인턴십 프로젝트

Security Storage 개발

RSA 암호를 이용한 데이터 검증

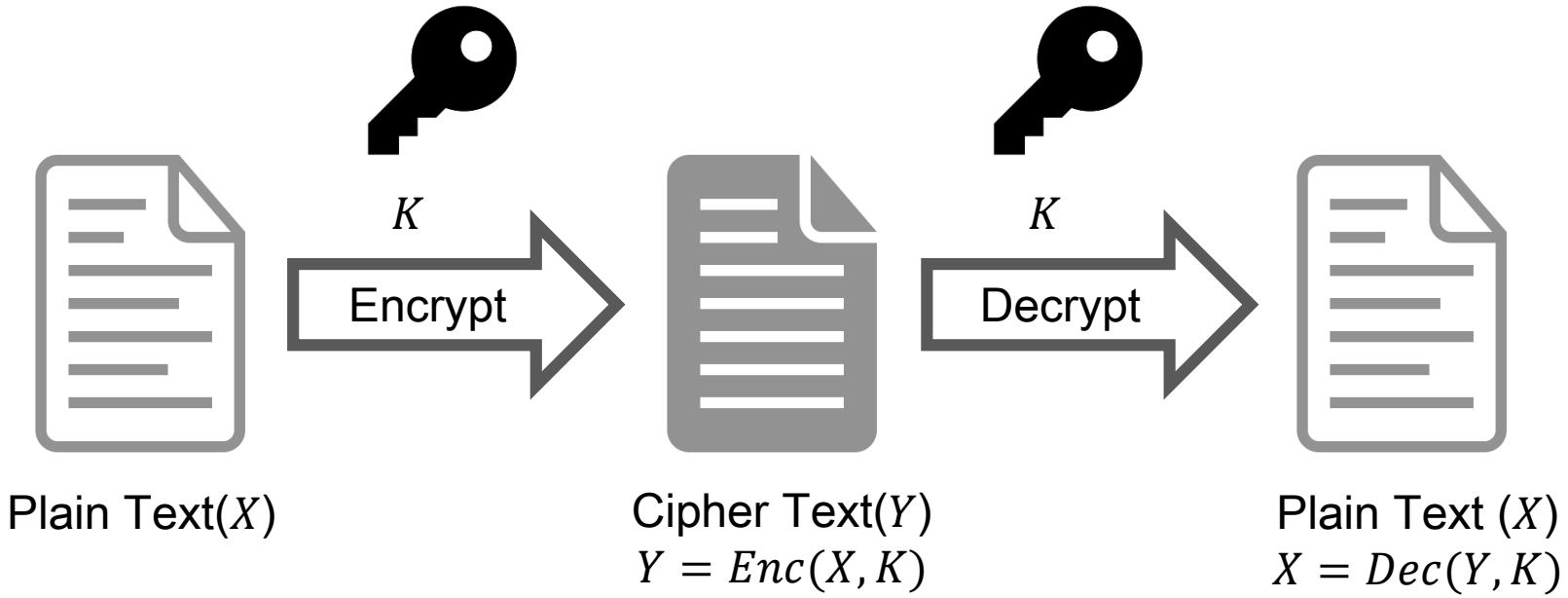
2022.06.29

안현준 인턴사원

1. Symmetric Key Algorithm

암호화($Enc()$)와 복호화($Dec()$)에 동일한 key(K) 사용
공개키 암호와 비교하여 빠른 연산 속도
n명의 사용자가 있다면 $\frac{n(n-1)}{2}$개의 키가 필요

