정보보호

공개키 암호

정보보호 연구실

이동섭

dajababa09@gmail.com

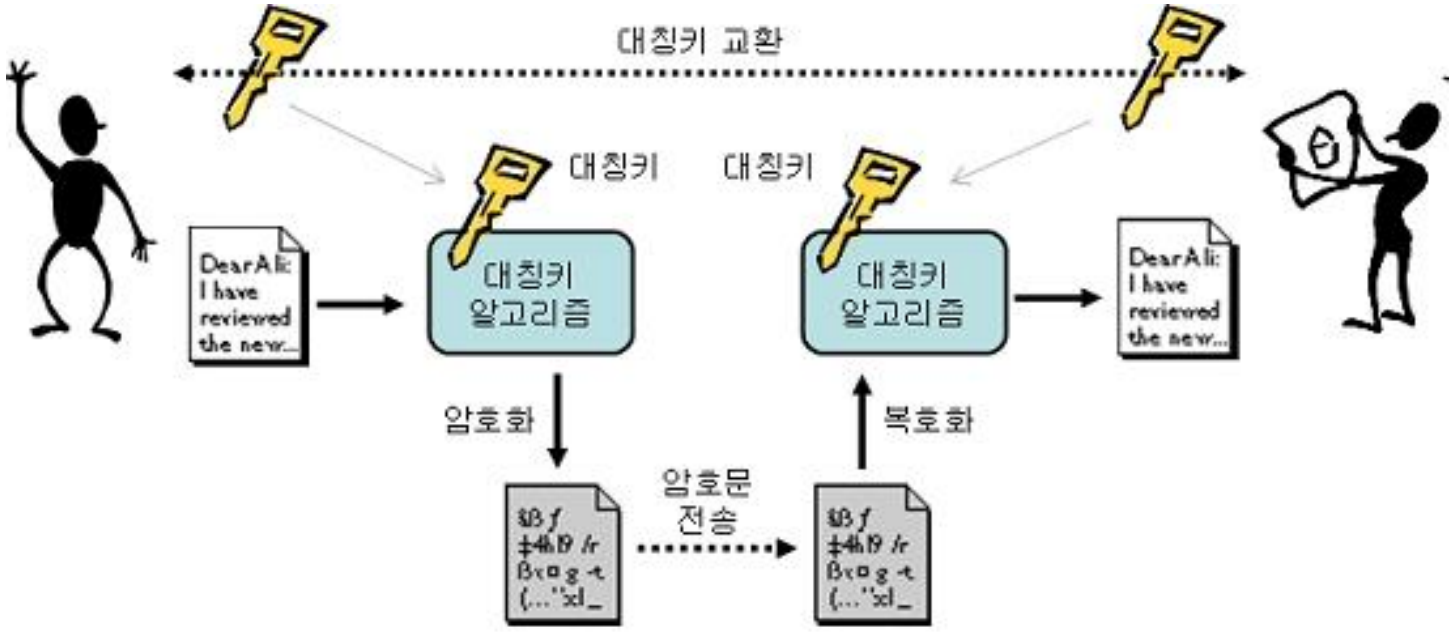
ISLAB
INFORMATION SECURITY LABORATORY CNU

