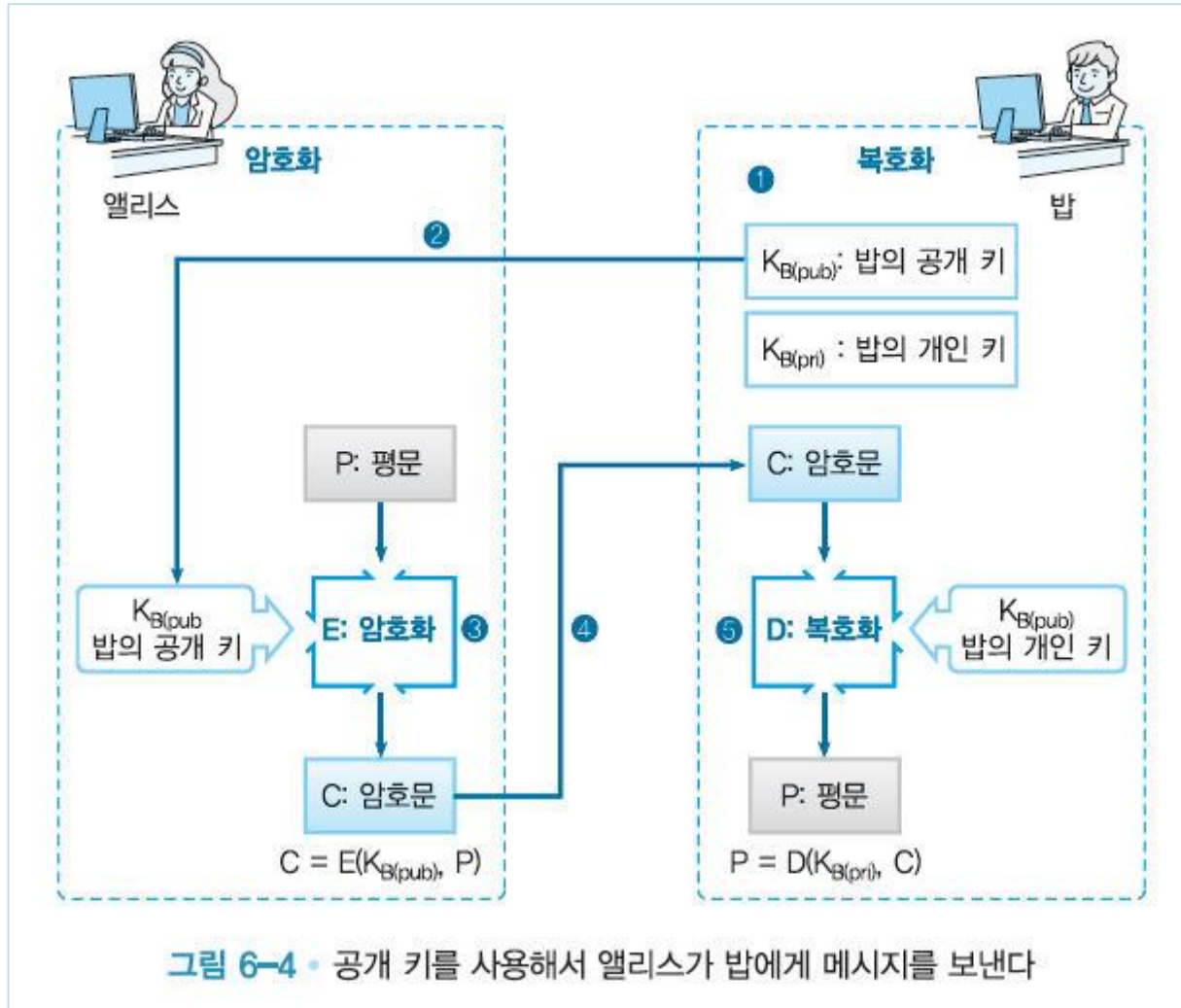


그림 6-4 · 공개 키를 사용해서 앨리스가 밥에게 메시지를 보낸다

## 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 인증에 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

# 제3절 정수론

**3.1 소수와 서로소**

**3.2 모듈러 연산**

**3.3 페르마와 오일러의 정리**

**3.4 이산대수**

# 3.1 소소수와 서로소

- 약수(divisors)
  - 어떤 수  $m$ 에 대해서  $a=mb$ 라면
    - $b \neq 0$  일때  $b$ 는  $a$ 를 나눈다=> 약수
    - 여기서  $a, b, m$ 은 정수
    - 나눗셈에서 나머지가 없을 때 **약수**라고 함
  - 예) 24의 양의 약수
    - 1, 2, 3, 4, 6, 8, 12, 24

## • 약수(계속)

- 만약  $a|1$  이라면, 그때  $a=\pm 1$
- 만약  $a|b$  이고  $b|a$  라면, 그때  $a=\pm b$
- 어떠한 0이 아닌  $b$ 도 0을 나눈다
- 만약  $b|g$ 이고  $b|h$ 라면, 그때 임의의  $m$ 과  $n$ 에 대해  $b|(mg+nh)$ 
  - 만약  $b|g$ 라면, 그때  $g$ 는 어떠한 정수  $g_1$ 에 대해  $g=b*g_1$ 을 만족
  - 만약  $b|h$ 라면, 그때  $h$ 는 어떠한 정수  $h_1$ 에 대해  $h=b*h_1$ 을 만족
  - 또한

$$mg+nh = mbg_1 + nbh_1 = b*(mg_1+nh_1)$$

=>  $b$ 는  $mg+nh$  를 나눈다

- 약수(계속)