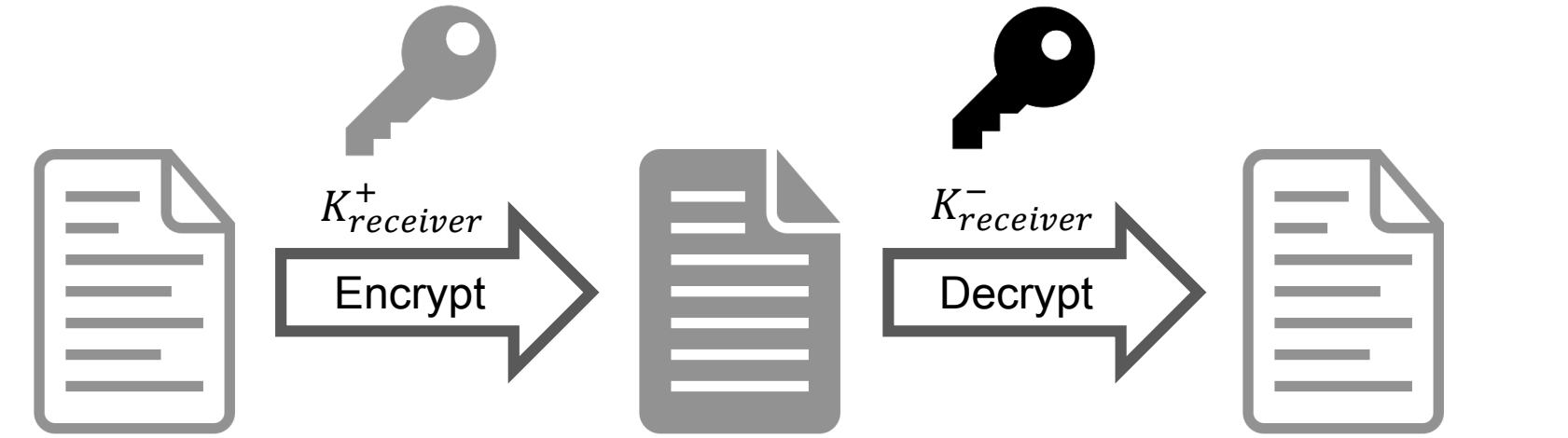
Challenges
서로 통신하기 전에 모두 대칭키를 소지하고 있어야 한다. (키 분배 문제)
key를 분실하거나, 공격자가 key를 탈취한다면? → 통신을 신뢰할 수 없다.



2. Asymmetric Key Algorithm

공개키(Public Key, K^+)와 개인키(Private Key, K^-)로 암호화/복호화
수신자의 공개키로 데이터를 암호화, 수신자의 개인키로 데이터를 복호화
암호문(Y)이 도청되어도, 수신자의 개인키가 없다면 평문(X)을 알 수 없음.
n명의 사용자가 있다면, $2n$개의 키가 필요하므로, 키 관리/분배가 용이

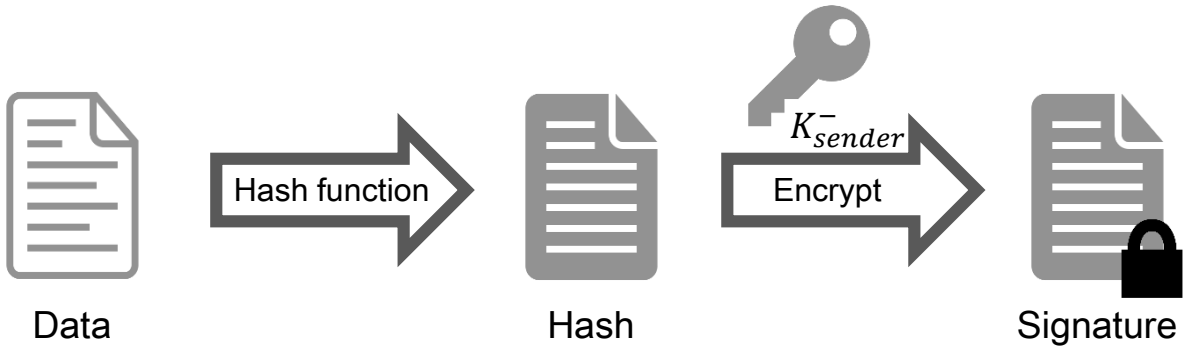
Challenges
공격자가 공개키를 탈취 후 키를 조작하여 유저와 통신을 시도한다면?
→ 상대방의 공개키가 변조되지 않았다는 증명(신뢰)이 필요



3. Digital Signature

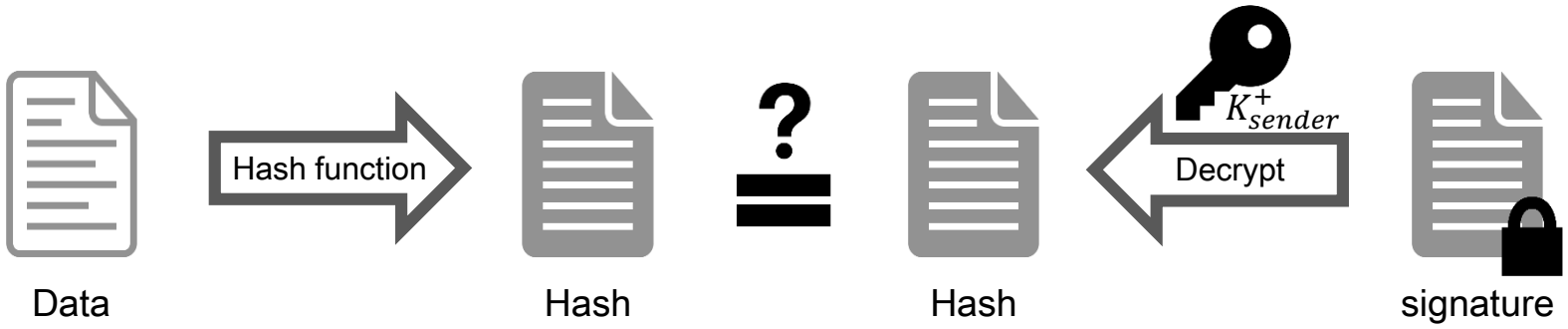
해시함수(H)로 메시지(M)의 해시값($H(M)$)을 구하고, 이 해시값을 개인키로 암호화하여 서명($sign$)을 얻는다. 송신자는 수신자에게 데이터와 서명을 전송한다.

