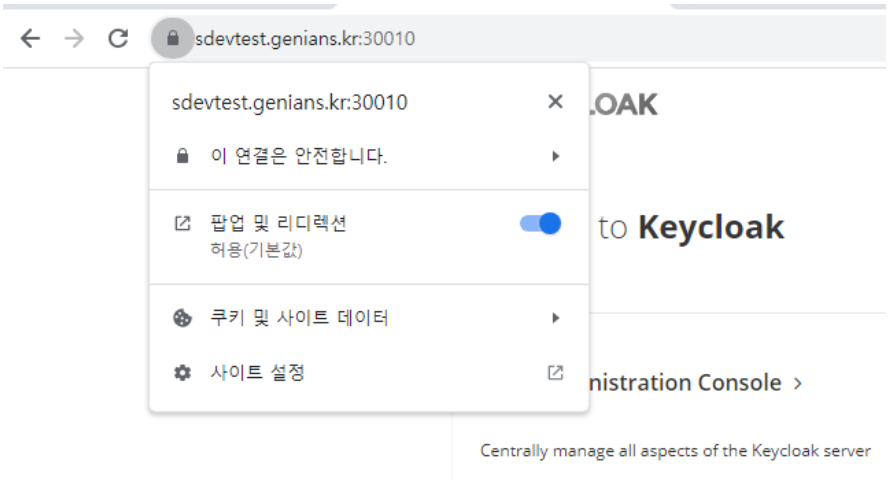
Keycloak - 인증서 생성 처리

왜 인증서가 필요한가?

네트워크를 통해 패킷을 주고 받을 때 중간에 패킷을 훔쳐(Sniffing) 보는 것을 방지하기 위하여 네트워크 데이터 암호화를 하여 패킷을 훔쳐 보더라도 데이터가 유출되는 것을 막을 수 있다.

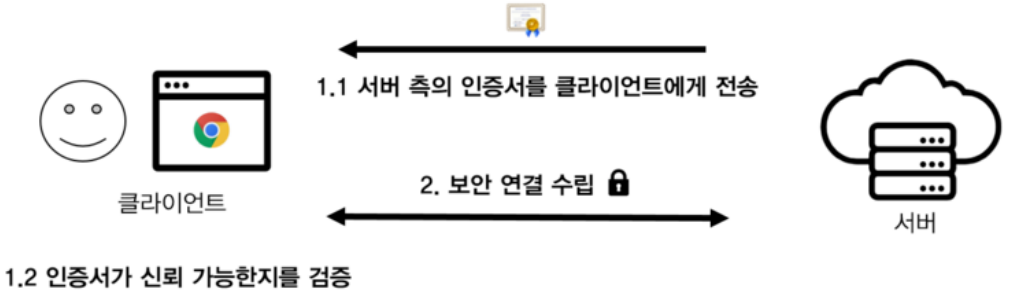
가장 보편적인 암호화 방식은 **SSL/TLS 으로 '인증서' 라고 하는 일종의 서명**을 사용한다. '인증서'는 신뢰할 수 있는 사람임을 확인하기 위한 용도로 신뢰할 수 있다는 검증 작업을 거치고 나면 데이터 암호화 작업이 수행된다. 이 과정을 SSL 보안 통신이라 한다.



흔히 사용하고 있는 보안 연결은 웹 브라우저와 웹 사이트 간의 연결을 예로 들 수 있는데 웹 브라우저의 주소 옆에 자물쇠는 인증서(CA)를 통해 브라우저 - 웹 서버 간 보안 연결이 수립됨을 의미한다.

인증서의 구조

서버와 클라이언트 간 보안 연결은 크게 두 단계로 나눈다. 첫 번째는 클라이언트의 CA를 통해 서버 인증서가 신뢰 가능한지를 확인하는 단계인데, 서버가 신뢰할 수 있는 사람임을 **'인증서'**를 통해 검증한다.



