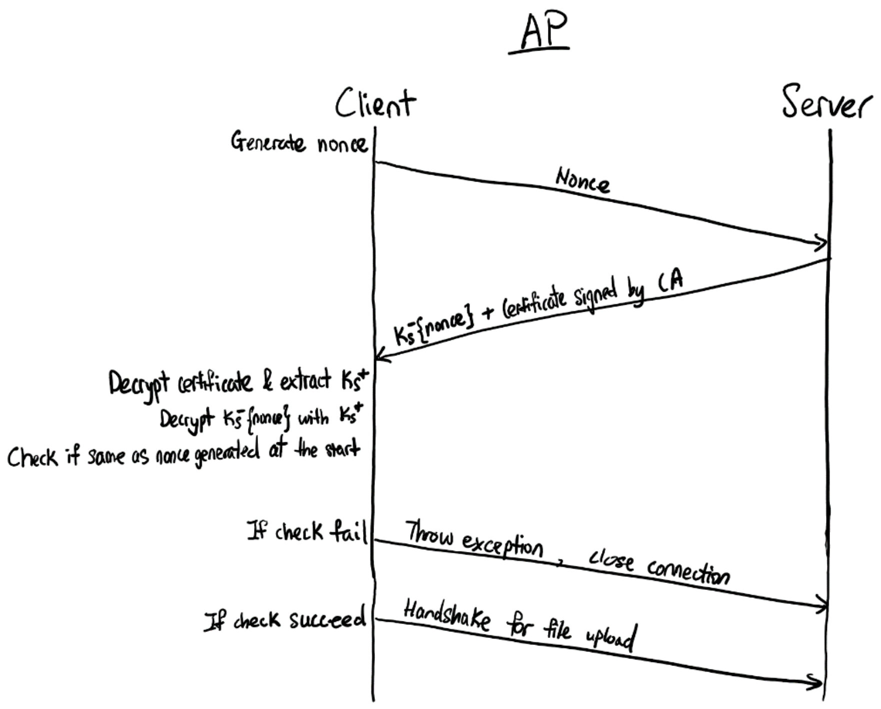
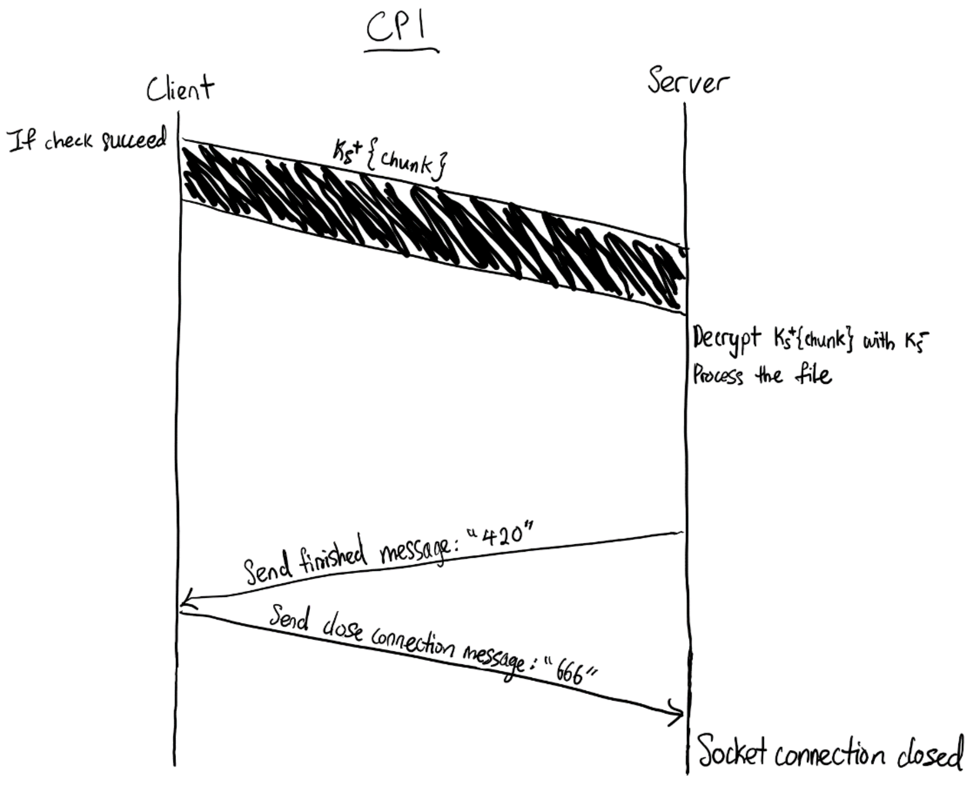
