2010-1학기 현대암호학

제 5장 공개키 암호

박 종 혁

Tel: 970-6702
Email: jhpark1@snut.ac.kr

5.0 주요 내용

- 키 배송 문제
- 공개 키 암호
- 시계 연산
- RSA
- 다른 공개 키 암호

5.1 키 배송 문제

- 대칭암호에서는 양측이 안전하게 통신을 하기 위해서 비밀키를 공유하는 것이 핵심이다.

5.1.1 키 배송

- 대칭 암호를 사용하려고 하면 바로 **키 배송 문제** (key distribution problem)에 부딪치게 된다.
- 앨리스가 자신의 메시지를 암호화할 때 사용했던 **대칭 키를 보내지 않으면 밥은 복호화 할 수 없다.**

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

5.1.2 키 사전 공유

- 키 배송 문제를 해결하기 위한 가장 간단한 방법은「안전한 방법으로 키를 사전에 건네주는」것이다. 이것을 키의 **사전 공유**라 부른다.

사전 공유의 한계

- 안전한 방법으로 통신 상대에게 키를 건네주지 않으면 안 된다
- 인원이 많아지면 통신을 위한 키의 수가 방대해지는 것도 문제이다

키를 보내 버리면 도청자 이브도 복호화할 수 있다(키 배송 문제)

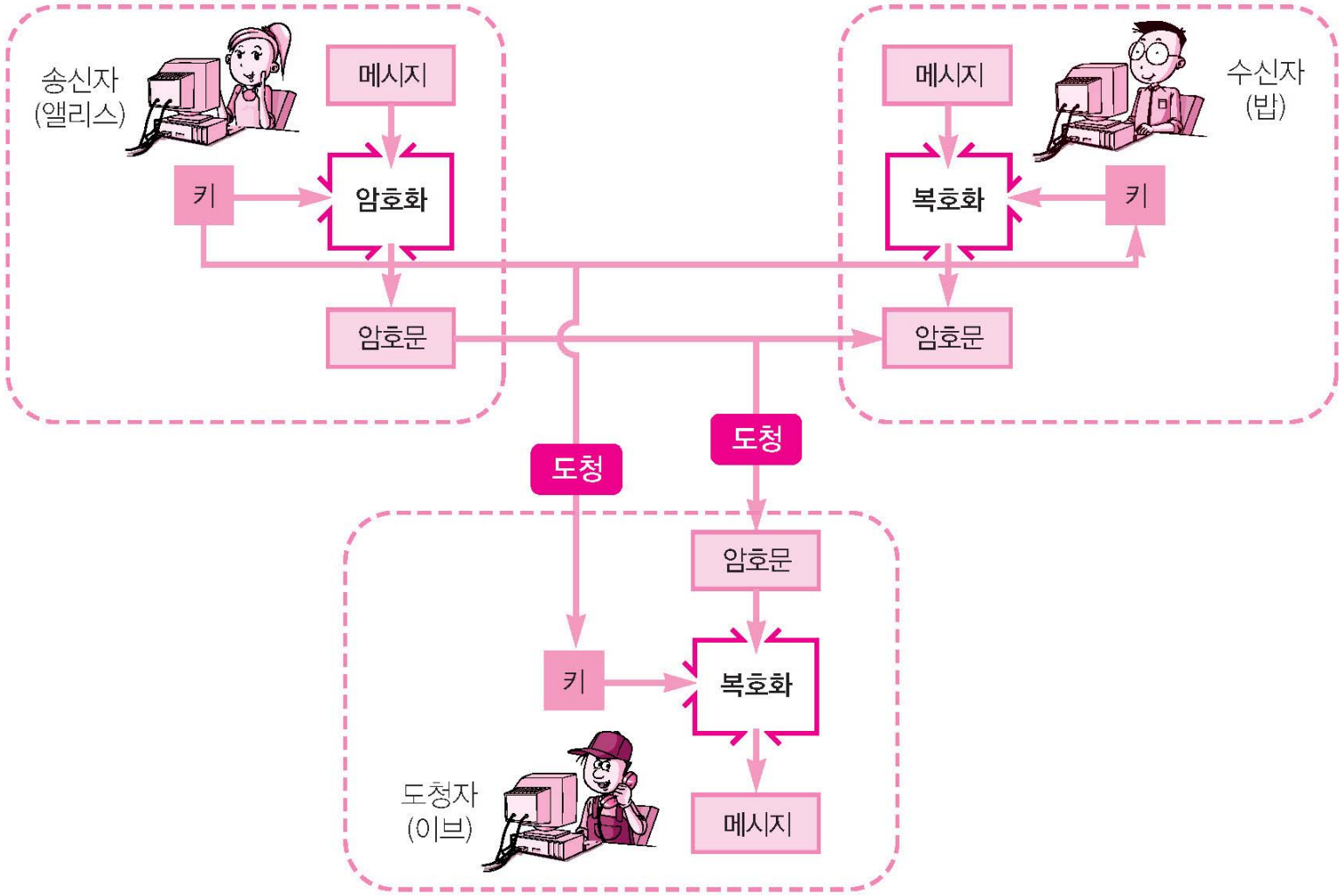


그림 5-1 키를 보내 버리면 도청자 이브도 복호화할 수 있다(키 배송 문제).

5.1.3 키 배포 센터

- **평문 블록**
  - 블록 암호 알고리즘에서 암호화의 대상이 되는 평문을 말한다.
  - 평문 블록의 길이는 블록 암호 알고리즘의 블록 길이와 같다.
- **암호문 블록**
  - 블록 암호 알고리즘을 써서 평문 블록을 암호화한 암호문을 말한다.