블록암호

■ 블록 암호화 방식

▶ 암호화에 사용되는 키와 복호화에 사용되는 키가 같음

▶ xor, 단순 치환 등 bit단위의 연산을 수행하기 때문에 속도가 빠름

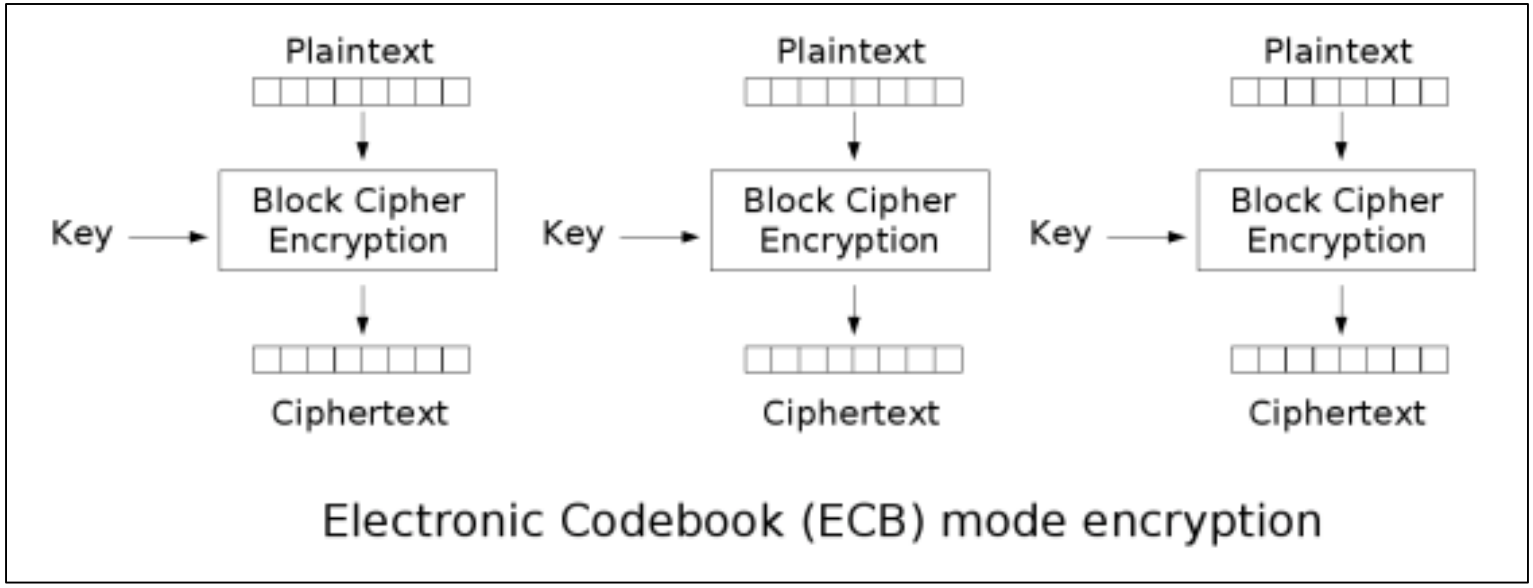


블록암호

■ 블록 암호화 방식

▶ data를 일정 블록 단위로 나눈 후 암호화 진행

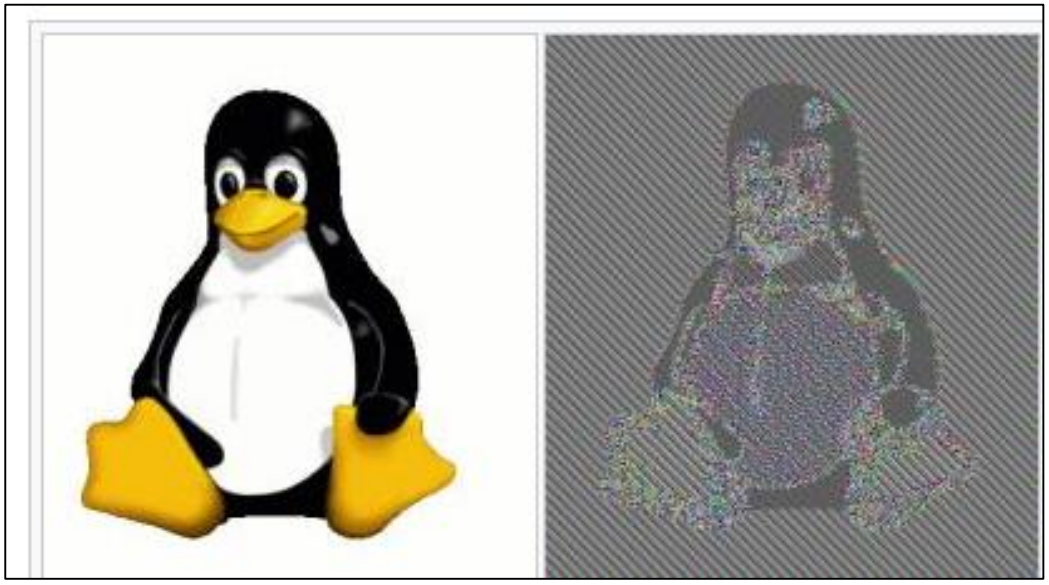
▶ 하지만 같은 평문을 암호화 한다면 같은 암호문이 생성됨



블록암호

■ 블록 암호화 방식

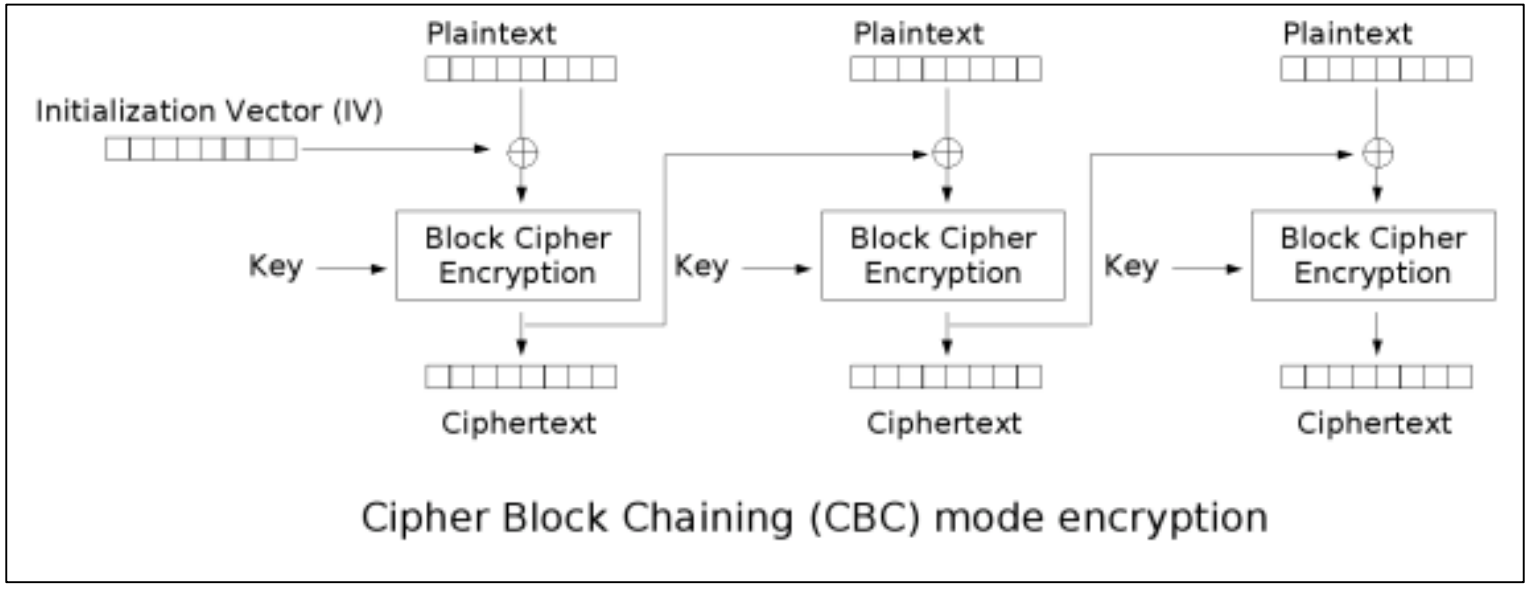
- ▶ data를 일정 블록 단위로 나눈 후 암호화 진행
- ▶ 하지만 같은 평문을 암호화 한다면 같은 암호문이 생성됨



블록암호

■ 블록 암호화 방식

- ▶ 이러한 문제를 해결하기 위해 다양한 모드 사용
- ▶ IV로 인하여 같은 평문을 암호화 한다고 해도 위치에 따라 값이 바뀜
- ▶ 하지만 CBC모드는 병렬처리를 할 수 없음(속도 감소)