$$sign = Enc(H(M), K^{-}_{sender})$$

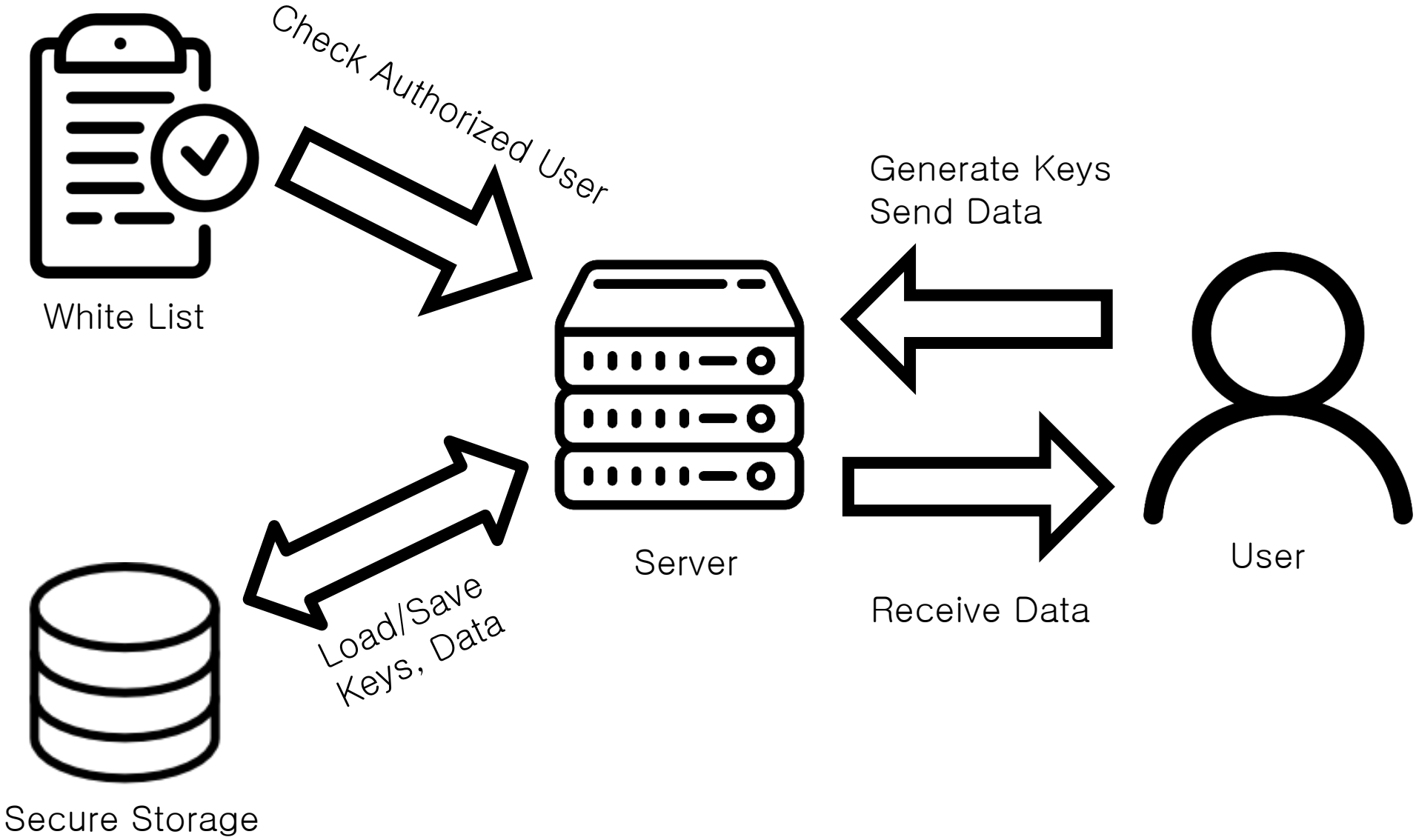


수신자는 공개키로 서명을 복호화하여 해시값을 통해 송신자의 공개키를 검증한다. 복호화된 해시값과 데이터의 해시값이 일치하면 송신자의 공개키를 신뢰할 수 있다.

