SECURITY

paco

목차 TABLE OF CONTENTS

- 암호화 방식
- SSL
- 보안 공격 / 방어

보안의 3요소 - CIA

Confidentiality:
권한이 있는 사람에게 접근이 가능하고, 없는 사람에게 접근이 불가능하게 하는 것

- User ID - Password
- 생체인증

보안의 3요소 - CIA

Confidentiality:
권한이 있는 사람에게 접근이 가능하고, 없는 사람에게 접근이 불가능하게 하는 것

- User ID - Password
- 생체인증

Integrity:
데이터의 무결성. 데이터가 옮겨지는 중에 바뀌어서는 안됨. 권한이 없는 사람에 의해 바뀌어서도 안됨.

- File permissions

보안의 3요소 - CIA

Confidentiality:
권한이 있는 사람에게 접근이 가능하고, 없는 사람에게 접근이 불가능하게 하는 것

- User ID - Password
- 생체인증

Integrity:
데이터의 무결성. 데이터가 옮겨지는 중에 바뀌어서는 안됨. 권한이 없는 사람에 의해 바뀌어서도 안됨.

- File permissions

Availability:
접근성이 용이해야 함.

- 서버 살리기
- 업데이트 해주기
- 백업

암호화

ENCRYPTION

https://www.youtube.com/watch?v=6I2cu6h7Ycs&t=3m40s

“

ENCRYPTION IS THE PROCESS OF ENCODING A MESSAGE OR INFORMATION IN SUCH A WAY THAT ONLY AUTHORIZED PARTIES CAN ACCESS IT AND THOSE WHO ARE NOT AUTHORIZED CANNOT.

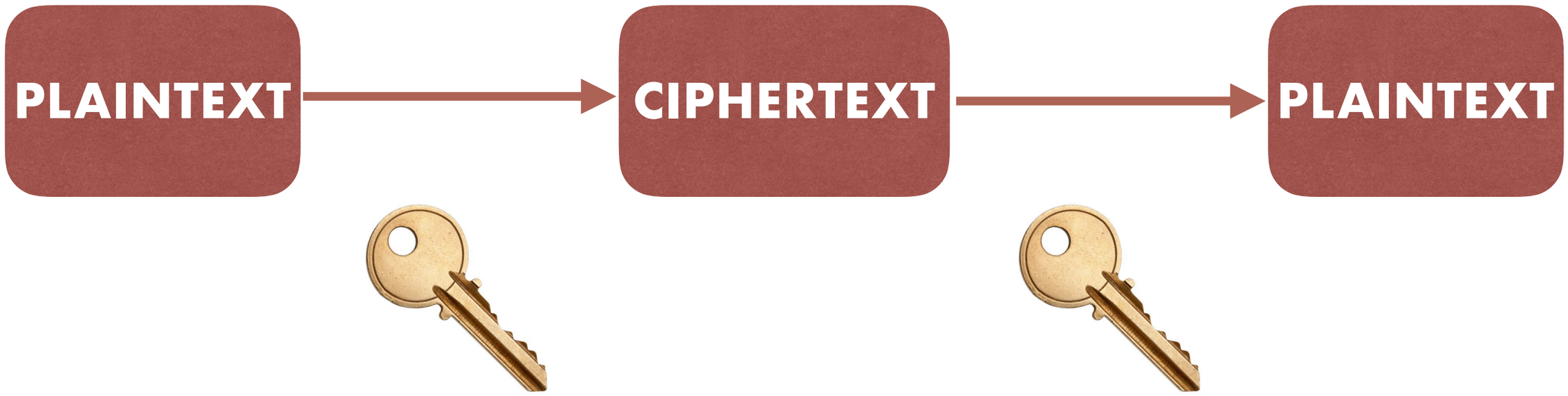
권한을 부여받은 사람만 어떤 내용을 접근/조회 수 있게 하는 과정

— Wikipedia —

”

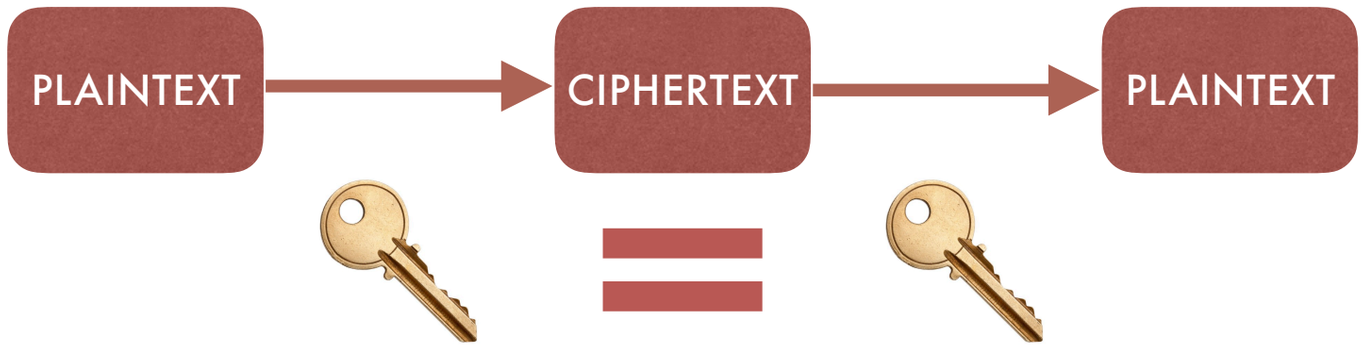
유용한 용어들

Plaintext - 평문: 암호화되지 않은 데이터
Ciphertext - 암호문: 암호화한 데이터
Key: 암호화할 때 필요한 정보
Alice & Bob: 암호화를 하는 모든 자리에 나타나는 영혼의 duo

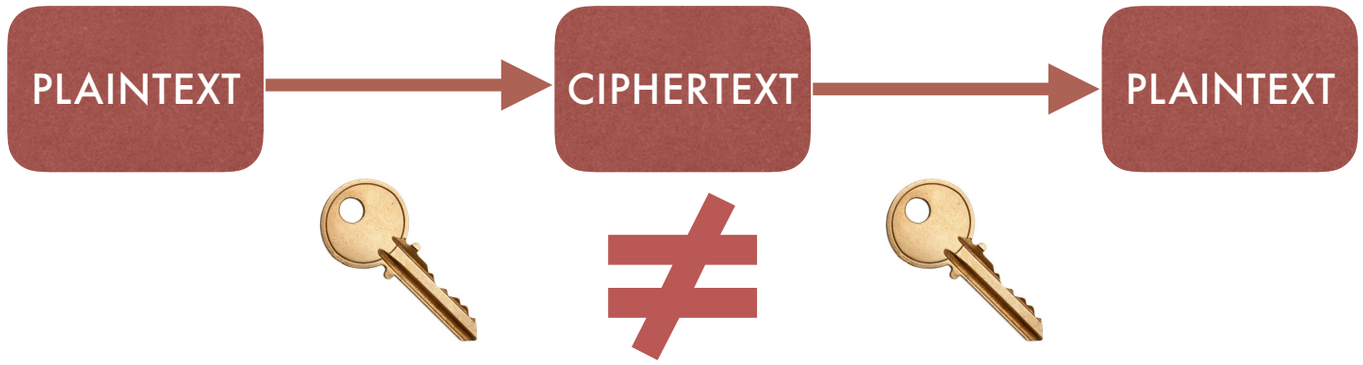


암호화의 종류

대칭키 암호화
Symmetric Key
Encryption



비대칭키 암호화
Asymmetric Key
Encryption



대칭키 암호화

- 암호화, 복호화에 쓰이는 키가 동일한 암호화 방식

특징

- 비대칭키 암호화보다는 오래된 암호화 방식
- 빠르고 효율적이기 때문에 대량의 데이터를 전송할 때 쓰임
- 키 하나가 노출되면 양단의 소통 정보가 노출될 위험이 있음

대칭키 암호화

- 암호화, 복호화에 쓰이는 키가 동일한 암호화 방식

종류

- **블록 암호 (Block Cipher)**
  - 블록(ex> 128 bit, 256 bit..) 단위로 암호화를 함
  - DES, 3DES, AES, Blowfish

- **스트림 암호 (Stream Cipher)**
  - 1 bit / byte 단위로 암호화를 함
  - RC4

대칭키 암호화

- 암호화, 복호화에 쓰이는 키가 동일한 암호화 방식

Difference

Stream Cipher	Block Cipher
Stream cipher operates on smaller Units of Plaintext	Block cipher operates on larger block of data
Faster than block cipher	Slower than Stream Cipher
Stream cipher processes the input element continuously producing output one element at a time	Block cipher processes the input one block of element at a time, producing an output block for each input block
Require less code	Requires more code
Only one time of key used.	Reuse of key is possible
Ex: One time pad	Ex: DES (Data Encryption Standard)
Application: SSL (secure connection on the web)	Application: Database, file encryption.
Stream cipher is more suitable for hardware implementation	Easier to implement in software.