- 예)  $b=7; g=14; h=63; m=3; n=2$

- 만약  $b|g$ 이고  $b|h$ 라면, 그때 임의의  $m$ 과  $n$ 에 대해  $b|(mg+nh)$

- $7|14$ 와  $7|63$ 에 대해서  $7|(3*14+2*63)$

- $mg+nh = mbg_1 + nbh_1 = b*(mg_1+nh_1)$

- $g = b*g_1, g_1 = g/b = 2$

- $h = b*h_1, h_1 = h/b = 9$

- $(3*14+2*63) = 7*(3*2+2*9)$  계산 가능

- 이 계산은  $7|7*(3*2+2*9)$ 가 성립함



- 소수(prime numbers)

- 정수  $p > 1$ 이 만약 단지 약수들로서  $\pm 1$ 과  $\pm p$ 만을 가진다면  $p$ 는 소수이다.

- 어떠한 정수  $a > 1$ 은 다음과 같은 유일한 방법으로 인수분해 가능

- $a = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_t^{\alpha_t}$
    - $p_t > p_{t-1} > \dots > p_1$  는 소수,  $\alpha_i > 0$

- 예) 91

- 2, 3, 4, 5, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89

- $7 \cdot 13$

- $10164 = 7 \cdot 11^2 \cdot 12$

- 소수(계속)

- 만약  $P$ 가 모든 소수들의 집합이라면 어떤 양의 정수는 다음의 형태에서처럼 유일하게 표시될 수 있다

$$a = \prod p^{a_p}, \text{ 여기서 } a_p \geq 0$$

- $a$ 의 어떤 특별한 값에 있어서 멱 지수  $a_p$ 의 대부분은 0
- 예)  $3600 \Rightarrow 2^4 * 3^2 * 5^2$

- 소수(계속)

- 정수 12

- $\{a_2 = 2, a_3 = 1\} \Rightarrow 2^2 * 3^1$

- 정수 18

- $\{a_2 = 1, a_3 = 2\} \Rightarrow 2^1 * 3^2$

- 두수의 곱셈은 해당하는 멱 지수를 더하는 것과 같다

- $12 * 18 = (2^2 * 3^1) * (2^1 * 3^2) = (2^3 * 3^3) = 216$

- 이러한 소수들의 관점에서  $a|b$ 의 의미는???

## • 소수(계속)

- 이러한 소수들의 관점에서  $a|b$ 의 의미는???

- $p^k$  형태를 가지는 어떠한 정수는 단지 그것보다 작거나 같은 지수를 가지는 소수  $p^j$ , 단  $j \leq k$  에 의해 **나누어 질 수 있다**

- 즉, 모든  $p$ 에 대해  $a|b \rightarrow a_p \leq b_p$

- 예)  $a=12, b=36$

$$12 = 2^2 * 3^1 \quad 36 = 2^2 * 3^2$$

-  $a_2 = 2 = b_2$

-  $a_3 = 1 \leq 2 = b_3$

- 서로소(relatively prime number)

- 어떤 두 수가 공통적인 소인수를 갖지 못할 때 두 수를 **서로소** 라고 한다.
- 공통적인 소인수 => 최대 공약수 => GCD
- 양의 정수  $c$ 가 다음의 조건을 만족한다면  $c$ 는  $a$ 와  $b$ 의 최대 공약수
  - $c$ 는  $a$ 와  $b$ 의 약수
  - $a$ 와  $b$ 에 대한 어떠한 약수는  $c$ 의 약수
- $\text{GCD}(a,b) = \max[k, \text{이때 } k \text{는 } k|a \text{이고 } k|b]$

- 서로소(계속)

- 암호학에서 요구하는 최대 공약수는 양수

- $\text{GCD}(a,b)=\text{GCD}(a,-b)=\text{GCD}(-a,b)=\text{GCD}(-a,-b)=\text{GCD}(|a|,|b|)$

- $\text{GCD}(60,24)=\text{GCD}(60,-24)=12$

- 모든 0이 아닌 정수들은 0을 나누기 때문에  $\text{GCD}(a,0)=|a|$

- 모든 정수의 소수 표현법을 이용하면 최대 공약수 결정이 쉬워짐

- $300 = 2^2 * 3^1 * 5^2$                        $18 = 2^1 * 3^2$

- $\text{GCD}(300,18) = 2^1 * 3^1 * 5^0 = 6$

- 서로소(계속)

- 8과 15는 서로 소인가?

