Programming Assignment 2

Yap Zhihan 1004570
Pan Feng 1003689

Authentication procedure (Before fix)

1. The server first has a pair of private and public keys.
2. The private key is only owned and known by the server.
3. Client will request for connection
4. Server will encrypt the message ("Hello, this is SecStore!") with private key and send it to client
5. Client then ask for certificate signed by CA
6. Server sent certificate.
7. Client decrypts the signed certificate, extracts the public key, and uses this public key to compute the message and checks if the result is correct.
8. Connection is established if the check comes true, close otherwise.

Vulnerability and problem (Playback Attack)

1. Cannot verify that the server is live (susceptible to the playback attack)
2. Someone can pretend to be the server.
    a. This is because the server will always send the same message ("Hello, this is SecStore!") to the client in step 4
    b. Even if an attacker does not know the private key of the server, he can send this encrypted message to the client and follow by step 5 to 8

Outcome

1. The client will think that the fake server is the real server
2. connect to the fake server
3. Send the files to the fake server

Fix

1. The client generates a nonce as the message to be sent by server in step 4, and sends it to the server when requesting the signed message.
2. The server then replies with its cert, along with the nonce, encrypted with the server's private key, rather than sending a fixed message.
3. The client then verifies the cert and extracts the server's public key and uses it to decrypt the nonce. To ensure that the server is live and has the private key.

4. Since only the server has the private key, the identity of the server is then confirmed.

Implementation

NonceGenerator in package AuthUtils

generates nonce with given nonce length

ClientWithAuthProtocol.java:

line 56: generate nonce

line 88: verifying decrypted message.

ServerWithAuthProtocol.java

line 63: encrypt received message

Authentication Protocol

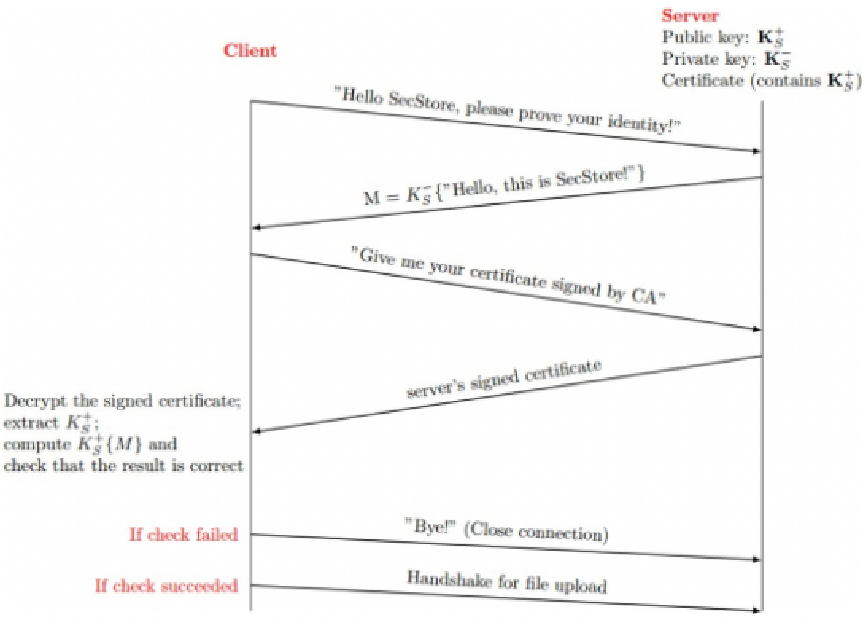
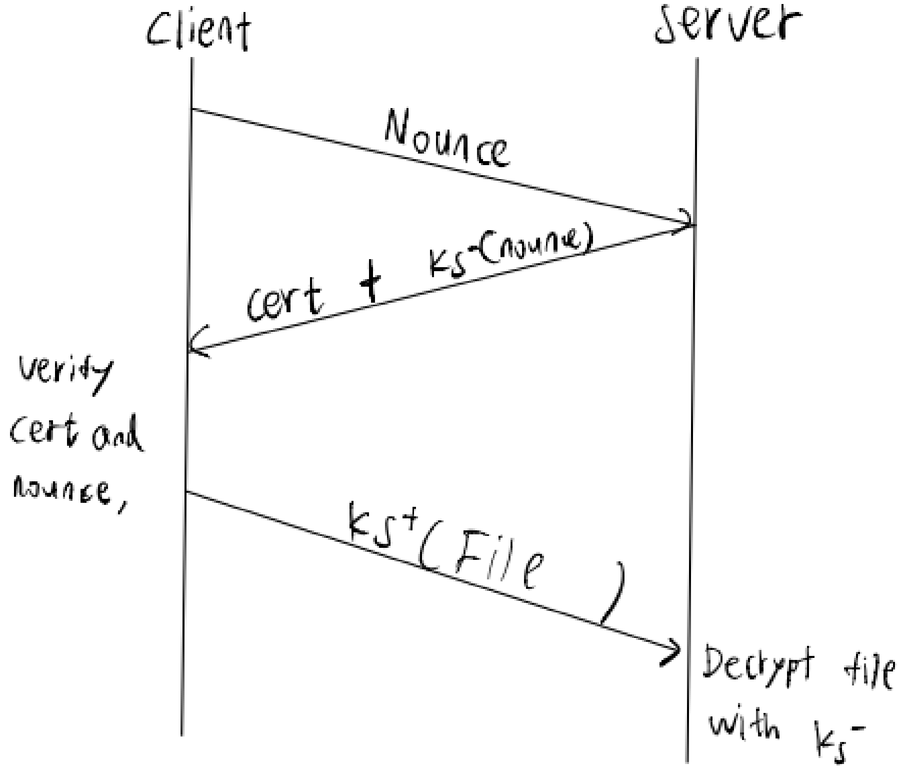