평문 블록과 암호문 블록

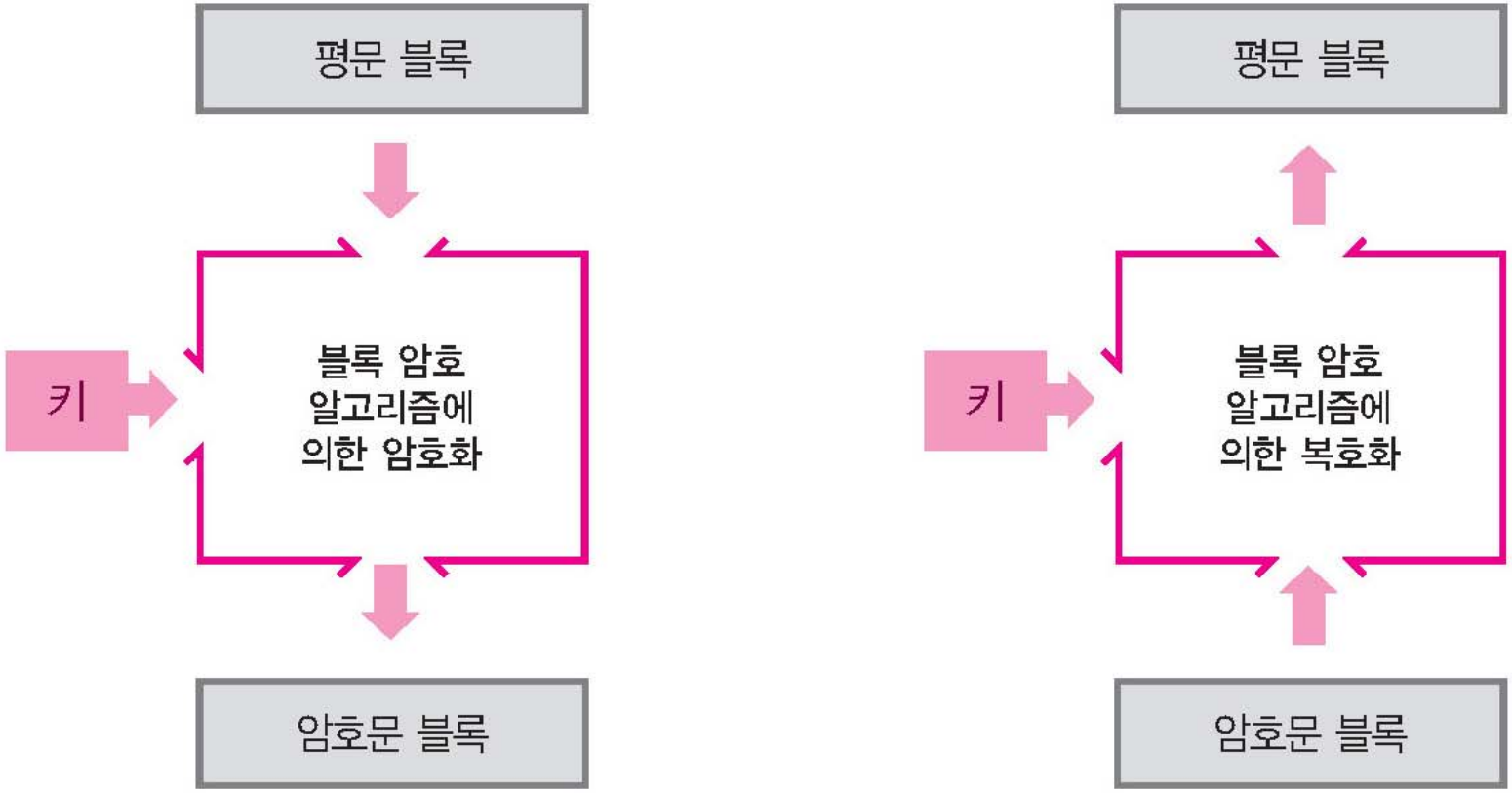


그림 4-1 평문 블록과 암호문 블록

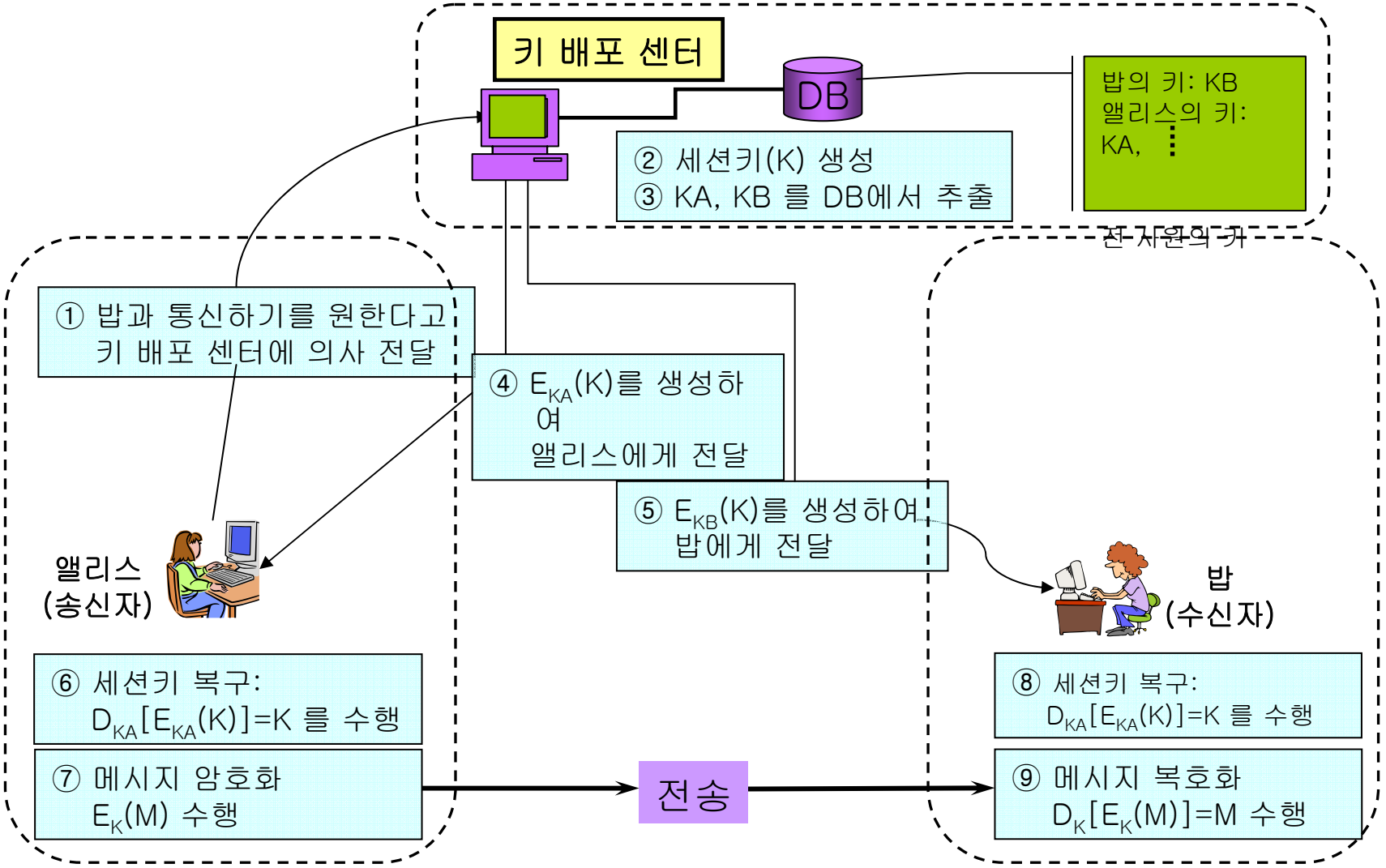
앨리스가 밥에게 암호 메일 보내기

1. 앨리스는 키 배포 센터에「밥과 통신하고 싶다」고 신청한다.
2. 키 배포 센터는 의사난수 생성기를 써서 세션 키를 만든다. 이것은 앨리스와 밥이 이번 통신에만 사용하기 위한 일시적인 키이다.
3. 키 배포 센터는 데이터베이스로부터 앨리스의 키와 밥의 키를 꺼낸다.
4. 키 배포 센터는 앨리스의 키를 써서 세션 키를 암호화해서 앨리스에게 보낸다.
5. 키 배포 센터는 밥의 키를 써서 세션 키를 암호화해서 밥에게 보낸다.

앨리스가 밥에게 암호 메일 보내기

6. 앨리스는 키 배포 센터로부터 온 세션 키(앨리스의 키로 암호화되어 있음)를 복호화해서 세션 키를 얻는다.
7. 앨리스는 세션 키를 써서 밥에게 보낼 메일을 암호화해서 밥에게 보낸다.
8. 밥은 키 배포 센터에서 온 세션 키(밥의 키로 암호화되어 있음)를 복호화해서 세션 키를 얻는다.
9. 밥은 세션 키를 써서 앨리스에게 온 암호문을 복호화한다.
10. 앨리스와 밥은 세션 키를 삭제한다.

키 배포 센터에 의한 키 배송



5.1.4 Diffie-Hellman 키 교환

- 송수신자가 어떤 특정 정보를 서로 교환한다.
  - 이 정보는 도청자 이브가 읽어도 괜찮다.
- 교환한 정보를 가지고 동일한 키를 각각 만들어 낼 수 있다.
  - 도청자 이브는 같은 키를 만들 수 없다
- 두 통신자 사이에 인증을 제공하지 못한다는 결점이 있다.

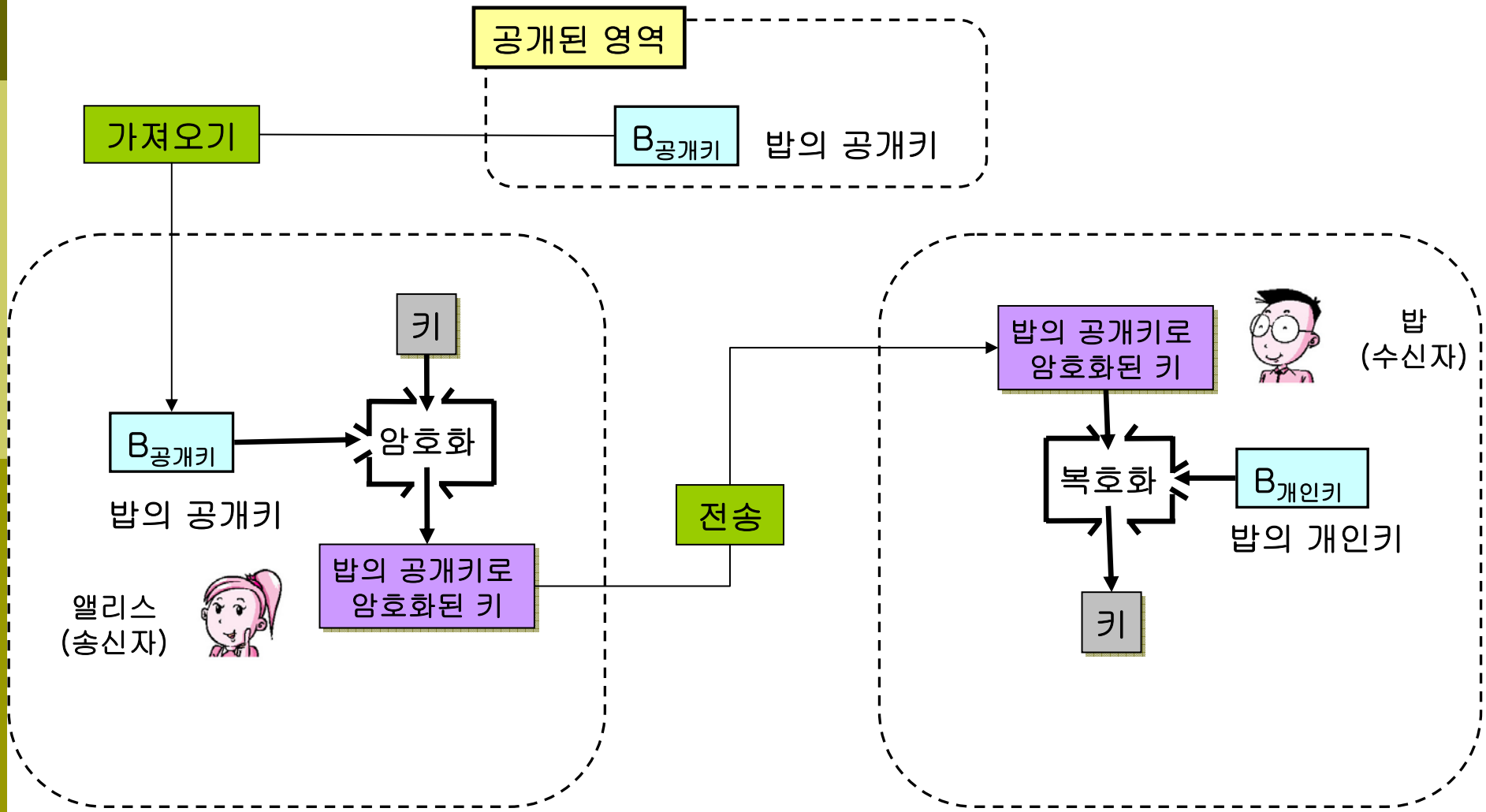
공개키 인증서를 이용해 세션키를 교환하는 방법

1. 메시지를 준비한다.
2. 일회용 세션 키를 이용하는 대칭암호 기법으로 메시지를 암호화한다.
3. 앨리스의 공개키를 이용해서 그 세션 키를 암호화 한다.
4. 암호화된 세션 키를 메시지에 첨부해서 앨리스에게 보낸다.

5.1.5 공개 키 암호를 이용한 키 배송

- 수신자는 미리 「암호화 키」를 송신자에게 알려 준다.
  - 이 「암호화 키」는 도청자에게 알려져도 괜찮다.
- 송신자는 그 「암호화 키」를 써서 보내고자 하는 메시지를 암호화해서 수신자에게 보낸다.
  - 복호화할 수 있는 것은 「복호화 키」를 가지고 있는 사람(수신자)뿐이다.

공개 키 암호를 이용한 키 배송



5.2 공개 키 암호

- 공개 키 암호를 사용하면 앞 절에서 말한 키 배송 문제를 해결할 수 있다.
- 내용
  - 공개 키 암호의 역사와 구조,
  - 공개 키 암호를 사용한 통신의 흐름,
  - 공개 키 암호만으로는 해결할 수 없는 문제

Diffie 와 Hellman 공개키 암호의 영향

- 수학적 함수를 근거로 해서 만들었다
  - 공개키 알고리즘은 비트 패턴의 단순한 조작이 아니다
- 비대칭 방식
  - 공개키 암호는 오직 한 개의 키만 사용하는 대칭암호 방식과 대조적으로 서로 다른 두 개의 키를 이용한다
- 공개키는 두 개의 키를 사용한다
  - 기밀성, 키 분배, 인증 분야에서 매우 성능이 뛰어나다

5.2.1 공개 키 암호에 대한 잘못된 상식

- 공개키 암호가 대칭암호 보다 암호해독에 있어서 더 안전하다고 생각하는 것이다.
- 공개키 암호 기술을 일반적으로 사용하기 때문에 대칭암호를 더 이상 사용하지 않게 된다는 생각이다.
- 대칭암호의 경우 키 분배 센터를 이용해야하는 성가신 교환절차가 있기 때문에 공개키를 사용하면 키 분배가 매우 쉽다고 생각한다는 것이다.