Aforajayshahnirma.wordpress.com

실습타임

실습1

DES3 알고리즘으로 암호화

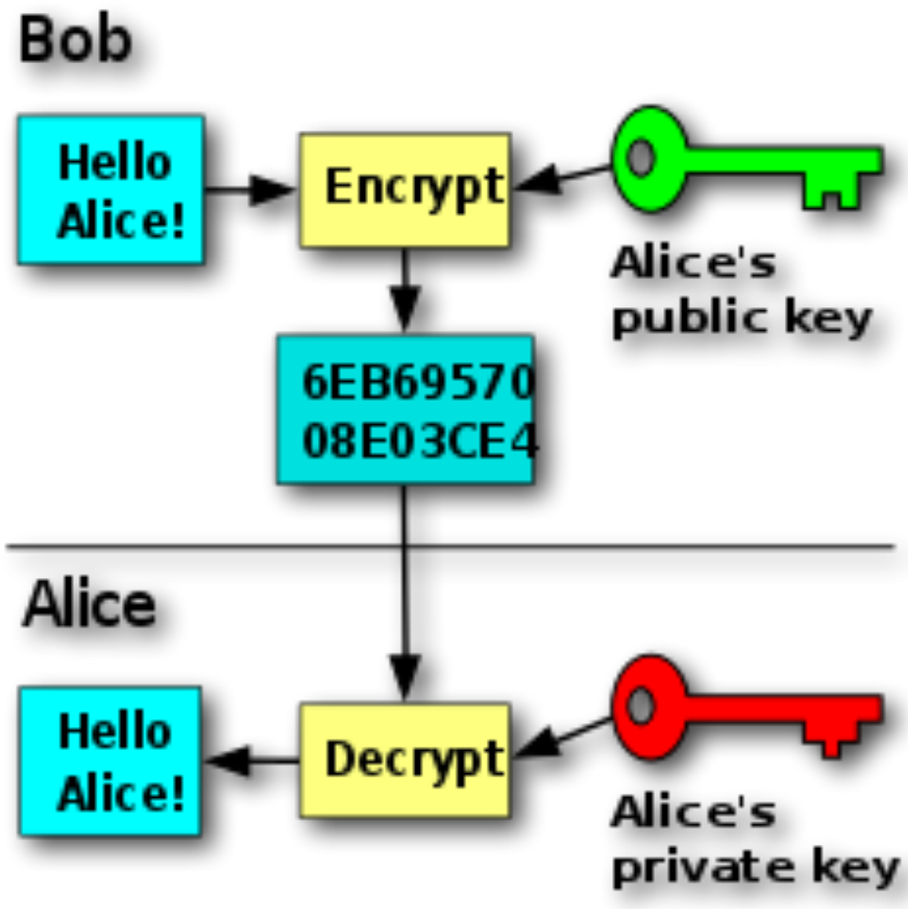
```
$ echo "this is wheel seminar" > plaintext
$ openssl des3 -in plaintext -out ciphertext
```

복호화

```
$ openssl des3 -d -in ciphertext -out decrypted
```

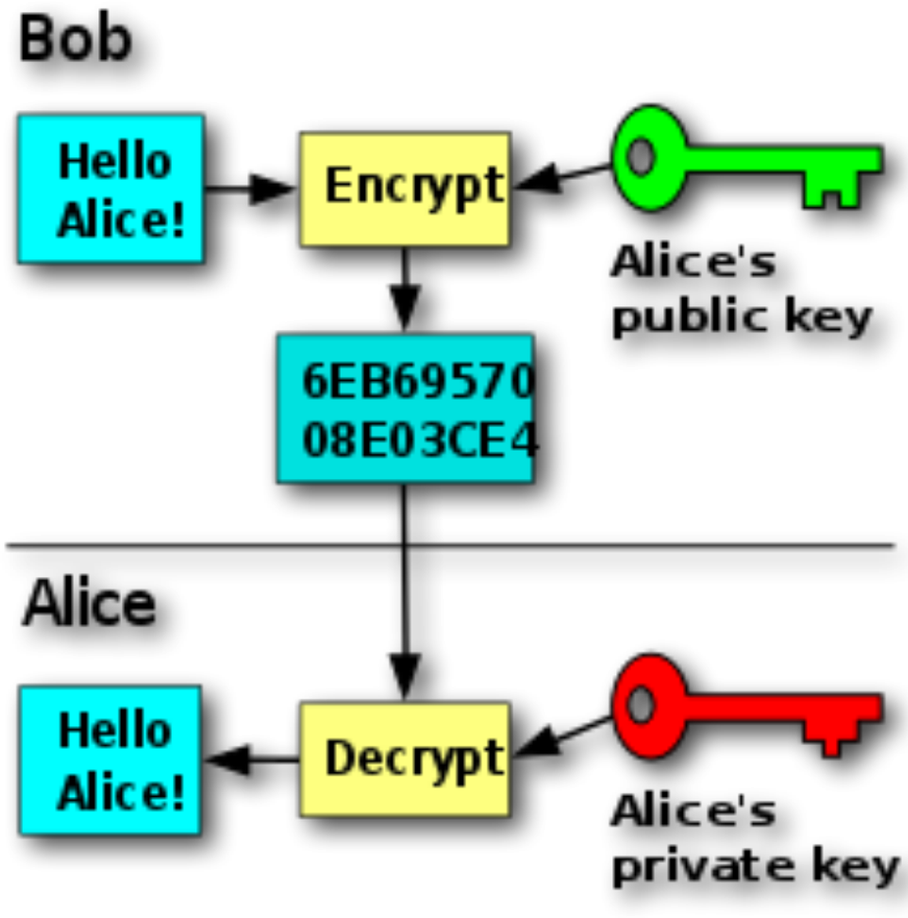
비대칭키 암호화

- 암호화, 복호화에 쓰이는 키가 다른 암호화 방식

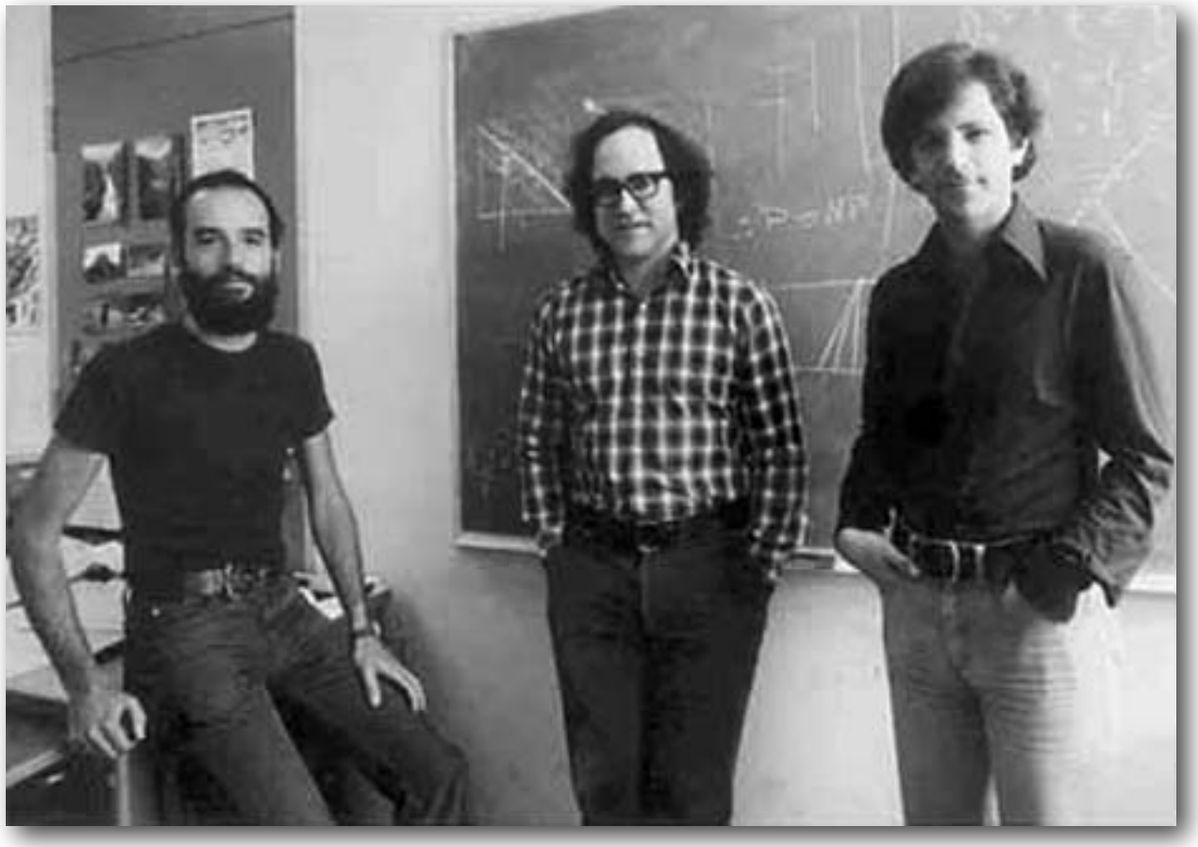


비대칭키 암호화

- 암호화, 복호화에 쓰이는 키가 다른 암호화 방식



RSA (Rivest-Shamir-Adleman)

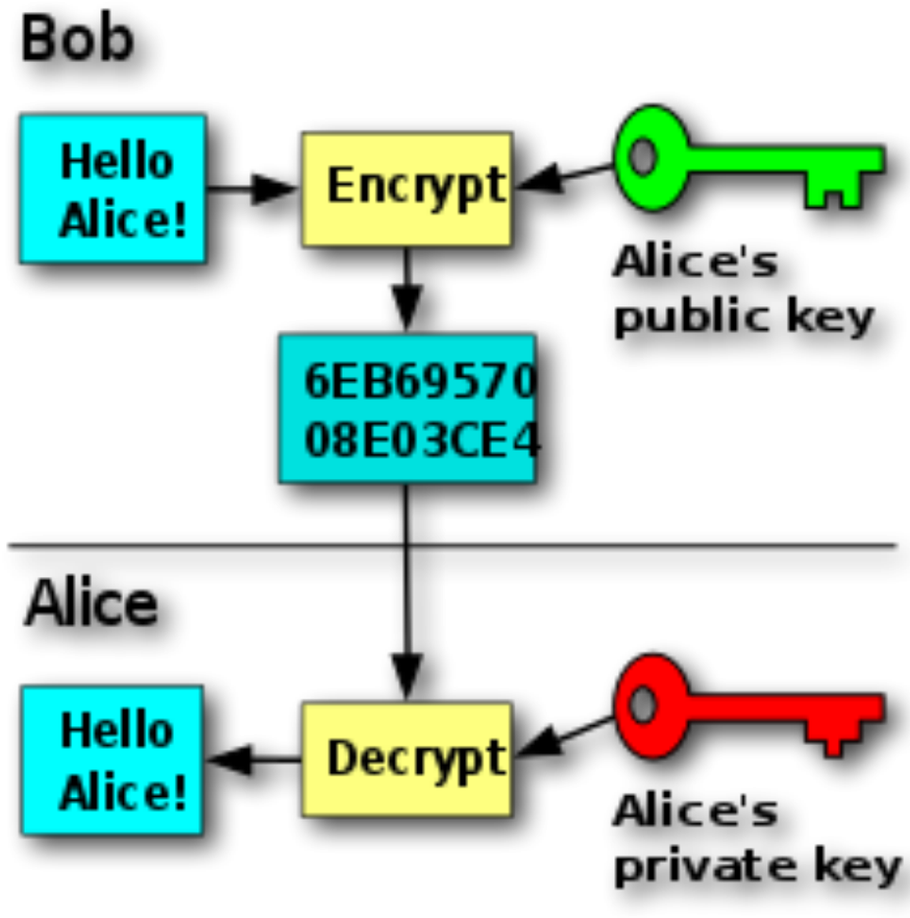


비대칭키 암호화의 구현으로는 RSA 알고리즘이 가장 널리 사용된다.