- 8은 약수로 1, 2, 4, 8 가짐

- 15는 약수로 1, 3, 5, 15 가짐

- => 동일한 약수를 가지지 못하므로 서로 소임

## 3.2 모듈러 연산

- 어떤 양의 정수  $n$ 과 어떤 정수  $a$ 가 주어지고, 만약  $a$ 를  $n$ 으로 나눈다면 다음과 같은 관계를 가지는 몫  $q$ 와 나머지  $r$ 을 얻는다

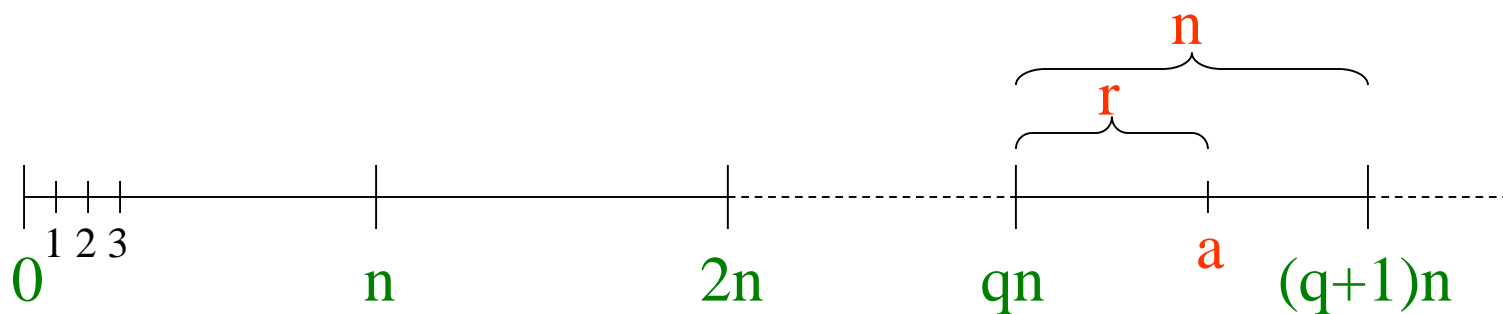
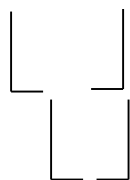
$$a = qn + r$$

$$0 \leq r < n$$

$$q = a/n$$

$$a \equiv r \pmod{n}$$

=> 여기서  $x$  는  $x$ 보다 작거나 또는 같은 가장 큰 정수





- 예제)

- $a = qn + r$

- $a=11, n=7$

- $11 = 1*7 + 4 = 4 \pmod{7}$

$$4 = 11 \pmod{7}$$

$$11 \equiv 4 \pmod{7}$$

- $a = -11, n=7$

- $-11 = 2*7 - 3 = 3 \pmod{7}$

$$3 = -11 \pmod{7}$$

$$-11 \equiv 3 \pmod{7}$$

- **합동**

- $(a \bmod n) = (b \bmod n)$ , 두 정수  $a$ 와  $b$ 는 modulo  $n$ 에 대해 합동
- $a \equiv b \pmod{n}$  으로 표기
- $a \equiv 0 \pmod{n}$  이라면 그때  $n|a$  이다

- **모듈러 연산자의 특성**

1. 만약  $n|(a-b)$  라면,  $a \equiv b \pmod{n}$
2.  $(a \bmod n) = (b \bmod n)$ 은  $a \equiv b \pmod{n}$
3.  $a \equiv b \pmod{n}$ 은  $b \equiv a \pmod{n}$  을 의미
4.  $a \equiv b \pmod{n}$ 과  $b \equiv c \pmod{n}$  은  $a \equiv c \pmod{n}$  을 의미

1. 만약  $n|(a-b)$  라면,  $a \equiv b \pmod n$

$$n = 5, a = 23, b = 8$$

$$23 - 8 = 15 = 5 \cdot 3 \text{ 이기 때문에 } 23 \equiv 8 \pmod 5$$

- 모듈러 산술 연산

- mod n 연산

- 정수들의 범위 =>  $\{0, 1, \dots, (n-1)\}$ 으로 표현가능
    - 즉, 이러한 집합의 범위 내에서 산술 연산이 가능

- 모듈러 연산의 특징

1.  $[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$
2.  $[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$
3.  $[(a \bmod n) * (b \bmod n)] \bmod n = (a*b) \bmod n$

- 예제)  $a = 11, b = 15, n = 8$

1.  $(a + b) \bmod n = ((11 \bmod 8) + (15 \bmod 8)) \bmod 8 = 3 + 7 \bmod 8$

$$= 10 \bmod 8 = 2$$

2.  $(a - b) \bmod n = ((11 \bmod 8) - (15 \bmod 8)) \bmod 8 = 3 - 7 \bmod 8$

$$= -4 \bmod 8 = 4$$

3.  $(a * b) \bmod n = ((11 \bmod 8) * (15 \bmod 8)) \bmod 8 = 3 * 7 \bmod 8$

$$= 21 \bmod 8 = 5$$

- 지수 연산 => 곱셈의 반복으로 수행가능

- 예)  $11^7 \bmod 13$

$$11^2 = 121 = 4 \bmod 13$$

$$11^4 = 4^2 = 3 \bmod 13$$

$$11^7 = 11 * 4 * 3 = 2 \bmod 13$$

## • 모듈러 연산의 속성

### – 교환법칙

- $(w+x) \bmod n = (x+w) \bmod n$
- $(w*x) \bmod n = (x*w) \bmod n$

### – 결합법칙

- $[(w+x)+y] \bmod n = [w+(x+y)] \bmod n$
- $[(w*x)*y] \bmod n = [w*(x*y)] \bmod n$

### – 분배법칙

- $[w*(x+y)] \bmod n = [(w*x)+(w*y)] \bmod n$

### – 항등원

- $(0+w) \bmod n = w \bmod n$
- $(1*w) \bmod n = w \bmod n$

### – 덧셈에 대한 역원 $(-w)$

## 3.3 페르마와 오일러의 정리

- 페르마 정리(Fermat Theorem)

