# RSA

유튜브 주소: https://www.youtube.com/watch?v=Jc_PKaBkFH8

RSA란?

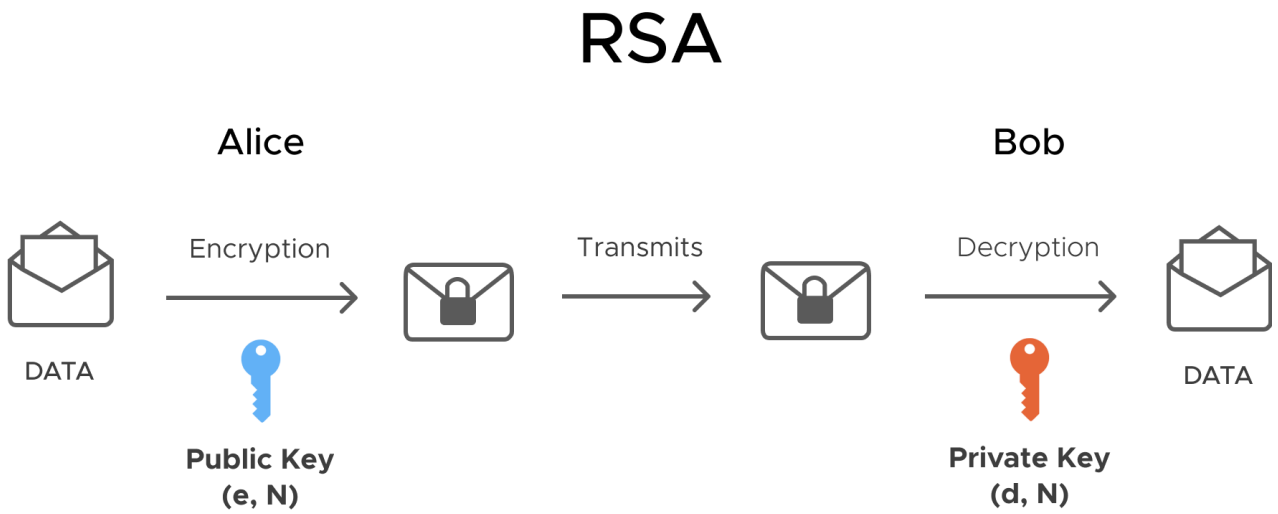
RSA 암호화,복호화 및 키 생성

RSA 구현

RSA 공격

# RSA란?

- 널리 사용되고 있는 공개키(비대칭 키) 암호 방식
- 인수분해 문제에 기반
  (두개의 큰 소수를 곱하는 것은 쉽지만 곱해서 나온 수를 인수분해 하는 것은 어려움)
- 대칭키 암호(AES, DES) 보다는 느림.
- 키 전송과 전자서명에 활용



# 오일러 파이 함수

- 오일러 파이 함수

  $\Phi(m) = Z_m$에서 $m$ 과 서로수인 정수의 수

  ex) $m = 6, Z_6 = \{0,1,2,3,4,5\}$ -> 6과 서로소인 정수 ={1,5}

  $\Phi(6) = 2$

$$\Phi(N) = \Pi_{i=1}^{n}\left(p_i^{e_i} - p_i^{e_i-1}\right)$$

Ex) $240 = 2^4 \times 3 \times 5$

$= p_1^{e_1} \times p_2^{e_2} \times p_3^{e_3}$

$\Phi(240)$

$= (2^4 - 2^3) \times (3^1 - 3^0) \times (5^1 - 5^0)$

- 오일러의 정리

  a와 m이 정수이고 gcd(a,m) =1 일 때,

  $a^{\Phi(m)} \equiv 1 \ (mod\ m)$

  ex) $m = 12, \quad a = 5,$

  $\Phi(12) = \Phi(2^2 * 3) = (2^2 - 2^1) \times (3^1 - 3^0) = (4-2)(3-1) = 4$

  $5^{\Phi(12)} = 5^4 = 25^2 = 625 \equiv 1\ mod\ 12$

# RSA 암호화,복호화 및 키 생성

- RSA 암호화 및 복호화

RSA 암호화(Encryption)