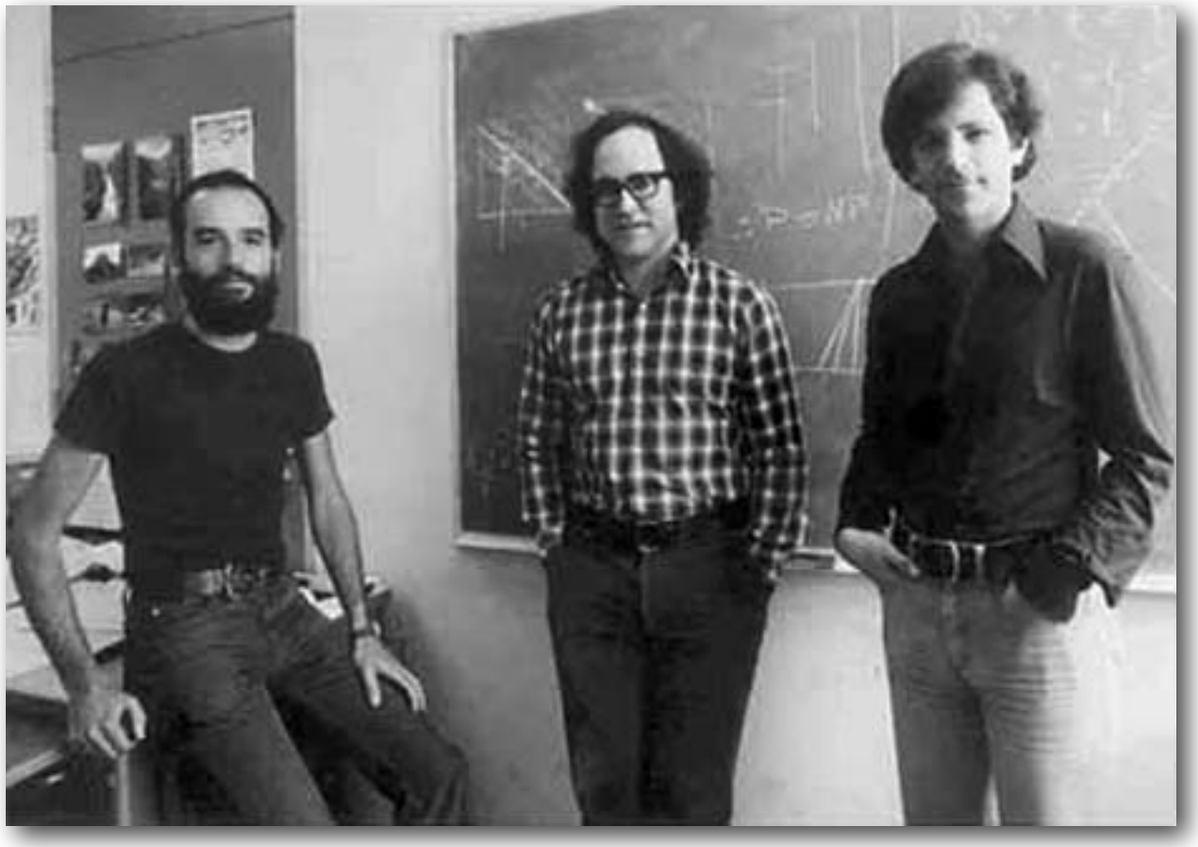
- 두 소수 p, q가 있을 때 각각의 값을 모르는 상태에서 pq를 소인수분해하는데 매우 오래 걸린다는 사실을 이용함.

비대칭키 암호화

- 암호화, 복호화에 쓰이는 키가 다른 암호화 방식



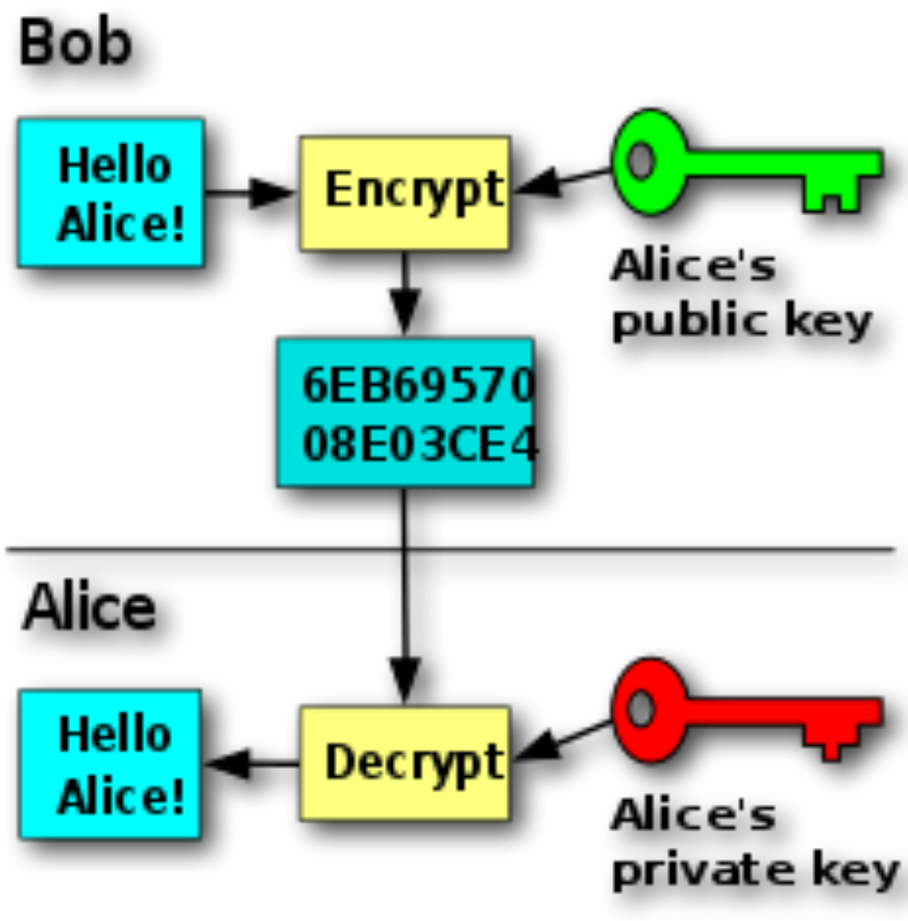
RSA (Rivest-Shamir-Adleman)



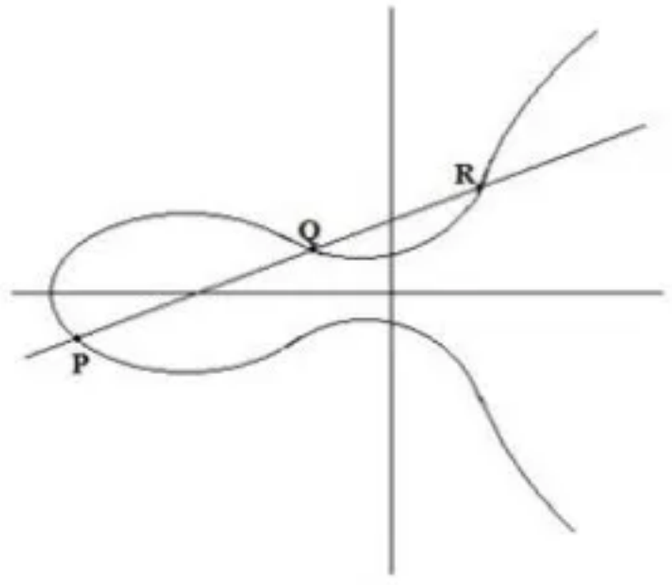
일반적인 컴퓨팅 파워로는 2048 bit RSA 암호를 복호화하기 위해서 64000000000000000(=6400조)년이 걸린다고 함.

비대칭키 암호화

- 암호화, 복호화에 쓰이는 키가 다른 암호화 방식



ECC - Elliptic-Curve Cryptography



복호화하기 위해서: 타원 곡선의 이산 로그를 찾아야 함 - Factoring보다 어려운 문제로 간주됨

더 짧은 키로도 RSA만큼의 안전성을 보장해준다.

비대칭키 암호화

- 암호화, 복호화에 쓰이는 키가 다른 암호화 방식

공개 키 암호

- 공개키로 암호화. 비밀키로 복호화
- 암호화는 누구나. 해독은 사용자만

공개 키 서명

- 비밀키로 암호화. 공개키로 복호화
- 암호화는 사용자만. 해독은 아무나

실습타임

실습 - 비대칭키 암호

영희의 Private Key 생성

```
$ openssl genrsa -out private.pem 2048
```

영희의 Private Key로부터 영희의 Public Key 뽑아내기
(그리고 철수에게 주기)

```
$ openssl rsa -pubout -in private.pem -out public.pem
```

철수가 메시지를 암호화한 뒤, 암호문을 영희에게 전달

```
$ echo "younghee. do you know wheel seminar?" > plaintext
$ openssl rsautl -encrypt -in plaintext -out ciphertext -inkey public.pem -pubin
```