$$Verify(sign, M, K^{+}_{sender}) = Dec(sign, K^{+}_{sender}) == H(M)$$



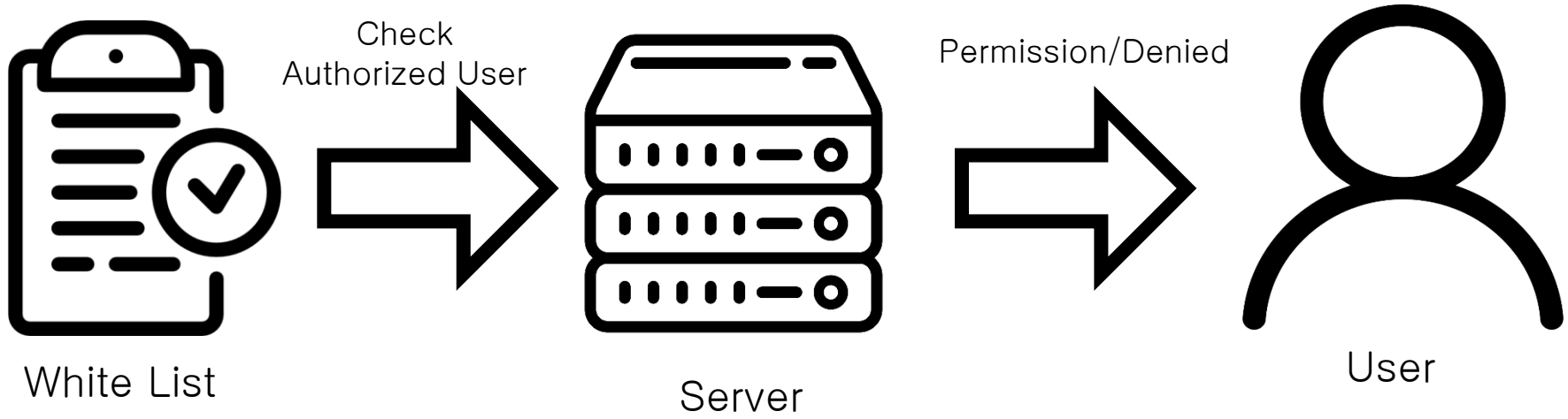
4. Program Scheme



4.1 Check Authorized User

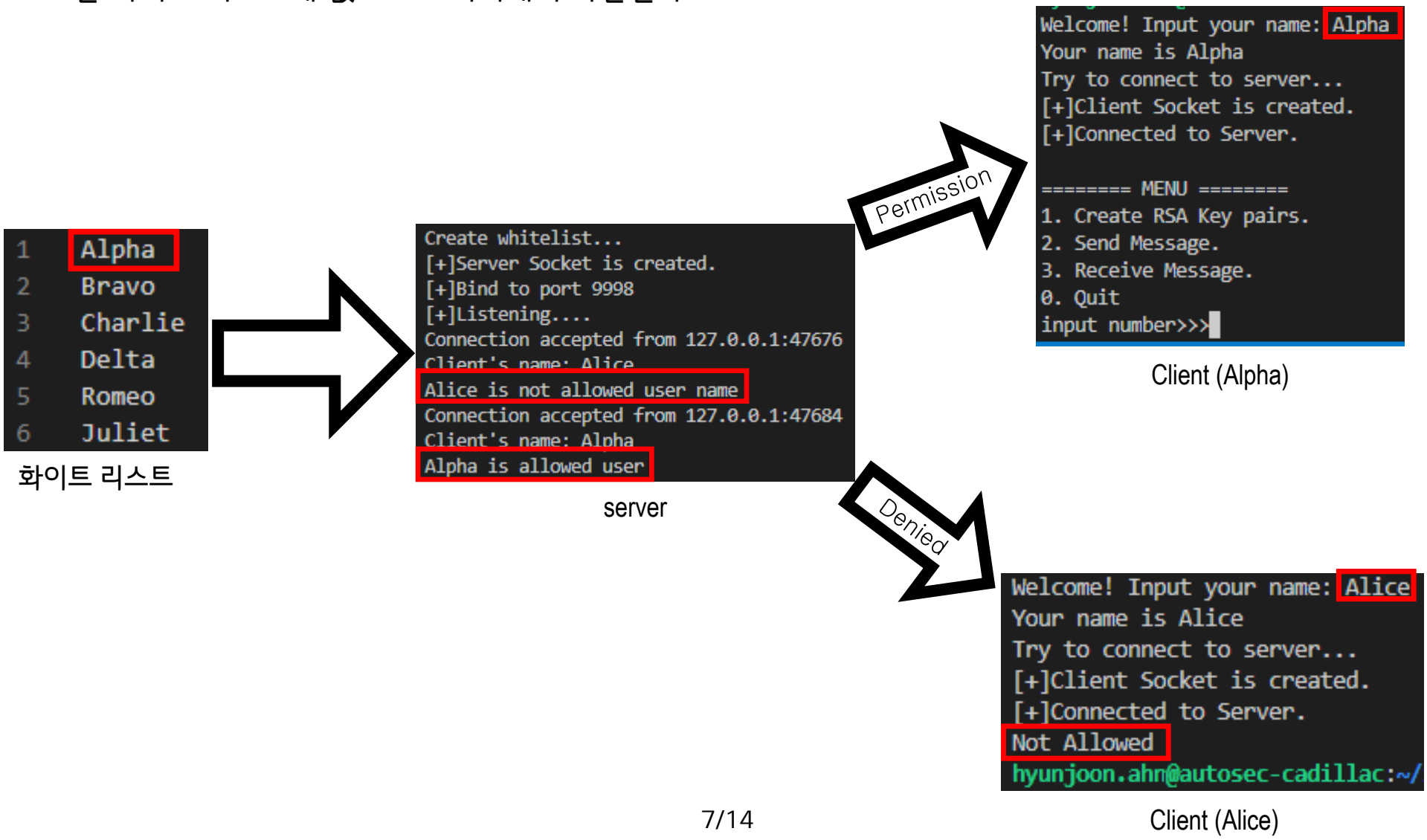
화이트리스트에 존재하는 유저만 서버에 접속하여 서비스를 이용하고, 접근 권한이 없는 유저는 서버와의 연결을 해제하여 차단한다.

화이트리스트: 기본 정책이 모두 차단인 상황에서, 예외적으로 접근이 가능한 대상들의 목록



4.1.1 Check Authorized User

Alpha는 화이트 리스트에 있으므로 서버의 서비스를 이용할 수 있다.