공개키(Public key) (n,e) =$k_{pub}$

암호화 $y = e_{k_{pub}}(x) \equiv x^e mod\, n \quad x, y \in Z_n$

e: 암호화 지수 or 공개지수

RSA 복호화(Decryption)

개인키(Private key) $d = k_{pr}$

복호화 $x = d_{k_{pr}}(y) \equiv y^d mod\, n \quad x, y \in Z_n$

d: 복호화 지수 or 개인 지수

# RSA 암호화,복호화 및 키 생성

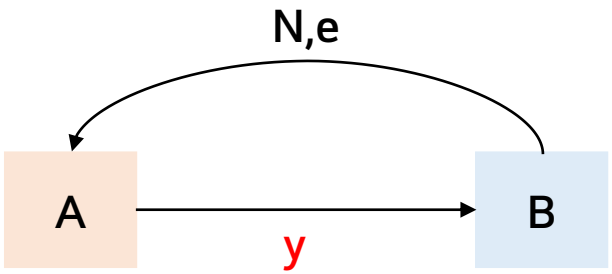
- RSA 키 생성 알고리즘

공개키(Public key) (n,e) =$\boldsymbol{k_{pub}}$ 개인키(Private key) $\boldsymbol{d = k_{pr}}$

1. 두 개의 큰 소수 $p$와 $q$ 선택
2. $n = p * q$
3. $\Phi(n) = (p-1) * (q-1)$
4. $\gcd(e, \Phi(n)) = 1$ $\quad e\epsilon\{1,2, \ldots . \Phi(n) - 1\}$
5. $d * e \equiv 1 \, mod \, \Phi(n)$

# RSA 암호화,복호화 및 키 생성

Ex)




$y = x^{e_B} \bmod N_B$

$y^{d_B} = (x^{e_B})^{d_B} \bmod N_B$
$= x^{e_B * d_B} \bmod N_B$

Bob

1. p =3 q=11
2. N= $p \times q = 33$
3. $\Phi(N) = (3-1) \times (11-1) = 20$
4. e = 3
5. d = $e^{-1} \equiv 7 \bmod 20$

$k_{pub_B} = (N_B, e_B) = (33, 3)$

$k_{pr_B} = d = 7$


전달할 메시지 x = 4

암호화: $y = 4^3 \equiv 31 \bmod 33$

복호화: $\boldsymbol{x = 31^7 mod\ 33 = 4}$

# RSA 구현

```c
#include <stdint.h>
#include <stdio.h>
#include <math.h>

int gcd(int num1,int num2){
    if(num1<num2){
        int temp=num1;
        num1=num2;
        num2=num1;
    }

    while(num2!=0){
        int temp = num1;
        num1= num2;
        num2= temp%num2;
    }
    return num1;
}

int publicKey(int phi_n){  //gcd(e,phi_n)=1, e
    int e=2;
    while(e<phi_n && gcd(e,phi_n)!=1)
        e+=1;

    return e;
}

int privateKey(int e,int phi_n){    //d*e ≡ 1 mod phi_n
    int d=1;
    while((e*d)%phi_n!=1 || d==e){
        d+=1;
    }
    return d;
}
```

실질적인 RSA는
패딩을 사용

```c
int encryption(int input,int e,int n){   // x^e mod n
    int temp;
    long long plus;
    temp=input%n;
    plus = (long long)pow(temp,e)%n;

    ret  n plus;
}

  yption(int input,int d,int n){    // y ^d mod n
    temp;

    n;
     d)%n;

11;
    p
    m=4;
  p*q;
  i_n =(p-1)*(q-1);
  nt e = publicKey(phi_n);
    int d = privateKey(e,phi_n);

    long long encrypt = encryption(m,e,n);
    long long decrypt = decryption(encrypt,d,n);

    printf("encryption result %lld\n",encrypt);
    printf("decryption result %lld",decrypt);
    return 0;

}
```

```
encryption result 31
decryption result 4
```

# RSA 공격

- 부채널 공격

  암호 연산 수행 시 장비에서 발생되는 전력 등의 정보를 이용하여 비밀키 획득

  ex) '0'과 '1'을 처리하는데 소비되는 전력이 서로 다르다는 점을 이용

- 낮은 지수 공격

  e 값이 매우 작고 n 값이 큰 경우 mod 연산을 안 거칠 수도 있음.