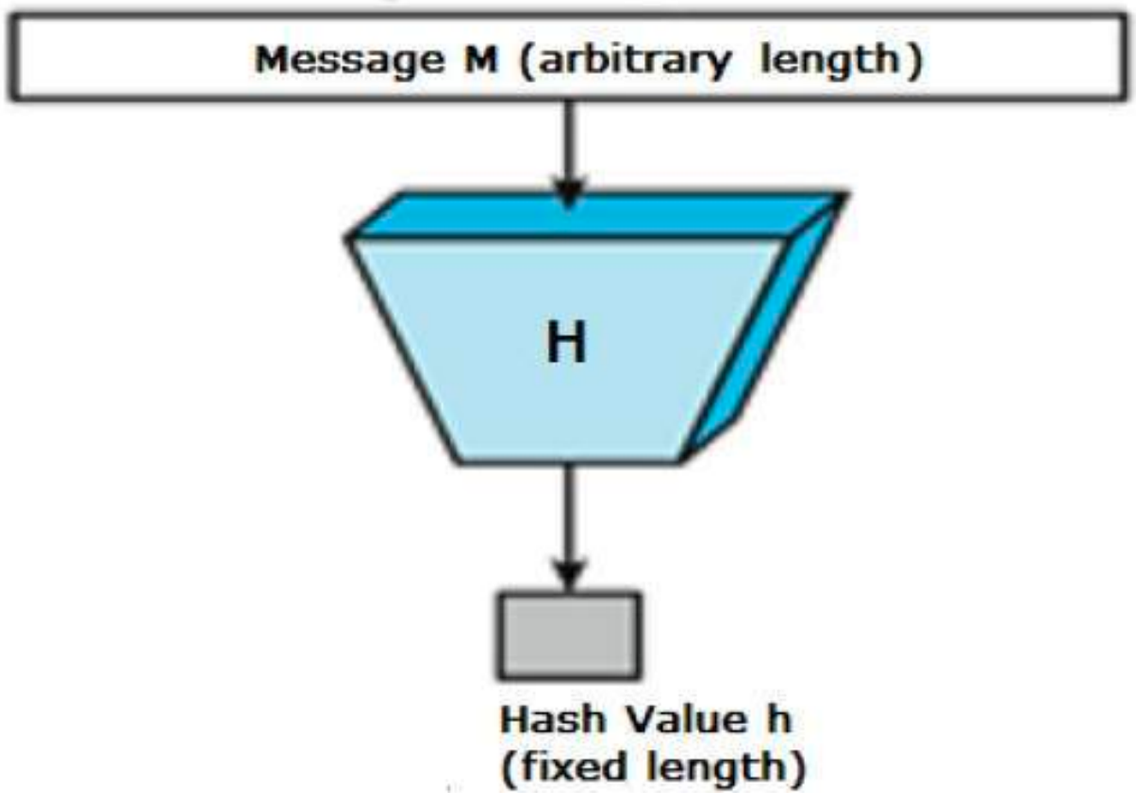
영희가 철수의 암호문을 복호화

```
$ openssl rsautl -decrypt -in ciphertext -inkey private.pem
```

```
$ echo -n "somemessage" | openssl sha256
```

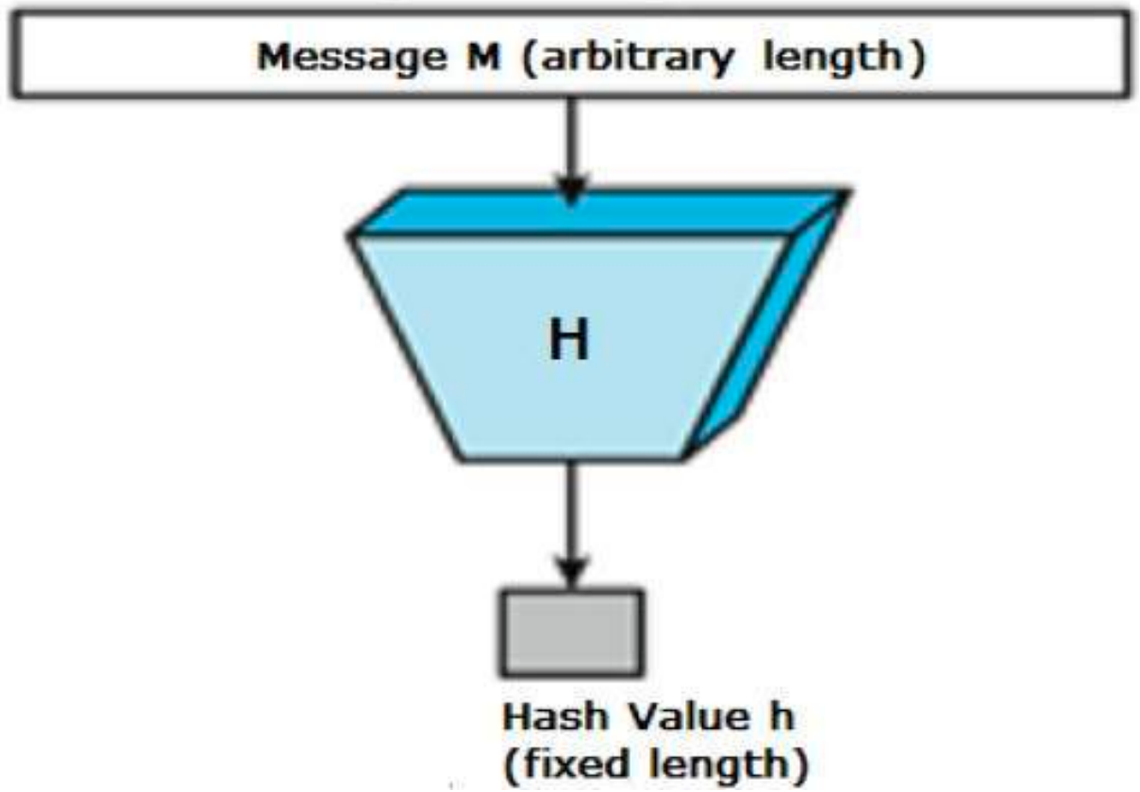
Hash 해시

- 어떤 데이터를 일정한 길이의 문자열로 변환하는 것



Hash 해시

- 어떤 데이터를 일정한 길이의 문자열로 변환하는 것
- 함수라 생각하면 편함



특징:

- 함수를 실행하는 것은 뚝딱
- 거꾸로 가는 것은 매우 어렵다

널리 쓰이는/쓰이던 알고리즘:

- MD5
  - 완전히 공략당함
- SHA-family
  - SHA-1
    - 완전히 공략당함
  - SHA-2
    - SHA-256
    - SHA-512

Hash 해시

- 어떤 데이터를 일정한 길이의 문자열로 변환하는 것
- 함수라 생각하면 편함

용도:

- 비밀번호 저장할 때
- 채굴

SHA1 Data & Hashes

Data: Hello
Hash: f7ff9e8b7bb2e09b70935a5d785e0cc5d9d0abf0

Data: The quick brown fox jumps over the lazy dog.
Hash: 408d94384216f890ff7a0c3528e8bed1e0b01621

Data: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
Hash: 99ed7eabae030ec036f35b16858af10fff840e53

Hash 해시 - Salt

- 원문의 앞이나 뒤에 임의의 문자열인 salt를 붙인 뒤 해시를 한다

공격방법:

- 흔할 것 같은 패스워드들을 해시 함수에 넘겨 각각 비교를 한다

```
e11c594e6a2f4eb499cceadfca988595:13LEXON
dcca2ed1630582435afa9d42ce361eb4:Admin11
d620cb449339aada319d5905e66d7924:azerty130
0426b809c0ee71d48407bc86461688d9:brain01
7bf7234ba9620f43cf43c87c13e9a4f6:c0nc0rd1
187655e4c9aff47ec2888f0cc2942efe:daniel179
a08c354f925b2a9df839290c8903c1e9:dodge15
86d3bc5436335ce3be44f4b8520c4506:francisca01
1dca20e382b4b72e4d8ae172eb3dd2c6:gustave01
6349546dd726f3f0a3b8adcc4a651ec1:hashpipe4
118e57b1eecd61e54b502055d82aa57f:hello world1
b3a8d3286f6d531d852659d45e2385af:hello world4
48f2064899478c034f36846ee1262dbf:hello2512
745c85844341eff2fe5aec3102dfd2ab:klopsiki3
c9944c427745e47c3d53ed63a70658d1:lilla3
8ffdb28c4deeac3e21fa786d2b252916:metro02
8c0d31cadefef386ed4ebb2daf1b80be:newports12
db66faf569e2fbef4d098dbdb32c937b:pistache7
cee3455a9e71ab468e3386661eec8ec5:quickshot4
10a88fb6a0a6acf23526d568d8e544d0:rowena01
7becb9424f38abff581f6f2a82ff436a:sail00
bc6f41d98e9040f9f961dbaf2420ed41:sarmad12
```

Hash 해시 - Salt

- 원문의 앞이나 뒤에 임의의 문자열인 salt를 붙인 뒤 해시를 한다

공격방법:

- 흔할 것 같은 패스워드들을 해시 함수에 넘겨 각각 비교를 한다

솔루션:

- 원문에 임의로 생성된 salt를 붙인 뒤 해시를 한다

```
e11c594e6a2f4eb499cceadfca988595:13LEXON
dcca2ed1630582435afa9d42ce361eb4:Admin11
d620cb449339aada319d5905e66d7924:azerty130
0426b809c0ee71d48407bc86461688d9:brain01
7bf7234ba9620f43cf43c87c13e9a4f6:c0nc0rd1
187655e4c9aff47ec2888f0cc2942efe:daniel179
a08c354f925b2a9df839290c8903c1e9:dodge15
86d3bc5436335ce3be44f4b8520c4506:francisca01
1dca20e382b4b72e4d8ae172eb3dd2c6:gustave01
6349546dd726f3f0a3b8adcc4a651ec1:hashpipe4
118e57b1eecd61e54b502055d82aa57f:hello world1
b3a8d3286f6d531d852659d45e2385af:hello world4
48f2064899478c034f36846ee1262dbf:hello2512
745c85844341eff2fe5aec3102dfd2ab:klopsiki3
c9944c427745e47c3d53ed63a70658d1:lilla3
8ffdb28c4deeac3e21fa786d2b252916:metro02
8c0d31cadefef386ed4ebb2daf1b80be:newports12
db66faf569e2fbef4d098dbdb32c937b:pistache7
cee3455a9e71ab468e3386661eec8ec5:quickshot4
10a88fb6a0a6acf23526d568d8e544d0:rowena01
7becb9424f38abff581f6f2a82ff436a:sail00
bc6f41d98e9040f9f961dbaf2420ed41:sarmad12
```

Hash 해시 - Salt

- 원문의 앞이나 뒤에 임의의 문자열인 salt를 붙인 뒤 해시를 한다

id	password_hash	salt
1	5cb40216084f69303a402e9d0df7fc427dd19c1fc39b77bbdc27cf803886250c	e8e4fddeda3c9c4ab28b648755eb9774
4	f577501e8c8e33f023b516ab93d3f4543e9da21b825dc0066538758e4189258b	374a28b338883a99cc5b3f6a279bdb0b
5	0c955b166d1bdbfac41d03107efb3acf68bc6417e54f67101130bc8ee0e3f0d5	173647e2394abcd546e6b143873f56c6
7	2626d6efecffc6fb7877df8c61ed0e89246dc80beeecfc6602e5f2cf4e47e1e3	6876851d532484e0ddbee8e4a06cbc82
8	9acc6b180bd34493a9e297b2c35e0a9f9dbc99bb07f6e812601820aa5954cda4	9dbadf08059e6d89bb9753d7c8be6015
9	e68507c0bf94c602cfb97e5163261752dd48eada4333cd9b2a4def30fe1ff785	d2e5660651ea519155ba0639a139612a
10	1f8fd95773afea93c86382a6a6a474c26a696a0c877a8a57dd98fea2e97c9640	5fdc198e267687fb57a9f46d66373faa
11	72228d19c34f13989098bee833c31a579aad14775a5e8983fe67282b6a559312	64f3b4975447ba052389dd41c58a1cdf
12	63d07e16e2bbc67ad14a78256053953868fb49b1cd098e0b4b61275394b88551	e497ccacce6f3851dd9ff65c39bd7990

```
10a88fb6a0a6acf23526d568d8e544d0:rowena01
7becb9424f38abff581f6f2a82ff436a:sail00
bc6f41d98e9040f9f961dbaf2420ed41:sarmad12
```