만약  $p$ 가 소수라면  $a$ 는  $p$ 에 의해 나누어지지 않는 양의 정수이다. 그때

$$a^{p-1} \equiv 1 \pmod{p}$$

가 성립한다



- 예제)  $a=7, p=19$

- $a^{p-1} \equiv 1 \pmod{p}$

- $7^2 = 49 \equiv 11 \pmod{19}$

- $7^4 = 121 \equiv 7 \pmod{19}$

- $7^8 = 49 \equiv 11 \pmod{19}$

- $7^{16} = 121 \equiv 7 \pmod{19}$

- $a^{p-1} = 7^{18} = 7^{16} * 7^2 \equiv 7 * 11 \equiv 1 \pmod{19}$

- 페르마의 다른 유용한 형태  
만약  $p$ 가 소수고  $a$ 가 양의 정수라면

$$a^p \equiv a \pmod{p}$$

가 성립한다

- 예)  $a=3, p=5, 3^5 = 243 \equiv 3 \pmod{5}$
- 예)  $a=10, p=5, 10^5 = 100000 \equiv 10 \pmod{5} \equiv 0 \pmod{5}$

# 오일러 정리

- 오일러의 Totient 함수

정수론에서 오일러의 totient 함수는  $\phi(n)$ 라고 표기된다

$\phi(n)$ :  $n$ 보다 작고  $n$ 과 서로 소인 양의 정수의 개수

$n$	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\phi(n)$	1	1	2	2	4	2	6	4	6	4	10	4	12	6

# 오일러 정리

- 오일러 함수의 특성

- $\varphi(1) = 1$

- 소수  $n$ 에 대해서

- $\varphi(n) = n-1$

- 소수  $p$ 와  $q$ 에 대해서  $n=pq$

- $\varphi(n) = \varphi(pq) = \varphi(p) * \varphi(q) = (p-1)(q-1)$

# 오일러 정리

- 오일러 정리

서로소인 모든  $a$ 와  $n$ 에 대한 관계를 나타낸다

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

$$a=3; n=10; \varphi(10) = 4; 3^4 = 81 \equiv 1 \pmod{10}$$

$$a=2; n=11; \varphi(11) = 10; 2^{10} = 1024 \equiv 1 \pmod{11}$$

- 오일러 정리의 추가적인 특성

$$a^{\varphi(n)+1} \equiv a \pmod{n}$$

## 3.4 이산대수

- 다음과 같은 등식을 고려해 보자

$$y = g^x \bmod p$$

$g$ ,  $x$  그리고  $p$ 가 주어진다면,  $y$ 를 계산하는 것은 쉬운 문제이다. 최악의 경우, 반복적인 곱셈과정을  $x$ 번 수행해야만 하며, 효율적인 계산을 위한 알고리즘이 존재한다.

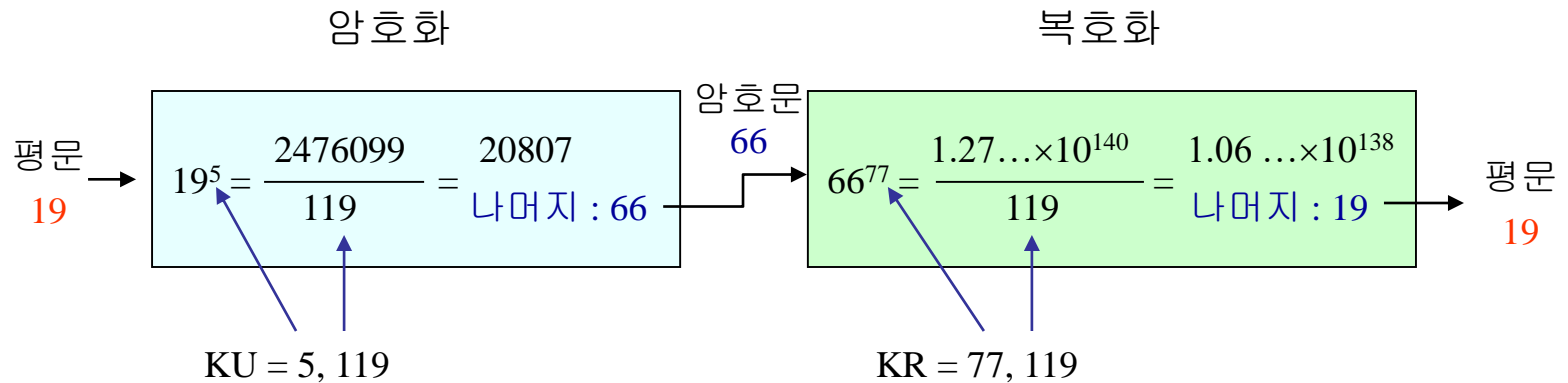
그러나  $y$ ,  $g$  그리고  $p$ 가 주어진다고 하더라도  $x$ 를 계산하는 것은 매우 어려운 문제이다. 그 어려움은 RSA알고리즘을 풀기 위해 요구되는 인수분해의 어려움과 같은 어려움을 가진다.