  보통 e값은 3인데 암호문 세 제곱근 구하면 그것이 평문 일 수도 있음.

  ex) 평문= 2, e=3 $8 \equiv 2^3 mod\ 100$ 모듈러 연산 X , 8의 세제곱근 =2

# RSA 공격

- N을 소인수 분해

  공격자는 d를 아는 것이 목적

  $d * e \equiv 1 \, mod \, \Phi(n)$

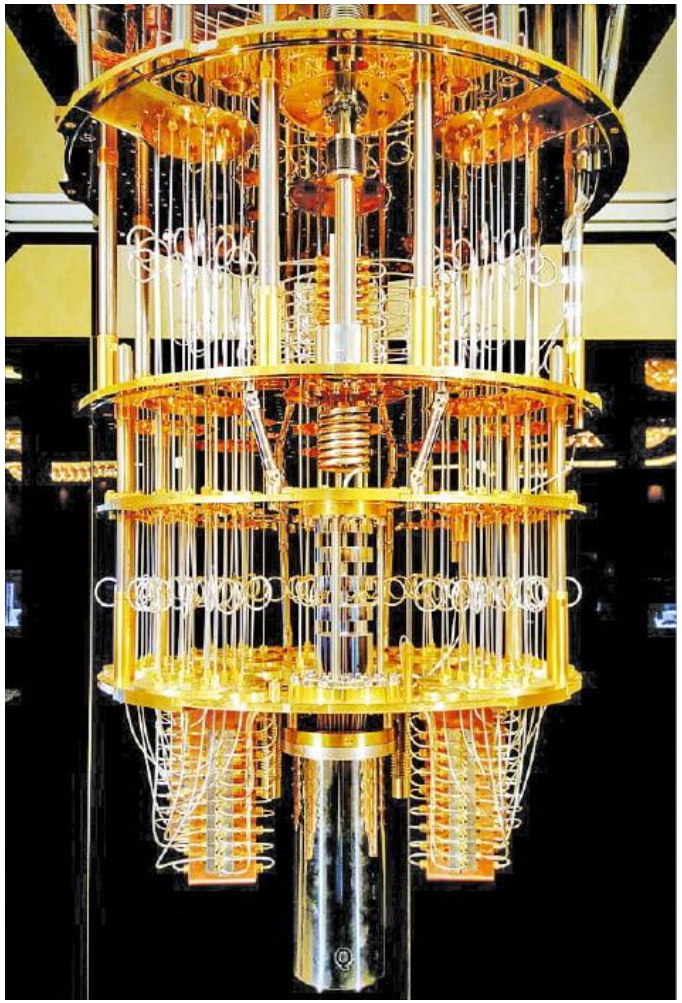
  But, $\Phi(n)$ 모름. $\Phi(n) = (p-1) * (q-1)$

  n을 인수분해 할 수 있으면 공격 가능

n은 매우 큰 정수이므로 인수분해 하기 매우 어려움(1024bits, 2048bits ….)

# 마무리

- RSA는 큰 정수를 인수분해 하는 것이 어렵다는 것을 가정
- 소인수 분해 공격 ➔ 양자 컴퓨터로 쇼어 알고리즘을 통해 공격 가능



- 쇼어 알고리즘

  Shor가 다항시간 내에 소인수 분해를 빠르게 처리 할 수 있는 양자 알고리즘을 제안

- 양자 컴퓨터가 개발된 미래에는 RSA를 사용할 수 없음
- 양자내성암호(PQC) 로 대체 해야함

* 양자내성암호(Post-Quantum Cryptography)
양자컴퓨팅 환경에서 해독 위협에 대응하는 새로운 공개키암호

# Q & A