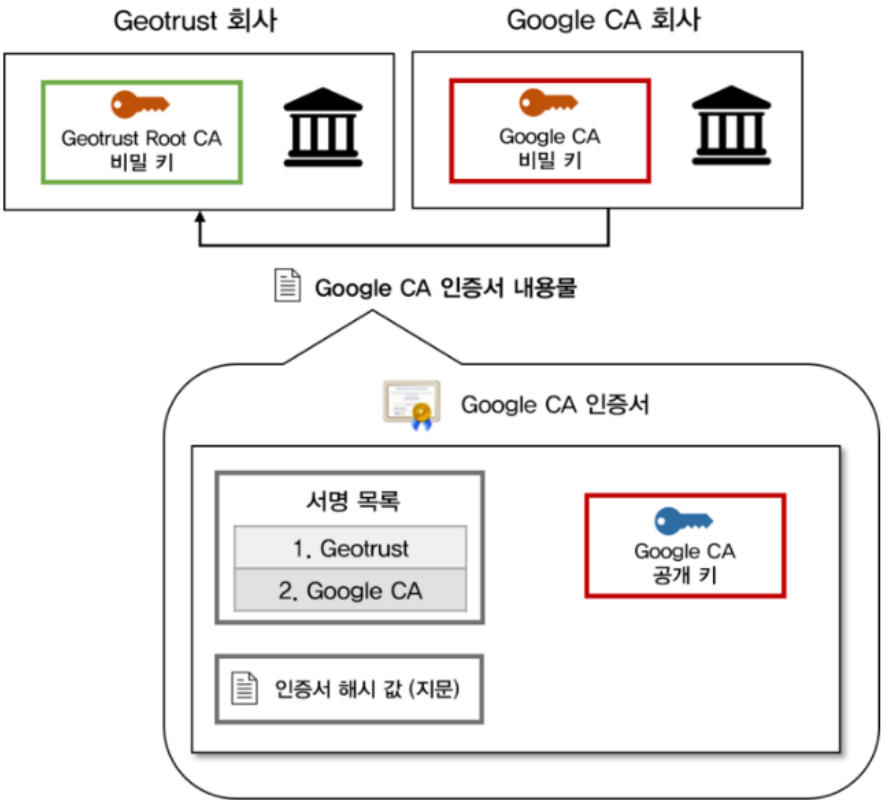
인증서 검증을 통해 서버가 신뢰할 수 있다고 판단되면, 클라이언트는 두 번째 단계인 보안 연결 수립 단계를 진행한다. 서버-클라이언트 간에 생성된 랜덤 값을 통해 대칭 키를 생성하고, 이를 통해 네트워크 데이터를 암호화해 전송한다.

1단계 : 신뢰 가능한 기관

무조건적으로 신뢰할 수 있는 기관이 몇 존재하는데, 해당 기관들은 최상위 인증 기관인 Root CA라는 인증서를 발급하는 기관이다. 해당 기관들은 고유한 비밀 키를 가지고 대응하는 공개 키를 배포한다. 암묵적으로 이 기관들은 신용할 수 있음을 약속한 뒤 배포된 공개 키로 복호화가 가능한 데이터는 기관의 비밀 키로 암호화되었기 때문에 신용할 수 있는 데이터라고 간주한다.

Geotrust ↔ Google CA 예시(2계층)

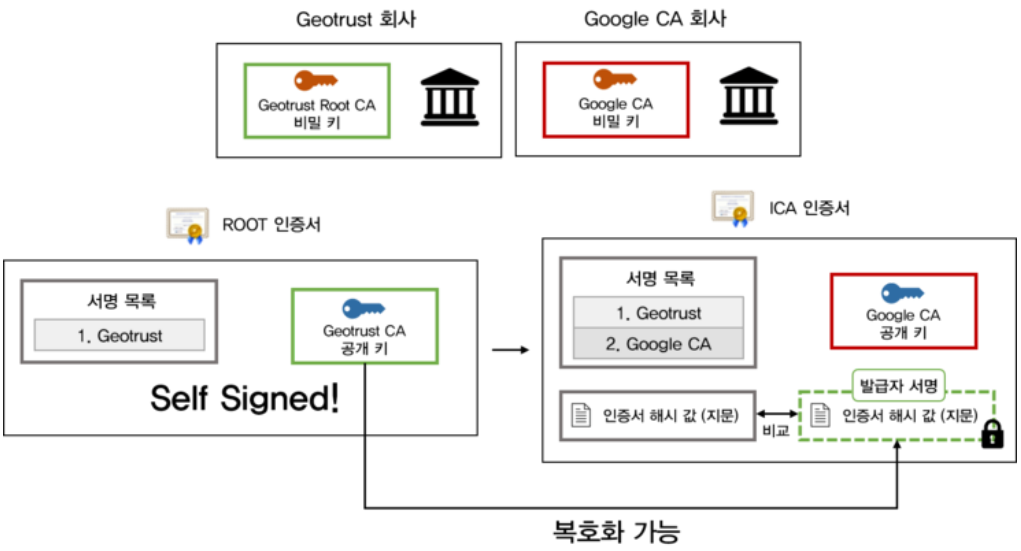
인증서는 **계층 구조**로 되어 있어서 일반적으로 3계층 구조로 되어 있다. 가장 최상위에 위치한 인증서는 Root 인증서로 신뢰 가능한 기관에서 발행한 인증서에 해당한다. 일반적으로 웹 브라우저 등에 미리 내장되어 있으며, 해당 인증서에 대응하는 공개 키 또한 인증서 내부에 포함되어 있다.



위 그림을 기준으로 설명하면, 가장 최상위 Root 인증서을 발행한 기관은 Geotrust 이며 인증서를 만들고 싶은 Google CA 회사는 Geotrust에게 인증서의 내용물을 종합해 해시화 한 값을 Geotrust 의 비밀키로 암호화를 해달라고 하는 것이다.

