50.005 Programming Assignment 2 Report

Victoria Yong 1004455, Lim Hng Yi 1004289

Overview of Protocols

Authentication Protocol (AP)

The authentication handshake protocol is implemented using asymmetric key cryptography. In our implementation, we tackle the issue of susceptibility to replay attacks by introducing a nonce (this is further explained in our Answer section of the report) . This protocol allows the client to verify the identity of the server before initiating a connection.

Confidentiality Protocol 1 (CP1)

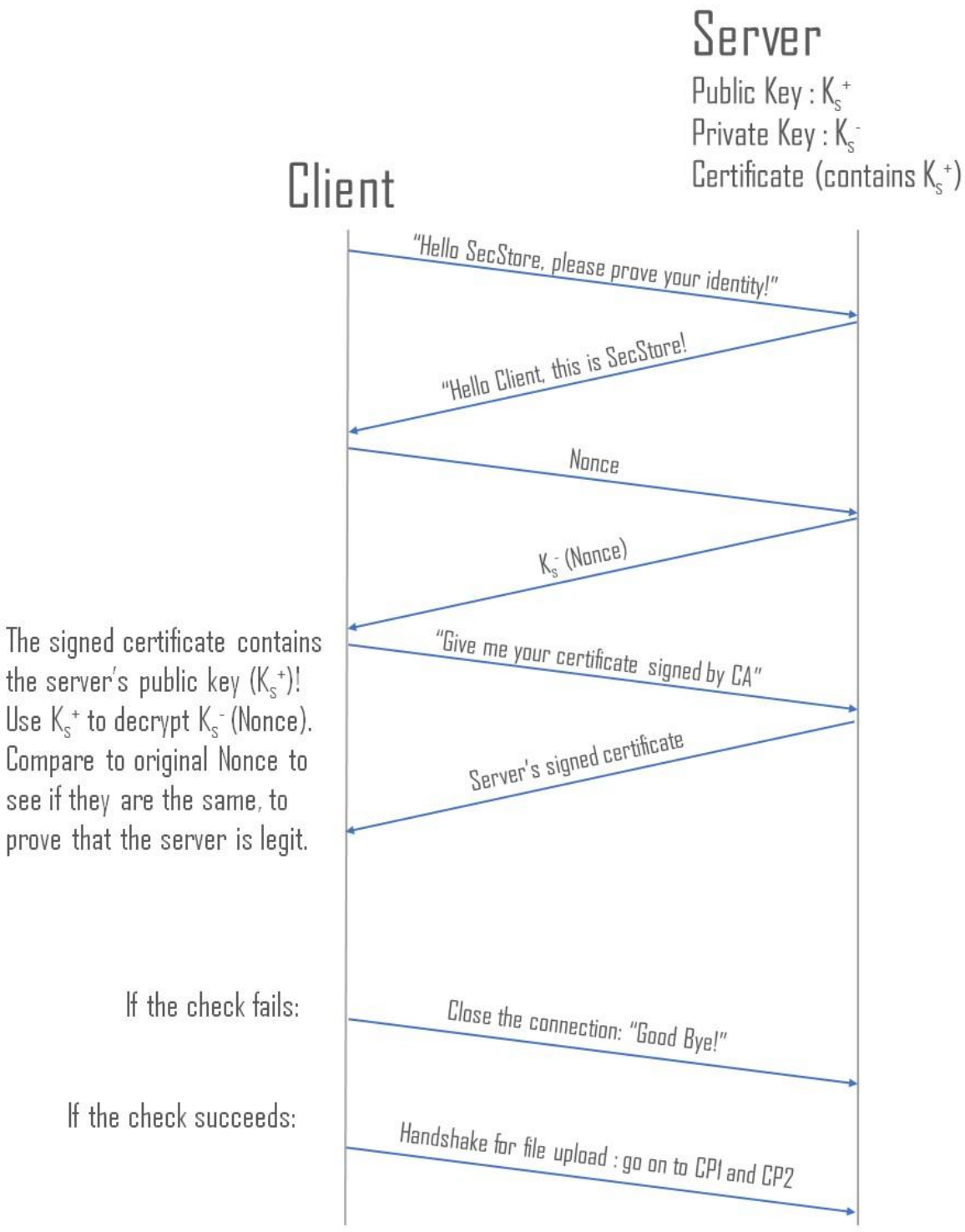
CP1 is a data transfer protocol, implemented through asymmetric key cryptography. It encrypts data using RSA. Files are encrypted by the client using the server's public key. The server then decrypts the file using its private key. In general, CP1 has a slower performance compared to CP2.