사실은

- 공개키 암호에서도 모종의 프로토콜 형태가 필요하다
- 종종 중앙 에이전트가 필요할 때도 있다
- 사용하는 절차들도 대칭암호와 비교해 볼 때 더 단순하다거나 더 효과적이라고 말하기 어렵다.

5.2.2 공개 키 암호란

- 공개 키 암호(public-key cryptography)에서는 「암호화 키」와 「복호화 키」를 나눈다.
- 송신자는 「암호화 키」를 써서 메시지를 암호화하고, 수신자는 「복호화 키」를 써서 암호문을 복호화 한다.
  - 「암호화 키」는 암호화를 위해 송신자가 사용하는 것이고,
  - 「복호화 키」는 복호화를 위해 수신자가 사용하는 것이다.

암호화 키와 복호화 키의 구별

- 송신자가 필요한 것은 암호화 키뿐이다.
- 수신자가 필요한 것은 복호화 키뿐이다.
- 도청자에게 알려지면 곤란한 것은 복호화 키이다.
- 암호화 키는 도청자에게 알려져도 괜찮다.

공개 키(public key)

- 공개 키 암호에서 암호화 키는 일반에게 공개할 수 있다.
- 공개해도 상관이 없으므로 이 키를 **공개 키 (public key)**라 부른다.
- 공개 키 전달방법
  - 메일, 신문의 광고란, 간판으로 해서 길가에 세워도, Web 페이지로 전 세계에서 읽을 수 있도록 해도 괜찮다.
- 도청자 이브에게 공개 키가 도청되는 것을 신경 쓸 필요가 없다.

개인 키(private key)

- 복호화 키는 절대로 공개해서는 안 된다.
- 이 키는 당신만이 사용하는 것이다. 그러므로 이 키를 **개인 키(private key)**라 부른다.
- 개인 키는 다른 사람에게 보이거나, 건네주거나 해서는 안 된다.
- 개인 키는 자신의 통신 상대에게도 보여서는 안 되는 것이다.

대칭암호의 비밀키와 개인키

- 대칭암호에서 사용하는 키는 **비밀키(secret key)** 라고 일컬어진다.
- 공개키 암호에서 사용되는 두 개의 키는 **공개키(public key)** 와 **개인키(private key)** 라고 한다
- 항상 개인키는 비밀로 해야 하지만 비밀키라고 부르지 않고 **개인키**라고 부르기로 한다.
  - 그 이유는 대칭암호에서 사용하는 비밀키와 혼동하지 않도록 하기 위한 것이다.

키 쌍(key pair)

- 공개 키와 개인 키는 둘이 한 쌍이 된다.
- 이 키 쌍을 가리켜 **키 쌍(key pair)**이라고 부른다.
- 공개 키로 암호화한 암호문은 그 공개 키와 쌍이 되는 개인 키가 아니면 복호화할 수 없다.
- 키 쌍을 이루고 있는 2개의 키는 서로 밀접한 관계—수학적인 관계—가 있다.
  - 이 때문에 공개 키와 개인 키를 각각 별개로 만들 수는 없다.

5.2.3 공개 키 암호의 역사

- 1976년 휘트필드 디피(Whitfield Diffie)와 마틴 헬만(Martin Hellman)이 공개 키 암호의 아이디어를 발표
  - 공개 키가 어떠한 특성을 가져야 하는지를 제시
- 1977년, 공개 키 암호의 구체적인 알고리즘으로서 랠프 메르클레(Ralph Merkle)와 마틴 헬만(Martin Hellman)에 의한 **배낭(napsack) 암호**가 만들어졌다.

공개 키 암호의 역사

- 1978년, 공개키 암호 알고리즘 RSA 발표
  - 론 라이베스트(Ron Rivest),
  - 아디 샤미르(Adi Shamir),
  - 레너드 애들먼(Leonard Adleman)

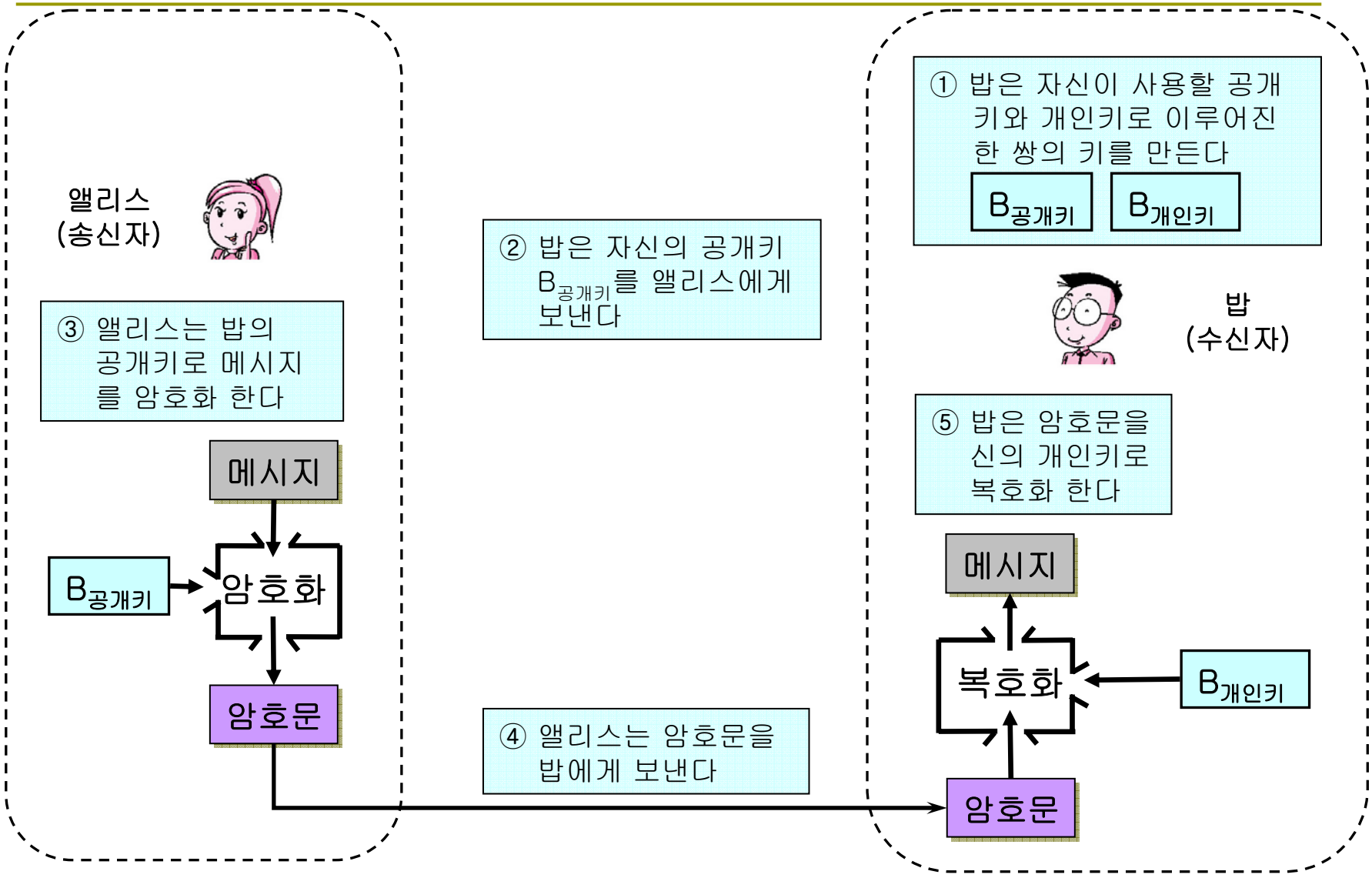
5.2.4 공개 키 암호 구조

- 다섯 가지 핵심요소
  - 평문(Plaintext):
  - 암호 알고리즘(Encryption algorithm):
  - 공개키와 개인키(Public and private key):
  - 암호문(Ciphertext):
  - 복호 알고리즘(Decryption algorithm):

5.2.5 공개 키를 사용한 통신의 흐름

1. 밥은 공개 키/개인 키로 이루어진 한 쌍의 키를 만든다.
2. 밥은 자신의 공개 키를 앨리스에게 보낸다.
3. 앨리스는 밥의 공개 키를 써서 메시지를 암호화한다.
4. 앨리스는 암호문을 밥에게 보낸다.
5. 밥은 자신의 개인 키를 써서 암호문을 복호화한다.

공개 키를 사용해서 앨리스가 밥에게 메시지 보내기



5.2.5 여러 가지 용어

- 공개 키 암호와 같은 의미로 **비대칭 암호**(asymmetric cryptography)라는 용어를 사용한다.
  - 이것은 대칭 암호와의 대비를 잘 나타내는 용어이다.
- 대칭 암호에서는 암호화와 복호화에서 같은 키를 사용
  - 대칭 암호에서는 암호화와 복호화가 마치 거울에 비친 것처럼 대칭을 이루고 있다.
- 그에 비해 비대칭 암호에서는 암호화와 복호화에서 다른 키를 사용한다.
  - 대칭이 아니고, 비대칭인 암호라고 부를 수 있다.