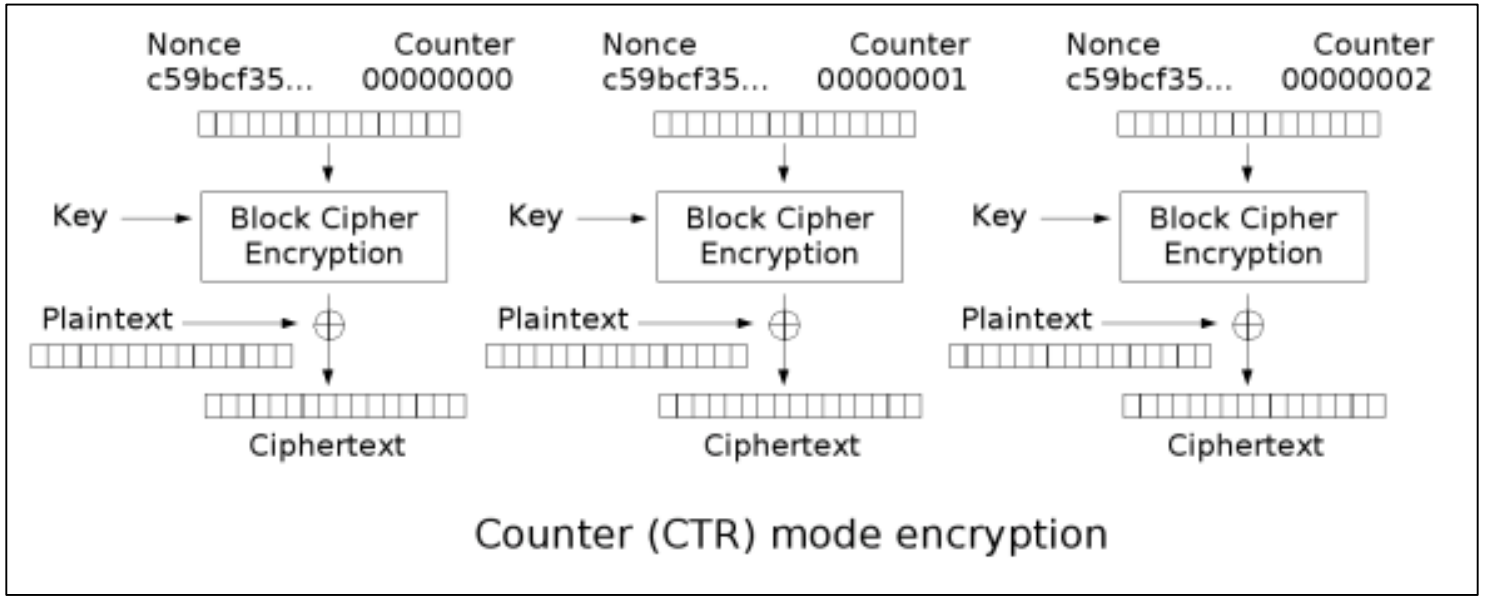


블록암호

■ 블록 암호화 방식

▶ CTR모드는 IV뒤에 count값을 추가함으로써 병렬처리가 가능하게 바뀜(속도 향상)

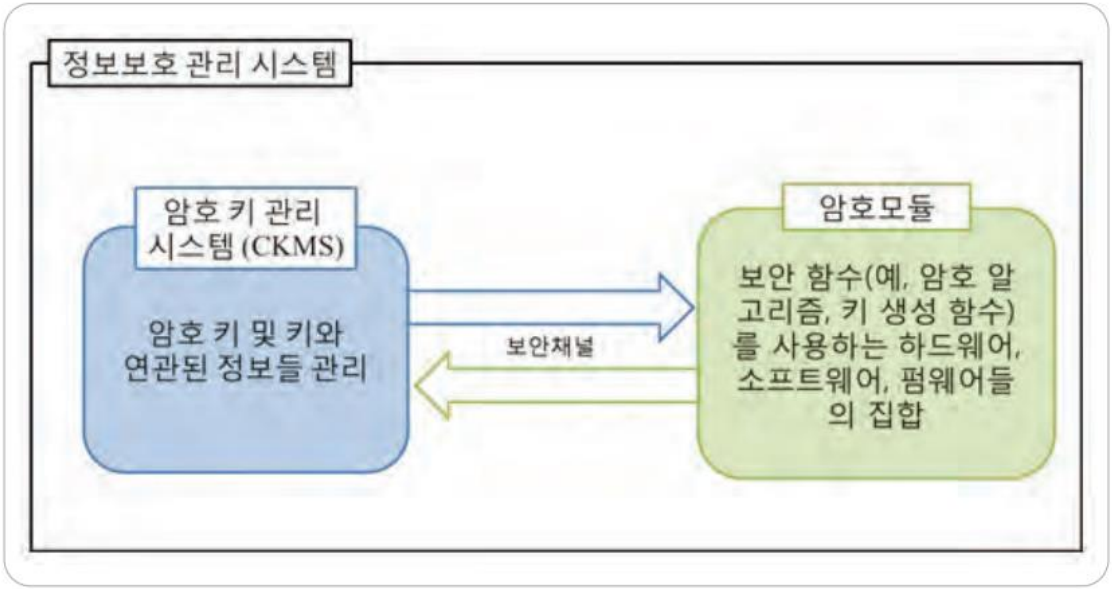
▶ 평문과 IV를 암호화 한 값에 XOR를 진행하는 형식이기 때문에 Stream암호화 비슷



필요성

■ 키 관리의 필요성

- ▶ 정보보호 시스템에서 민감한 정보를 안전하게 보관하고 전송하기 위해 암호화
- ▶ 암호화된 정보의 안전성은 암호 알고리즘, 키의 보안 강도, 키 관련 메커니즘과 관련이 있음
- ▶ 암호 키가 유출된다면 암호화를 통한 정보보호는 의미가 없음

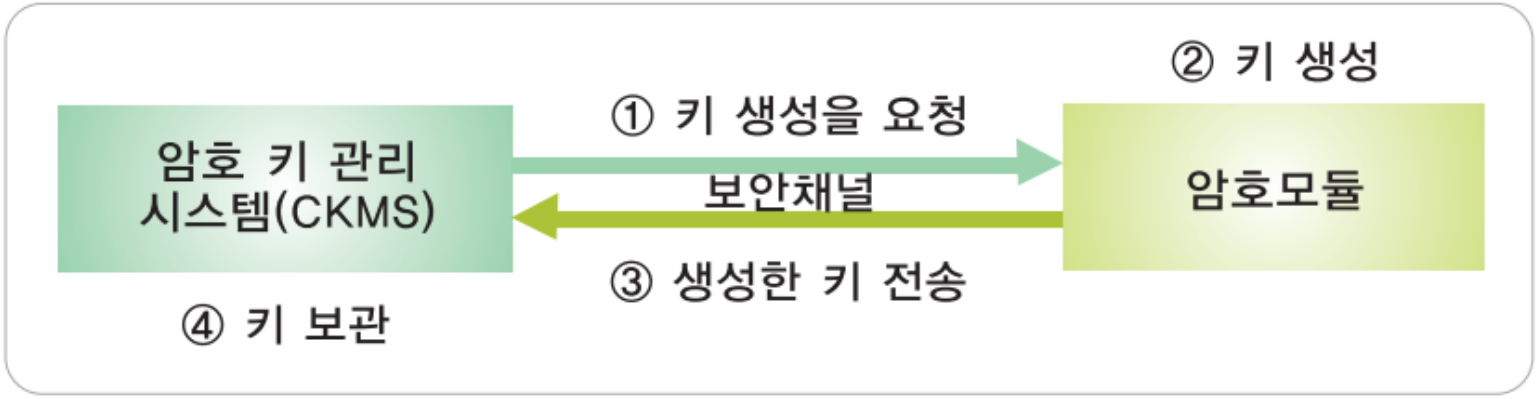


필요성

■ 암호 키 관리 시스템

▶ 키 생성 요청 : 키 생성에 필요한 정보(키 길이, 키 유형 등)을 암호 모듈에 전송

▶ 생성 된 암호 키를 암호 키 관리 시스템에 전송



암호 키 수명 주기

■ 암호 키의 상태

▶ 준비 상태 (Pre-activation state)

