RSA 암호

2학년 임성빈

목차

RSA 암호 개요

RSA 암호의 탄생 (1978년)

로널드 라이베스트(Ron Rivest), 아디 샤미르(Adi Shamir), 레너드 애들먼(Leonard Adleman)

RSA

RSA 암호 개요

RSA 암호의 특징

- 공개키 방식
- 어려운 큰 수의 소인수분해

RSA 암호 개요

공개키 방식

공개키로 암호화하고 개인키로 복호화 한다

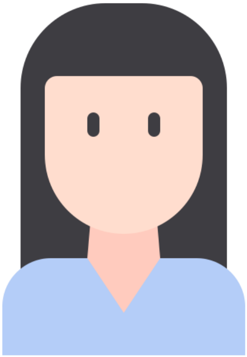
공개키 : 공개하는 키로 데이터를 보내는 사람이 암호화를 수행할 때 쓰인다

개인키 : 개인만이 알고 있어야 하는 키로, 복호화를 수행할 때 쓰인다

RSA 암호 개요

공개키 방식

밥

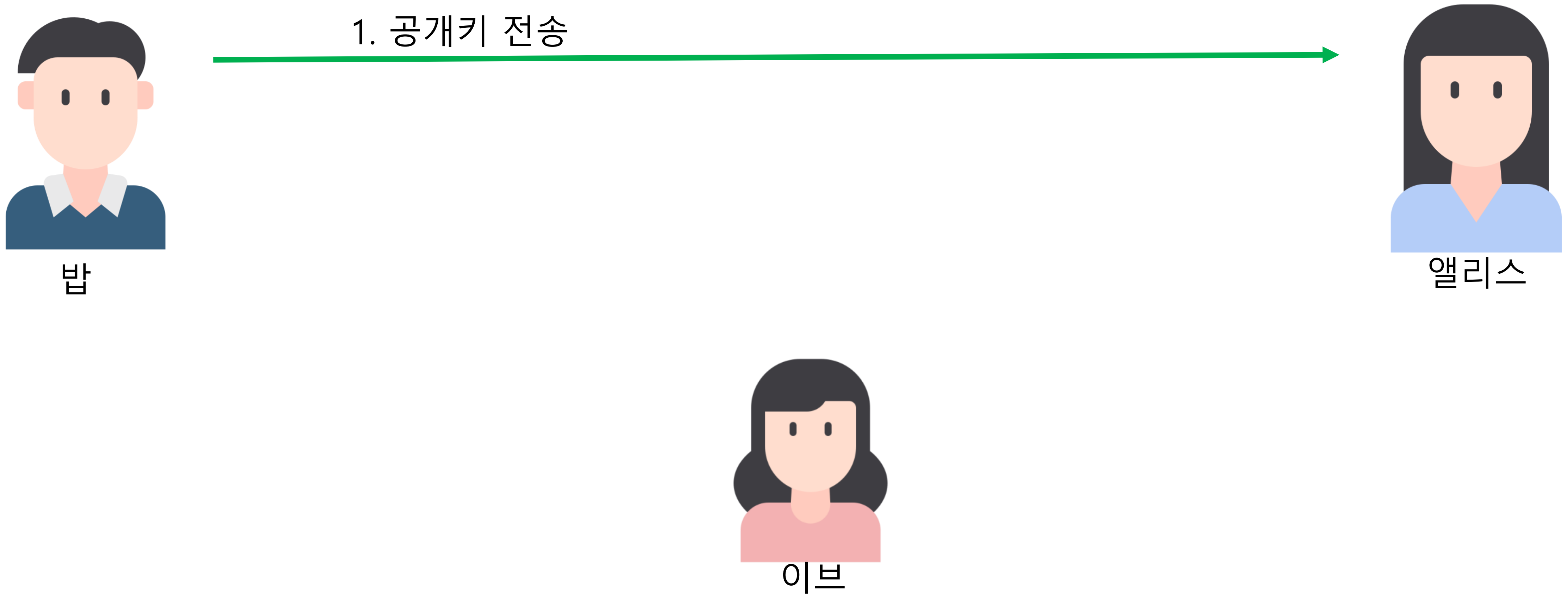