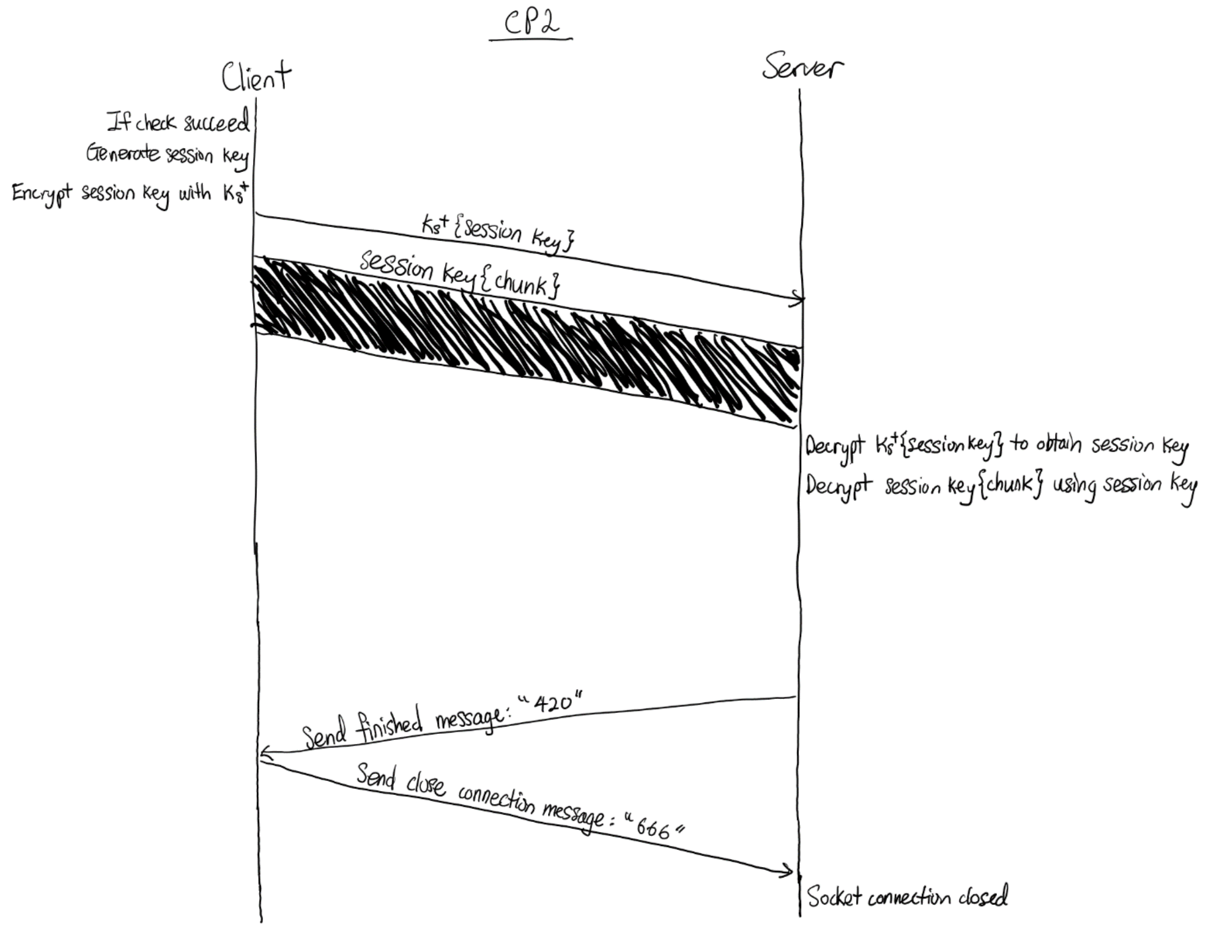
CSE Programming Assignment 2