## • RSA 알고리즘 예

### - 암호화와 복호화

: 평문 메시지  $M = 19$  일 경우

- 암호문 :  $19^5 = 66 \pmod{119} \Rightarrow 66$
- 복호문 :  $66^{77} = 19 \pmod{119} \Rightarrow 19$



**4.1 RSA란 무엇인가?**

**4.2 RSA에 의한 암호화**

**4.3 RSA에 의한 복호화**

**4.4 키 쌍의 생성**

**4.5 구체적 계산**



# 4.1 RSA란 무엇인가?

- RSA는 공개 키 암호 알고리즘의 하나
  - RSA 이름
    - 개발자 3명의 이름
    - Ron Rivest, Adi Shamir, Leonard Adleman의 이니셜 (Rivest-Shamir-Adleman)
  - 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

$$\text{암호문} = (\text{평문})^E \bmod N$$

(RSA에 의한 암호화)

# E와 N은 무엇일까?

- (E, N): 공개 키
  - E와 N이라는 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

$$\text{평문} = (\text{암호문})^D \bmod N$$

(RSA의 복호화)

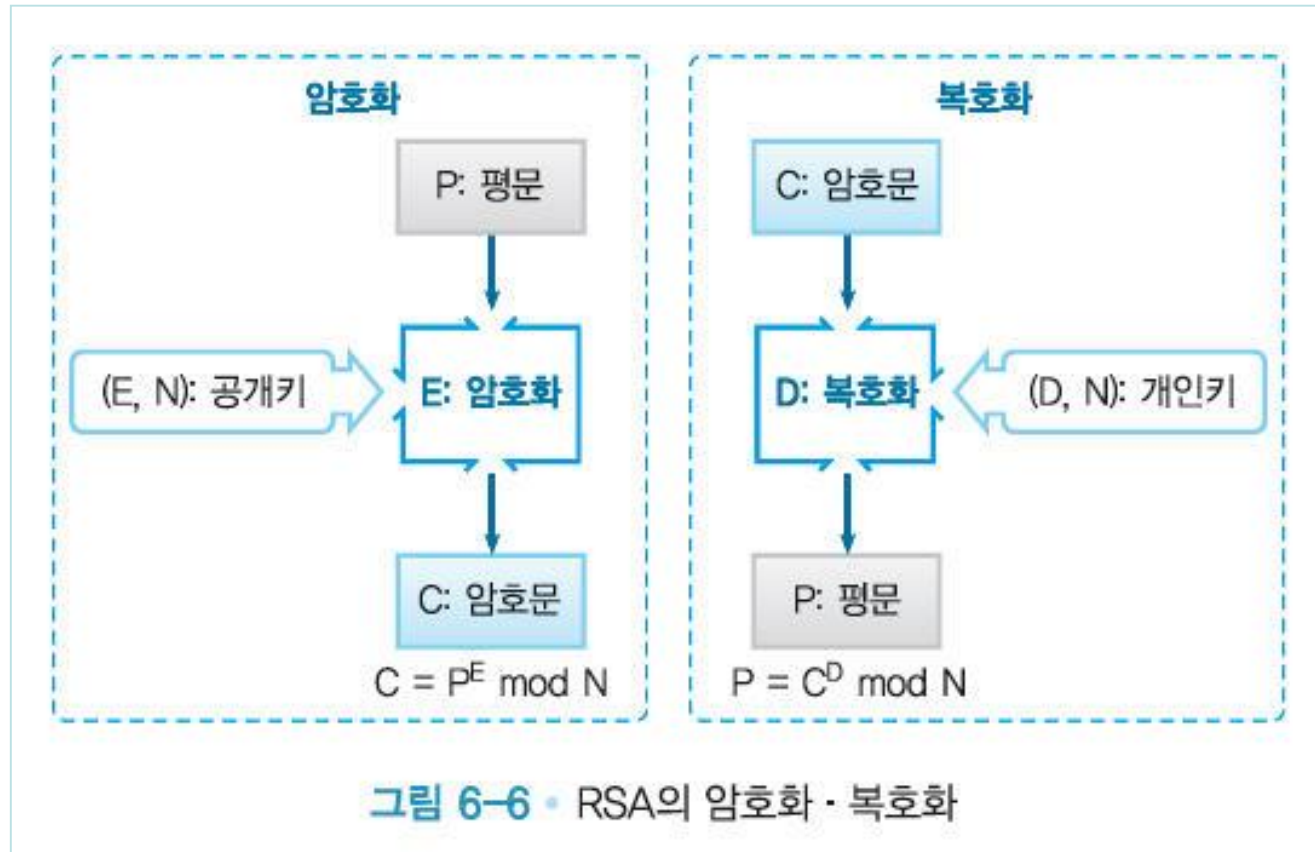
# D와 N은 무엇일까?

- (D, N): 개인 키
  - D와 N이라는 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		$\text{암호문} = (\text{평문})^E \bmod N$ <p>(평문을 E제곱해서 N으로 나눈 나머지)</p>
복호화		$\text{평문} = (\text{암호문})^D \bmod N$ <p>(암호문을 D제곱해서 N으로 나눈 나머지)</p>

# RSA의 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)
- $N = p \times q$  (p, q는 소수)

# L 구하기

- L 은 RSA의 암호화나 복호화에 사용안함
- 키 쌍을 만들 때 임시로 사용
- $L = \text{lcm}(p-1, q-1)$   
(L은  $p-1$  과  $q-1$ 의 최소공배수)

# E 구하기

- 다음 두 식을 만족하는 수 E를 하나 찾아낸다
- $1 < E < L$
- $\text{gcd}(E, L) = 1$  (E와 L은 서로 소)