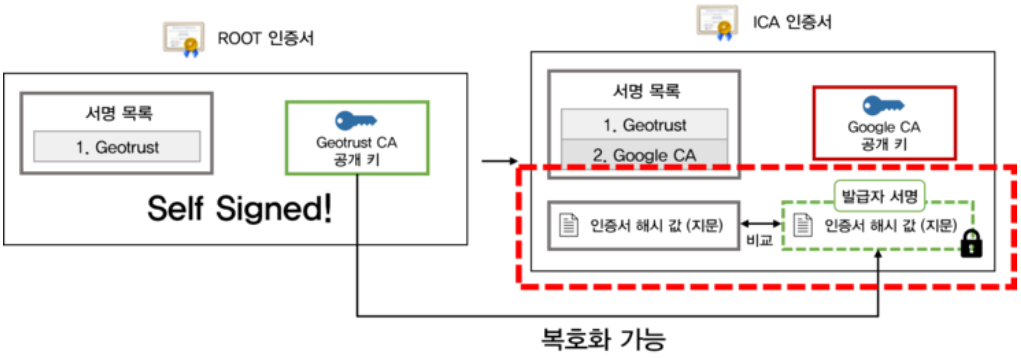
이는 Geotrust 공개 키는 모두가 가지고 있고 해당 공개키로 복호화 되는 데이터는 신뢰할 수 있는 데이터라고 간주하는 것이다.

Chain of Trust



이러한 원리를 **Chain of Trust**라고 부른다. 상위 계층의 인증서 (Geotrust) 가 신뢰할 수 있는 기관이라면, 해당 인증서 (Geotrust) 의 비밀 키로 암호화된 하위 인증서 (Google CA) 또한 신뢰 가능하다고 간주한다.

그렇다면 굳이 인증서의 내용물로 해시 값을 만든 뒤, 이 값을 Geotrust의 비밀 키로 암호화한 이유는 무엇일까?



Geotrust의 공개 키를 이용해, Geotrust의 하위 인증서인 Google CA 인증서의 암호화된 해시값을 복호화했고, Google CA의 인증서를 생성할 당시에 사용한 해시 값을 얻었다고 가정해보자.

그런데, **(1) 복호화된 해시 값과 (2) 인증서에 들어 있는 내용물을 다시 해시값**으로 만들어 비교를 해보니 **동일한 값이 아닌 경우가 발생한다면,** 이는 곧 Google CA 인증서의 내용물이 변조되었음을 의미한다.

따라서, 상위 인증서 기관의 공개 키로 하위 기관의 인증서 해시 값을 복호화 한다는 것은 (1) Chain of Trust의 원리에 의해 하위 인증서가 신뢰할 수 있는지를 알 수 있으며, (2) 하위 인증서의 내용물이 변조됐는지를 알 수 있게 된다.

SSL 인증서 생성

CSR : 인증서 서명 요청

PKI (공개키 기반) 은 개인키 (Private Key) 와 공개키 (Public Key) 가 쌍으로 이룬다. 인증서는 나의 공개키가 나의 공개키가 맞다고 인증기관 (CA) 의 개인키로 전자서명을 한 것이며, 나와 보안통신을 하려는 상대방이 내 인증서를 확인하고 그 인증서 안에 있는 나의 공개키를 이용하여 나에게 암호화된 데이터를 만들어 보낼 수 있다.

이때 인증기관 (CA) 에 나의 인증서를 만들어 달라고 요청하기 위해서 CSR (Certificate Signing Request) 이라는 ASN.1 이라는 형식의 파일을 만들어 인증기관에 요청하게 된다. CSR 에는 내 공개키 정보와 사용하려는 알고리즘 정보 등이 들어있다.

개인키는 외부에 유출되면 안되기 때문에 CSR 형식의 파일을 만들어서 인증기관에 전달하여 인증기관으로 하여금 나의 인증서를 만들어서 전달 받게 된다.

Root CA 인증서 생성

OS : WSL2 Ubuntu 22.04 LTS

1. RSA 키 생성

```
openssl genrsa -aes256 -out genian_keycloak-rootca.key 2048
```

이 명령은 AES-256 비트로 암호화된 indienote-rootca.key 라는 개인키 파일을 생성한다. 생성될 개인키의 길이는 2048 비트이다.

생성할 때 개인키를 암호화할 암호를 입력하게 되는데 암호를 꼭 기억해 두자.