Confidentiality Protocol 2 (CP2)

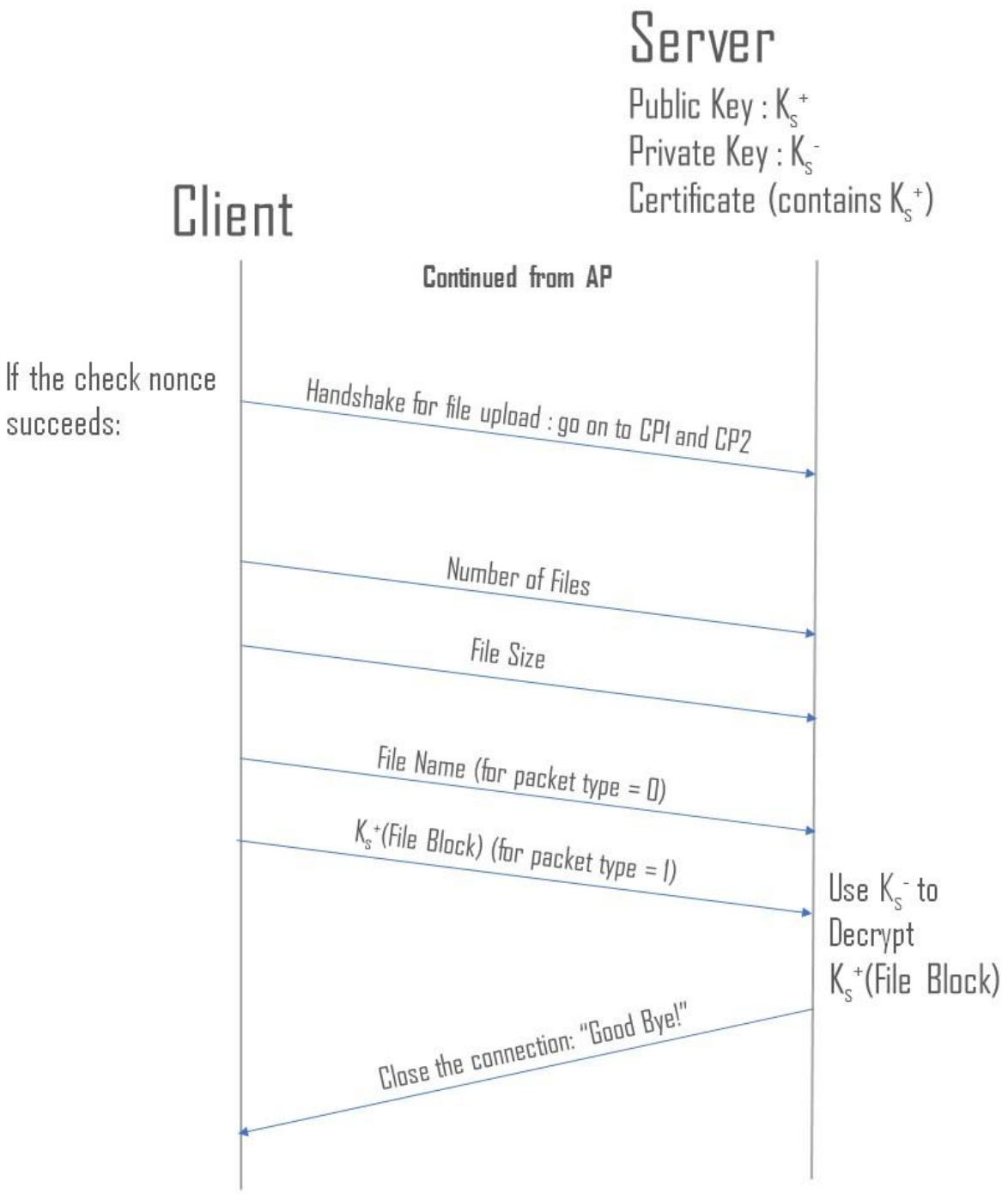
CP2 is a data transfer protocol, implemented through symmetric key cryptography using RSA, with a shared session key between the client and server. The client first generates a session key, encrypts it using the server's public key, and sends it to the server. Files sent to the server will then be encrypted using the session key. On the server side, the encrypted key received from the client is decrypted using the server's private key. The server now has access to the session key, and files sent by the client can be decrypted using the session key.