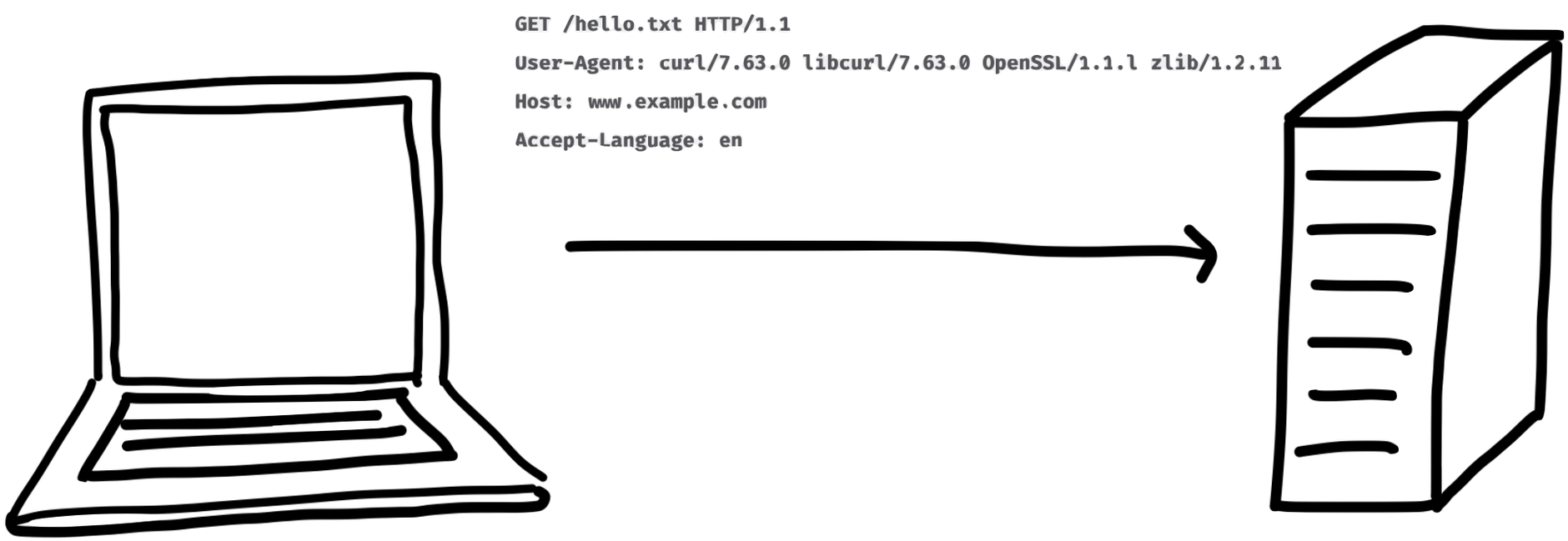
```
echo -n "somemessage" | openssl sha256
```

TLS

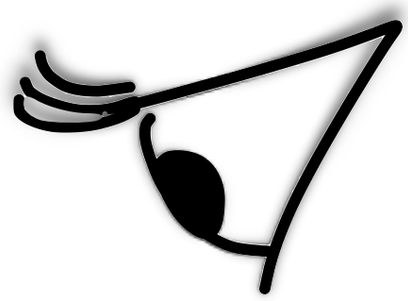
TRANSPORT LAYER SECURITY

TLS - Transport Layer Security

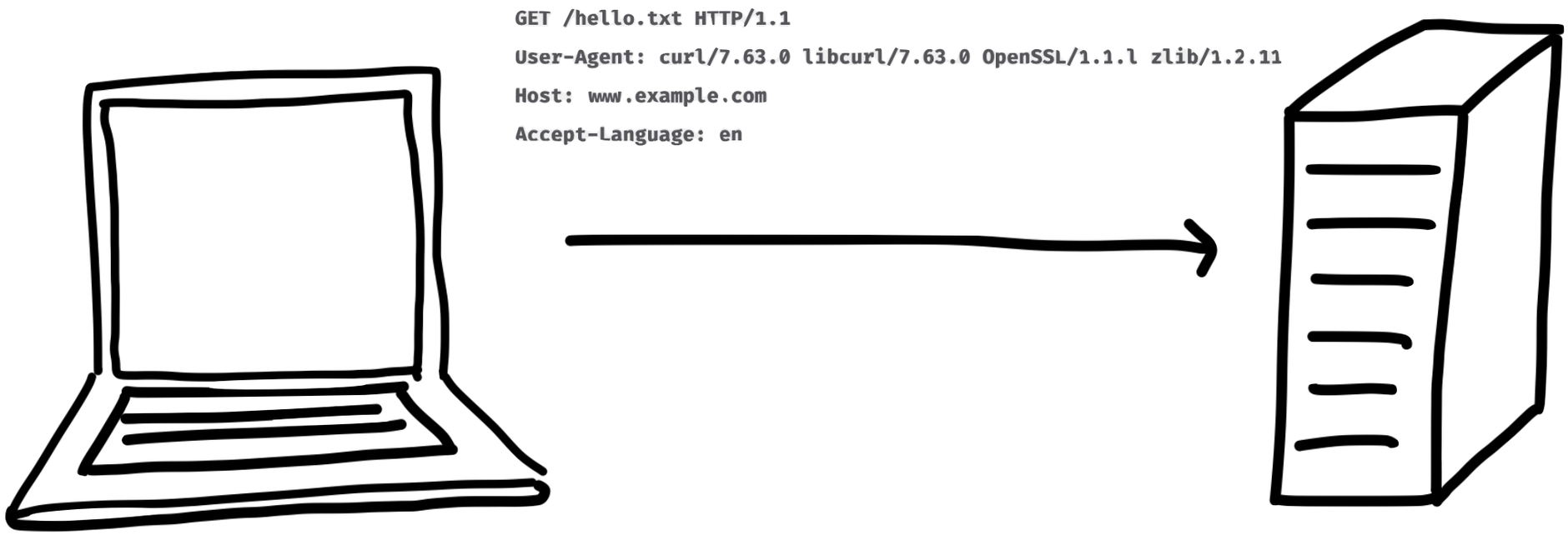
왜 씀?



TLS - Transport Layer Security

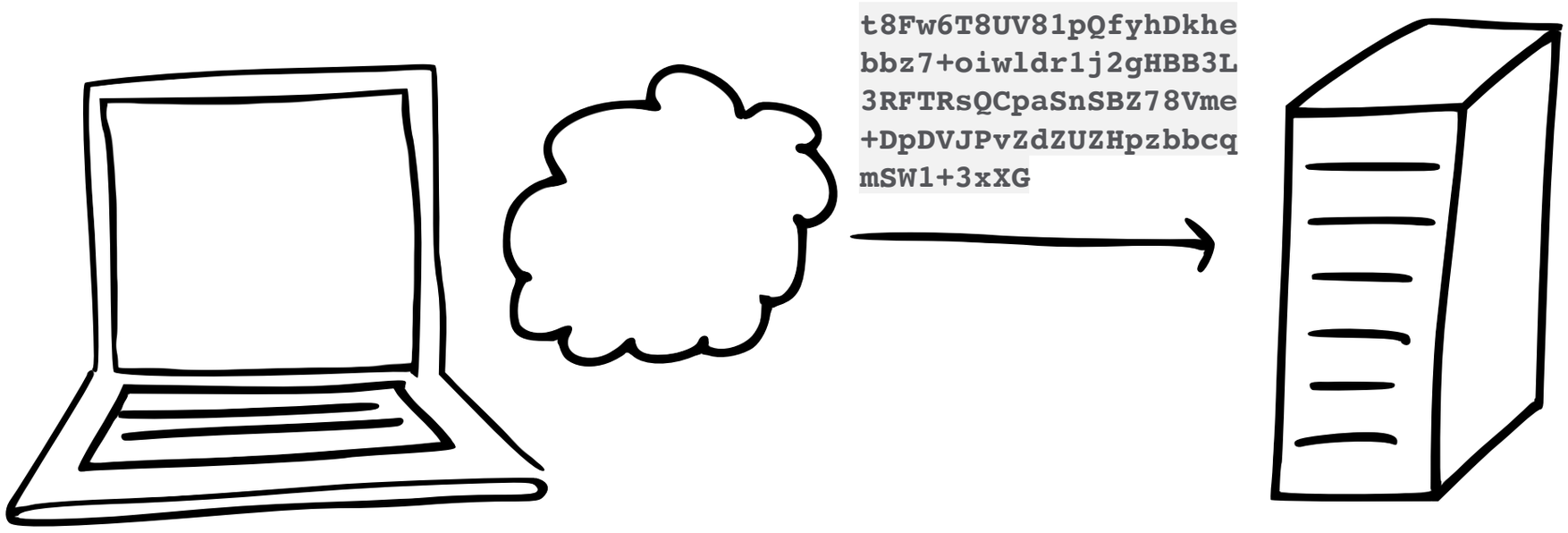


왜 씀?