2. 개인키 권한 설정

```
chmod 600 genian_keycloak-rootca.key
```

3. CSR 파일을 생성하기 위해서 genian_keycloak-rootca.conf 파일을 생성

```
[ req ]
default_bits            = 2048
default_md              = sha1
default_keyfile         = genian_keycloak-rootca.key
distinguished_name      = req_distinguished_name
extensions              = v3_ca
req_extensions          = v3_ca

[ v3_ca ]
basicConstraints       = critical, CA:TRUE, pathlen:0
subjectKeyIdentifier   = hash
##authorityKeyIdentifier = keyid:always, issuer:always
keyUsage               = keyCertSign, cRLSign
nsCertType             = sslCA, emailCA, objCA
```

```
[req_distinguished_name ]
countryName                     = Country Name (2 letter code)
countryName_default             = KR
countryName_min                 = 2
countryName_max                 = 2

# 회사명 입력
organizationName              = Organization Name (eg, company)
organizationName_default      = Genian_keycloak_ms

# 부서 입력
#organizationalUnitName          = Organizational Unit Name (eg, section)
#organizationalUnitName_default  = CA Project

# SSL 서비스할 domain 명 입력
commonName                      = Common Name (eg, your name or your server's hostname)
commonName_default              = sdevtest.genians.kr (서비스할 domain 명을 입력)
commonName_max                  = 64
```

4. CSR 파일을 생성한다.

```
openssl req -new -key genian_keycloak-rootca.key -out genian_keycloak-rootca.csr -config genian_keycloak-rootca.conf
```

```
root@DESKTOP-0SI26G1:~# openssl req -new -key indienote-rootca.key -out indienote-rootca.csr -config indienote-rootca.co
nf
Enter pass phrase for indienote-rootca.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [KR]:
Organization Name (eg, company) [Genian_keycloak_ms]:
Common Name (eg, your name or your servers hostname) [localhost:30004]:
root@DESKTOP-0SI26G1:~#
```

```
-rw-r--r--  1 root root  1156 Oct 17 00:09 indienote-rootca.conf
-rw-r--r--  1 root root  1086 Oct 17 00:10 indienote-rootca.csr
-rw-------  1 root root  1874 Oct 16 23:05 indienote-rootca.key
```

5. n 년짜리 Self Signed 인증서 생성

```
openssl x509 -req \
-days 365 \
-extensions v3_ca \
-set_serial 1 \
-in genian_keycloak-rootca.csr \
-signkey genian_keycloak-rootca.key \
-out genian_keycloak-rootca.crt \
-extfile genian_keycloak-rootca.conf
```

서명에 사용할 해시 알고리즘을 변경하려면 -sha256, -sha384, -sha512 처럼 해시를 지정하는 옵션을 추가해 주면 된다. 생략하면 기본값으로 -sha256 이 적용된다.

```
root@DESKTOP-0SI26G1:~/ssl# openssl x509 -req \
-days 365 \
-extensions v3_ca \
-set_serial 1 \
-in genian_keycloak-rootca.csr \
-signkey genian_keycloak-rootca.key \
-out genian_keycloak-rootca.crt \
-extfile genian_keycloak-rootca.conf
Enter pass phrase for genian_keycloak-rootca.key:
Certificate request self-signature ok
subject=C = KR, O = genian_keycloak, CN = localhost
root@DESKTOP-0SI26G1:~/ssl# ls -al
total 44
drwxr-xr-x  2 root root 4096 Oct 17 04:00 .
drwx------ 29 root root 4096 Oct 17 05:07 ..
-rw-r--r--  1 root root 1157 Oct 17 05:54 genian_keycloak-rootca.conf
-rw-r--r--  1 root root 1196 Oct 17 05:56 genian_keycloak-rootca.crt
-rw-r--r--  1 root root 1078 Oct 17 05:55 genian_keycloak-rootca.csr
-rw-------  1 root root 1874 Oct 16 23:05 genian_keycloak-rootca.key
-rw-r--r--  1 root root 1625 Oct 17 05:03 genian_keycloak.conf
-rw-r--r--  1 root root 1371 Oct 17 05:11 genian_keycloak.crt
-rw-r--r--  1 root root  985 Oct 17 05:54 genian_keycloak.csr
-rw-------  1 root root 1704 Oct 17 04:00 genian_keycloak.key
-rw-------  1 root root 1874 Oct 17 03:59 genian_keycloak.key.enc
```

6. 정상적으로 Root CA 인증서가 생성되었는지 확인

```
openssl x509 -text -in genian_keycloak-rootca.crt
```