앨리스

이브

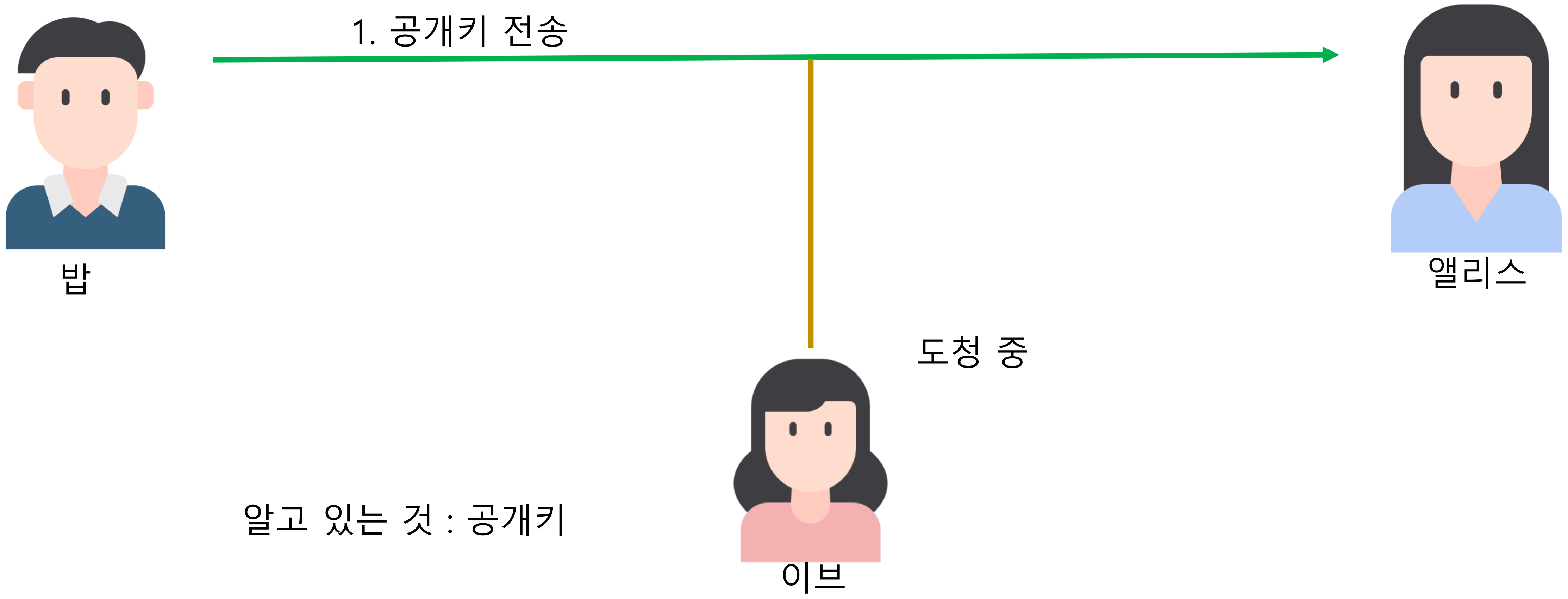
RSA 암호 개요

공개키 방식



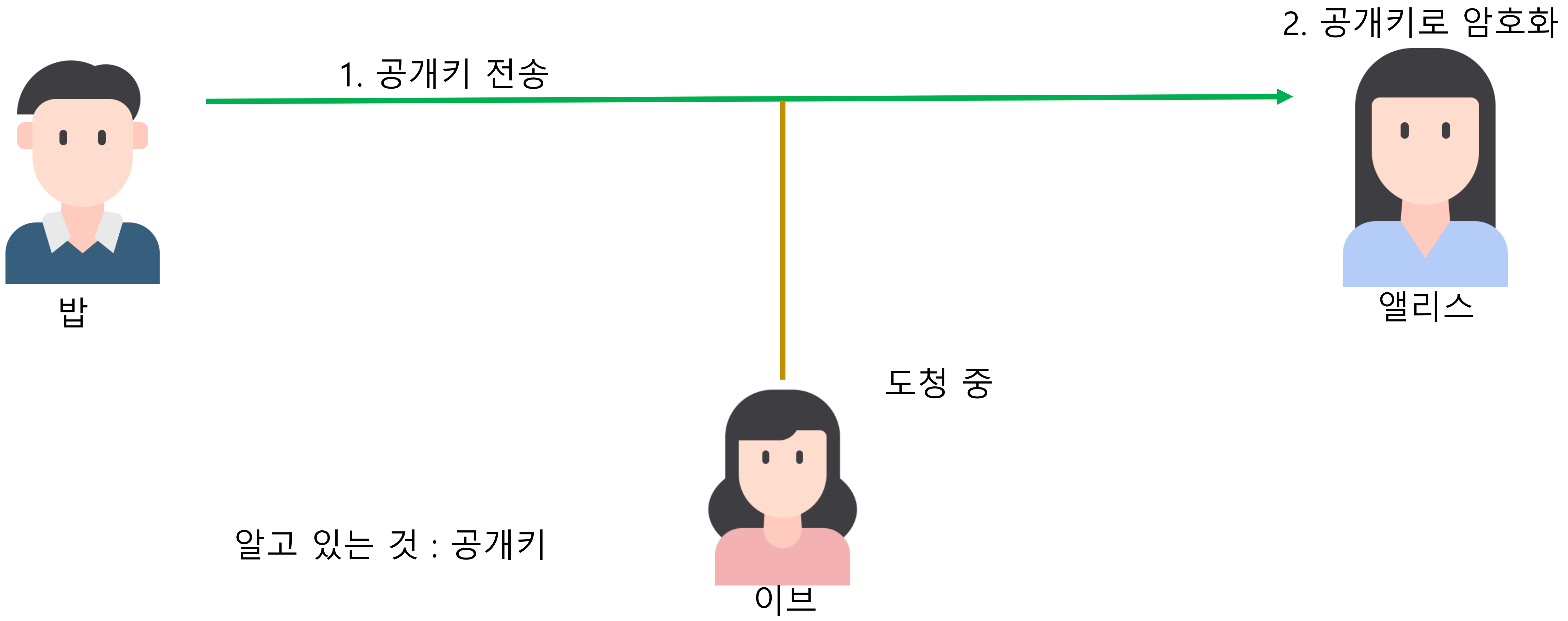
RSA 암호 개요

공개키 방식



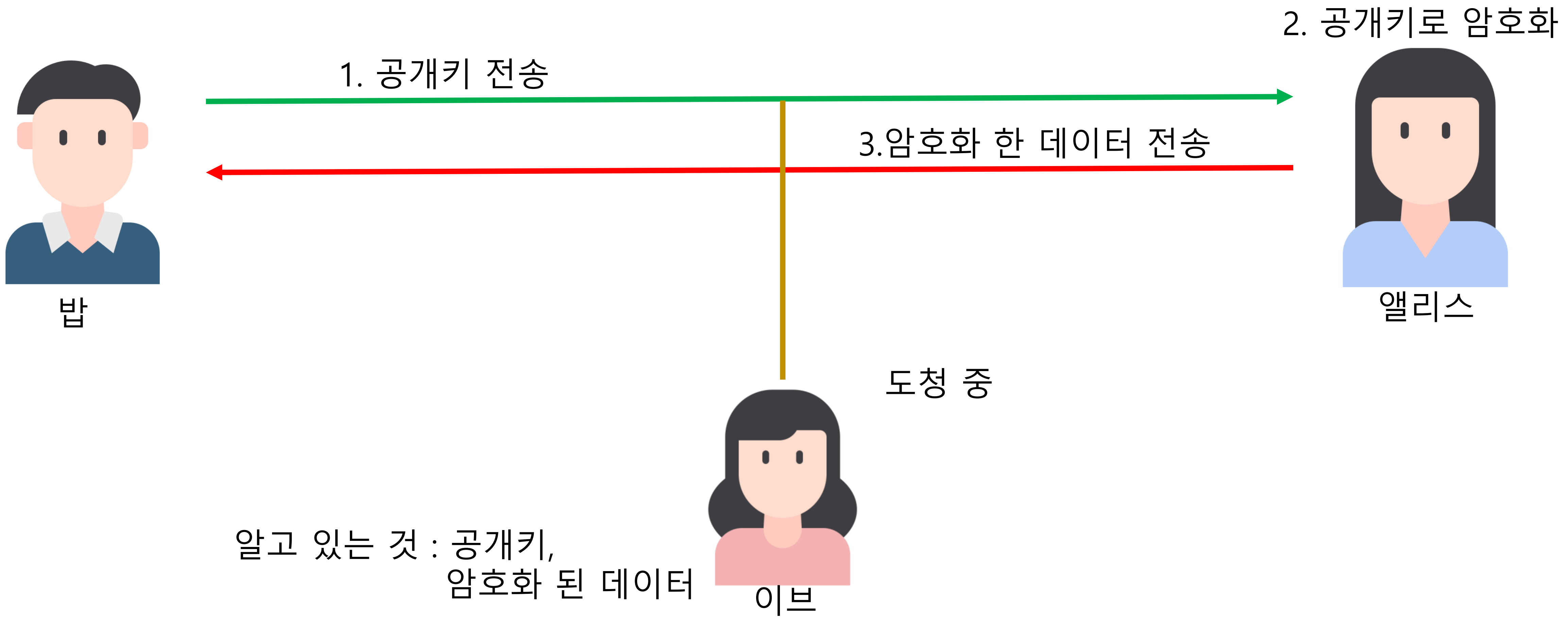
RSA 암호 개요

공개키 방식