5.2.6 공개 키 암호로도 해결할 수 없는 문제

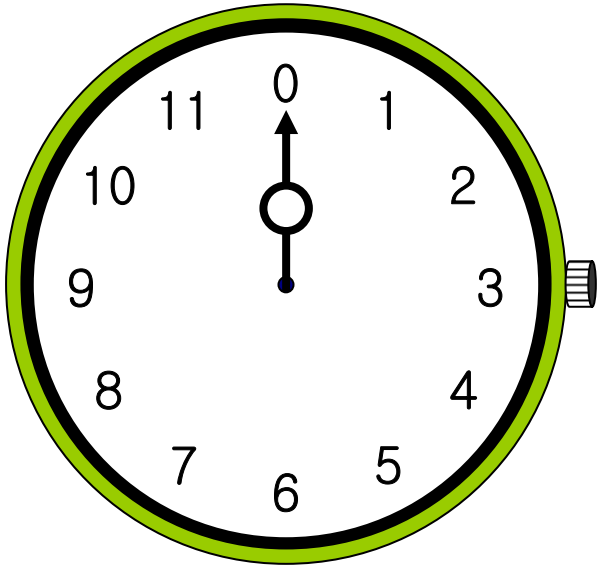
- 공개 키에 대한 검정
  - 입수한 공개키에 대한 판단이 필요
  - **공개 키의 인증**에 관한 문제이다.
  - 중간자공격(man-in-the-middle attack) 공격이 유효하다
- 처리 속도문제
  - 대칭 암호에 비해 처리 속도가 몇 백 배나 늦다는

5.3 시계 연산

- RSA 알고리즘을 이해하기 위해 필요한 가장 기본적이며 최소한의 연산인 시간 연산

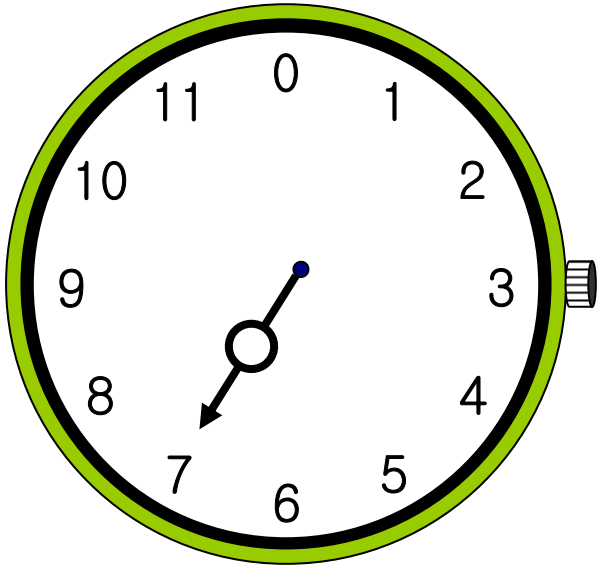
5.3.1 덧셈

- 바늘이 하나만 달린 시계를 떠올리기 바란다.
- 12의 자리에는 0이라고 쓰여 있다.
- 즉 0부터 11까지의 숫자가 표시된 시계이다



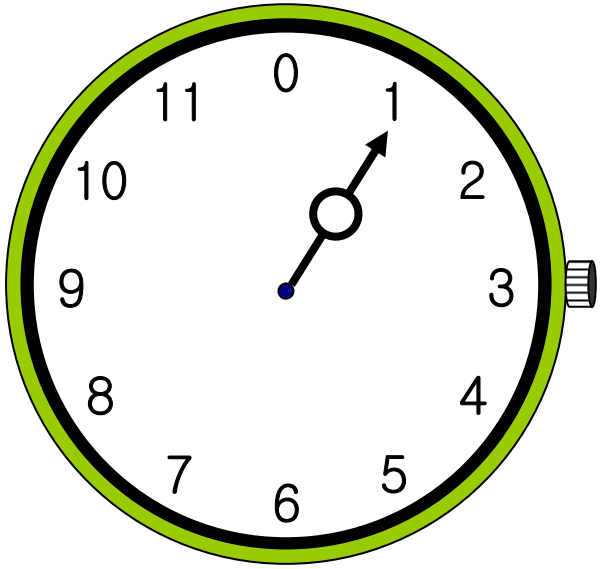
예

- 바늘이 7을 가리키고 있다고 하고 오른쪽으로 6눈금 보내면 바늘은 어디를 가리키는가?



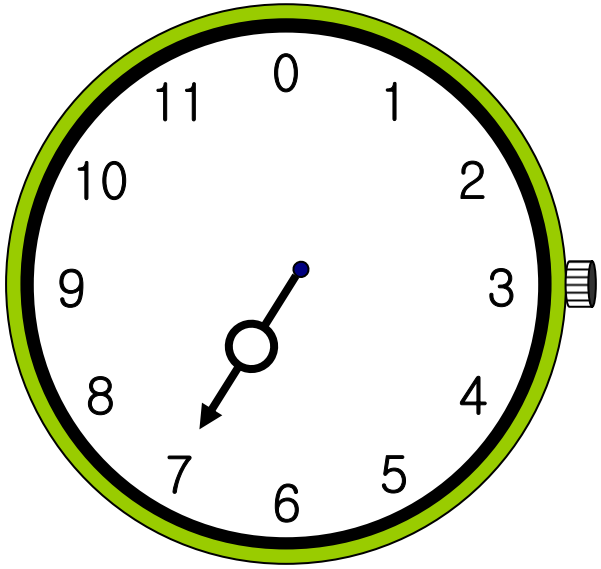
예

- 13일까?
- 아니다, 이 시계에는 13은 없다.
- 바늘은 1을 가리킨다.



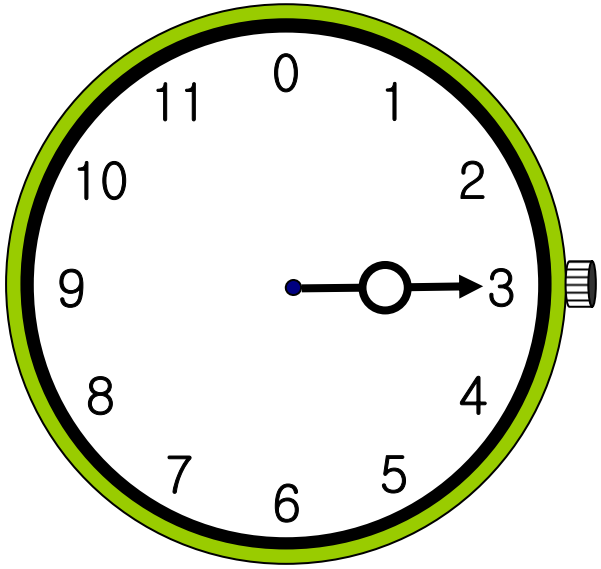
예

- 7을 가리키고 있다고 하고 오른쪽으로 20눈금을 보내면 바늘은 어디를 가리키는가?



예

- 음, 바늘은 27을 가리키는 게 아니고...
- 12마다 0으로 돌아가니까...
- 3을 가리키게 된다.



수행한 내용을 수식으로 보면

$$(7 + 20) \div 12$$

$$= 27 \div 12$$

$$= 2 \text{ 나머지 } 3$$

5.3.1.1 mod 계산

- 나눗셈을 해서 나머지를 구하는 계산을 위한 기호(연산자)의 표현

mod

mod 연산의 예

- mod를 사용하면 27을 12로 나누었을 때의 나머지를 다음과 같이 표기할 수 있다.
- 27 mod 12 란 27을 12로 나눈 나머지
- 27을 12로 나눈 나머지는 3이므로

$$27 \bmod 12 = 3$$

수학적 표현

- “27과 3은 12를 제수로 해서 합동이다” 라고 표현한다.
- 이야기를 정리해 보면
  - 시계를 오른쪽으로 돌린다는 것은 덧셈에 해당한다.
  - 단, 단순한 덧셈이 아니라 「나눗셈의 나머지(mod)」를 생각할 것.

mod 12 덧셈 연산표

표 5-1 mod 12 덧셈 연산표

a / b	0	1	2	3	4	5	6	7	8	9	10	11
0	0	1	2	3	4	5	6	7	8	9	10	11
1	1	2	3	4	5	6	7	8	9	10	11	0
2	2	3	4	5	6	7	8	9	10	11	0	1
3	3	4	5	6	7	8	9	10	11	0	1	2
4	4	5	6	7	8	9	10	11	0	1	2	3
5	5	6	7	8	9	10	11	0	1	2	3	4
6	6	7	8	9	10	11	0	1	2	3	4	5
7	7	8	9	10	11	0	1	2	3	4	5	6
8	8	9	10	11	0	1	2	3	4	5	6	7
9	9	10	11	0	1	2	3	4	5	6	7	8
10	10	11	0	1	2	3	4	5	6	7	8	9
11	11	0	1	2	3	4	5	6	7	8	9	10

5.3.2 뺄셈

- 뺄셈이라는 것은 덧셈의 역의 연산이므로 시계의 바늘을 왼쪽으로 돌리면 되는 것
- 그런데 여기서 시계를 왼쪽으로 돌리는 것을 금지한다.
- 바늘이 7을 가리키고 있을 때, 어떻게 하면 바늘이 0을 가리키게 되는가?

예

- 7에 뭔가를 더해서 12로 나눈 나머지는 0이 되는 수는 무엇인가를 알면 된다

$$(7 + \square) \bmod 12 = 0$$

- 여기에서 □에 들어갈 숫자는 무엇일까?
- 단 □는 0, 1, 2, ..., 10, 11 중 하나이어야 한다.
- 음수 -7은 여기서 생각하지 않는다.