Server
Public key: $\mathbf{K}_S^+$
Private key: $\mathbf{K}_S^-$
Certificate (contains $\mathbf{K}_S^+$)
Client
"Hello SecStore, please prove your identity!"
M = K_S^-{"Hello, this is SecStore!"}
"Give me your certificate signed by CA"
server's signed certificate
Decrypt the signed certificate;
extract K_S^+;
compute $K_S^+\{M\}$ and
check that the result is correct
If check failed
"Bye!" (Close connection)
If check succeeded
Handshake for file upload

Fig. 1: Basis of Authentication Protocol

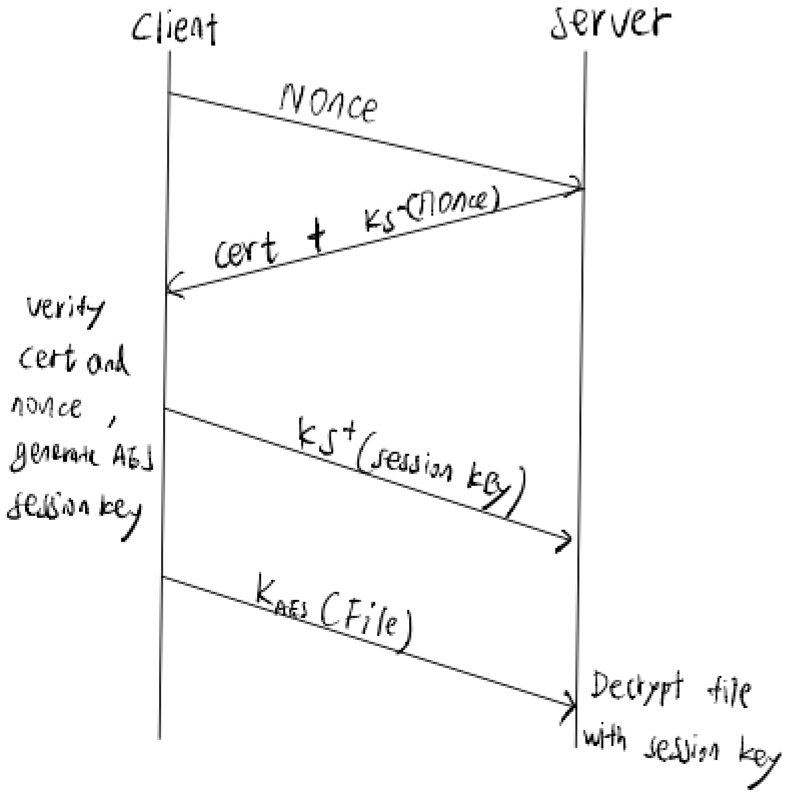
CP1

CP1