# D 구하기

- 다음 두 식을 만족하는 수 E를 하나 찾아낸다
- $1 < D < L$
- $E \times D \bmod L = 1$

# RSA 키 쌍 생성

(1) N을 구한다

의사난수 생성기로 p와 q를 구한다 p와 q는 소수

$$N = p \times q$$

(2) L을 구한다

$L = \text{lcm}(p-1, q-1)$  L은 p-1과 q-1의 최소공배수

(3) E를 구한다

$$1 < E < L$$

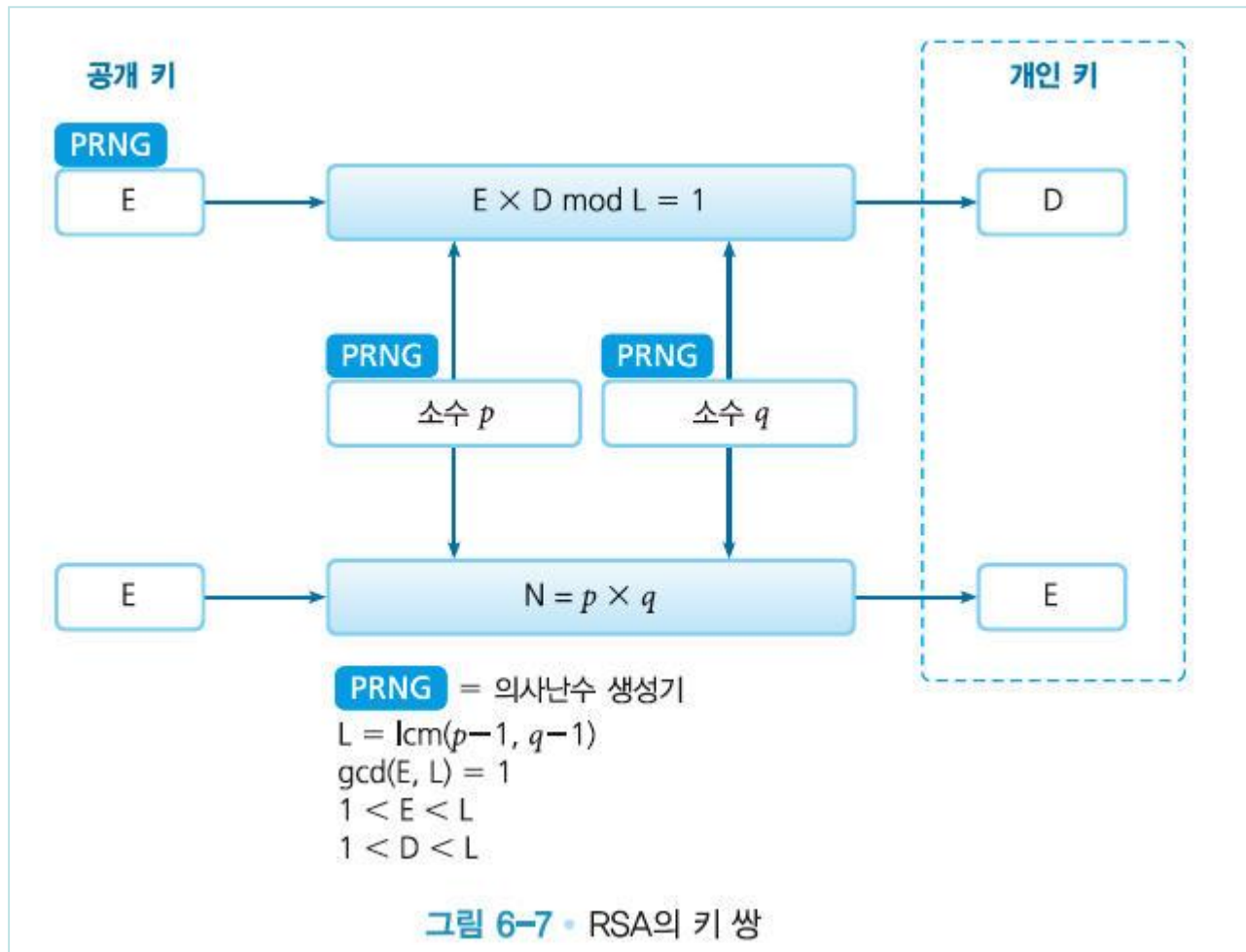
$\text{gcd}(E, L) = 1$  E와 L과의 최대공약수는 1(E와 L은 서로 소)

(4) D를 구한다

$$1 < D < L$$

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

# RSA 키 쌍



## 4.5 구체적 계산

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

# RSA 예

- p 와 q 선택하기  
2개의 소수  $p=17$ ,  $q=19$  선택
- N 구하기  
 $N = p \times q = 17 \times 19 = 323$
- L 구하기  
 $L = \text{lcm}(p-1, q-1) = \text{lcm}(16, 18) = 144$  (16과 18의 최소공배수)
- E 구하기(선택하기)
  - $\text{gcd}(E, L) = 1$  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  $E=5$ 를 선택(다른 수를 선택해도 무방)
- D 구하기  
 $E \times D \bmod L = 5 \times 29 \bmod 144 = 145 \bmod 144 = 1$  이므로  $D=29$