예

- 7 + 5를 더해서 12로 나눈 나머지는 0이 된다

$$(7 + 5) \bmod 12 = 0$$

(X + Y) mod 12 = 0 이 되는 X와 Y의 짝

표 5-2 (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

mod 12 뺄셈 연산표

표 5-3 mod 12 뺄셈 연산표

a \ b	0	1	2	3	4	5	6	7	8	9	10	11
0	0	1	2	3	4	5	6	7	8	9	10	11
1	11	0	1	2	3	4	5	6	7	8	9	10
2	10	11	0	1	2	3	4	5	6	7	8	9
3	9	10	11	0	1	2	3	4	5	6	7	8
4	8	9	10	11	0	1	2	3	4	5	6	7
5	7	8	9	10	11	0	1	2	3	4	5	6
6	6	7	8	9	10	11	0	1	2	3	4	5
7	5	6	7	8	9	10	11	0	1	2	3	4
8	4	5	6	7	8	9	10	11	0	1	2	3
9	3	4	5	6	7	8	9	10	11	0	1	2
10	2	3	4	5	6	7	8	9	10	11	0	1
11	1	2	3	4	5	6	7	8	9	10	11	0

5.3.3 곱셈

- 보통 산수에서는 곱셈은 「덧셈을 반복한 것」이다.
- 예를 들면 7× 4라는 것은 7을 4회 더한 것이다.

$$7 \times 4 = 7 + 7 + 7 + 7$$

시계 연산의 곱셈도 덧셈의 반복

- (7× 4는 28이므로)

$$7 \times 4 \bmod 12 = 28 \bmod 12$$

- (28÷ 12는 몫이 2이고 나머지가 4이므로)

$$28 \bmod 12 = 4$$

mod 12 곱셈 연산표

표 5-4 mod 12 곱셈 연산표

a \ b	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11
2	0	2	4	6	8	10	0	2	4	6	8	10
3	0	3	6	9	0	3	6	9	0	3	6	9
4	0	4	8	0	4	8	0	4	8	0	4	8
5	0	5	10	3	8	1	6	11	4	9	2	7
6	0	6	0	6	0	6	0	6	0	6	0	6
7	0	7	2	9	4	11	6	1	8	3	10	5
8	0	8	4	0	8	4	0	8	4	0	8	4
9	0	9	6	3	0	9	6	3	0	9	6	3
10	0	10	8	6	4	2	0	10	8	6	4	2
11	0	11	10	9	8	7	6	5	4	3	2	1

5.3.4 나눗셈

- 뺄셈을 생각할 때 덧셈의 역연산을 생각한 것처럼 곱셈의 역연산을 생각한다.
- 예를 들면 다음 식에 대해 생각해 본다.
- 7에 □를 곱해서 12의 mod를 취했더니 1이 되었다. 이때의 □는 무엇일까?

$$7 \times \square \bmod 12 = 1$$

바늘의 조작으로 생각하면

- 「7눈금 오른쪽으로 돌리는」 조작을 몇 회 반복하면 10이 되는가? 하는 문제이다.
- 이것은 금방은 알 수 없다. □에 0, 1, 2, …을 순서대로 넣어서 7× □ mod 12를 계산해 본다.

7× □ mod 12 계산표

표 5-5 7 × □ mod 12 계산표

□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

mod 12의 세계에서는

- 이 표에서 □는 7이라는 것을 알 수 있다.

$$7 \times 7 \bmod 12 = 1$$

- 지금 생각한 것은 「mod 12의 세계에서는 7에 무엇을 곱하면 1이 되는가?」라는 문제였다.
- 바꿔 말하면 「mod 12의 세계에서는 1÷ 7의 답은 무엇인가?」라는 문제가 된다.
- 즉, 12로 나누어 남는 수를 취하는 세계에서 나눗셈을 생각한 것이 되는 것이다.

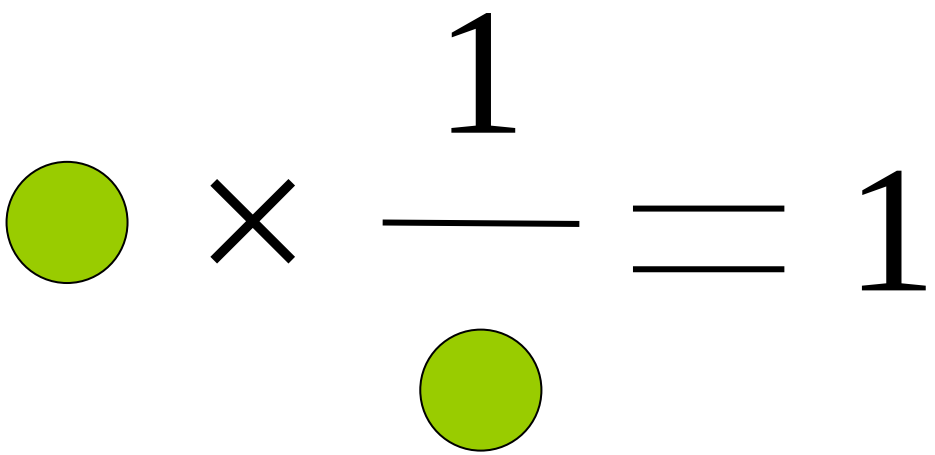
mod 세상의 역수

$$\bigcirc \times \square \qquad = 1$$

라는 관계식을 보면서 손으로 잠깐 mod 12의 부분을 가려 본다.

- ○와 □는 역수의 관계에 있다는 것이다.
- 역수란, 곱하면 1이 되는 수를 말한다.

보통의 산수라면



mod 세상에서는...

- 0부터 11까지 중에서 어떤 수에도 역수는 있는 것일까?
- 시계 연산에서 「어떤 수의 역수가 존재하는지 어떤지」하는 문제는, 공개 키 알고리즘 RSA에서 「공개 키와 쌍을 이루는 개인 키가 존재하는지 어떤지」하는 문제와 직결되어 있다.

0의 역수는 있는가?

- 0 눈금의 회전(즉 돌리지 않는 것)을 아무리 반복해도 1까지 바늘을 나아가게 할 수는 없으므로,

$$0 \times \square \bmod 12 = 1$$

을 충족시키는 □는 존재하지 않는다.

1의 역수는 어떤가?

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

을 충족시키는 □는 분명히 1이다.

2의 역수는 어떤가?

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

을 충족시키는 □는 존재하지 않는다.

- 왜냐 하면 2눈금의 회전을 반복해도 0, 2, 4, 6, 8, 10, 0, 2, 4, 6, 8, ...처럼 짝수인 곳만을 가리키기 때문이다.
- 절대로 1은 가리키지 않는다.

다른 수들은?

마찬가지로

$$3\times \square \bmod 12 = 1$$

나

$$4\times \square \bmod 12 = 1$$

를 충족시키는 □도 없다.

여기서 일람표로 만들어 본다

- 0은 특별하므로 제외하도록 한다.
- 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

모두 다 역수를 갖지는 않는다

- 역수를 가지고 있는 수는 1, 5, 7, 11밖에 없다
- 소수인가라고 생각할 수도 있지만 꼭 그렇지도 않다(2와 3은 소수이지만 역수를 안 가진다).
- 그 수와 12의 최대공약수가 1인지 아닌지로 판단할 수가 있는 것이다.

최대공약수를 살펴보면...

- 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

최대공약수를 살펴보면...

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

서로 소(素)

- 12와의 최대공약수가 1인 수(5, 7, 11)는 수학에서는 12와 **서로 소(素)인 수**라고 부른다.

5.3.5 거듭제곱

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

시계에서의 거듭제곱이란?

- 「「「「7눈금 오른쪽으로 돌린다」를 7회 반복한다」를 7회 반복한다」를 7회 반복한다」

- 바늘은 어디를 가리키게 되는가?
- 실제 계산은 간단하다.
- 여하튼 거듭제곱 계산을 하고 나서 나눗셈의 나머지를 취하면 된다(mod를 취하면 되는).

실제 계산

$74 \bmod 12 = 7 \times 7 \times 7 \times 7 \bmod 12$

$= 2401 \bmod 12$

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

$= 1$

(∵ 2401 ÷ 12는 몫이 200이고 나머지가 1)

좀 쉬운 계산

$$
\begin{aligned}
&74 \bmod 12 \\
&\quad = 7 \times 7 \times 7 \times 7 \bmod 12 \\
&\quad = ((7 \times 7 \bmod 12) \times (7 \times 7 \bmod 12)) \bmod 12 \\
&\quad = ((49 \bmod 12) \times (49 \bmod 12)) \bmod 12 \\
&\quad = (1 \times 1) \bmod 12 \\
&\quad = 1 \bmod 12 \\
&\quad = 1
\end{aligned}
$$

5.3.6 대수

- 거듭제곱의 역연산을 대수라고 한다.
- 보통의 수학에서 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.

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

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

이 되는 X는 무엇일까?

다음과 같이 하면 X는 9라는 것을 알 수 있다.

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

우리가 주의 해야 할 점

- 이산대수 구하기는 어렵다
  - 위의 예에서는 쉽게 X의 값이 9라는 것을 알 수 있다
  - 하지만 7과 같은 작은 수가 아니고 매우 큰수라면 해당 이산대수를 구하기는 매우 어렵다
  - 특히 숫자가 백 자리 이상으로 커지면 이산 대수를 구하는 것은 고속연산을 수행할 수 있는 컴퓨터를 이용하더라도 상당히 어렵고 시간이 엄청나게 많이 걸린다.

좋은 방법이 아직 알려지지 않았다

- 현재까지 이산 대수를 구하는 고속 알고리즘은 아직 발견되지 않았다.
- 이산 대수는 키 교환 프로토콜의 하나인 Diffie-Hellman 키 교환이나, 공개 키 알고리즘의 하나인 ElGamal 방식에서 사용되고 있다.

5.3.7 시계의 바늘에서 RSA로

여기서 알아야 할 중요한 것은 단 하나이다. 그것은,

$$7^4 \bmod 12$$

와 같은 식이 나와도 당황하지 말라는 것이다. 침착하게 7을 4제곱해서 12로 나눈 나머지라고 읽을 수 있다면 여러분은 RSA를 이해할 준비가 된 것이다.

5.4 RSA

- 공개 키 암호에서는 「암호화 키」와 「복호화 키」 두 개의 키를 사용한다.
- 그러나 어떻게 하면 그렇게 할 수 있는 것일까?
- 이 절에서는 현재 가장 많이 사용되고 있는 공개 키 암호 알고리즘인 RSA에 대해서 설명을 한다

5.4.1 RSA란

- RSA는 공개 키 암호 알고리즘의 하나이다.
- RSA는 공개 키 암호와 디지털 서명에 사용할 수 있다.
- 1983년 RSA사는 미국에서 RSA 알고리즘에 대한 특허를 취득하였다.
  - 그러나 현재는 이미 특허 기한이 종료되었다.

5.4.2 RSA에 의한 암호화

- RSA에서는 평문도 키도 암호문도 숫자이다
- RSA 암호화

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

- RSA의 암호문은 평문을 나타내는 수를 E제곱해서 mod N을 취한 것이다.

E와 N은 무엇일까?

- RSA의 암호화는, 평문을 E제곱해서 mod N을 취하는 것이므로, E와 N이라는 한 쌍의 수를 알면 누구라도 암호화를 행할 수 있다.
- 따라서 E와 N이 RSA에서 사용하는 암호화 키가 된다.
- 즉, **(E, N) 이 공개 키**인 것이다.
  - 단 E와 N은 어떤 수라도 되는 것은 아니다.
  - 이들은 면밀한 계산에 의해 만들어진 수다.