Alice는 화이트 리스트에 없으므로 서버에서 차단된다.

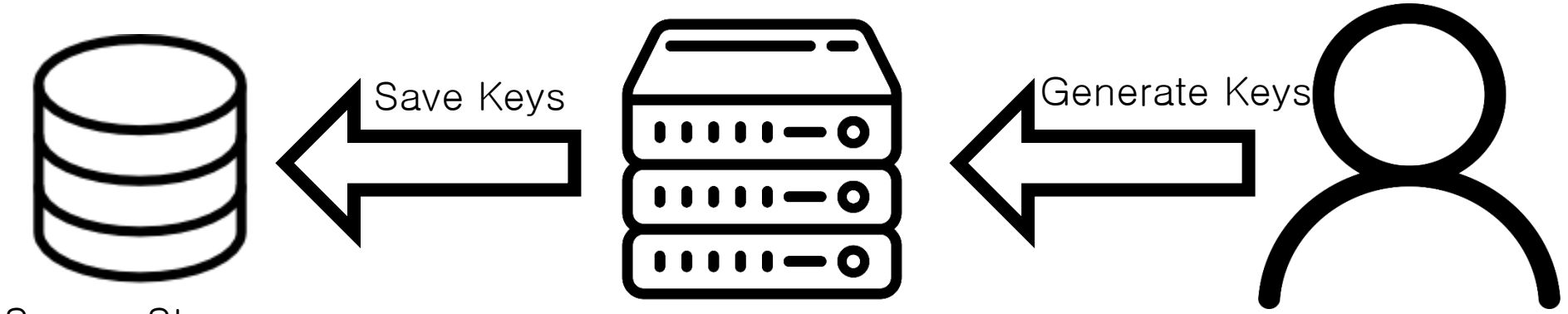


4.2 Generate RSA Keys

공개키를 먼저 생성하고, 공개키에서 개인키를 생성한다.
전자서명이 가능한 공개키 암호시스템으로 RSA 암호를 이용

1. 서로 다른 두 소수 p, q를 선택하여 $N = pq$과 $\varphi(N) = (p-1)(q-1)$을 계산한다.
2. $1 < e < \varphi(N)$ 면서 $\varphi(N)$ 과 서로소인 e를 찾는다. 이때 (N, e)를 공개키(K^+)로 한다.
3. $ed \equiv 1 \pmod{\varphi(N)}$인 d를 구한다. 이때 (N, d)를 비밀키(K^-)로 한다.