# RSA 예

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

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

$$\begin{aligned} \text{평문}^E \bmod N &= 123^5 \bmod 323 \\ &= 225 \end{aligned}$$

$$\begin{aligned}\text{암호문}^D \bmod N &= 225^{29} \bmod 323 \\ &= 123\end{aligned}$$

# 225<sup>29</sup> mod 323의 계산

- $29=10+10+9$
- $225^{29} = 225^{10+10+9} = 225^{10} \times 225^{10} \times 225^9$   
 $225^{10} = 332525673007965087890625$   
 $225^9 = 1477891880035400390625$

$$225^{10} \bmod 323 = 332525673007965087890625 \bmod 323 = 16 \dots\dots (1)$$

$$225^9 \bmod 323 = 1477891880035400390625 \bmod 323 = 191 \dots\dots (2)$$

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

- 따라서  $225^{29} \bmod 323 = 123$

# 제5절 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  $p, q, L$ 을 모름

## 5.1 암호문으로부터 평문 구하기

$$\text{암호문} = (\text{평문})^E \pmod N$$

- 에서 평문을 구하려면 **이산 대수 문제**를 풀어야 함
- 이산 대수 문제는 매우 곤란
- 현재까지 아직 이산 대수를 구하는 빠른 방법을 알지 못함

## 5.2 전사 공격

- 전사공격(brute-force attack)
  - $D$ 의 후보가 되는 수를 순서대로 모두 시도해서 복호화 해본다
  - $D$ 의 비트 수가 크면 클수록 어려워진다
  - 비트 수가 충분히 크면 전사공격으로 수  $D$ 를 찾아내는 것은 현실적으로는 불가능
  - RSA에서는  $p$ 와  $q$ 의 비트 수로서 512 비트 이상을 사용
  - $N$ 은 1024 비트 이상을 이용
  - $E$ 나  $D$ 는  $N$ 과 같은 정도의 크기로 할 수 있으므로  $D$ 를 찾으려면 1024 비트 이상의 전사공격이 필요
  - 현실적으로 불가능



## 5.3 E와 N으로부터 D 구하기

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

- L은  $\text{lcm}(p-1, q-1)$ 이므로 E로부터 D를 계산할 때는 p와 q를 사용
- 암호해독자는 p와 q를 전혀 모름
- 해독자는 D를 구할 수 없음
- RSA의 안전성을 위해 소수 p와 q를 암호 해독자가 모르게 해야 함

# N의 소인수 분해

- $N = p \times q$ 라는 관계식을 공격자는 알고 있고  $N$ 은 공개되어 있다
- $N$ 으로부터  $p$ 와  $q$ 를 구할 수는 없는 것일까?
- $p$ 와  $q$ 는 소수이기 때문에  $N$ 으로부터  $p$ 와  $q$ 를 구한다는 것은 자연수  $N$ 을 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 추측하기

- 소인수분해를 하지 않아도  $p$ 와  $q$ 가 암호 해독자에게 알려질 가능성이 있다
- $p$ 와  $q$ 는 의사난수 생성기로 생성하기 때문에 의사난수 생성기의 품질이 나쁘면  $p$ 와  $q$ 를 암호 해독자가 추측할 수 있다
- 난수 생성기가 강력해서 암호 해독자가 추측할 수 없어야 한다

# 기타 공격

- $N$ 을 소인수분해 해서  $p$ 와  $q$ 를 구할 수 있으면  $D$ 를 구할 수 있다
- 「 $D$ 를 구하는 것」이 「 $N$ 을 소인수분해 하는 것」과 수학적 같  
은지 아닌지가 증명되어 있지 않다
- $N$ 을 소인수분해 하지 않아도( $p$ 와  $q$ 를 몰라도)  $E$ 와  $N$ 으로부터  
 $D$ 를 구하는 방법이 있을 수 있다

## 5.4 중간자 공격

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

## 중간자공격 절차 (1/3)

- 1) 앨리스는 밥의 공개 키 요청
- 2) 멜로리는, 앨리스의 요청을 도청
- 3) 밥은 자신의 공개 키( $K_B(\text{pub})$ )를 앨리스에게 전송
- 4) 멜로리는 밥의 이 메일이 앨리스에게 도달하지 못하도록 하고,  
밥의 공개 키를 보존
- 5) 멜로리는 자신의 공개 키( $K_M(\text{pub})$ )를 밥의 공개키라고 속여서  
앨리스에게 전송