(BY: Koh Ting Yew, Ng Pei Shi Doreen)

Specifications of the protocols:


AP
Client
Server
Generate nonce
Nonce
$K_S^-\{nonce\}$ + certificate signed by CA
Decrypt certificate & extract K_S^+
Decrypt $K_S^-\{nonce\}$ with K_S^+
Check if same as nonce generated at the start
If check fail
Throw exception, close connection
If check succeed
Handshake for file upload


CP1
Client
Server
If check succeed
$K_S^+\{chunk\}$
Decrypt $K_S^+\{chunk\}$ with K_S^-
Process the file
Send finished message: "420"
Send close connection message: "666"
Socket connection closed


CP2
Client
Server
If check succeed
Generate session key
Encrypt session key with Ks+
Ks+{session key}
Session key{chunk}
Decrypt Ks+{session key} to obtain session key
Decrypt session key{chunk} using session key
Send finished message: "420"
Send close connection message: "666"
Socket connection closed

The issue with Figure 1:

The protocol in figure 1 is susceptible to a playback attack. An attacker can simply record and playback the message, M = Ks-{"Hello, this is SecStore!"} and the server's signed certificate. The client will not know whether this message certificate came from the attacker or the server. The client will then extract the Ks+ and use it to compute Ks+{M}, and perform the check. The client will then see that the check has succeeded and begin the handshake for file upload. Thinking that it is the server, if client uploads unencrypted data to the attacker, this poses a serious security issue. Even if client uploads encrypted data which the attacker cannot decipher, it is still undesirable as the files are being sent to an unintended destination, and the client is fooled into thinking that he/she has sent the file to the client.

The fix:

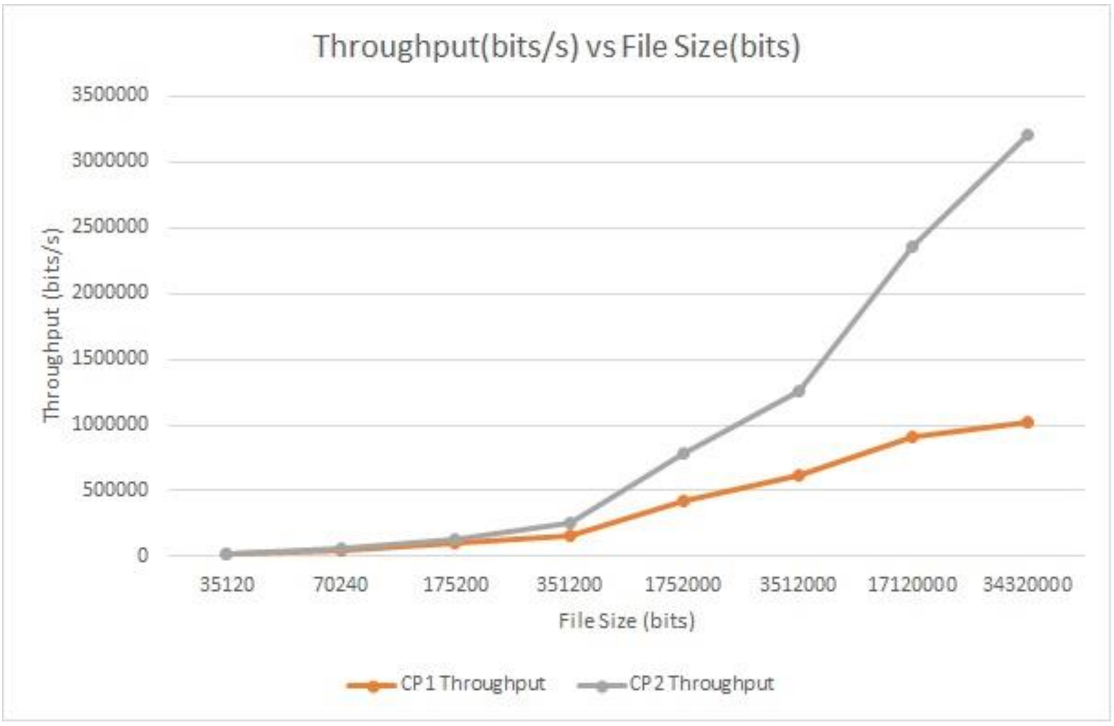
C: client

S: server

- C should send a nonce over to S.
- S replies with Ks-(nonce), its cert signed by CA, and maybe some message (can be plaintext or encrypted with Ks-).
- C obtains CA's public key, use it to verify S's certificate, then extracts Ks+.

- C uses Ks+ to decrypt Ks-(nonce) to get nonce, and compares it with what it sends earlier to S. If they are both the same, then there's no dirty work here. If they are not the same, it means there is an attacker and the connection will be closed immediately.
- C can trust the accompanying plain text sent by S. Or had S sent over encrypted text, then C can decrypt the text with Ks+, but at least now it can trust the contents of the decryption.
- C can proceed with the handshake, sending data encrypted with Ks+ to S.

Plots of achieved data throughput of CPI1 and CPI2 against a range of file sizes:


Throughput(bits/s) vs File Size(bits)
Throughput (bits/s)
File Size (bits)
CP1 Throughput
CP2 Throughput