RSA 키 생성은 **`RSA_generate_key_ex()`** API를 이용함.



Alpha_prikey.pem
Alpha_pubkey.pem

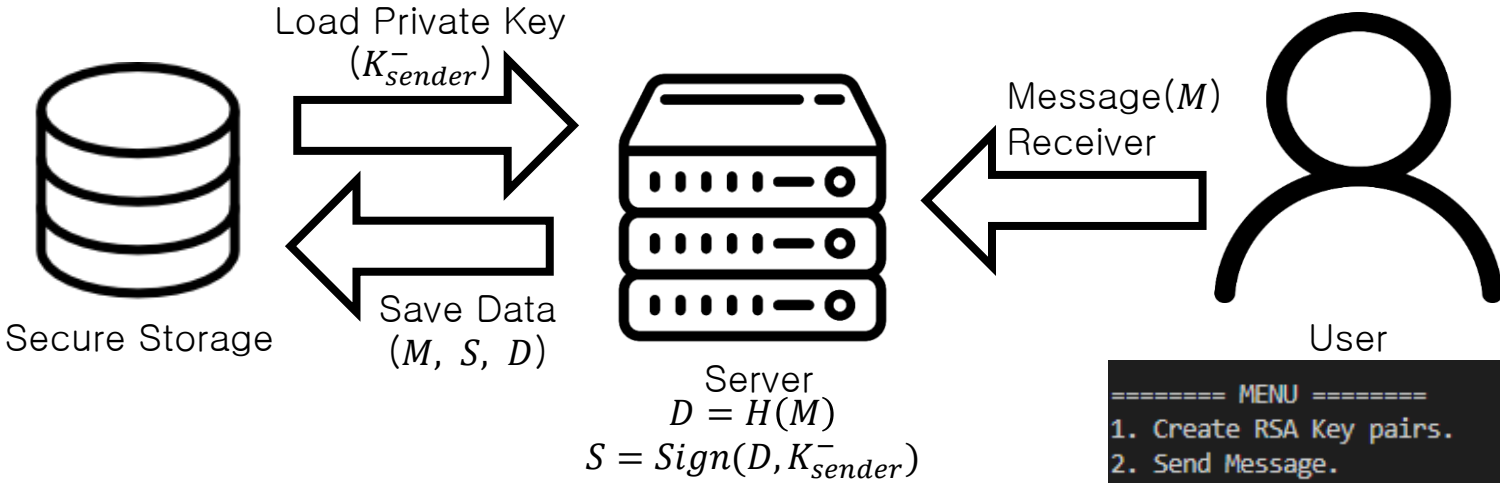
생성된 RSA 키는 PEM 파일로
Secure Storage에 저장된다

```
======== MENU ========
1. Create RSA Key pairs.
2. Send Message.
3. Receive Message.
0. Quit
input number>>>1
Generating RSA (2048 bits) keypair...
done.
```

4.3 Send Message

사용자는 메시지(M)와 수신인을 입력하여 서버로 전송한다.
서버는 암호 해시함수를 통해 메시지의 해시값($D = H(M)$, Digest message)을 얻고, 사용자의 개인키로 서명을 얻는다. 전자서명을 포함한 메타데이터는 Secure Storage에 저장된다.

암호 해시함수로는 SHA-2를 이용한 **`SHA256_*()`**를 이용함




```
sender: Alpha
receiver: Bravo
digest: e4ad0102dc25234
message: hello World!
```

```
sender: Alpha
receiver: Brian
digest: 9e768df9a9e250
message: Life is Good
```