– 암호 키의 사용이 허가되지 않은 상태. ‘개인키 소유 증명‘ 또는 ‘키 확인‘ 만을 위해 사용되어야 함

▶ 운영 상태 (Active State)

– 정보를 암호화로 보호하는 데 사용하거나 보호되고 있는 정보를 처리하는 데 사용

▶ 정지 상태 (Deactivated State)

– 암호 키의 유효기간이 만료되었으나 암호화된 정보를 처리하기 위해 암호 키가 필요한 상태

▶ 폐기 상태 (Destroyed State)

– 키 파기 기능을 이용하여 암호 키를 폐기

▶ 위험 상태 (Compromised State)

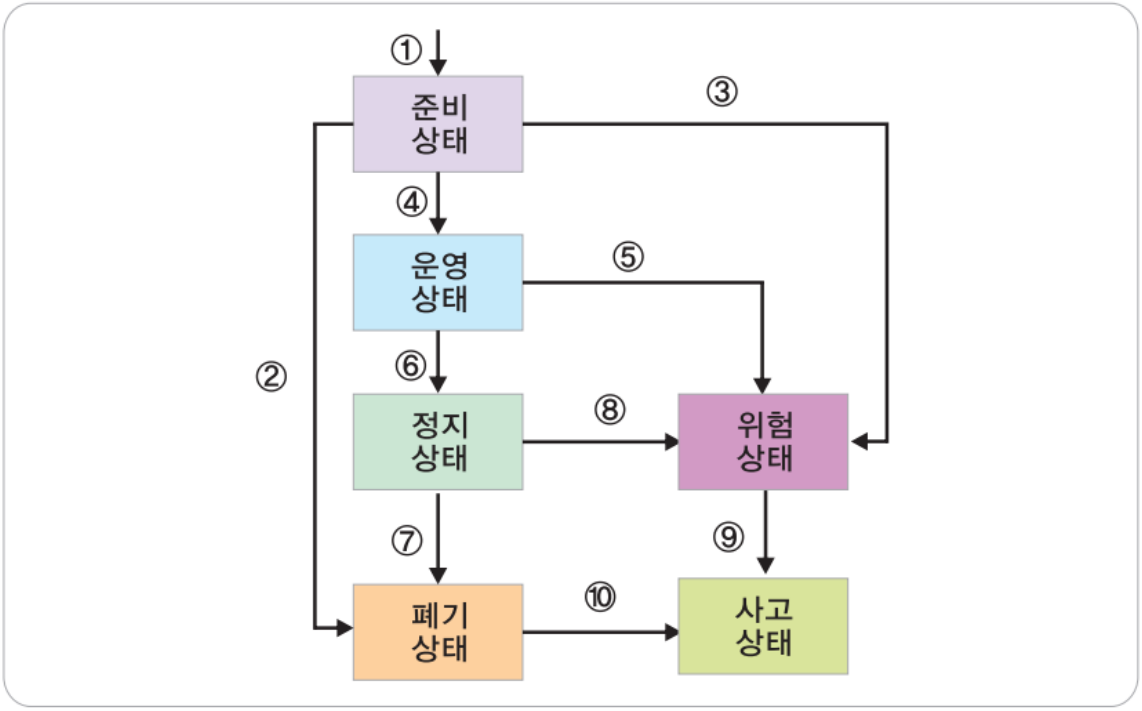
– 허가되지 않은 실체에게 암호 키가 노출 된 상태

▶ 사고 상태 (Destroyed Compromised State)