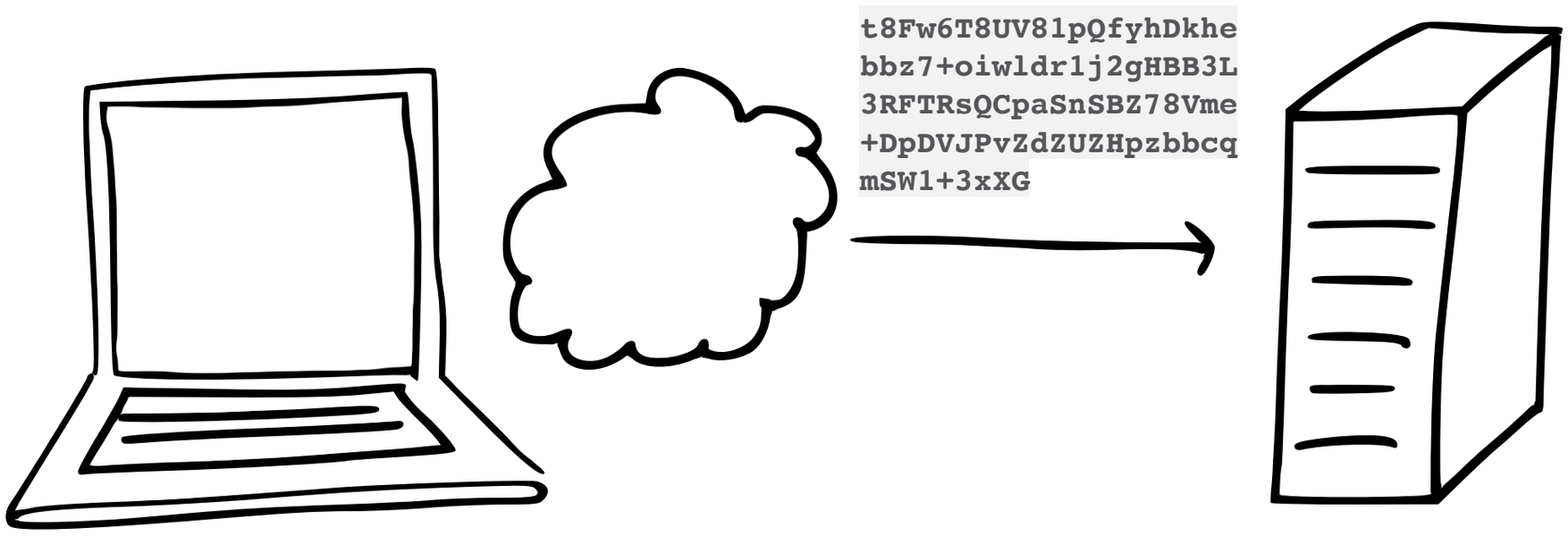
TLS - Transport Layer Security

왜 씀?



TLS - Transport Layer Security

왜 씀?



TLS - Transport Layer Security

특징

- SSL (Secure Socket Layer)을 바탕으로 만든 보안 프로토콜
- TCP/IP 상으로는 Application Layer에 해당
- 현재는 TLS v1.3 이용
- Google은 TLS를 적용한 웹 사이트들에게 더 높은 랭크를 부여한다.

TLS - Transport Layer Security

특징

- SSL (Secure Socket Layer)을 바탕으로 만든 보안 프로토콜
- TCP/IP 상으로는 Application Layer에 해당
- 현재는 TLS v1.3 이용
- Google은 TLS를 적용한 웹 사이트들에게 더 높은 랭크를 부여한다.

SSL이랑은 무슨 사이?

- SSL 1.0은 1995 Netscape에서 개발함. 첫 버전에 보안상 버그가 많아 이후 2.0, 3.0까지 개발이 됨
- IETF (Internet Engineering Task Force)가 SSL 3.0을 기반으로 프로토콜을 개발하기 시작. 그것이 TLS 1.0 — SSL 3.0 ≠ TLS 1.0 !
- 현재 널리 사용되는 프로토콜이 실제로는 TLS이지만 아직까지 SSL이라고 부른다.

https://www.ssl2buy.com/wiki/ssl-vs-tls
https://www.networkworld.com/article/2303073/lan-wan-what-is-transport-layer-security-protocol.html

TLS Handshake

: 서버와 클라이언트가 처음으로 소통하기 시작할 때 TLS Handshake를 한다.

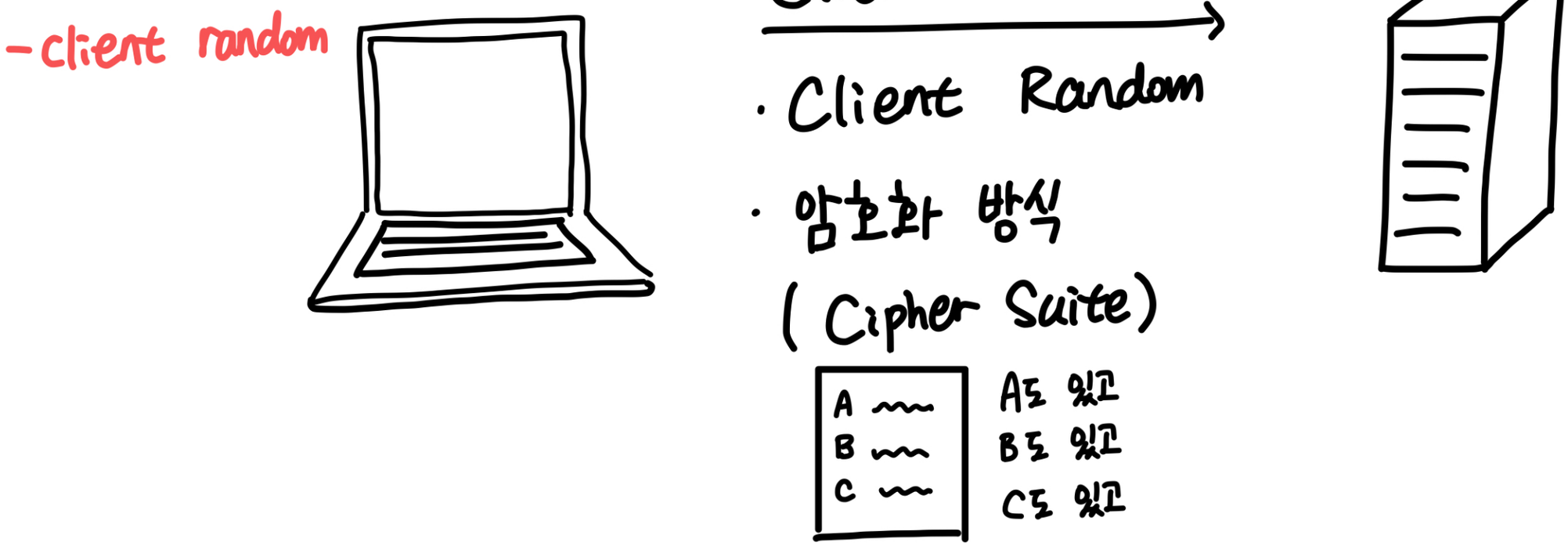
TLS Handshake가 성공적으로 이루어졌을 때, 서버와 클라이언트는 서로를 신뢰하고 소통을 한다.

(아주 단순화된) 단계들:

