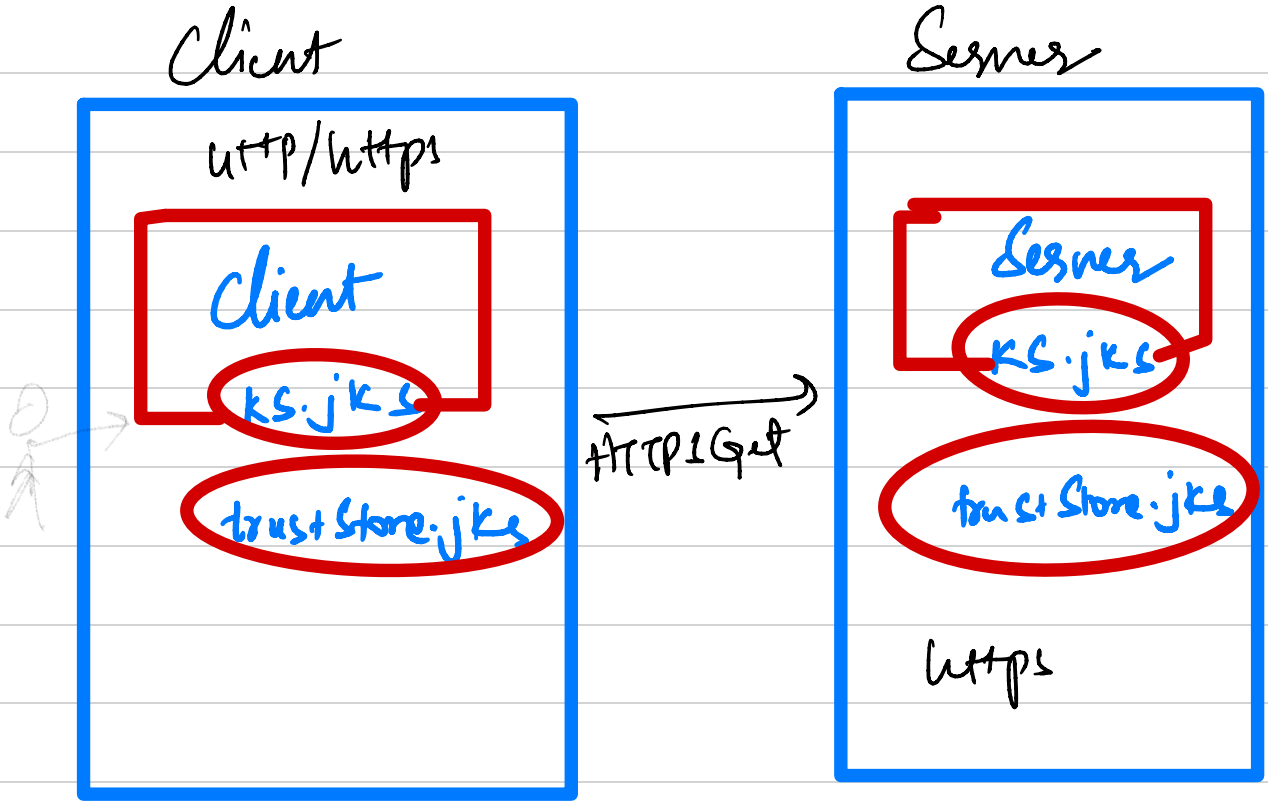
Mutual Authentication



- JKS :-
  Java KeyStore

Generate key files using Java tools & openssl.
(Server keystore)

```
keytool -genkey -alias bunny-server -keystore
server-keystore.jks -storetype jks -keypass password
-storepass password -keyalg RSA -deststoretype
pkcs12
```

Similarly for client keystore.

```
keytool -genkey -alias bunny-client -keystore
client-keystore.jks -storetype jks -keypass password
-storepass password -keyalg RSA -deststoretype
pkcs12
```

1) Now:-

Extract certificates from Trust stores

i) keytool -exportcert -alias bunny-server -keystore server-keystore.jks -file server-cert.cer -storepass password

Now we should see 3 files

Similarly do this for client.

ii) keytool -exportcert -alias bunny-client -keystore client-keystore.jks -file client-cert.cer -storepass password

2) In server-truststore we should have

certificate & public key of client

i) keytool -importcert -keystore server-keystore.jks -file client-cert.cer -alias bunny-client -storepass password -trustcacerts -deststoretype pkcs12

3) In client truststore we should have server's publickey & certificate.

i) keytool -importcert -keystore client-keystore.jks -file server-cert.cer -alias tommy-server -storepass password -trustcacerts -deststoretype pkcs12

Cons of this approach:-

i) If multiple clients interact with the server we have do the same process for all the clients which is cumbersome!

fix:- Using Certificate Authority. (Another approach).

Approach: 2 way SSL

Approach 2 Using Openssl Approach with CA Approach

1) Create Certificate Authority

2) Create certs for server

3) Create certificates for clients.

Step 1:- openssl genrsa -out ca-key.pem 4096

↳ Creates private key

↓

Generate public key certificate with pk

openssl req -new -x509 -sha256 -days 1000

-key ca-key.pem -out ca-cert.pem

Server Certificate

↳ Create Certificate Signing Req
↳ CA will approve/Sign & send back to us.

i) Create private key for server

openssl genrsa -out server-key.pem 4096

ii) Create certificate Signing Request. with same pk

openssl req -new -sha256 -key server-key.pem -out server-csr.csr

↳ output is certificate signing request

iii) Create Server certificates

↳ Generally prereqs for domain name / alternate name for servers.

```
touch server-extfile.cnf
echo "subjectAltName = DNS:*.bunny-servers.com
    >> server-extfile.conf
```

```
openssl x509 -req -sha256 -days 1000
-in server-csr.csr -CA ca-cert.pem -CAkey
ca-key.pem -out server-cert.pem -extfile
server-extfile.cnf -CAcreateserial
```

iv) Repeat the same for the client as well.

```
openssl genrsa -out client-key.pem 4096
```

```
openssl req -new -sha256 -key client-key.pem
-out client-csr.csr

touch client-extfile.cnf
echo "subjectAltName = DNS:*.bunny-client.com

openssl x509 -req -sha256 -days 1000 -in
client-csr.csr -CA ca-cert.pem -CAkey ca-key.pem
-out client-cert.pem -extfile client-extfile.cnf
-CAcreateserial
```

(V) Now combine ca-cert server cert server-key

```
cat ca-cert.pem server-cert.pem server-key.pem
    > server-combined.pem
```

vi) Now create server cert.

openssl pkcs12 -export -in server-combined.pem
-out server-cert.p12 -name bunny-server

prompts for password

vii) Convert the .p12 to server-keystore.jks

keytool -importkeystore -srckeystore server-cert.p12
-srcstoretype pkcs12 -destkeystore server-keystore
.jks

prompts for password

vii) Now Generate keystore for the client

cat ca-cert.pem client-cert.pem client-key.pem > client-combined.pem

openssl pkcs12 -export -in client-combined.pem -out client-cert.p12 -name bunny-client.

prompts password

keytool -importkeystore -srckeystore client-cert.p12 -srcstoretype pkcs12 -destkeystore client-keystore.jks

prompts password

(iii) Import CA-cert to truststore

which could be shared b/w

Server & Client.

cp ca-cert.pem ca-cert.cer

keytool -importcert -keystore server-truststore.jks

-file ca-cert.cer -alias bunny-client -storepass

password -trustcacerts -deststoretype pkcs12

keytool -importcert -keystore client-truststore.jks

-file ca-cert.cer -alias bunny-server

-storepass password -trustcacerts -deststoretype

pkcs12