5.4.3 RSA에 의한 복호화

- RSA에 의한 복호화는 암호화와 마찬가지로 간단하다.
- RSA의 복호화

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

- 암호문을 나타내는 수를 D제곱해서 mod N을 취하면 평문을 얻을 수 있다.
- 바꿔 말하면 암호문을 D회 곱해서 그 결과를 N으로 나눈 나머지를 구하면 그것이 평문이다.

D와 N은 무엇일까?

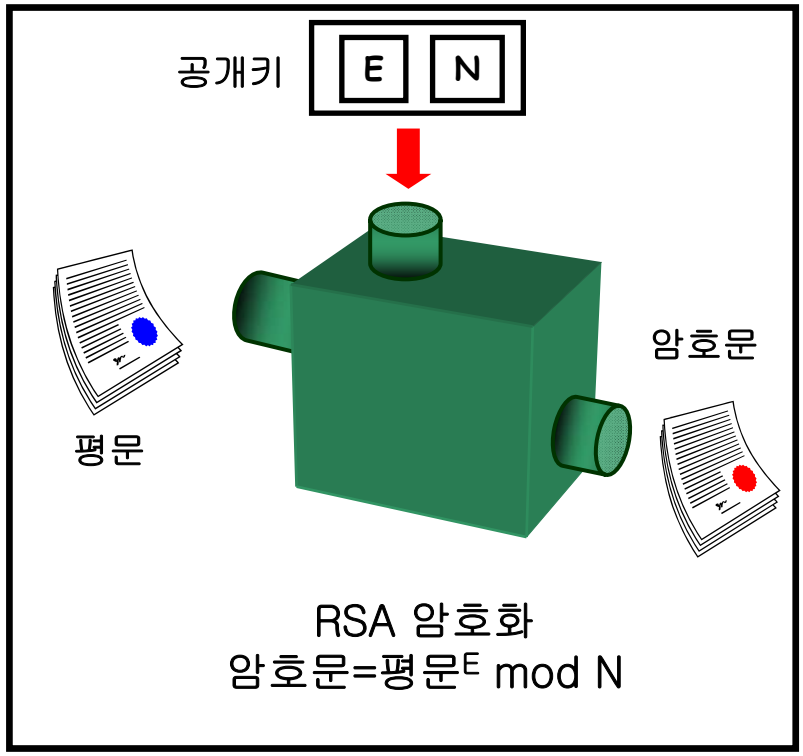
- 여기서 사용되고 있는 수 N은 암호화 때에 사용한 수 N과 같은 수이다.
- 수 D와 수 N을 합친 것이 RSA의 복호화 키가 된다.
- **D와 N의 쌍인 (D, N)이 개인 키**에 해당된다.
  - 궁극적으로 D와 N 모두를 알고 있는 사람만이 이 암호문을 복호화할 수 있다
  - D는 어떤 수라도 괜찮은 것은 아니다.
  - 복호화의 키인 수 D는 수 E와 깊은 수학적 연관이 있다.

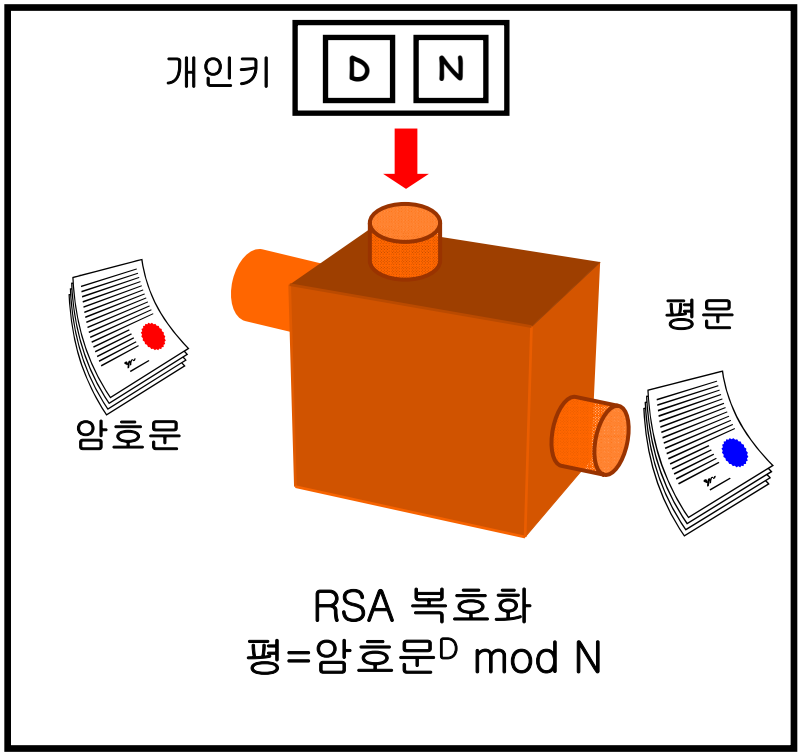
RSA의 암호화 · 복호화

표 5-6 RSA의 암호화 · 복호화

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

RSA의 암호화와 복호화





5.4.4 키 쌍의 생성

- E, D, N이라는 3개의 수는 어떻게 준비를 하는 것일까?
- E와 N이 공개 키, D와 N이 개인 키이므로 E, D, N 3개의 수를 구하는 것은 **키 쌍을 생성하는 것**과 다름없다.

RSA의 키 쌍 생성 순서

(1) N을 구한다

(2) L을 구한다

(L은 키 쌍을 생성할 때만 등장하는 수이다)

(3) E를 구한다

(4) D를 구한다

N 구하기

- 우선 처음에 큰 소수 p와 q 2개 준비한다.
- 2개의 수를 서로 곱한다.
- 그 결과로 나온 수가 우리가 구하는 N이다

$$N = p \times q \quad (p, q\text{는 소수})$$

L을 구한다

- L이라는 수는 RSA의 암호화나 복호화에는 등장하지 않는 수로서 키 쌍을 만드는 동안에만 보조적으로 사용하는 수이다
- L은 p-1과 q-1의 최소공배수(least common multiple; lcm)이다. 즉,

$$L = \mathrm{lcm}(p-1, q-1)$$

E를 구한다

- E는 1보다 크고, L보다도 작은 수이다.
- E와 L과의 최대공약수(greatest common divisor; gcd)는 1이다. 즉,

$$1 < E < L$$

$$\gcd(E, L) = 1$$

- E와 L의 최대공약수가 1이라는 조건은 암호화에서 사용하는 D의 존재를 보증하기 위해 필요한 조건이다.

D를 구한다

- D는 E로부터 계산해서 구한다.
- D와 E와 L은 다음과 같은 관계를 갖는다.

$$1 < D < L$$

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

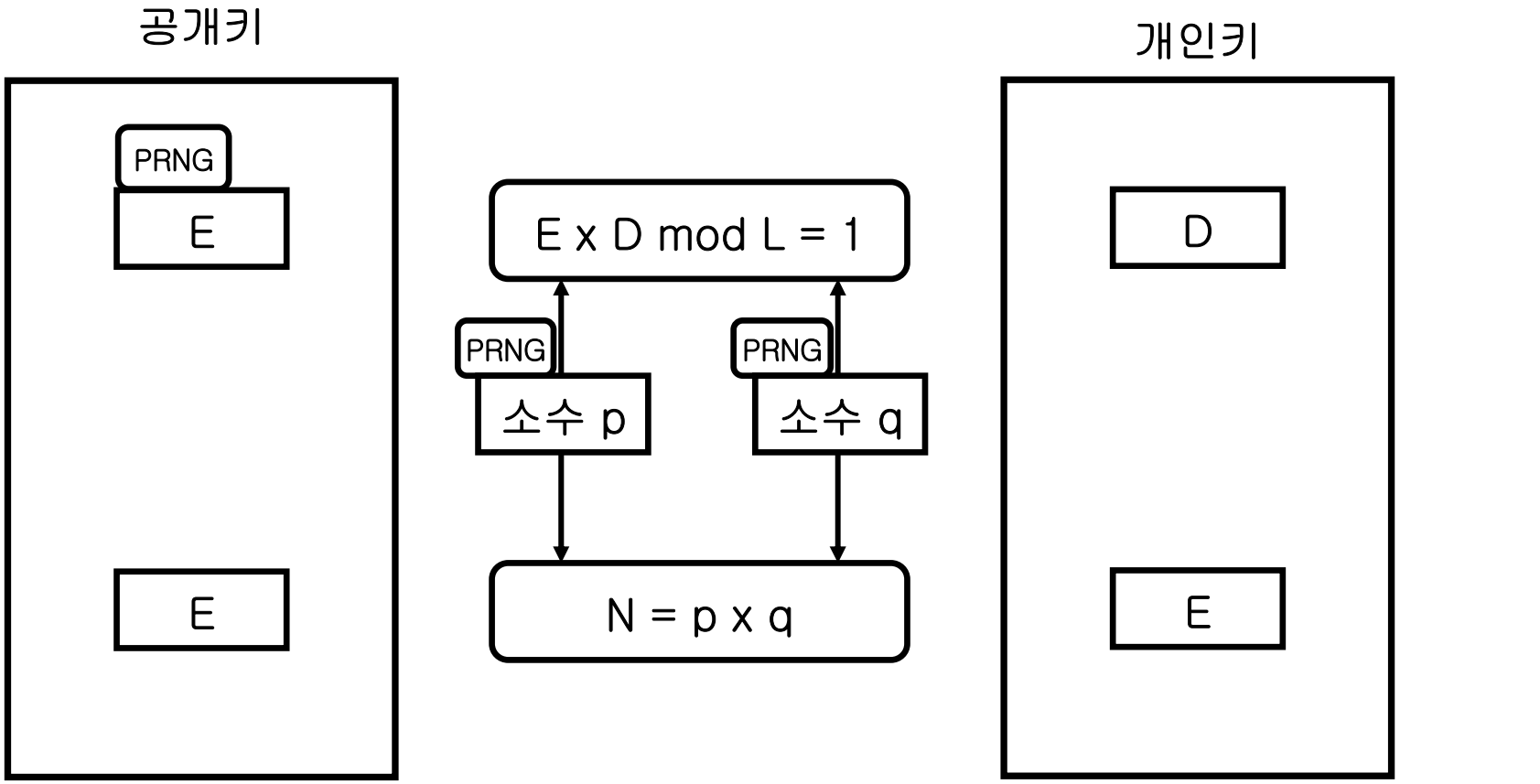
- $E \times D \bmod L = 1$은 암호문을 복호화하면 원래의 평문으로 돌아가는 것을 보증하기 위해 사용한다.