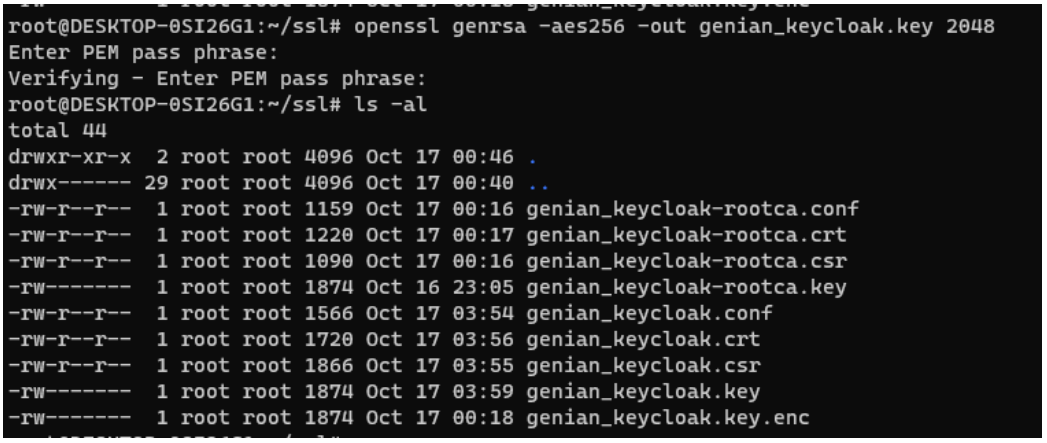
```
root@DESKTOP-0SI26G1:~# openssl x509 -text -in indienote-rootca.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = KR, O = Genian_keycloak_ms, CN = localhost:30004
        Validity
            Not Before: Oct 16 15:17:12 2023 GMT
            Not After : Oct 15 15:17:12 2024 GMT
        Subject: C = KR, O = Genian_keycloak_ms, CN = localhost:30004
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:b9:e4:1a:af:3c:30:bf:49:96:9e:a4:b1:9a:4b:
                    8d:44:42:ee:68:5b:33:00:59:2b:3e:43:a7:50:6e:
                    d6:03:d1:7c:90:1e:0c:71:af:67:b6:4b:a9:96:c2:
                    b3:82:b6:f4:0e:1b:6e:99:96:2d:2b:25:7d:dd:1d:
                    c0:df:6a:1e:f4:73:a8:6c:0a:43:46:c8:f0:45:55:
                    6c:54:f5:e7:14:fa:0e:af:4c:9e:12:f2:45:4e:14:
                    a2:5f:29:7e:e4:88:20:4e:d6:d5:34:9a:c3:20:8d:
                    3c:2f:81:aa:5e:86:f8:ca:79:d5:61:aa:36:8e:c1:
                    5a:aa:ed:94:b5:6c:92:64:0b:07:97:1f:87:4c:5b:
                    14:41:09:43:63:04:a5:9d:f6:28:8c:e4:d9:29:ed:
                    1e:fb:4e:cc:fd:d4:8e:a8:b6:5c:1b:b8:80:92:42:
                    07:0a:cd:71:05:d2:21:0d:77:1c:05:54:7a:26:7e:
                    fd:49:75:92:d7:e9:25:49:87:e7:a5:e1:8e:75:10:
                    6b:5b:7f:24:da:d8:72:5d:54:c6:48:4e:79:60:a4:
```

SSL 인증서 발급을 위한 개인키 생성

앞에서 생성한 Root CA 개인키로 서명한 SSL 인증서를 발급

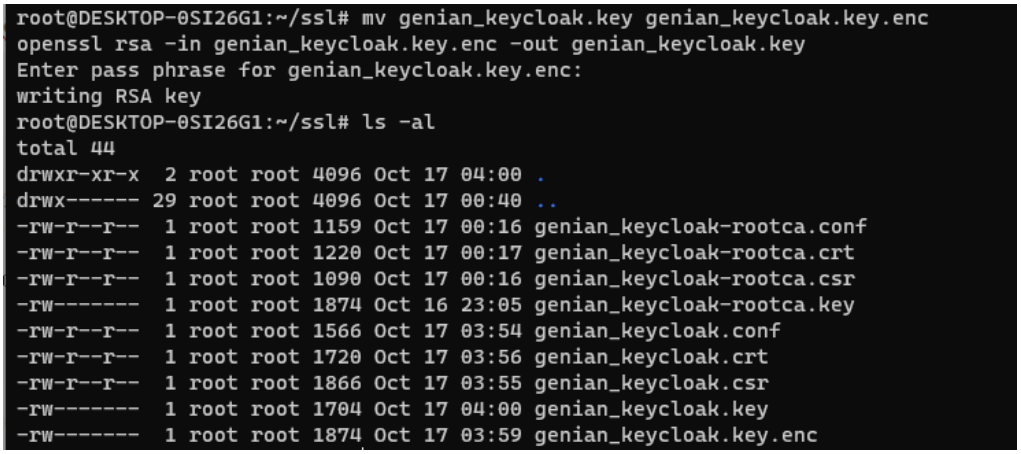
1. SSL 인증서에서 사용할 RSA Private Key 를 생성

```
openssl genrsa -aes256 -out genian_keycloak.key 2048
```



2. Key 에서 암호를 제거 (웹서버 같은데서 사용할 것으로 암호를 원하지 않을 때)

```
mv genian_keycloak.key genian_keycloak.key.enc
openssl rsa -in genian_keycloak.key.enc -out genian_keycloak.key
```