1. Client hello
2. Server hello
3. Authentication & Send premaster secret
4. Generate session keys
5. Done!

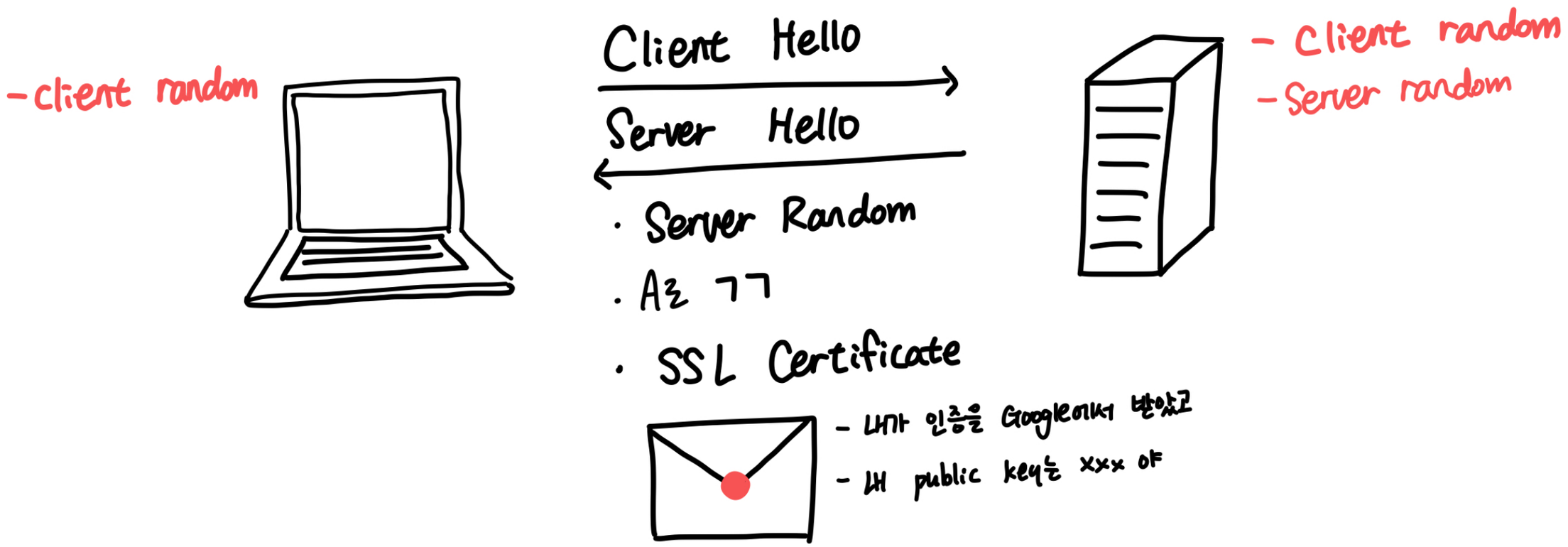
TLS Handshake

1. Client Hello



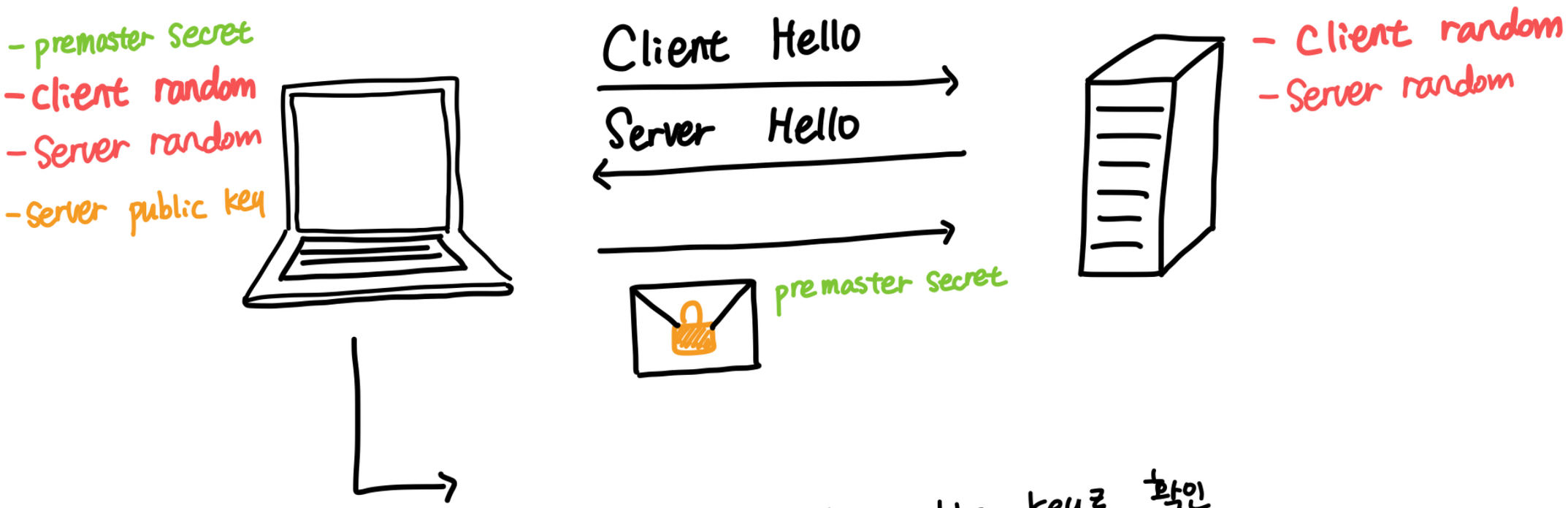
TLS Handshake

2. Server Hello



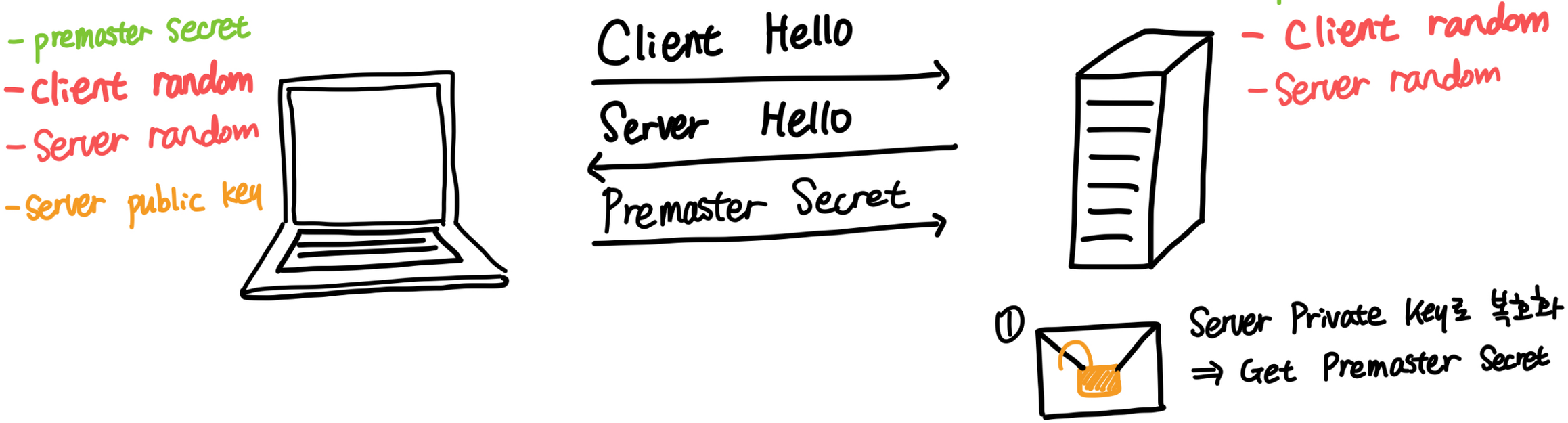
TLS Handshake

3. Authentication & Send premaster secret



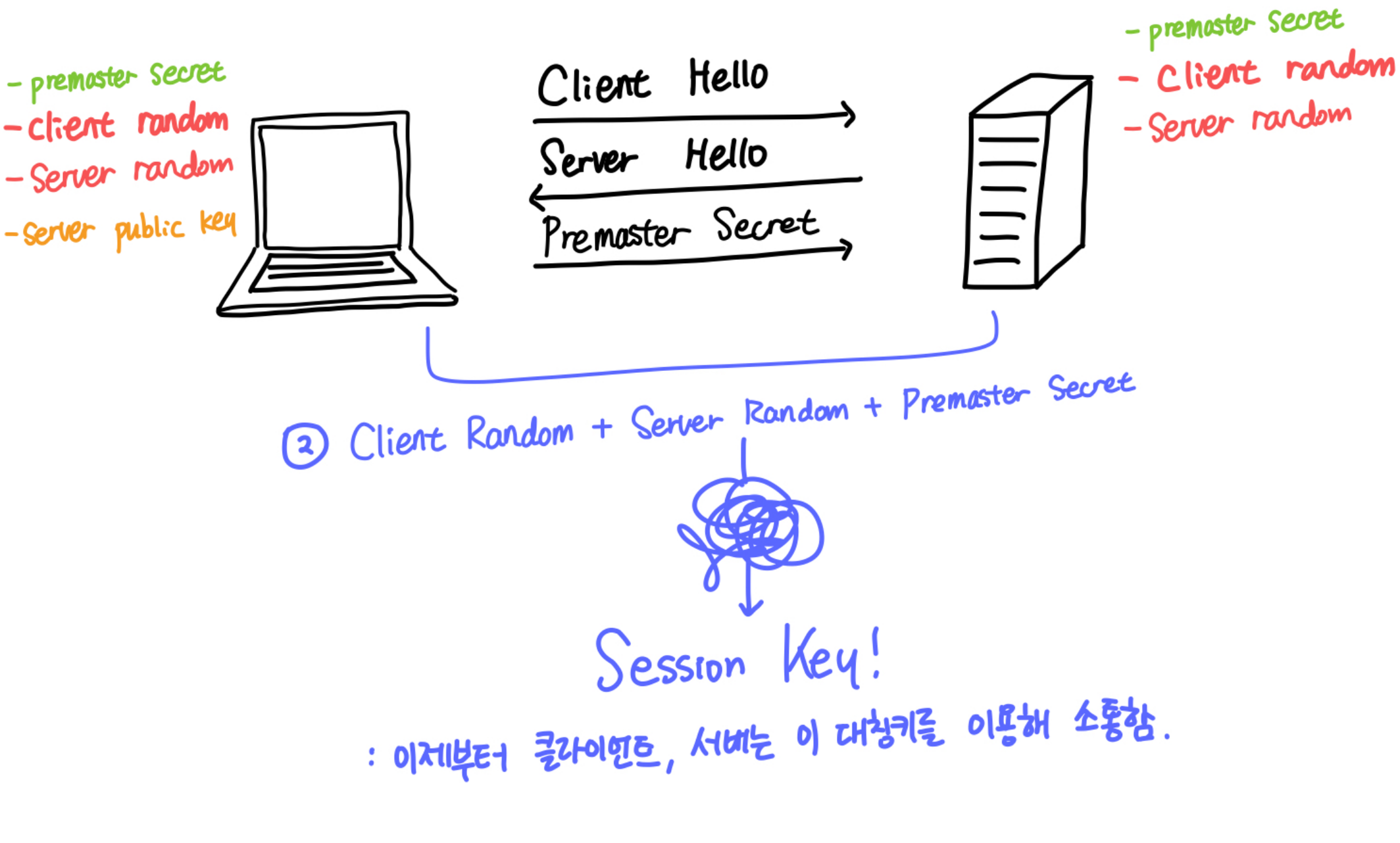
TLS Handshake

4. Generate session keys



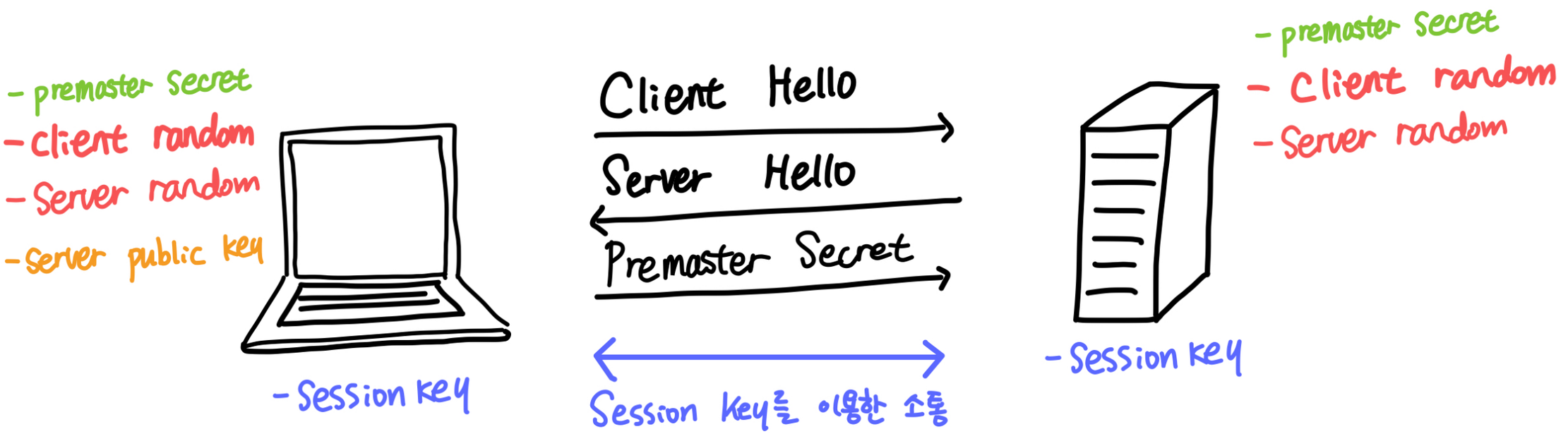
TLS Handshake

4. Generate session keys

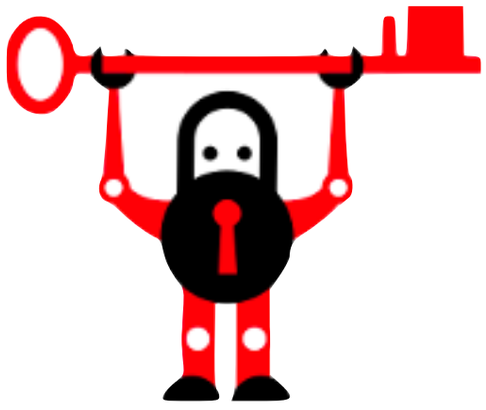


TLS Handshake

5. Done!



Let’s Encrypt / Certbot



> WE GIVE PEOPLE THE DIGITAL CERTIFICATES THEY NEED IN ORDER TO ENABLE HTTPS (SSL/TLS) FOR WEBSITES, FOR FREE, IN THE MOST USER-FRIENDLY WAY WE CAN. WE DO THIS BECAUSE WE WANT TO CREATE A MORE SECURE AND PRIVACY-RESPECTING WEB.
>
> *— Let’s Encrypt —*

실습타임

1. EC2 인스턴스를 띄운다
    - 22, 80, 443 포트를 열어주세요!
2. nginx를 설치한다
3. sparcs-kaist Route53에서 레코드 세트를 생성한다
4. 새로운 index html 파일을 생성한다
5. 새로운 nginx configuration 파일을 만든다
6. certbot을 설치하고 실행한다
7. 다시 접속한다

실습타임

1. EC2 인스턴스를 띄운다

- 22, 80, 443번 포트를 열어주세요!