RSA의 키 쌍 생성(정리해보자)

표 5-7 RSA의 키 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 $p-1$과 $q-1$의 최소공배수
(3) E를 구한다	
$1 < E < L$	
$\gcd(E, L) = 1$	E와 L과의 최대공약수는 1(E와 L은 서로 소)
(4) D를 구한다	
$1 < D < L$	
$ED \bmod L = 1$	

RSA의 키 쌍



PRNG = 의사난수 생성기 L= lcm(p−1,q−1), gcd(E,L) = 1, 1 < E < L, 1 < D < L

5.4.5 구체적 계산

- 구체적인 수를 써서 RSA의 키 쌍 생성・암호화・복호화를 실제로 해 보자.
- 그렇지만 너무 큰 수를 사용하면 계산이 복잡하고 힘들기 때문에 작은 수를 이용하여 계산을 해 보자.

키 쌍의 생성 예

- N을 구한다
  - 2개의 소수 p=17, q=19 를 고른다.
  - N = p × q = 17× 19 = 323
- L을 구한다
  - L = lcm(p−1, q−1)
    = lcm(16. 18) (p−1은 16, q−1은 18)
    = 144 (16과 18의 최소공배수)

키 쌍의 생성 예

- E를 구한다
  - gcd(E, L) = gcd(E, 144) = 1 이 되는 E를 구한다
  - 이과 같은 E는 많이 있다. 예를 들면, 다음과 같은 수이다.

    5, 7, 11, 13, 17, 19, 23, 25, 29, 31, ...

  - 여기서는 E로서 5를 골라 보겠다.
  - 여기까지 해서 공개키는 (E,N)=(5, 323)이 되었다.

키 쌍의 생성 예

- D를 구한다
  - D는 식 E× D mod L = 1 을 충족시켜야만 된다.
  - E에 뭔가를 곱해서 mod L을 취하면 1이 되는 수를 찾아보자.
  - D = 29는, 이 관계식을 충족시킨다. 왜냐 하면,

    E× D mod L = 5× 29 mod 144
    = 145 mod 144
    = 1
  - 여기까지 해서 개인키는 (D,N)=(29, 323)이 되었다.

5.4.6 암호화

- 평문은 N 미만의 수, 즉 323 미만의 수가 아니면 안 된다.
- 평문으로서 123을 암호화해 보자.

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

- 따라서 암호문은 225가 된다.

5.4.7 복호화

- 암호문 225를 복호화 하겠다.

- 복호화에서는 개인 키 D = 29, N = 323을 사용한다.

$$\text{암호문}^{D} \bmod N = 225^{29} \bmod 323$$
$$= 123$$

$225^{29} \bmod 323$의 계산

- $225^{29} = 225^{10+10+9}$

  $= 225^{10}+225^{10}+225^{9}$

  - $225^{10} = 33252567300796508789062 5$
  - $225^{9} = 1477891880035400390625$

- $225^{10} \bmod 323 = 33252567300796508789062 5 \bmod 323 = 16$
- $225^{9} \bmod 323 = 1477891880035400390625 \bmod 323 = 191$

225^{29} mod 323의 계산

- 225^{29} mod 323

  $= 225^{10} \times 225^{10} \times 225^{9}$ mod 323

  $= \underline{(225^{10} \text{ mod } 323)} \times \underline{(225^{10} \text{ mod } 323)} \times$

  $\underline{(225^{9} \text{ mod } 323)}$ mod 323

  $= 16 \times 16 \times 191$ mod 323

  $= 48896$ mod 323

  $= 123$

5.5 RSA에 대한 공격

- 기밀성은 어떻게 되어 있는 것일까?
- 즉, 암호해독자는 왜 평문을 복원할 수는 없는 것일까?
- 여기서 암호해독자가 알고 있는 것과 모르는 것을 정리해 본다.

암호해독자가 아는 것과 모르는 것

- 【암호해독자가 알고 있는 것】
  - 암호문 : 도청해서 얻은 것이라고 하면 알기 쉽겠군.
  - 수 E와 N : 공개 키로서 공개되어 있으므로 암호해독자는 E와 N을 알고 있다.

- 【암호해독자가 모르는 것】
  - 평문 : 지금부터 해독하려고 하는 내용이다.
  - 수 D : 개인 키 중 적어도 D는 모르고 있다.
  - 기타 : 암호해독자는 키 쌍을 만들었을 때의 p, q, L을 모르고 있다.

5.5.1 암호문으로부터 평문 구하기

- RSA의 암호문은 다음과 같은 형태이다

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

- 만약 mod N이 없고,

$$\text{암호문} = \text{평문}^{E}$$

이었다면 암호문으로부터 평문을 구하는 것은 그다지 어렵지 않다.

이산대수 구하기는 매우 어렵다

- 하지만 mod N이 붙어 있으면 평문을 구하는 것은 이산 대수를 구한다는 매우 곤란한 문제가 된다.
- 인류는 아직 이산 대수를 구하는 빠른 방법을 알지 못하기 때문이다.

5.5.2 전사(brute force) 공격

- 수 D를 알면 암호문을 복호화 할 수 있다.
- 그렇기 때문에 RSA에 대한 공격 방법으로서, D의 후보가 되는 수를 순서대로 시도해서 복호화 한다는 전사공격을 생각할 수 있다.
- 전사공격은 D의 비트 수가 크면 클수록 시도해 보아야 할 수도 많아지고 계산도 복잡해지므로 점점 더 시간이 걸리며 어려워진다.
- 따라서 비트 수가 충분히 크면 전사공격으로 수 D를 찾아내는 것은 현실적으로는 불가능해진다.

전사공격을 어렵게 하기

- 통상 RSA에서는 p와 q의 비트 수로서 512 비트 이상을 N은 1024 비트 이상을 이용한다.
- E나 D는 N과 같은 정도의 크기로 할 수 있으므로 D를 찾아내기 위해서는 1024 비트 이상의 전사공격이 필요해진다.
- 이 비트 길이의 전사공격에 의해 D를 찾아내는 것은 지극히 곤란하다.

5.5.3 E와 N으로부터 D 구하기

- 암호해독자는 D를 알지 못한다.
- 그러나 공개 키인 E와 N은 알고 있다.
- 키 쌍을 만들 때 D는 원래 E로부터 계산해서 얻은 것이었다.
- 그렇다면 암호해독자도 E로부터 D를 구할 수 있는 것은 아닐까?

어렵다

- 아니다. 키 쌍을 만들었을 때의 방법을 생각해 보라. D와 E의 관계식,

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

- 에서 사용된 것은 L이다.
- L은 lcm(p-1, q-1)이므로 E로부터 D를 계산할 때는 p와 q를 사용하게 된다.
- 하지만, 암호해독자는 p와 q를 전혀 모른다.
- 따라서 키 쌍을 만들었을 때와 똑같은 계산을 해서 D를 구할 수는 없다.

RSA의 중요한 포인트

- 소수 p와 q를 암호해독자가 알게 해서는 안 된다는 것이다.
- p와 q가 암호해독자의 손에 넘어 간다는 것은 개인 키가 암호해독자의 손에 들어가는 것과 같다.

N을 소인수분해하는 공격

- p와 q를 암호해독자가 알게 해서는 안 된다.
- 하지만 N = p× q라는 관계식이 있고, 게다가 N은 공개되어 있다.
- N으로부터 p와 q를 구할 수는 없는 것일까?
- p와 q는 소수이기 때문에 N으로부터 p와 q를 구하는 것은 자연수 **N을 소인수분해하는 것**과 같다.
- **큰 수의 소인수분해를 고속으로 할 수 있는 방법이 발견되면 RSA는 해독할 수 있다**

하지만

- 그러나 현재 큰 수의 소인수분해를 고속으로 행하는 방법은 아직 발견되지 않았다.
- 소인수분해가 정말로 어려운 문제인가 하는 것은 증명된 것이 아니며, 소인수분해를 간단히 행하는 방법이 존재하는지의 여부도 아직 모른다.

p와 q를 추측하는 공격

- 소인수분해를 하지 않아도 p와 q가 암호해독자에게 알려질 가능성은 있다.
- p와 q는 의사난수 생성기로 생성하는 것이기 때문에 의사난수 생성기의 품질이 나쁘면 p와 q를 암호해독자가 추측해 버릴 우려가 있다.
- 생성하는 난수를 암호해독자가 추측할 수 있어서는 안 된다.

기타 공격

- N을 소인수분해해서 p와 q를 구할 수 있으면 D를 구할 수 있다.
- 그러나 「D를 구하는 것」이 「N을 소인수분해하는 것」과 같은 것인지 어떤지는 수학적으로 증명되어 있는 것은 아니다.
- 어쩌면 N을 소인수분해하지 않아도(p와 q를 몰라도) E와 N으로부터 D를 구하는 방법이 발견될지도 모른다.
- 이와 같은 방법은 아직 발견되지 않았고 애초부터 존재하는 것이지 어떤지도 모른다.

5.5.4 중간자(man-in-the-middle) 공격

- RSA를 해독하는 것은 아니지만, 기밀성에 대한 매우 유효한 공격 방법이다.
- man-in-the-middle 공격이란 적극적 공격자 맬로리가 송신자와 수신자 사이에 들어가서, 송신자에 대해서는 수신자처럼, 수신자에 대해서는 송신자처럼 행세하는 공격이다.

중간자 공격 절차

(1) 앨리스는 밥에게 메일을 보내서 공개 키를 받으려고 한다.

- 「밥에게 : 당신의 공개 키를 주십시오. 앨리스로부터」

(2) 맬로리는, 앨리스가 밥에게 공개 키를 요청한 것을 도청으로 감지한다.

(3) 밥은 앨리스로부터 온 메일을 읽고 자신의 공개 키를 앨리스에게 보낸다.

- 「앨리스에게 : 이것이 내 공개 키야. 밥으로부터」

(4) 맬로리는 밥의 이 메일을 빼앗아서 앨리스에게 도달하지 못하도록 한다. 그리고 밥의 공개 키를 살짝 보존한다.

- 이 밥의 공개 키는 나중에 맬로리가 사용한다.

중간자 공격 절차

(5) 맬로리는 밥 행세를 하며 맬로리 자신의 공개 키를 앨리스에게 보낸다.