3. 개인키 유출 방지를 위해서 파일 권한을 600 으로 변경

```
chmod 600 genian_keycloak.key*
```

SSL 인증서 발급

1. CSR 생성을 위한 config 파일 genian_keycloak.conf 생성

```
[ req ]
default_bits           = 2048
default_md             = sha1
default_keyfile        = Genian_keycloak-rootca.key
```

```
distinguished_name      = req_distinguished_name
extensions              = v3_user
## 인증서 요청시에도 extension 이 들어가면 authorityKeyIdentifier 를 찾지 못해 에러가 나므로 막아둔다.
## req_extensions = v3_user

[ v3_user ]
# Extensions to add to a certificate request
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
#authorityKeyIdentifier = keyid,issuer
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
## SSL 용 확장키 필드
extendedKeyUsage = serverAuth,clientAuth
subjectAltName          = @alt_names

[alt_names]
## Subject AltName의 DNSName field 에 SSL Host 의 도메인 이름을 적어준다.
## 멀티 도메인일 경우 *.indienote.com 처럼 쓸 수 있다.
DNS.1   = localhost
IP.1 = 127.0.0.1
IP.2 = 221.151.133.217
## 공인 ip 주소 추가


[req_distinguished_name ]
countryName                     = Country Name (2 letter code)
countryName_default             = KR
countryName_min                 = 2
countryName_max                 = 2

# 회사명 입력
organizationName              = Organization Name (eg, company)
organizationName_default      = Genian_keycloak

# 부서 입력
organizationalUnitName          = Organizational Unit Name (eg, section)
organizationalUnitName_default  = Genian_keycloak SSL

# SSL 서비스할 domain 명 입력
commonName                      = Common Name (eg, your name or your server's hostname)
commonName_default              = localhost
commonName_max                  = 64
```

2. CSR 파일 생성

```
openssl req -new -key genian_keycloak.key -out genian_keycloak.csr -config genian_keycloak.conf
```

```
root@DESKTOP-0SI26G1:~/ssl# openssl req -new -key genian_keycloak.key -out genian_keycloak.csr -config genian_keycloak.c
onf
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [KR]:
Organization Name (eg, company) [Genian_keycloak]:
Organizational Unit Name (eg, section) [Genian_keycloak SSL]:
Common Name (eg, your name or your servers hostname) [localhost:30004]:
root@DESKTOP-0SI26G1:~/ssl# ls -al
total 40
drwxr-xr-x  2 root root 4096 Oct 17 00:44 .
drwx------ 29 root root 4096 Oct 17 00:40 ..
-rw-r--r--  1 root root 1159 Oct 17 00:16 genian_keycloak-rootca.conf
-rw-r--r--  1 root root 1220 Oct 17 00:17 genian_keycloak-rootca.crt
-rw-r--r--  1 root root 1090 Oct 17 00:16 genian_keycloak-rootca.csr
-rw-------  1 root root 1874 Oct 16 23:05 genian_keycloak-rootca.key
-rw-r--r--  1 root root 1567 Oct 17 00:40 genian_keycloak.conf
-rw-r--r--  1 root root  993 Oct 17 00:44 genian_keycloak.csr
-rw-------  1 root root 1704 Oct 17 00:23 genian_keycloak.key
-rw-------  1 root root 1874 Oct 17 00:18 genian_keycloak.key.enc
```

3. n년 genian_keycloak 용 SSL 인증서를 Root CA 의 개인키로 서명하여 발급

```
openssl x509 -req -days 365 -extensions v3_user -in genian_keycloak.csr \
-CA genian_keycloak-rootca.crt -CAcreateserial \
-CAkey  genian_keycloak-rootca.key \
-out genian_keycloak.crt -extfile genian_keycloak.conf
```

```
root@DESKTOP-0SI26G1:~/ssl# openssl x509 -req -days 365 -extensions v3_user -in genian_keycloak.csr -CA genian_keycloak-
rootca.crt -CAcreateserial -CAkey  genian_keycloak-rootca.key -out genian_keycloak.crt -extfile genian_keycloak.conf
Certificate request self-signature ok
subject=C = KR, O = Genian_keycloak, OU = Genian_keycloak SSL, CN = localhost:30004
Enter pass phrase for genian_keycloak-rootca.key:
root@DESKTOP-0SI26G1:~/ssl# ls -al
total 44
drwxr-xr-x  2 root root 4096 Oct 17 00:46 .
drwx------ 29 root root 4096 Oct 17 00:40 ..
-rw-r--r--  1 root root 1159 Oct 17 00:16 genian_keycloak-rootca.conf
-rw-r--r--  1 root root 1220 Oct 17 00:17 genian_keycloak-rootca.crt
-rw-r--r--  1 root root 1090 Oct 17 00:16 genian_keycloak-rootca.csr
-rw-------  1 root root 1874 Oct 16 23:05 genian_keycloak-rootca.key
-rw-r--r--  1 root root 1567 Oct 17 00:40 genian_keycloak.conf
-rw-r--r--  1 root root 1371 Oct 17 00:47 genian_keycloak.crt
-rw-r--r--  1 root root  993 Oct 17 00:44 genian_keycloak.csr
-rw-------  1 root root 1704 Oct 17 00:23 genian_keycloak.key
-rw-------  1 root root 1874 Oct 17 00:18 genian_keycloak.key.enc
```