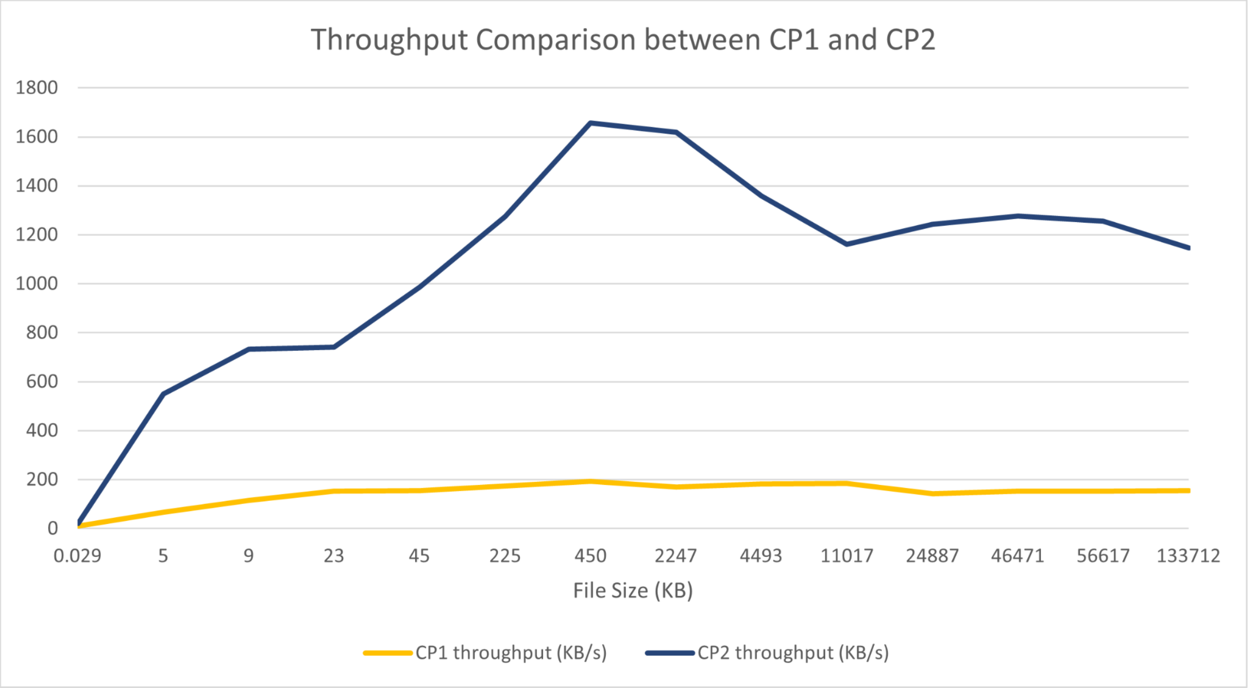

Client
Server
Nounce
cert + ks-(nounce)
verify
cert and
nounce,
ks+(File)
Decrypt file
with ks-

CP2

CP2


Client
Server
Nonce
cert + Ks-(nonce)
verify cert and nonce, generate AES session key
Ks+(session key)
KAES(File)
Decrypt file with session key

Plot of throughput with CP1 and CP2 against file size.


Throughput Comparison between CP1 and CP2
1800
1600
1400
1200
1000
800
600
400
200
0
0.029
5
9
23
45
225
450
2247
4493
11017
24887
46471
56617
133712
File Size (KB)
CP1 throughput (KB/s)
CP2 throughput (KB/s)