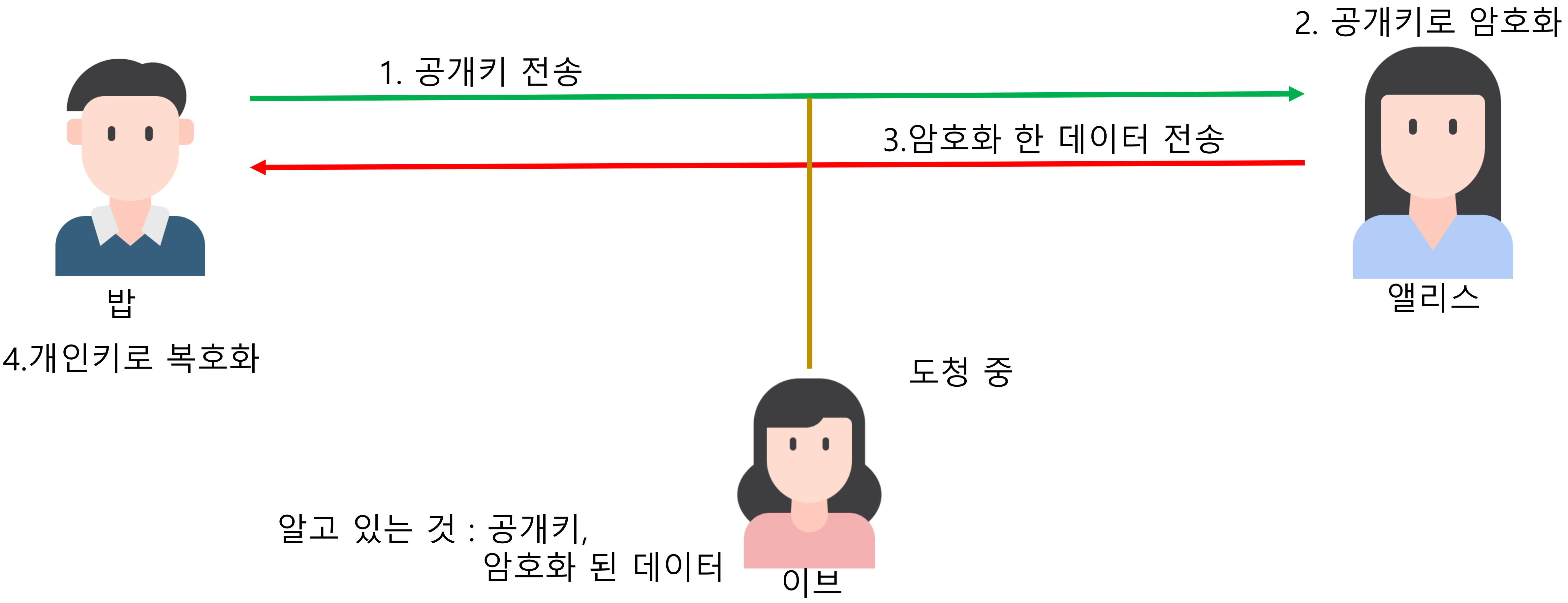
RSA 암호 개요

공개키 방식



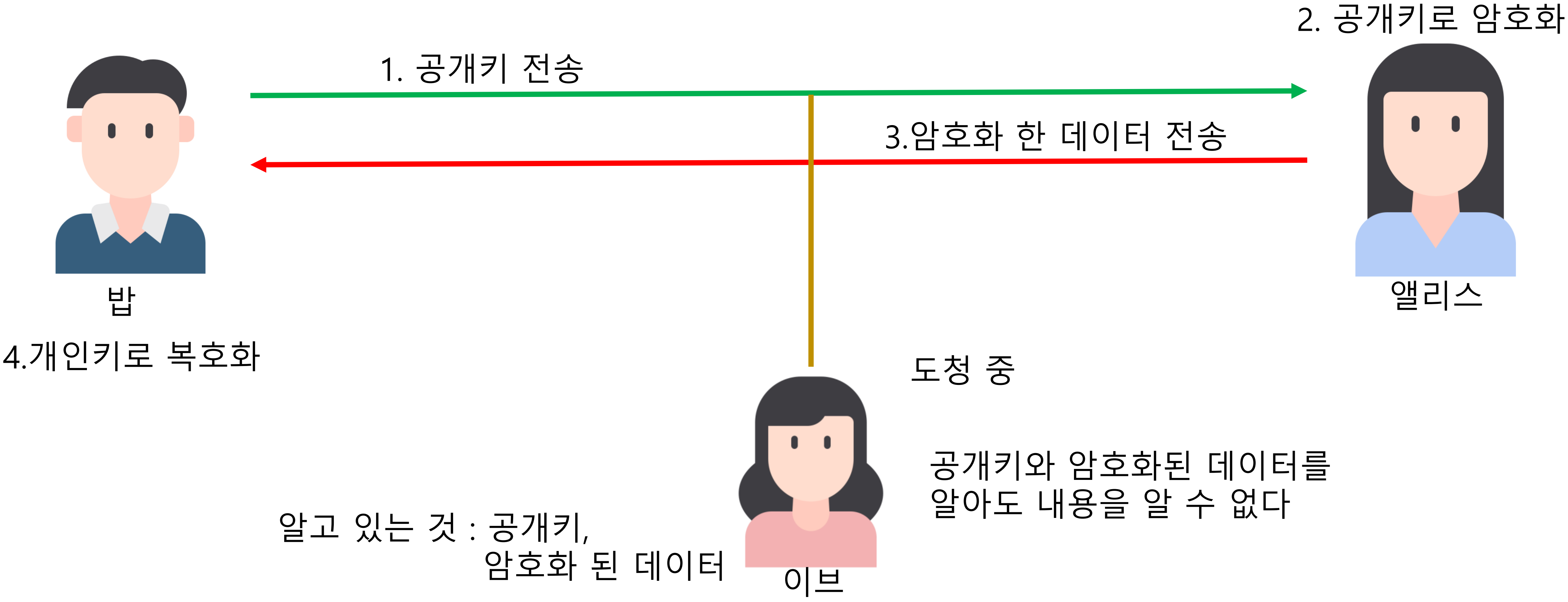
RSA 암호 개요

공개키 방식



RSA 암호 개요

공개키 방식



RSA 암호 개요

어려운 큰 수의 소인수분해

11659를 소인수분해 하면 몇이 나올까?