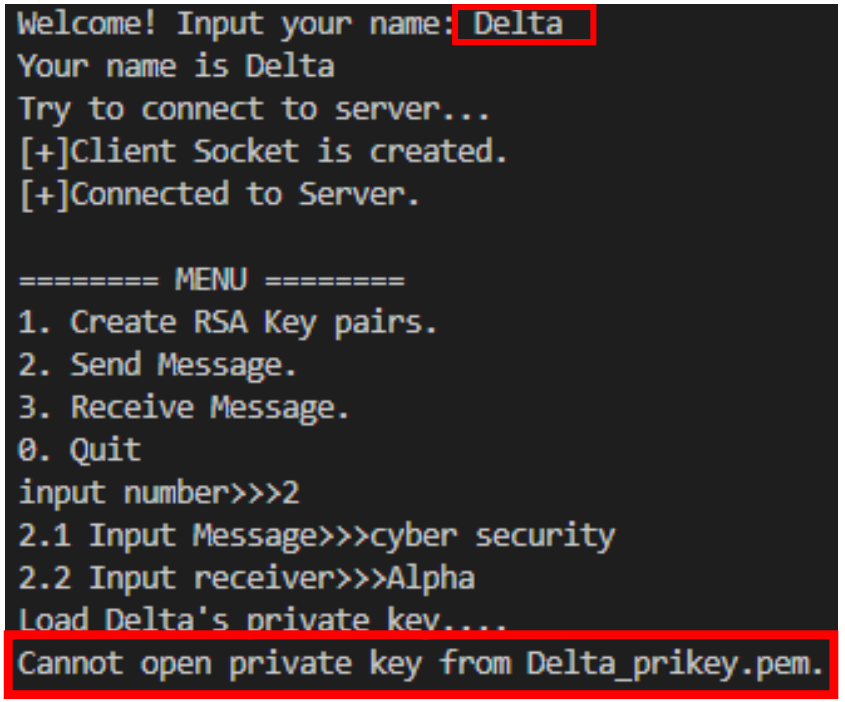
```
sender: Alpha
receiver: Bravo
digest: e9c8c7054d62fae0d
message: LG Electronics
```

```
======== MENU ========
1. Create RSA Key pairs.
2. Send Message.
3. Receive Message.
0. Quit
input number>>>2
2.1 Input Message>>>hello world!
2.2 Input receiver>>>Bravo
Load Alpha's private key....
Successfully load private key.
Digest message process...
```

데이터들은 링크드 리스트로 Secure Storage에 저장된다.

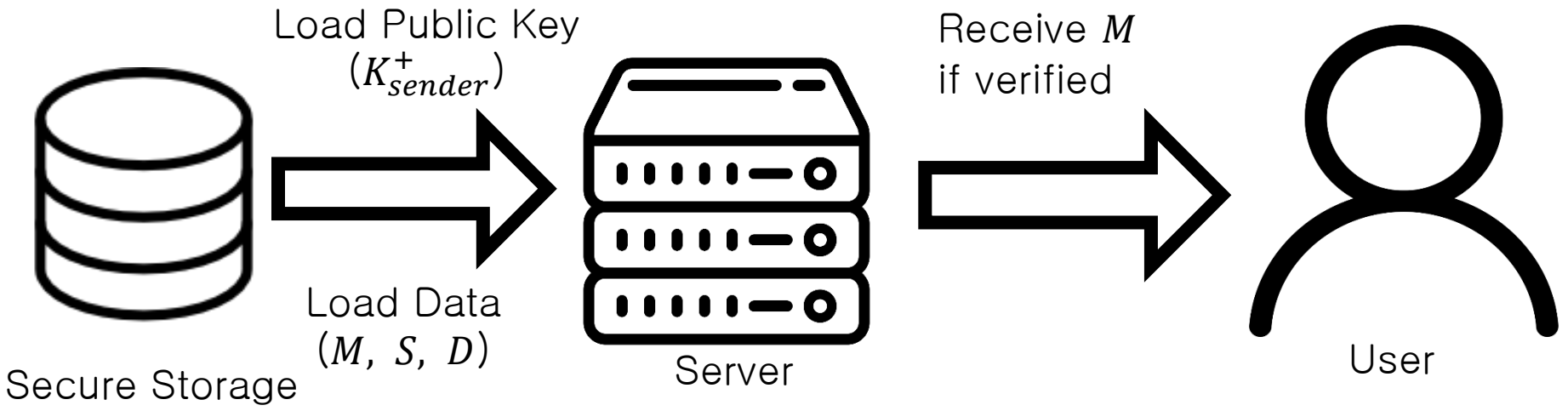
4.3.1 Send Message (without RSA keys)

접근권한이 있는 사용자라도, RSA key를 갖고 있지 않으면 메시지 전송 서비스를 이용할 수 없다.



4.4 Receive Message

수신인은 서버로부터 메시지(M)의 해시값($H(M)$)과 송신인의 서명을 받는다.
송신자의 공개키로 해시값을 비교하여 공개키를 검증하면 수신된 메시지를 출력한다.
출력된 데이터는 Secure Storage의 링크드 리스트에서 삭제된다.