- 「앨리스에게 : 이것이 내 공개 키야. 밥으로부터」

(실은 맬로리가 자신의 공개키를 앨리스에게 보내면서 밥의 행세를 하는 것이다.)

(6) 앨리스는 자신의 메시지를 밥의 공개 키(실은 맬로리의 공개 키)로 암호화한다.

- 「밥에게 : 사랑해. 앨리스로부터」

그렇지만 앨리스가 가지고 있는 것은 밥의 공개 키가 아니고 맬로리의 공개 키이다. 그러므로 앨리스는 자신의 메시지를 맬로리의 공개 키로 암호화한 것이 된다.

(7) 앨리스는 암호화한 메시지를 밥에게 보낸다.

중간자 공격 절차

(8) 맬리로는 앨리스의 암호 메일을 빼앗는다. 이 암호 메일은 맬로리의 공개 키로 암호화되어 있기 때문에 맬로리는 복호화할 수 있다. 맬로리는 앨리스의 러브레터를 읽는다.

(9) 맬로리는 앨리스 행세를 하며 위조 메일을 만든다.

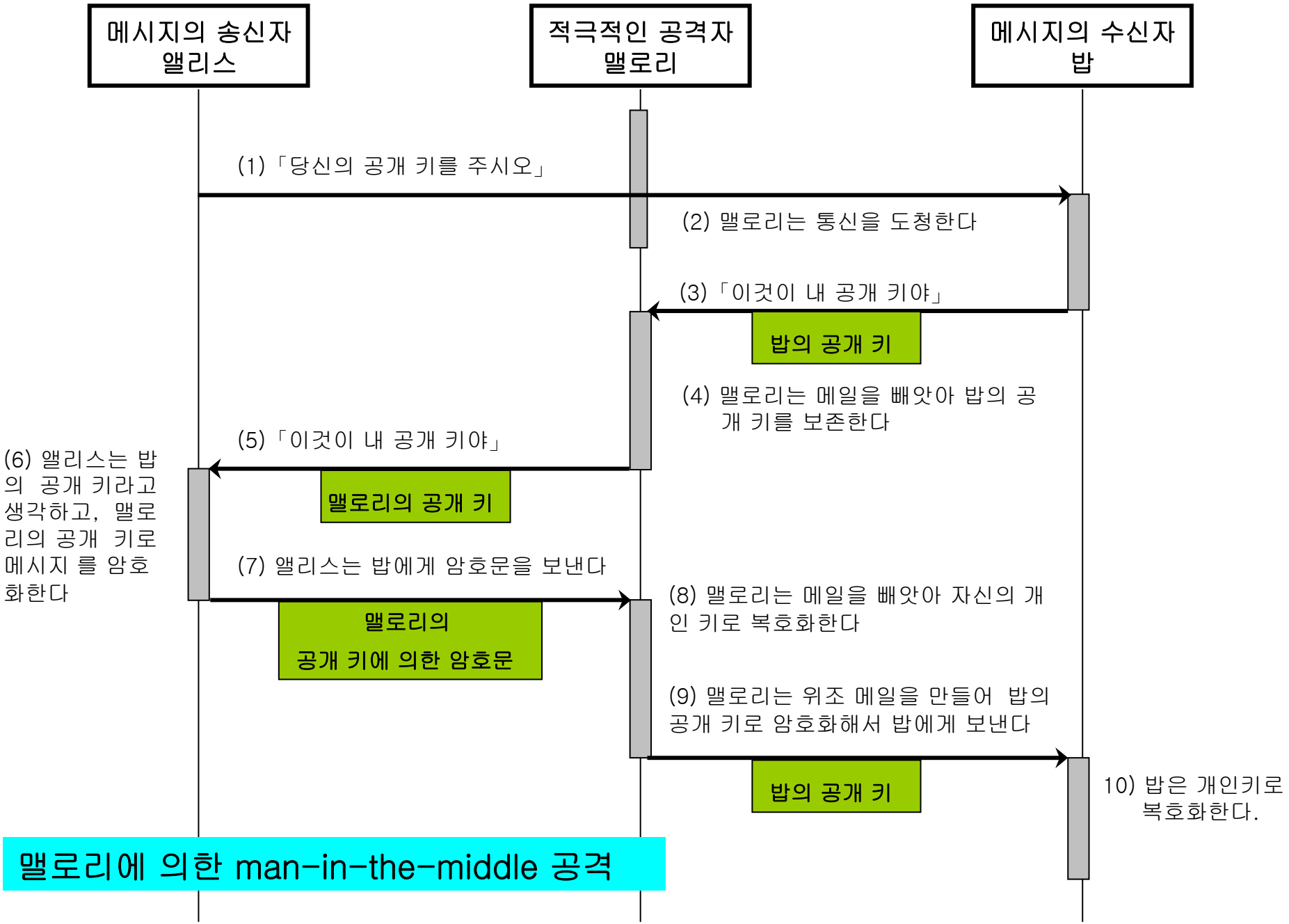
「밥에게 : 당신 따위 정말 싫어. 앨리스로부터」(실은 맬로리)

그리고 아까 (4)에서 보존해 둔 밥의 공개 키를 써서 이 위조 메일을 암호화하여 밥에게 보낸다.

(10) 밥은 받은 암호 메일을 자신의 개인 키로 복호화한다. 그리고,

「밥에게 : 당신 따위 정말 싫어. 앨리스로부터」(실은 맬로리)

라는 메일을 읽고 쇼크를 받는다.


메시지의 송신자
앨리스
적극적인 공격자
맬로리
메시지의 수신자
밥
(1)「당신의 공개 키를 주시오」
(2) 맬로리는 통신을 도청한다
(3)「이것이 내 공개 키야」
밥의 공개 키
(4) 맬로리는 메일을 빼앗아 밥의 공개 키를 보존한다
(5)「이것이 내 공개 키야」
맬로리의 공개 키
(6) 앨리스는 밥의 공개 키라고 생각하고, 맬로리의 공개 키로 메시지 를 암호화한다
(7) 앨리스는 밥에게 암호문을 보낸다
맬로리의
공개 키에 의한 암호문
(8) 맬로리는 메일을 빼앗아 자신의 개인 키로 복호화한다
(9) 맬로리는 위조 메일을 만들어 밥의 공개 키로 암호화해서 밥에게 보낸다
밥의 공개 키
10) 밥은 개인키로 복호화한다.
맬로리에 의한 man-in-the-middle 공격

5.6 다른 공개 키 암호

- RSA는 현재 가장 많이 보급되어 있는 공개 키 암호 알고리즘이다
- RSA 이외에도 공개 키 암호는 많이 있다.
  - ElGamal 방식
  - Rabin 방식
  - 타원 곡선 암호
- 이들 암호는 모두 암호와 디지털 서명에 이용할 수 있다.

5.6.1 ElGamal 방식

- **ElGamal 방식**은 Taher ElGamal에 의한 공개 키 알고리즘이다.
- RSA는 소인수분해가 곤란하다는 것을 이용했지만, ElGamal 방식에서는 mod N으로 이산 대수를 구하는 것이 곤란하다는 것을 이용한다.
- ElGamal 방식에 의한 암호화에서는 암호문의 길이가 평문의 2배가 되어 버린다는 결점이 있다.
- 암호 소프트웨어 GnuPG에 구현되어 있다.

5.6.2 Rabin 방식

- **Rabin 방식**은 M. O. Rabin에 의한 공개 키 알고리즘이다.
- Rabin 방식은 mod N으로 평방근을 구하는 것이 곤란하다는 것을 이용하고 있다.
- RSA는 큰 수 N의 소인수분해를 하지 않아도 해독할 수 있는 가능성이 있지만, Rabin 방식에 의한 공개 키 암호의 해독은 소인수분해를 행하는 것과 같은 정도로 어렵다는 것이 수학적으로 증명되어 있다.

5.6.3 타원 곡선 암호

- **타원 곡선 암호**(elliptic curve cryptosystems; ECC)는 최근 주목받고 있는 공개 키 암호 알고리즘이다.
- RSA에 비해 키의 비트 수를 적게 할 수 있는 것이 특징이다.
- 타원 곡선이라 불리는 곡선을 정하고, 그 곡선 상에 있는 점에 대하여 특수한 「연산」을 정의한다.
- 타원 곡선 암호에서는 이 연산의 역연산이 어렵다는 것을 이용한다.

5.7 공개 키 암호에 관한 Q&A

- Q&A의 형식으로 공개 키 암호에 관한 의문에 답하겠다.
- 주로 오해하기 쉬운 내용을 선택했다.

5.7.1 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 때문이다.

5.7.2 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.
  - 1024비트 길이의 공개 키 암호와 128비트 길이의 대칭 암호에서는, 128비트 길이의 대칭 암호 쪽이 전사 공격에 강하다는 것을 알 수 있다.

전사공격에 대해서 같은 강도를 갖는 키 길이 비교

표 5-8 전사공격에 대해서 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

5.7.3 대칭 암호의 미래

- 의문:
  - 공개 키 암호가 생겼기 때문에 앞으로 대칭 암호는 사용되지 않게 되는 것인가?
- 답:
  - 아니다.
  - 일반적으로 같은 정도의 기밀성을 갖는 키 길이를 선택했을 경우, 공개 키 암호는 대칭 암호보다도 몇 백 배나 느려진다.
  - 공개 키 암호는 긴 메시지를 암호화하기에는 적합하지 않다.
  - 목적에 따라 대칭 암호와 공개키 암호 두 가지 모두 사용되게 될 것이다.

5.7.4 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 10의 150거듭제곱으로 전 우주에 존재하는 원자의 개수보다도 많은 수이다.

5.7.5 RSA와 소인수분해

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

- 의문:
  - RSA를 해독하는 것은 큰 수를 소인수분해하는 것과 같은 것인가?
- 답:
  - 같은 것인지 아닌지 아직 모른다.
  - 분명히 소인수분해를 고속으로 할 수 있다면 RSA는 해독된다.
  - 그러나 RSA를 해독하려면 소인수분해를 꼭 해야 한다는 것이 증명된 것은 아니다.
  - 어쩌면 소인수분해를 하지 않아도 해독할 수 있는 방법이 발견될지도 모른다.

5.7.6 RSA의 비트 길이

- 의문:
  - 소인수분해 되지 않기 위해서는 N은 몇 비트의 길이가 필요한가?
- 답:
  - 몇 비트가 되어도 언젠가는 소인수분해 된다.