– 위험 상태의 암호 키가 폐기되었거나, 암호 키가 폐기된 후에 암호 키가 손상되었음이 발견된 상태

암호 키 수명 주기

■ 암호 키의 상태 전환



① 암호 키는 생성됨과 동시에 준비상태

② 준비 상태의 암호 키가 일정 기간 동안 한번도 사용되지 않으면 폐기 상태로 전환

③ 준비 상태의 암호 키가 일정 기간 동안 한번도 사용되지 않았지만 암호 키에 대한 무결성이나 기밀성이 의심되면 위험 상태로 전환

④ 준비 상태의 암호 키가 사용되면 운영 상태로 전환

⑤ 운영 상태의 암호 키에 대한 무결성이나 기밀성이 의심되는 경우에, 운영상태에서 위험 상태로 전환

⑥ 운영 상태의 암호 키가 정보의 암호화에 사용되지 않는 경우에 정지 상태로 전환

⑦ 암호 키가 더 이상 필요하지 않을 때 폐기 상태로 전환

⑧ 정지 상태의 암호 키에 대한 무결성이나 기밀성이 의심 되는 경우에 위험 상태로 전환

⑨ 위험 상태의 암호 키가 더 이상 필요하지 않을 때 사고 상태로 전환

⑩ 폐기 상태의 키가 폐기 전에 손상되었던 것이 확인 되면 사고 상태로 전환

암호 키 유효기간

■ 키 유형에 따라 권장하는 키 유효기간

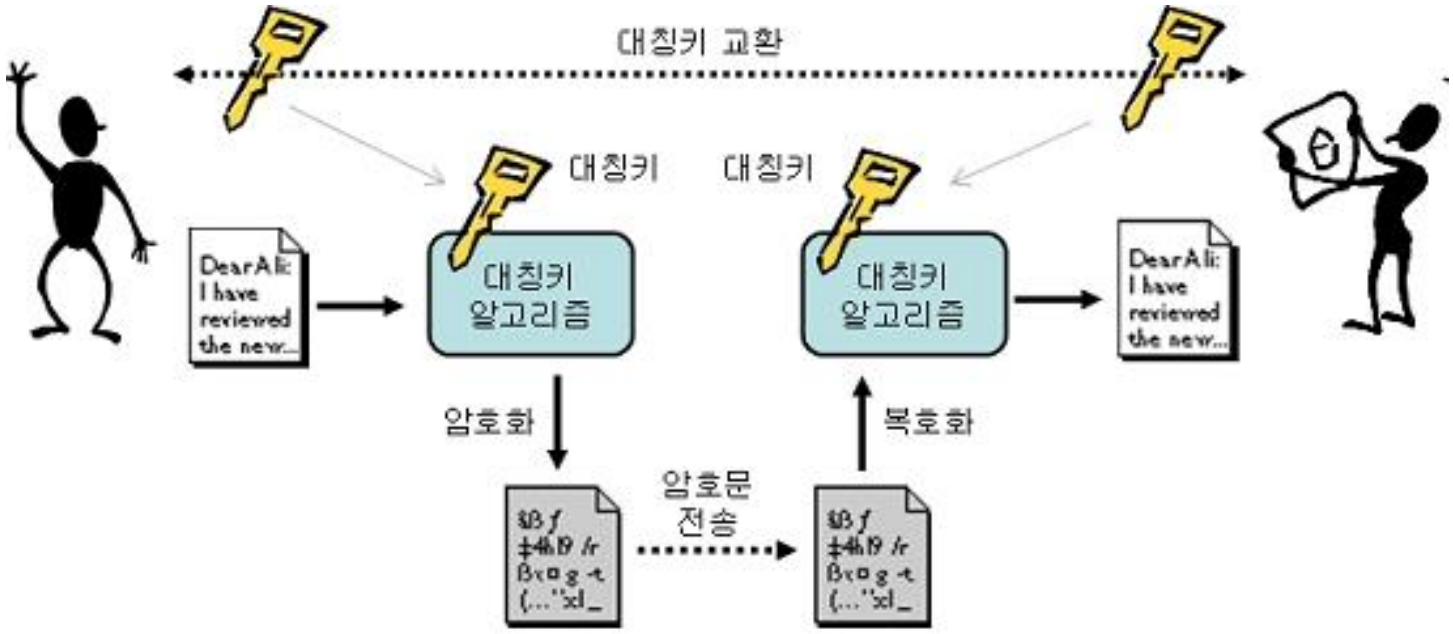
키 유형	키 유효기간	
	발신자 사용기간	수신자 사용기간
1. 개인 서명 키	1~3년	
2. 공개 서명 검증 키	키 크기에 따라 다름	
3. 대칭 인증 키	2년 이하	(발신자 사용기간 + 3년)이하
4. 개인 인증 키	1~2년	
5. 공개 인증 키	1~2년	
6. 대칭 암호 키	2년 이하	(발신자 사용기간 + 3년)이하
7. 대칭키 암호화 키	2년 이하	(발신자 사용기간 + 3년)이하
8. 대칭/공개 RNG 키	리시딩에 따라 다름	
9. 대칭 마스터 키	약 1년	
10. 개인키 전송 키	2년 이하	
11. 공개키 전송 키	1~2년	
12. 대칭키 합의 키	1~2년	
13. 개인 고정 키 합의 키	1~2년	
14. 공개 고정 키 합의 키	1~2년	
15. 개인 임시 키 합의 키	하나의 키 합의 트랜잭션	
16. 공개 임시 키 합의 키	하나의 키 합의 트랜잭션	
17. 대칭 인가 키	2년 이하	
18. 개인 인가 키	2년 이하	
19. 공개 인가 키	2년 이하	

https://www.kisa.or.kr/public/laws/laws3_View.jsp?cPage=1&mode=view&p_No=259&b_No=259&d_No=83&ST=total&SV=