2. nginx를 설치한다

```
$ sudo apt-get update
$ sudo apt-get install nginx
```

3. sparcs-kaist Route53에서 레코드 세트를 생성한다
4. 새로운 index html 파일을 생성한다
5. 새로운 nginx configuration 파일을 만든다

```
# your_domain.sparcs.org
server {
  root /var/www/html;
  server_name your_domain.sparcs.org
  index your_index.html
}
```

6. certbot을 설치하고 실행한다

```
# Add Certbot PPA
$ sudo apt-get update
$ sudo apt-get install software-properties-common
$ sudo add-apt-repository universe
$ sudo add-apt-repository ppa:certbot/certbot
# Install Certbot
$ sudo apt-get update
$ sudo apt-get install certbot python-certbot-nginx
# Run Certbot
$ sudo certbot --nginx
```

보안 공격/방어

여러 가지 공격 방법

- Brute Force Attack
- Dictionary Attack
- Dos/DDos
- Web based attack
  - SQL injection
  - XSS
  - CSRF
  - …

BRUTE FORCE ATTACK

:가능한 모든 방법을 다 쓰는 공격 방법
ex> 자물쇠 0000 ~ 9999 모두 대입

방어 솔루션

- n번 틀릴 때마다 m의 시간 간격을 준다
- 패스워드로 가능한 문자의 수를 늘린다
- 암호의 길이를 늘린다

DICTIONARY ATTACK

:여러 단어들을 조합하여 대입을 시도하는 공격 방법

ex> paco1234, ilovesparcs

방어 솔루션

- 더욱 다양한 문자를 패스워드에 넣도록 장려한다
- n번 틀릴 때마다 m의 시간 간격을 준다
- 데이터베이스에 저장할 때 salt를 추가한다

DOS/DDOS ATTACK

:타겟 서버에 대량의 요청을 보내 네트워크 자원을 모두 소모시키고, 서
실제 유저들에게는 서비스를 제공하지 못하게 하는 공격 방법

방어 솔루션

- 공격 대상 영역 축소
- 규모에 대한 대비
- 정상 및 비정상 트래픽 파악
- 방화벽 배포

https://aws.amazon.com/ko/shield/ddos-attack-protection/

SQL INJECTION

:가장 흔한 웹 공격 중 하나로 입력란에 SQL 코드를 입력해 데이터베이스에서 공격자가 원하는 정보를 가져오거나 데이터베이스에 피해를 입히는 공격 방법

```
SELECT * FROM Users WHERE UserId=[UserId]
```

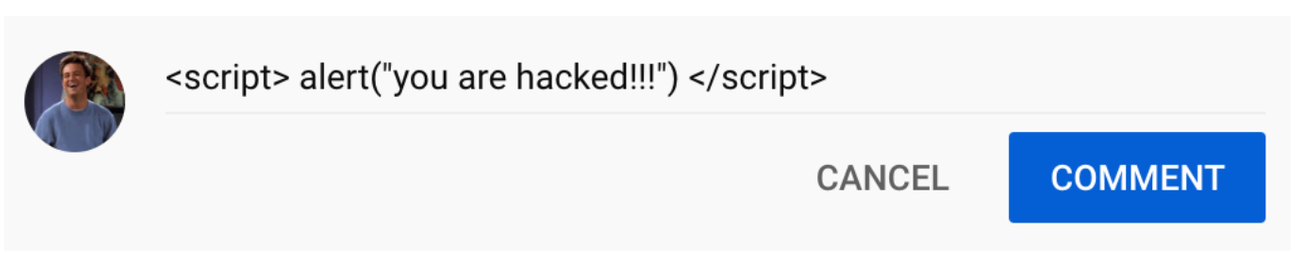
UserId: 105 or 1=1

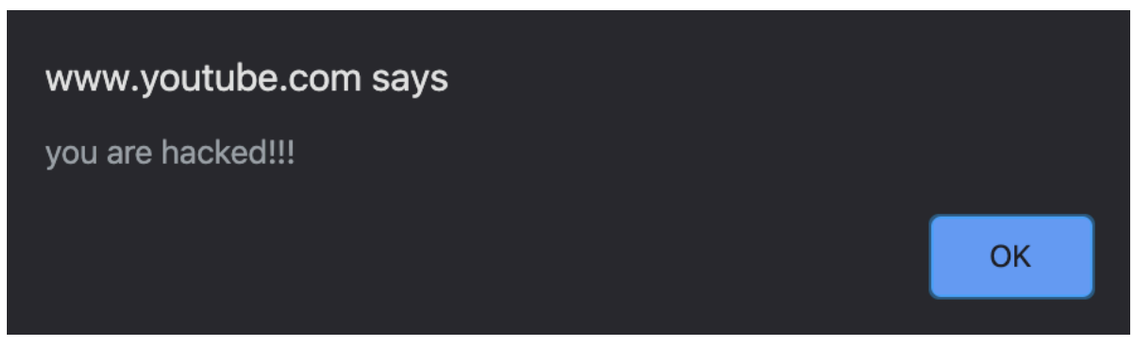
```
SELECT * FROM Users WHERE UserId=105 or 1=1
```

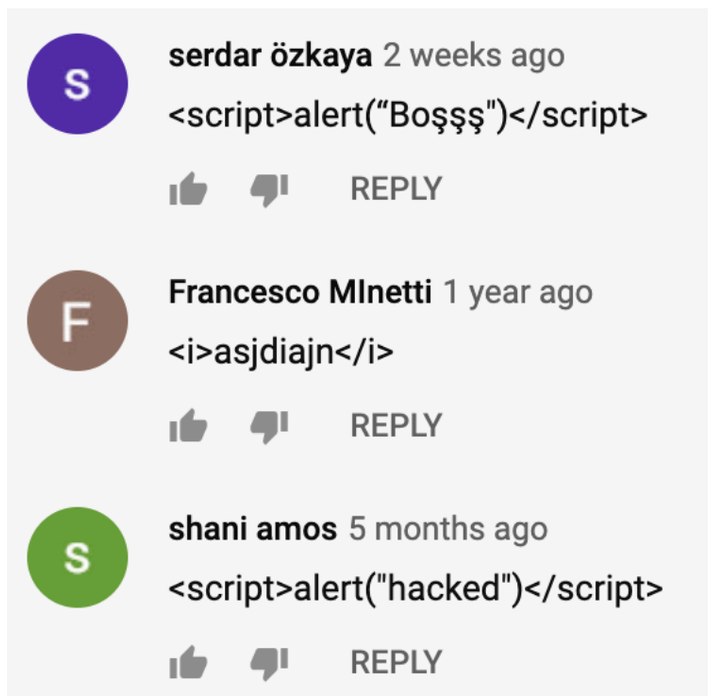
모든 유저를 가져옴

XSS (CROSS SITE SCRIPTING) ATTACK

:클라이언트 쪽에서 입력한 스크립트가 서버에 저장되며 다른 유저들이 페이지를 조회할 때 실행되게 하는 공격. 가장 많이 쓰이는 공격으로 알려져 있다.







SQLI, XSS 방어 솔루션

방어 솔루션:

SQLI, XSS 방어 솔루션

방어 솔루션:

1. 권한 설정
2. 인코딩
   - 민감한 문자들은 escape를 하여 공격을 방지
3. Invalidation
   - Input element에 들어올 수 있는 값의 범위를 최대한 좁히고 그 외의 값들은 invalidate시킨다

이외에도...

- 여러 가지 공격들
- fail2ban을 이용한 접근 거부 시간 설정
- iptables를 이용한 방화벽 설정

이 있어요!

{ 감사합니다 }

paco

Wheel Seminar - Security

실습 - 대칭키 암호

DES3 알고리즘으로 암호화

```
$ echo "this is wheel seminar" > plaintext
$ openssl des3 -in plaintext -out ciphertext
```

복호화

```
$ openssl des3 -d -in ciphertext -out decrypted
```

실습 - 비대칭키 암호

영희의 Private Key 생성

```
$ openssl genrsa -out private.pem 2048
```

영희의 Private Key로부터 영희의 Public Key 뽑아내기
(그리고 철수에게 주기)

```
$ openssl rsa -pubout -in private.pem -out public.pem
```

철수가 메시지를 암호화한 뒤, 암호문을 영희에게 전달

```
$ echo "younghee. do you know wheel seminar?" > plaintext
$ openssl rsautl -encrypt -in plaintext -out ciphertext -inkey public.pem -
pubin
```

영희가 철수의 암호문을 복호화

```
$ openssl rsautl -decrypt -in ciphertext -inkey private.pem
```

실습 - 해시 생성

```
$ echo -n "somemessage" | openssl sha256
```

실습 - Certbot을 이용해 인증서 달기

1. EC2 인스턴스를 띄운다

- 22, 80, 443번 포트를 열어주세요!

2. nginx를 설치한다

```
$ sudo apt-get update
$ sudo apt-get install nginx
```

3. sparcs-kaist Route53에서 레코드 세트를 생성한다
4. 새로운 index html 파일을 생성한다
5. 새로운 nginx configuration 파일을 만든다

```
# your_domain.sparcs.org
server {
  root /var/www/html;
  server_name your_domain.sparcs.org
  index your_index.html
}
```

6. certbot을 설치하고 실행한다

```
# Add Certbot PPA
$ sudo apt-get update
$ sudo apt-get install software-properties-common
$ sudo add-apt-repository universe
$ sudo add-apt-repository ppa:certbot/certbot
# Install Certbot
$ sudo apt-get update
$ sudo apt-get install certbot python-certbot-nginx
# Run Certbot
$ sudo certbot --nginx
```