4. 생성된 SSL 인증서 확인

```
openssl x509 -text -in genian_keycloak.crt
```

```
root@DESKTOP-0SI26G1:~/ssl# openssl x509 -text -in genian_keycloak.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            5b:f6:fb:c8:a8:35:13:c4:39:7d:88:c8:1e:62:60:1d:e2:40:68:af
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = KR, O = Genian_keycloak_ms, CN = localhost:30004
        Validity
            Not Before: Oct 16 15:47:10 2023 GMT
            Not After : Oct 15 15:47:10 2024 GMT
        Subject: C = KR, O = Genian_keycloak, OU = Genian_keycloak SSL, CN = localhost:30004
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:b0:4c:a6:03:06:21:07:0f:10:b1:18:b9:7a:91:
                    5a:5d:17:71:96:7a:fa:cf:19:c2:79:0a:a0:0a:bc:
                    7b:17:e3:67:f6:8e:cf:67:1a:1b:aa:b3:21:db:cf:
                    21:9f:d4:79:10:d8:14:23:41:e1:32:49:1d:3a:40:
                    00:c3:03:7a:40:fc:b2:dc:92:17:bd:9d:a3:88:2b:
                    e5:3a:b0:23:b4:18:82:7d:c0:05:5f:26:52:6d:c0:
                    19:32:67:fb:ed:27:6f:11:ad:8c:ac:ca:1c:97:df:
                    00:45:e5:5d:fb:15:09:08:56:7e:79:d7:23:77:a0:
                    ba:a1:8b:2c:c0:c1:49:d3:ac:29:6b:ea:c3:52:fa:
                    4e:4c:71:49:44:2c:27:67:52:2d:cc:0c:4f:31:9e:
                    8d:ce:7f:83:2f:f4:ae:f7:18:74:66:d4:73:ff:49:
                    06:c7:55:db:df:78:37:db:5c:63:f8:8f:9f:97:a6:
                    8c:2a:74:10:22:cc:7a:d1:23:92:5f:0e:03:cb:9b:
```

CSR 의 이상유무 확인

```
openssl req -text -in genian_keycloak.csr
```

자체 서명된 인증서를 신뢰할 수 있는 인증 기관 저장소에 추가

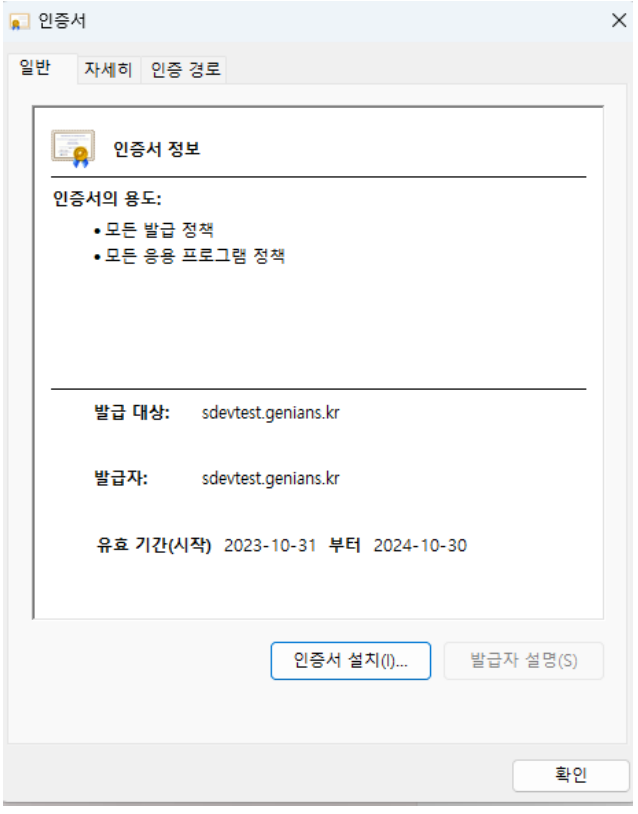
생성한 자체 root ca 인증서를 꼭 신뢰할 수 있는 인증 기관 저장소에 추가해야 한다.

```
sudo cp /path/to/your/certificate.crt /usr/local/share/ca-certificates/ #인증서는 Root CA 인증서를 신뢰할 수 있는 인증 기관 저장소에 추가해야함.
sudo update-ca-certificates
```