대칭키

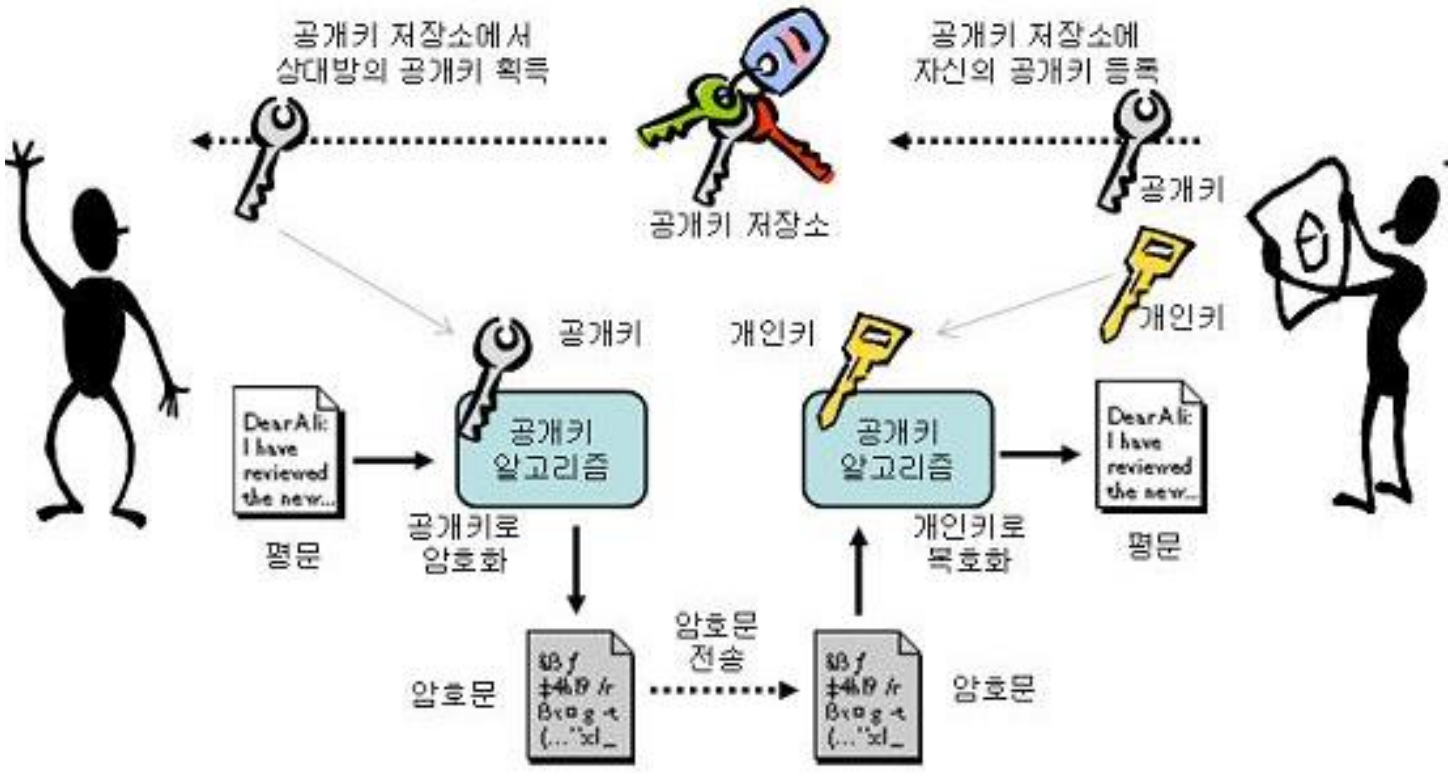
■ 대칭키 암/복호화 방식

▶ 암호화에 사용되는 키와 복호화에 사용되는 키가 같음



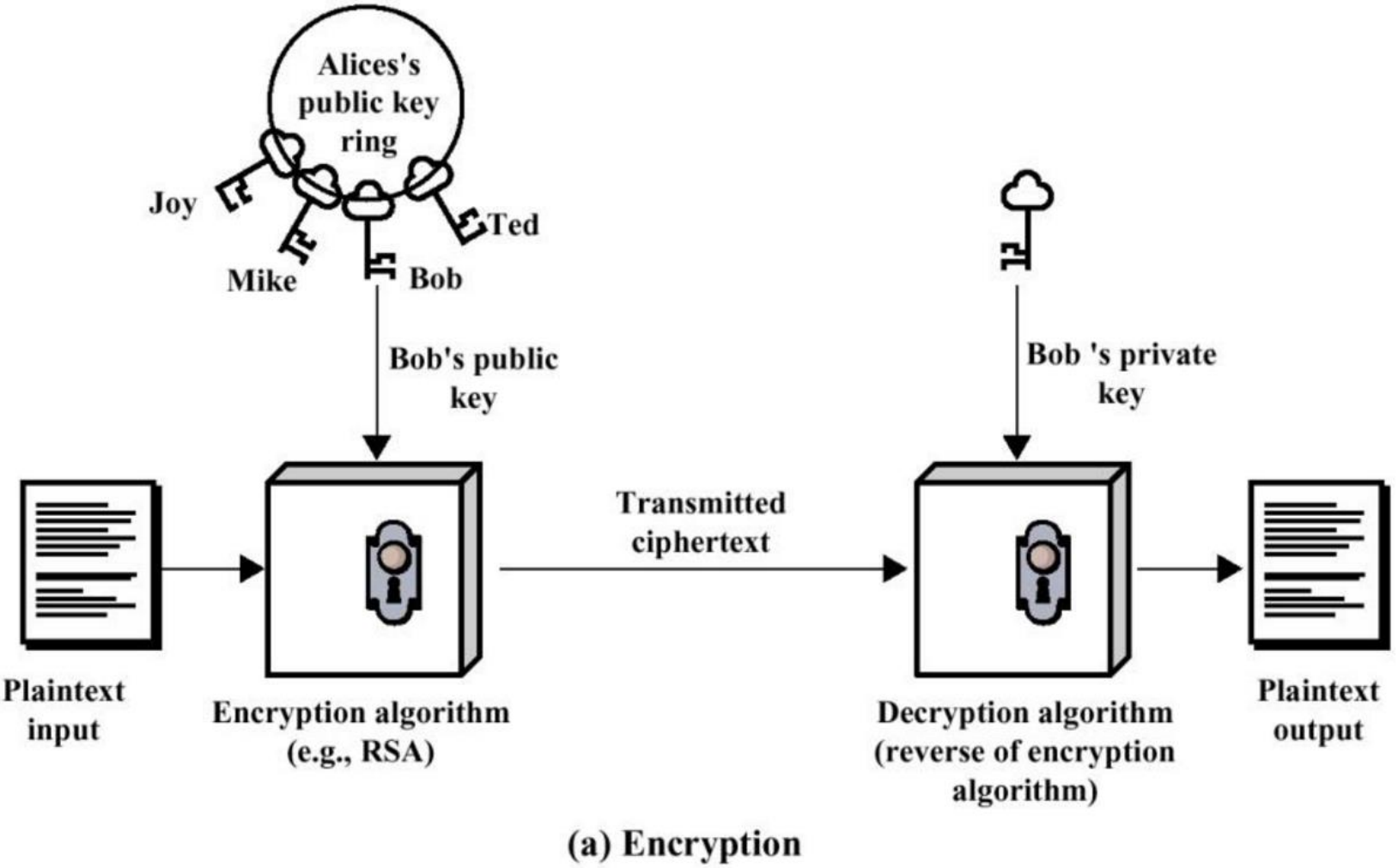
공개키

■ 공개키 암호의 단순 모델



공개키

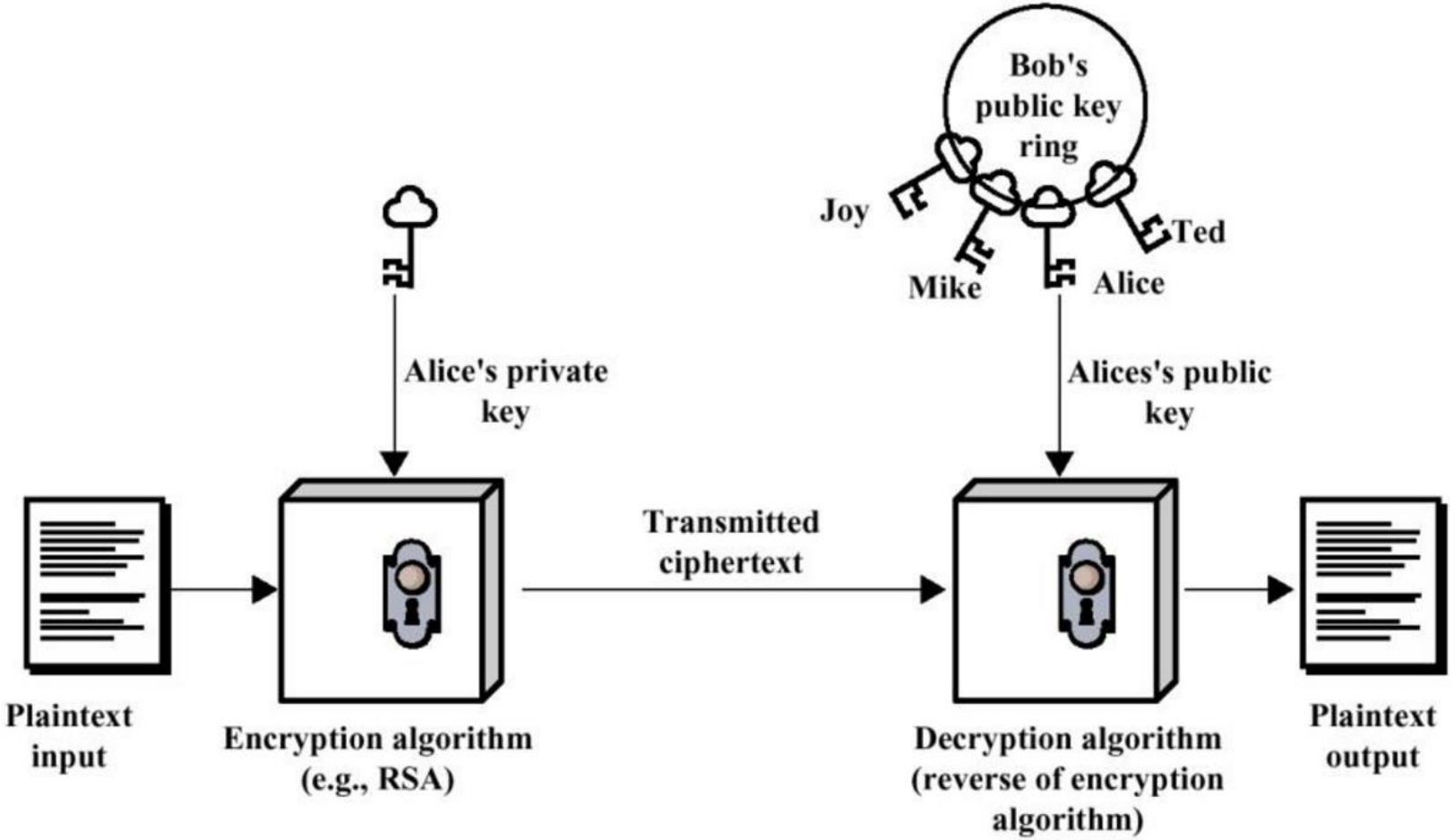
■ 공개키 암호 시스템 – 기밀성



(a) Encryption

공개키

■ 공개키 암호 시스템 – 인증

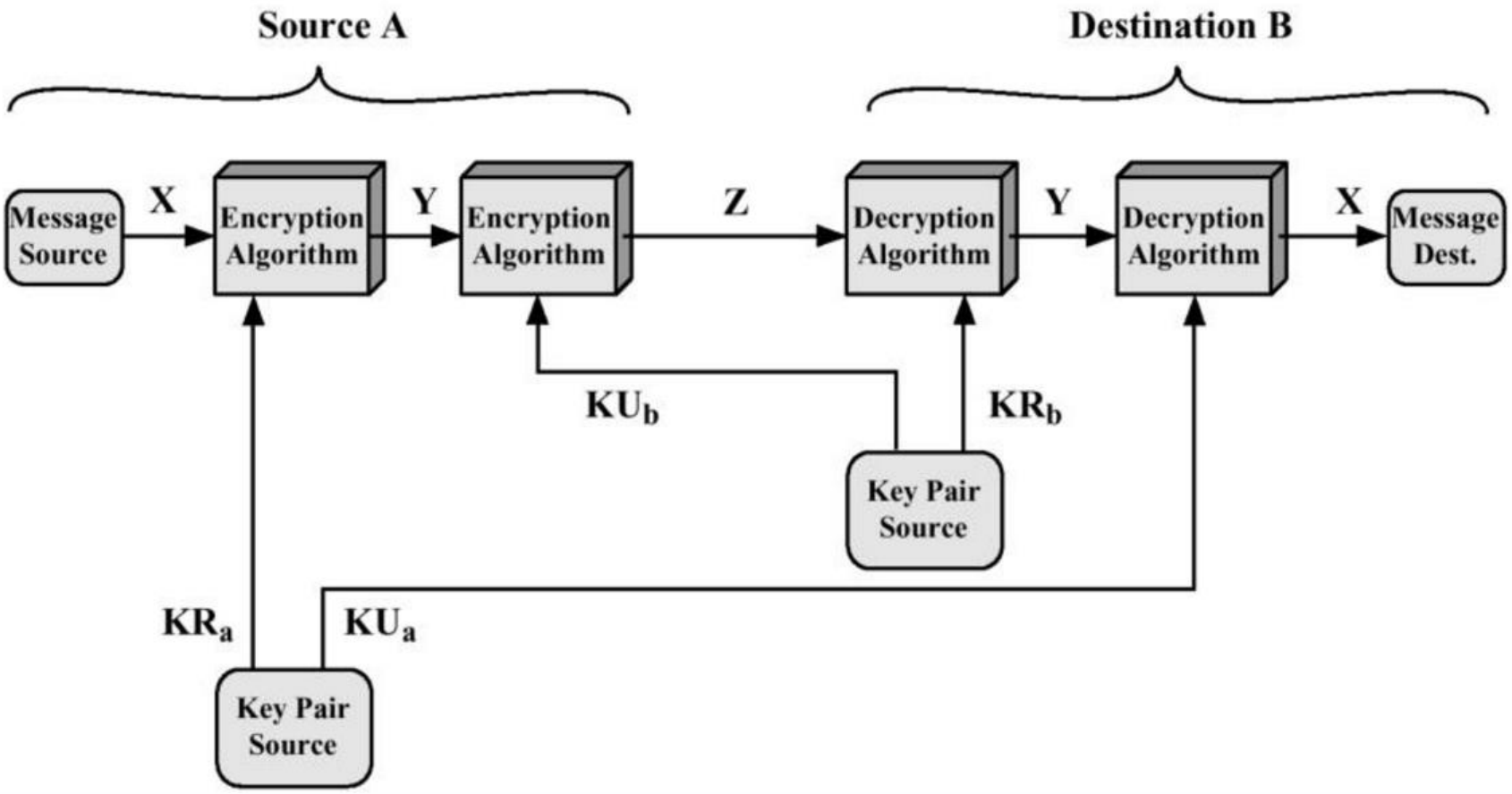


(b) Authentication

공개키

■ 공개키 암호 시스템 – 기밀성 & 인증

▶ 송신자의 개인키로 서명, 수신자의 공개키로 암호화하여 기밀성과 인증 제공



OpenSSL

대칭키 vs 공개키

■ 대칭키와 공개키 암호화 방식의 차이점

비교 항목	대칭키	공개키
키 개수	1개	2개
키 보관 형태	비밀 보관	개인 키 = 비밀 보관 / 공개 키 = 배포
키 교환	비밀 키를 비밀리에...	공개 키만 공개
키 길이	128, 192, 256 등	1024, 2048 비트 등
암호화 속도	빠름	느림
암호화할 평문 길이	제한 없음	제한 있음
기밀성	○	○
인증	△	○
무결성	△	○
부인 방지	X	○