## 중간자공격 절차 (2/3)

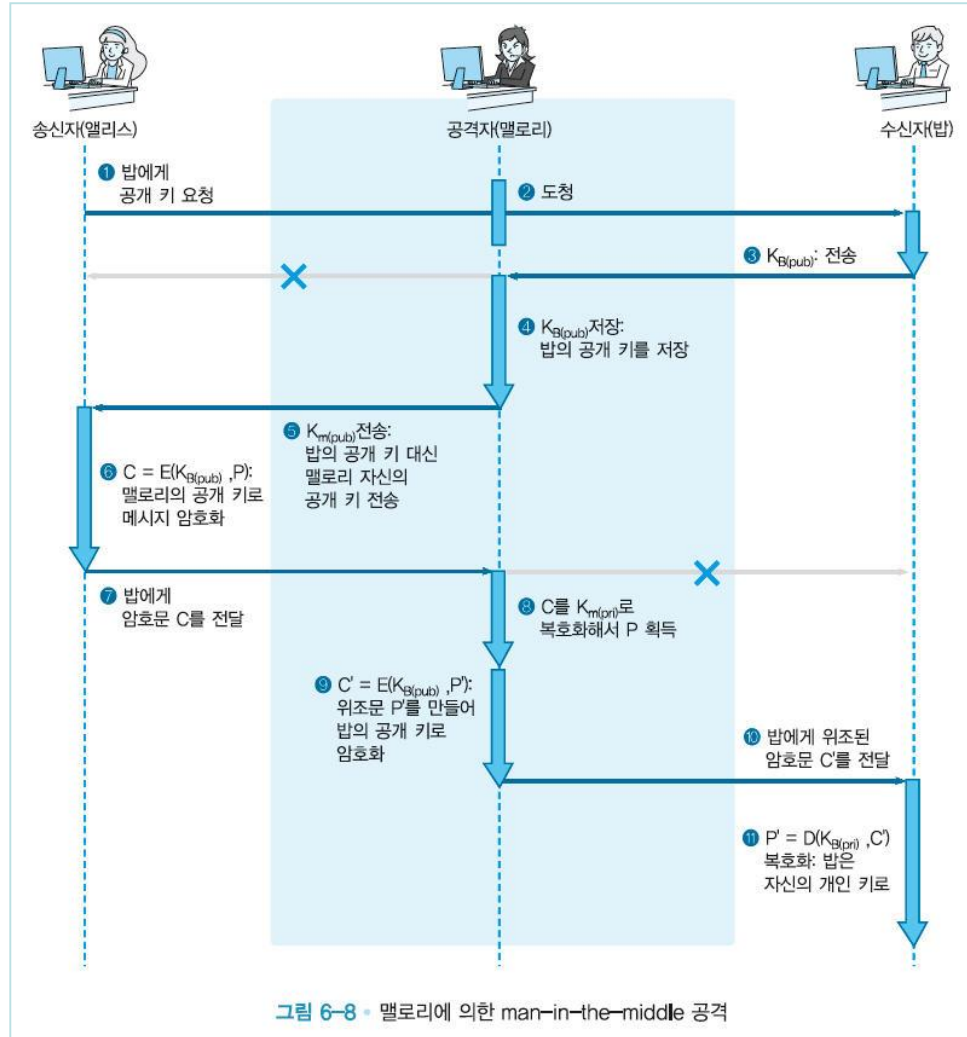
- 6) 앨리스는 자신의 메시지(P)를 밥의 공개 키(실은 맬로리의 공개 키)로 암호화(  $C = E(K_{M(\text{pub})}, P)$  )
- 7) 앨리스는 암호화한 메시지(C)를 밥에게 전송
- 8) 맬리로는 앨리스의 암호 메일을 갈취해서 자신의 개인키 ( $K_{M(\text{pri})}$ )로 복호화(  $P = D(K_{M(\text{pri})}, C)$  ) 하고 평문(P)을 확보



## 중간자공격 절차 (3/3)

- 9) 맬로리는 앨리스 행세를 하며 위조 메일( $P'$ )을 만들고 위의 단계 (4)에서 보존해 둔 밥의 공개 키( $K_{B(\text{pub})}$ )를 써서 이 위조 메일을 암호화( $C' = E(K_{B(\text{pub})}, P')$ )하여 밥에게 전송
- 10) 밥은 받은 암호 메일( $C'$ )을 자신의 개인 키로 복호화하고 메일( $P'$ )을 읽게 된다

# 메모리에 의한 중간자 공격



## 제6절 선택 암호문 공격

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

# 복호 오라클 서비스의 의미

- 난센스처럼 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) (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

# 제7절 기타 공개키 암호

**7.1 ElGamal 방식**

**7.2 Rabin 방식**

**7.3 타원곡선 암호**

## 7.1 ElGamal 방식

- ElGamal 방식은 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 방식은 M. O. Rabin에 의한 공개 키 알고리즘
- Rabin 방식은 mod  $N$ 으로 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

# 제8절 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.

# 전사 공격에 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 백 배나 느리다
  - 공개 키 암호는 긴 메시지를 암호화하기에는 적합하지 않다
  - 목적에 따라 대칭 암호와 공개키 암호 두 가지 모두 사용

## 8.4 RSA와 소수

- 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

## 8.5 RSA와 소인수 분해

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



## 8.5 RSA와 소인수 분해

- 의문: RSA를 해독하는 것은 큰 수를 소인수분해 하는 것과 같은 것인가?
- 답: 같은 것인지 아닌지 아직 모름
  - 분명히 소인수분해를 고속으로 할 수 있다면 RSA는 해독된다
  - 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  $N$ 은 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 소인수 분해 할 수 있었다

# 640비트 N

- RSA사가 제시한 640비트 N(193자리 10진수)

3107418240490043721350750035888567930  
0373460228427275457201619488232064405  
1808150455634682967172328678243791627  
2838033415471073108501919548529007337  
7248227835257423864540146917366024776  
52346609

- 는 2005.11.2일에 인수분해 되었다

# 704비트 N

- RSA사가 제시한 704비트 N(212자리 10진수)

74037563479561712828046796097429573142593  
18888923128908493623263897276503402826627  
68919964196251178439958943305021275853701  
18968098286733173273108930900552505116877  
06329907239638078671008609696253793465056  
3796359

는 아직 인수분해 되지 않았다

- 상금은 3만불. 한 번 시도해보길...

**Q & A**

**Thank You!**