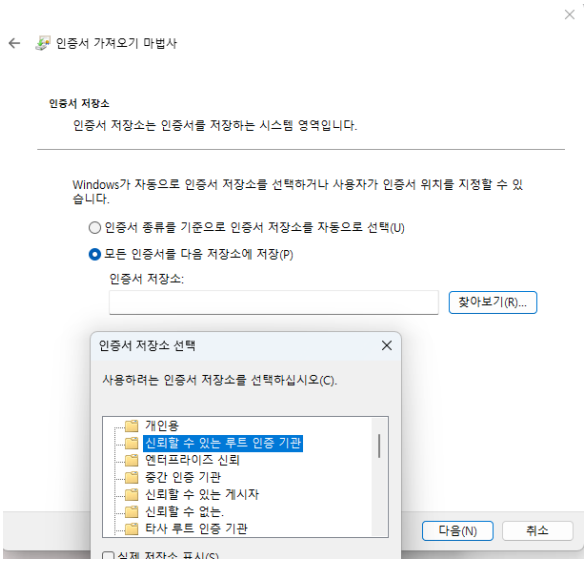
```
root@DESKTOP-0SI26G1:~/ssl#  cp /root/ssl/genian_keycloak.crt /usr/local/share/ca-certificates/
root@DESKTOP-0SI26G1:~/ssl# update-ca-certificates
Updating certificates in /etc/ssl/certs...
rehash: warning: skipping ca-certificates.crt,it does not contain exactly one certificate or CRL
1 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...

Adding debian:genian_keycloak.pem
done.
done.
```

웹 브라우저에 신뢰할 수 있는 기관에 인증서 등록

우분투 환경에서 root ca 인증서를 신뢰할 수 있는 기관 저장소에 저장했다. 하지만 로컬 윈도우에서 웹 브라우저에 인증서를 적용하기 위해서는 원격 우분투에 있는 인증서를 로컬 윈도우에 설치를 진행한다. (RootCA 인증서와 ssl 인증서 둘 다 설치해야함.)



하단의 인증서 설치를 누른다.

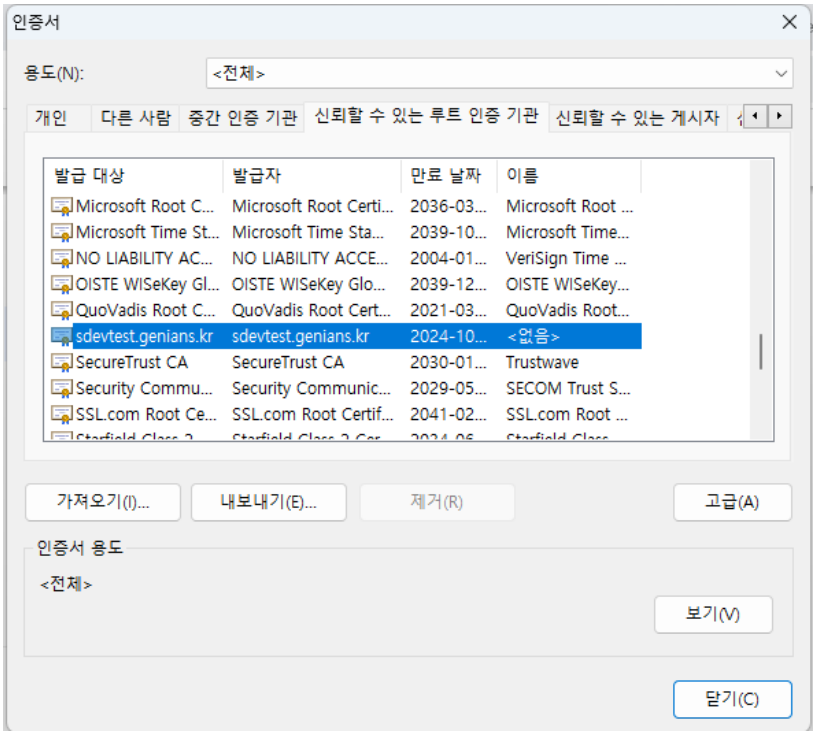


신뢰할 수 있는 루트 인증 기관에 등록하면 인증서 설치는 끝났다. Root CA 인증서를 신뢰할 수 있는 루트 인증 기관에 등록했기 때문에 ca 인증서로 발급한 ssl 인증서 또한 신뢰할 수 있는 인증서가 되었다.

로컬 윈도우에 인증서를 등록하고 나서, 실제 사이트에 인증서를 등록하기 전에 해야 할 단계가 남아 있다. 기존에 진짜 신뢰할 수 있는 기관에서 만든 인증서들은 일반적으로 웹 브라우저에 내장되어 있는 경우여서 자체 생성된 ca 인증서를 등록해주는 과정이 필요하다.

chrome 브라우저 기준으로 설명을 드리는 것으로 브라우저에 인증서 등록은 다음과 같다.

설정 > 개인 정보 보호 및 보안 > 보안 탭으로 들어가면 기기 인증서 관리가 있다. 여기에 등록을 하면된다.



웹 브라우저에 인증서를 등록하고 사이트 서버를 재시작하게 되면 아래 그림과 같이 자물쇠가 걸린 것을 확인할 수 있다.