실습

OpenSSL 사용

■ 공개키 방식 사용법

▶ 개인키 생성

```
cillic@cillic-virtual-machine:~/securityClass/8$ openssl genrsa -out prkey.pem 1024
Generating RSA private key, 1024 bit long modulus
.++++++
.....................++++++
e is 65537 (0x10001)
cillic@cillic-virtual-machine:~/securityClass/8$ cat prkey.pem
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQC6azD2XVo/frw5uz5cOQm60Lzl4/5LgT7QeSJhuWA931GTsg2p
60WOhTDT+yFGArZa4FehBgOXjpia+Qv3CqBj9x2np1swc6O07Pvs1B6OQOW56Gr4
yXlgvydB8ukOAtwY++hf7QJK9kfxG+p6rn/p786/1q5rP5SsTjsbf8McswIDAQAB
AoGBALfz2mbhi5KoXtihGlIqvp0oKZ64slCTgwZ44iKIqJ2lZnxV8QHBdevDJoGn
iBfryUI3qoDszFKblZ8nLlVimQ0ao+6cxebT9Q66wqejxS5bMUOQDxr8AZyeOQKk
ue4egFUuBbJdZ0P5Ssx6ZNfrAE6/Ti+ol3Q3odflG+38CKgRAkEA8Toxz933+zWF
eM1pYYQkxesB52dcjC9WSWK+phowiAYEDTw6/lws2T9QCw+SAY3vkovpVnIyJnVs
ZfBkyedXVQJBAMXVvi+CdprCbb8WheRytQo1nn9ogoUsnk4Qk76zkk4aTgl9/m1K
5ikzAqnRBu92xdX0L3hssD0Qx23IdwJoE+cCQQCw8jghuzew/KSIZVzcnDsBNw6x
eH9JkVug/x7L5sActxT3xGuds21uemOziVtXao8Glzy1HZUt4sp9yCWWeOz1AkAY
aXqxdCDckfdm6AmXmgv6+YjYYfYRJuNV8Le6hYAdQSO2CiUWKXLjTJcci12WuQpK
j3o4GvlYX7SsOFtvIZarAkB8N7inslXz6SYE90Ma/XoLb3AZfGvtKyz2cBWmLyXT
L+vKYT3hzAc/8k1OYNELwgWlhCDL1L9ng+qf7te43O9S
-----END RSA PRIVATE KEY-----
cillic@cillic-virtual-machine:~/securityClass/8$
```

실습

OpenSSL 사용

■ 공개키 방식 사용법

▶ 공개키 생성

– 앞서 생성한 개인키를 바탕으로 공개키 생성 가능

```
cillic@cillic-virtual-machine:~/securityClass/8$ openssl rsa -in prkey.pem -out pukey.pem -outform PEM -pubout
writing RSA key
cillic@cillic-virtual-machine:~/securityClass/8$ cat pukey.pem
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC6azD2XVo/frw5uz5cOQm60Lzl
4/5LgT7QeSJhuWA931GTsg2p60WOhTDT+yFGArZa4FehBgOXjpia+Qv3CqBj9x2n
p1swc6O07Pvs1B6OQOW56Gr4yXlgvydB8ukOAtwY++hf7QJK9kfxG+p6rn/p786/
lq5rP5SsTjsbf8McswIDAQAB
-----END PUBLIC KEY-----
cillic@cillic-virtual-machine:~/securityClass/8$
```

실습
Python

■ 공개키쌍 생성

▶ 개인키를 먼저 생성한 후 개인키에서 공개키 추출

```
from Crypto.PublicKey import RSA

prikey = RSA.generate(1024)
pubKey = prikey.publickey()

priFile = open("./mypriKey.pem","wb+")
priFile.write(prikey.exportKey('PEM'))
priFile.close()

pubFile = open("./mypubKey.pem","wb+")
pubFile.write(pubKey.exportKey('PEM'))
pubFile.close()
```

실습
Python

■ PEM

▶ 개인키

```
RSAPrivateKey ::= SEQUENCE {
  version           Version,
  modulus           INTEGER,  -- n
  publicExponent    INTEGER,  -- e
  privateExponent   INTEGER,  -- d
  prime1            INTEGER,  -- p
  prime2            INTEGER,  -- q
  exponent1         INTEGER,  -- d mod (p-1)
  exponent2         INTEGER,  -- d mod (q-1)
  coefficient       INTEGER,  -- (inverse of q) mod p
  otherPrimeInfos   OtherPrimeInfos OPTIONAL
}
```

실습
Python

■ PEM

▶ 공개키

```
RSAPublicKey ::= SEQUENCE {
    modulus           INTEGER,  -- n
    publicExponent    INTEGER   -- e
}
```

실습
Python

■ 키 읽기

▶ 공개키 또한 같은 방법으로 불러오기 가능

```
f = open("mypriKey.pem", "r")
prikey = RSA.importKey(f.read())
f.close()
```

실습
Python

■ 암, 복호화

▶ 공개키.encrypt(), 개인키.decrypt() 로 암 복호화 진행

```
encrypt = pubkey.encrypt(test,32)
decrypt = prikey.decrypt(encrypt)
```

실습
Python

■ 실습 진행

1. 공개키 쌍 생성
   clientPriKey.pem clientPubKey.pem

2. serve는 client의 공개키로 대칭키를 암호화하여 전송 (IV 제외)

3. client는 client의 개인키로 대칭키를 복호화

실습
Python

■ 실습 진행

▶ Mcipher.py

```
#키 읽기
readPEM(filename):

#암호화
RSAEncrypt(pubKey, data):

#복호화
RSADecrypt(pubKey, data):
```

제출 요령

- **보고서 (*.pdf)**
  - 문서는 PDF로 변환하여 제출
  - 과제 해결 과정
    - 과제를 어떻게 이해했는지
    - 어떻게 해결했는지

- **소스코드 (*.py)**
  - 과제 해결에 작성한 코드

❖ 소스코드

1) 실습 부분 코드

- 채팅프로그램에 RSA추가