RSA 암호 개요

어려운 큰 수의 소인수분해

11659를 소인수분해 하면 몇이 나올까?

$$131 \times 89$$

RSA 암호 개요

어려운 큰 수의 소인수분해

11659를 소인수분해 하면 몇이 나올까?

$$131 \times 89$$

큰 두 소수의 곱의 소인수분해가 어려움을 이용한다

RSA 암호의 이해를 돕기 위한 약간의 정수론

mod 연산

$$a \bmod m = b$$

RSA 암호의 이해를 돕기 위한 약간의 정수론

mod 연산

$$a \bmod m = b$$

mod를 C언어의 %연산자라고 생각하자

RSA 암호의 이해를 돕기 위한 약간의 정수론

mod 연산

$$a \bmod m = b$$

mod를 C언어의 %연산자라고 생각하자

$$a\%m = b$$

RSA 암호의 이해를 돕기 위한 약간의 정수론

연습문제

$$17 \bmod 3 \ = \ ?$$

$$49 \bmod 7 \ = \ ?$$

RSA 암호의 이해를 돕기 위한 약간의 정수론

연습문제

$$17 \% 3 \ = \ ?$$

$$49 \bmod 7 \ = \ ?$$

RSA 암호의 이해를 돕기 위한 약간의 정수론

연습문제

$$17 \% 3 = 2$$

$$49 \bmod 7 = ?$$

RSA 암호의 이해를 돕기 위한 약간의 정수론

연습문제

$$17 \% 3 = 2$$

$$49 \% 7 = ?$$

RSA 암호의 이해를 돕기 위한 약간의 정수론

연습문제

$$17 \% 3 = 2$$

$$49 \% 7 = 0$$

RSA 암호의 이해를 돕기 위한 약간의 정수론

합동식

$$a \equiv b \mod m$$

RSA 암호의 이해를 돕기 위한 약간의 정수론

합동식

$$a \equiv b \bmod m$$

정의: a는 법 m에 대하여 b와 합동이다

RSA 암호의 이해를 돕기 위한 약간의 정수론

합동식

$$a \equiv b \bmod m$$

a와 b는 m으로 나누었을 때의 나머지가 서로 같다

RSA 암호의 이해를 돕기 위한 약간의 정수론

합동식

$$19 \equiv 7 \bmod 4$$

RSA 암호의 이해를 돕기 위한 약간의 정수론

합동식

$$19 \equiv 7 \bmod 4$$

19 % 4 = 3, 7 % 4 = 3

RSA 암호의 이해를 돕기 위한 약간의 정수론

합동식

$$19 \equiv 7 \bmod 4$$

19 % 4 = 3, 7 % 4 = 3

19와 7이 4로 나눈 나머지가 3으로 같다.

즉, 19와 7은 법3 에 대하여 합동이다.

RSA 암호의 원리

키 생성 과정

1. 두 소수 p, q와, N = pq를 구한다.

2. p – 1, q – 1과 서로소인 정수 e를 찾는다.
   (단, $3 < e < (p-1)(q-1)$)

3. ed를 (p-1)(q-1)으로 나눈 나머지가 1이 되도록 하는 d를 찾는다.
   (단, $1 < d < (p-1)(q-1)$)

4. N과 e를 공개하고 d는 숨긴다.

RSA 암호의 원리

1. 두 소수 p, q와, N = pq 를 구한다

$p = 7, q = 11$이라 정하자

$N = pq = 77$

구한 값

$p = 7, q = 11$

$N = 77$

RSA 암호의 원리

2. p – 1 q – 1과 서로소인 정수 e를 찾는다.

구한 값

p = 7, q = 11

N = 77

e = 17

p와 q보다 큰 소수를 사용하면 된다

e = 17

RSA 암호의 원리

3. ed를 (p-1)(q-1)으로 나눈 나머지가 1이 되도록 하는 d를 찾는다.

$$e\,d \equiv 1 \bmod (p-1)(q-1)$$

$$17d \equiv 1 \bmod 60$$

$$d \equiv 43 \bmod 60$$

$$d = 43$$

구한 값

p = 7, q = 11

N = 77

e = 17

$d = 43$

RSA 암호의 원리

3. ed를 (p-1)(q-1)으로 나눈 나머지가 1이 되도록 하는 d를 찾는다.

$$e\,d \equiv 1 \bmod (p-1)(q-1)$$

$$17d \equiv 1 \bmod 60$$

$$d \equiv 43 \bmod 60$$

$$d = 43$$

구한 값

p = 7, q = 11

N = 77

e = 17

$d = 43$

17d % 60 = 1이 되는 d를 구하면 된다

RSA 암호의 원리

4. N과 e를 공개하고 d는 숨긴다.

공개키 : e, N (17, 77)

개인키 : d, N (43, 77)

구한 값

p = 7, q = 11

N = 77

e = 17

$d = 43$

RSA 암호의 원리

RSA 암호화

전송하려는 평문을 a = 20이라 할 때, 공개키 e, N를 사용하여 암호화 한다.

$$x \equiv a^e \bmod N(\text{단}, a < N)$$

$$x \equiv 20^{17} \bmod 77$$

$$x = 48$$

구한 값

p = 7, q = 11

N = 77

e = 17

$d = 43$

$x = 48$

$a = 20$

RSA 암호의 원리

RSA 암호화

전송하려는 평문을 a = 20이라 할 때, 공개키 e, N를 사용하여 암호화 한다.

$$x \equiv a^e \bmod N(\text{단, } a < N)$$

$$x \equiv 20^{17} \bmod 77$$

x = 48

구한 값

p = 7, q = 11

N = 77

e = 17

$d = 43$

x = 48

$a = 20$

$a^e \,\%\, N$ 의 결과 x가 암호문

RSA 암호의 원리

RSA 암호화

전송하려는 평문을 a = 20이라 할 때, 공개키 e, N를 사용하여 암호화 한다.

$$x \equiv a^{e} \bmod N(\text{단, } a < N)$$

$$x \equiv 20^{17} \bmod 77$$

x = 48

구한 값

p = 7, q = 11

N = 77

e = 17

$d = 43$

x = 48

$a = 20$

$a^{e} \,\%\, N$ 의 결과 x가 암호문

암호문 $x = 48$ 을 전송한다.