4.4.1 Receive Message (Verified)

공개키가 검증이 되면, 수신 메시지를 client의 화면에 출력한다.

```
sender: Alpha
receiver: Bravo
digest: 7f83b1657ff1fc53b92dc
message: Hello World!

sender: Alpha
receiver: Delta
digest: 9e768df9a9e2505c0d26f
message: Life is Good

sender: Alpha
receiver: Bravo
digest: e9c8c7054d62fae0d8804
message: LG Electronics
```

server

$$Verify(D, S, K^{+}_{sender})$$

```
Welcome! Input your name: Bravo
Your name is Bravo
Try to connect to server...
[+]Client Socket is created.
[+]Connected to Server.

======== MENU ========
1. Create RSA Key pairs.
2. Send Message.
3. Receive Message.
0. Quit
input number>>> 3
You've got 2 message(s).

Message from Alpha
----Hello World!

Message from Alpha
----LG Electronics
```

client

```
======== MENU ========
1. Create RSA Key pairs.
2. Send Message.
3. Receive Message.
0. Quit
input number>>>3
No messages.
```

메시지가 없는 경우

4.4.2 Receive Message (Not verified)

만약 송신자의 공개키가 변조되었을 경우, 공개키 검증에 실패하여 메시지를 수신하지 않는다.

```
-----BEGIN RSA PUBLIC KEY-----
MIIBCgKCAQEAudcg/Xv0jmH+Qbz1kK8HVRfrz5u4TTTLP1cLNvuhu6WxIPZsq3Ng
h6tMfznS3W4mbzShZ4dHBJdJWlIlD5hwLllj89/PTwkfgHnhXuHI9M4wundG95TQ
otz1E+d8QAjMV8M8D9evcnKobj4uBbUp724pod2N3ZzcUlWbBNsGSStkhphw829Z
97a5/mkdjeq0JiIZqO37BxUgPb0fCWNnUGBH5vxjSGXSAaXHZPN1kNrP3Rw3Kh0P
INcmx1H2C1EHBQ8LroBZ4RBTmyAf6KwhQlPHBmw8As8TxEZSUuouE7uYatEo9YKc
vAK+3WXAurdmvtOFH7nqer7wzdZuccSBNQIDAQAB
-----END RSA PUBLIC KEY-----
```

원본 공개키

```
-----BEGIN RSA PUBLIC KEY-----
MIIBCgKCAQEAudcg/Xv0jmH+Qbz1kK8HVRfrz5u4TTTLP1cLNvuhu6WxIPZsq3Ng
h6tMfznS3W4mbzShZ4dHBJdJWlIlD5hwLllj89/PTwkfgHnhXuHI9M4wundG95TQ
otz1E+d8QAjMV8M8D9evcnKobj4uBbUp724pod2N3ZzcUlWbBNsGSStkhphw829Z
97a5/mkdjeq0JiIZqO37BxUgPb0fCWNnUGBH5vxjSGXSAaXHZPN1kNrP3Rw3Kh0P
INcmx1H2C1EHBQ8LroBZ4RBTmyAf6KwhQlPHBmw8As8TxEZSUuouE7uYatEo9YKc
vAK+3WXAurdmvtOFH7nqer7wzdZuccSBNQIDAQAC
-----END RSA PUBLIC KEY-----
```

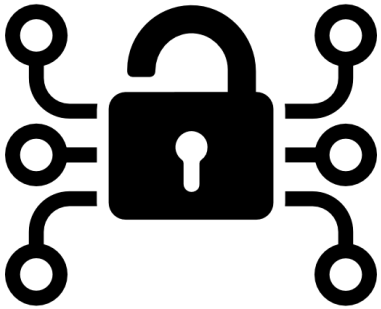
변조된 공개키

```
input number>>> 3
You've got 2 message(s).

Message from Alpha
----not verified

Message from Alpha
----not verified
```

공개키 검증에 실패하여 메시지가 출력되지 않음

5. 배운점

1. 기본적인 암호 이론을 학습하여 전자서명의 필요성을 학습
- 대칭키 암호 알고리즘
- 공개키 암호 알고리즘
- 공개키 서명
- PKI(공개키 기반 구조)



2. OpenSSL의 API를 이용하여 간단한 암호화 통신 시스템을 구현
- RSA key 생성
- 암호화 및 복호화
- 암호 해시함수와 디지털 서명으로 공개키 인증

OpenSSL
Cryptography and SSL/TLS Toolkit

3. 코드 리뷰
- OpenSSL의 다양한 구조체의 동적 메모리 할당과 해제
- 에러 핸들링