RSA 암호의 원리

RSA 복호화

암호문 x 를 개인키 d, N를 이용하여 복호화 한다

$$a' \equiv x^d \bmod N$$

$$a' \equiv 48^{43} \bmod 77$$

$$a' \equiv 20 \bmod 77$$

$$a' = 20$$

구한 값

p = 7, q = 11

N = 77

e = 17

$d = 43$

x = 48

$a = 20, a' = 20$

RSA 암호의 원리

RSA 복호화

암호문 x 를 개인키 d, N를 이용하여 복호화 한다

$$a' \equiv x^d \bmod N$$

$$a' \equiv 48^{43} \bmod 77$$

$$a' \equiv 20 \bmod 77$$

$$a' = 20$$

$x^d \,\%\, N$ 의 결과 a' 가 복호문

구한 값

p = 7, q = 11

N = 77

e = 17

$d = 43$

$x = 48$

$a = 20, a' = 20$

RSA 암호의 원리

RSA 복호화

암호문 x 를 개인키 d, N를 이용하여 복호화 한다

$$a' \equiv x^d \bmod N$$

$$a' \equiv 48^{43} \bmod 77$$

$$a' \equiv 20 \bmod 77$$

$$a' = 20$$

구한 값

p = 7, q = 11

N = 77

e = 17

$d = 43$

$x = 48$

$a = 20, a' = 20$

x^d % N 의 결과 a' 가 복호문

$a' = a = 20$ 이므로 복호화가 성공적으로 수행되었다.

RSA 암호 구현 (python)

최소 공배수를 구하는 함수

```python
def gcd(a, b): #최소 공배수
    while b != 0:
        r = a % b
        a, b = b, r
    return a
```

유클리드 호제법을 이용한 알고리즘을 통해 최소 공배수를 구한다

유클리드 호제법 : $\mathrm{a} = bq + r \Rightarrow \mathrm{gcd}\{a, b\} = \mathrm{gcd}\{a, r\}$

RSA 암호 구현 (python)

페르마의 소정리

```python
def ferrmatLittleTheorem(p): #페르마 소정리
    for i in range(2, p):
        if(gcd(i, p) == 1):
            a = i
            break
    if (mod(a, p-1, p) == 1):
        return 1
```

소수를 빠르게 찾기 위해 사용

페르마의 소정리 : p가 소수일 때, $p \nmid a \Rightarrow a^{p-1} \bmod p$

RSA 암호 구현 (python)

소수 판정 함수

```python
def is_prime(p): #소수 판정
    if (p < 2):
        return 0
    elif(ferrmatLittleTheorem(p)):
        q = int(p**0.5) + 1
        for i in range(2 , q):
            if p % i == 0:
                return 0
        return 1
```

아래의 정리를 이용하여 소수를 판정한다

정수 n 이 합성수이면 n 은 $\sqrt{n}$ 이하인 적당한 소인수를 가진다.

RSA 암호 구현 (python)

소수 찾기

```python
def get_prime():  #소수 찾기
    while(True):
        number = random.randint(2**13 + 1, 2**14 + 1)
        if(number % 2 == 0):
            number = number + 1
        if(is_prime(number)):
            return number
```

8193 ~ 16385 사이의 소수를 반환

RSA 암호 구현 (python)

Mod 연산

```python
def mod(x, e, N):  #x**e mod N 연산
    y = x % N
    m, r = 1, 1
    d = bin(e)
    while m < len(d) - 1:
        if d[len(d) - m] == '1':
            r = (r * y) % N
        y = y**2 % N
        m = m + 1
    return r
```

x에 e제곱을 할 때, e가 크면 단순 계산으로는 시간이 많이 걸리므로

mod연산의 성질을 이용하여 연산속도를 크게 줄인다

RSA 암호 구현 (python)

공개키, 개인키 생성

```python
def get_key(): # 비밀키와 공개키 생성
    e = 65537 #p, q보다 큰 소수
    p, q = get_prime(), get_prime()
    N = p * q
    pi = (p - 1)* (q - 1)
    for d in range(N, 0, -1):
        if (e * d) % pi == 1:
            return [e, N], [d, N]
```

공개키, 개인키를 만들어 반환한다.

RSA 암호 구현 (python)

암호화, 복호화 함수

```python
def encryption(plaintext, publicKey):  #암호화, publickey = [e, N]
    return mod(plaintext, publicKey[0], publicKey[1])

def decryption(ciphertext, privateKey):    #복호화, privateKey = [d, N]
    return mod(ciphertext, privateKey[0], privateKey[1])
```

입력 받은 평문을 암호화 하고, 암호문을 복호화 한다.

RSA 암호 구현 (python)

6739204를 암호화하고 복호화 한 결과 출력

```python
publicKey, privateKey = get_key()
plain = 6739204
cipher = encryption(plain, publicKey)
print(f"plain = {plain}, cipher = {cipher}, plain = {decryption(cipher,privateKey)}")
```

키 생성 후, 6739204를 암호화와 복호화를 한 결과를 출력한다.

RSA 암호 구현 (python)

실행 결과

```
pythonFiles\lib\python\debugpy\launcher' '53499' '--' 'c:\programing\RSA\RSA.py'
plain = 6739204, encrytion = 155877658, decryption = 6739204
PS C:\programing>
```

암호화 하기 전 : 6739204

암호화 후 : 155877658

복호화 한 결과 : 6739204

암호화 하기 전(6739204) = 복호화 한 결과(6739204)

RSA 암호의 미래

RSA 암호의 원리는 큰 수의 어려운 소인수 분해

RSA 암호의 미래

RSA 암호의 원리는 큰 수의 어려운 소인수 분해

큰 수의 소인수 분해가 쉬워지면 RSA 암호는 무용지물이 된다.

RSA 암호의 미래

쇼어 알고리즘

양자 컴퓨터를 이용하여 소인수 분해를 하는 알고리즘

현재의 컴퓨터는 GNFS(Number Field Sieve)라는
소인수분해 알고리즘을 사용한다

RSA 암호의 미래

쇼어 알고리즘

소인수 분해를 할 숫자가 N일 때 소인수 분해의 시간 복잡도

GNFS 알고리즘의 시간 복잡도 : $O\left(e^{\left((\log N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}}\right)}\right)$

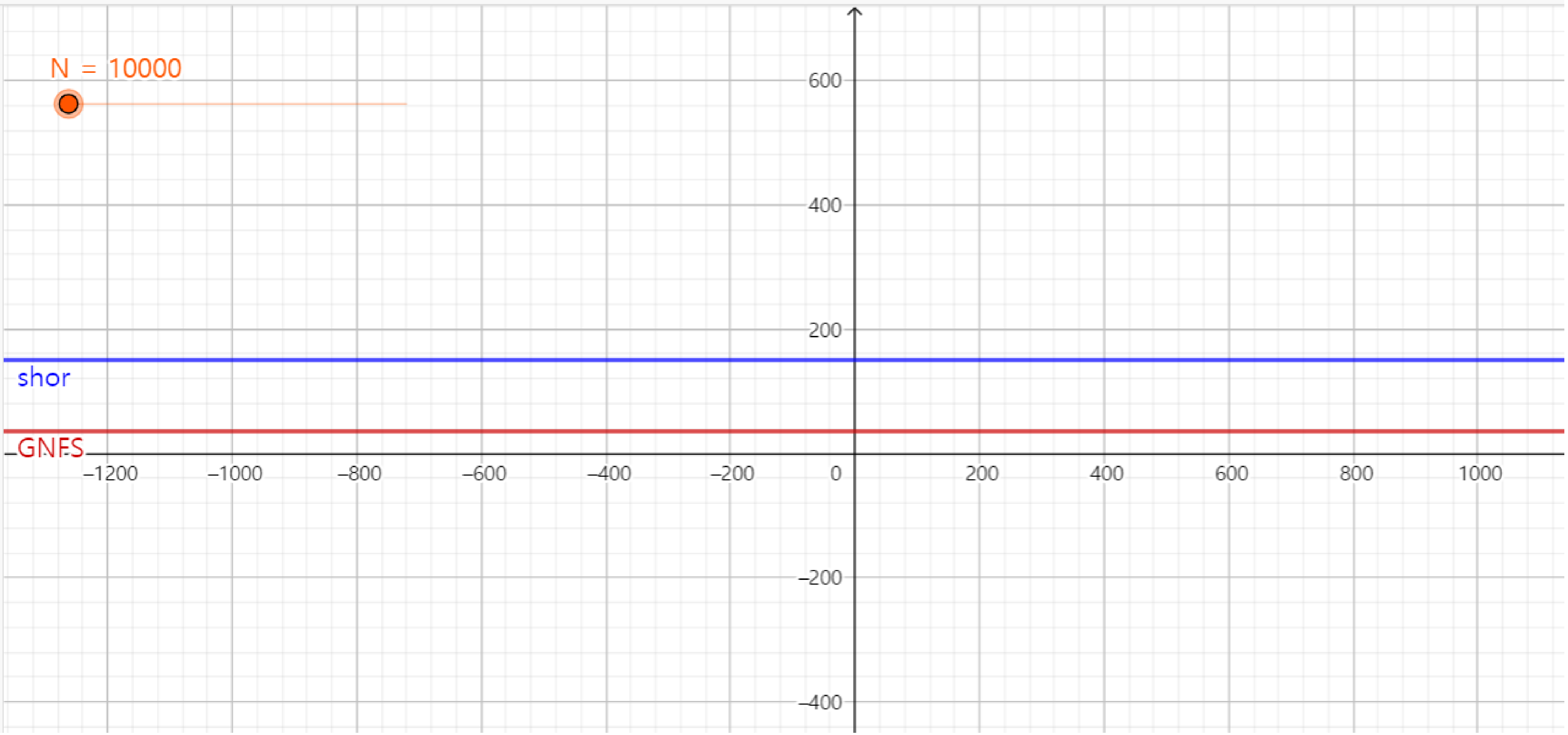
쇼어 알고리즘으로 인한 시간 복잡도 : $O((\log N)^2(\log\log N)\ (\log\log\log N))$

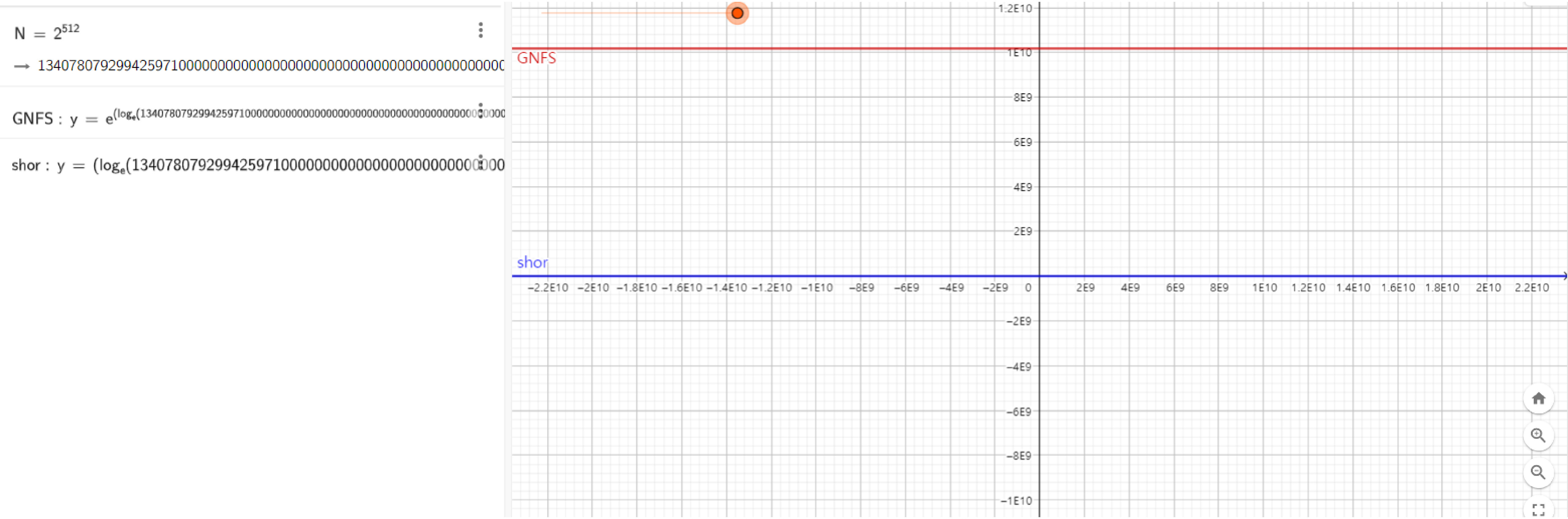
$N = 10000$

$\text{GNFS}: y = e^{(\log_e(10000))^{\frac{1}{3}}(\log_e(\log_e(10000)))^{\frac{2}{3}}}$

$\text{shor}: y = (\log_e(10000))^2 \ \log_e(\log_e(10000)) \ \log_e(\log_e(\log_e(10000)))$

입력...


N = 10000
shor
GNFS
600
400
200
0
−200
−400
−1200
−1000
−800
−600
−400
−200
200
400
600
800
1000


N = 2^512
→ 134078079299425971000000000000000000000000000000000000000
GNFS : y = e^(log_e(134078079299425971000000000000000000000000000000000000000
shor : y = (log_e(134078079299425971000000000000000000000000000000
GNFS
shor
1.2E10
1E10
8E9
6E9
4E9
2E9
0
−2E9
−4E9
−6E9
−8E9
−1E10
−2.2E10 −2E10 −1.8E10 −1.6E10 −1.4E10 −1.2E10 −1E10 −8E9 −6E9 −4E9 −2E9 0 2E9 4E9 6E9 8E9 1E10 1.2E10 1.4E10 1.6E10 1.8E10 2E10 2.2E10

RSA 암호의 미래

쇼어 알고리즘

소인수 분해를 할 숫자가 N일 때 소인수 분해의 시간복잡도

GNFS 알고리즘의 시간 복잡도 : O$\left(e^{\left((\log N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}}\right)}\right)$

쇼어 알고리즘으로 인한 시간 복잡도 : O$((\log N)^2(\log\log N)\ (\log\log\log N))$

N의 크기가 커질수록 쇼어 알고리즘의 연산속도가 매우 빠르다

RSA 암호의 미래

따라서 쇼어 알고리즘을 구현할 수 있는 성능의 양자컴퓨터가 개발 된다면 RSA 암호는 안전하지 못하다.

Q n A