Specifications for Protocols:

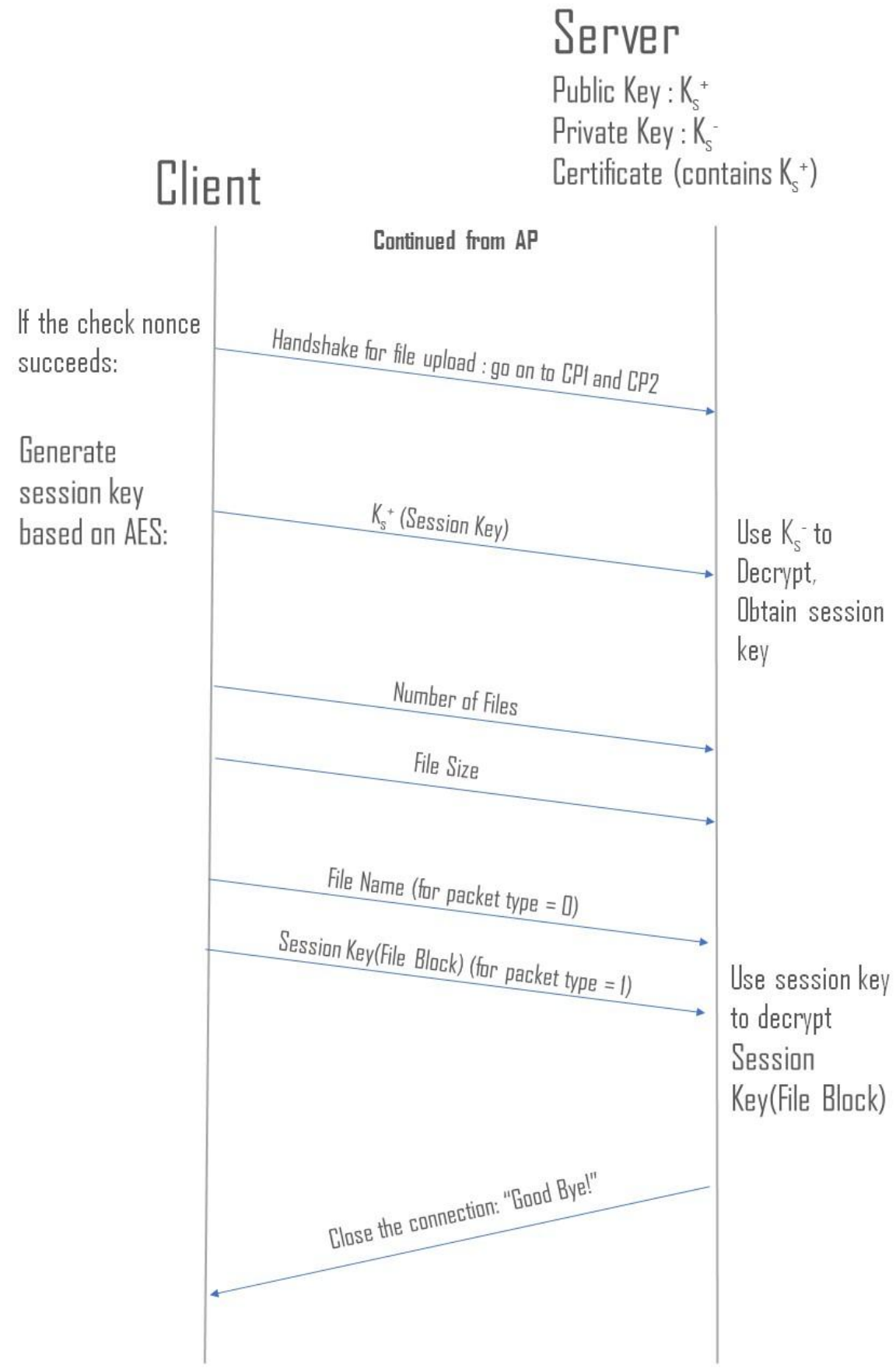
AP



CP1


Server
Public Key : K_s^+
Private Key : K_s^-
Certificate (contains K_s^+)
Client
Continued from AP
If the check nonce succeeds:
Handshake for file upload : go on to CP1 and CP2
Number of Files
File Size
File Name (for packet type = 0)
K_s^+(File Block) (for packet type = 1)
Use K_s^- to Decrypt K_s^+(File Block)
Close the connection: "Good Bye!"

CP2


Server
Public Key : K_s^+
Private Key : K_s^-
Certificate (contains K_s^+)
Client
Continued from AP
If the check nonce succeeds:
Handshake for file upload : go on to CP1 and CP2
Generate session key based on AES:
K_s^+ (Session Key)
Use K_s^- to Decrypt, Obtain session key
Number of Files
File Size
File Name (for packet type = 0)
Session Key(File Block) (for packet type = 1)
Use session key to decrypt Session Key(File Block)
Close the connection: "Good Bye!"

Answers

Question:

Answer:

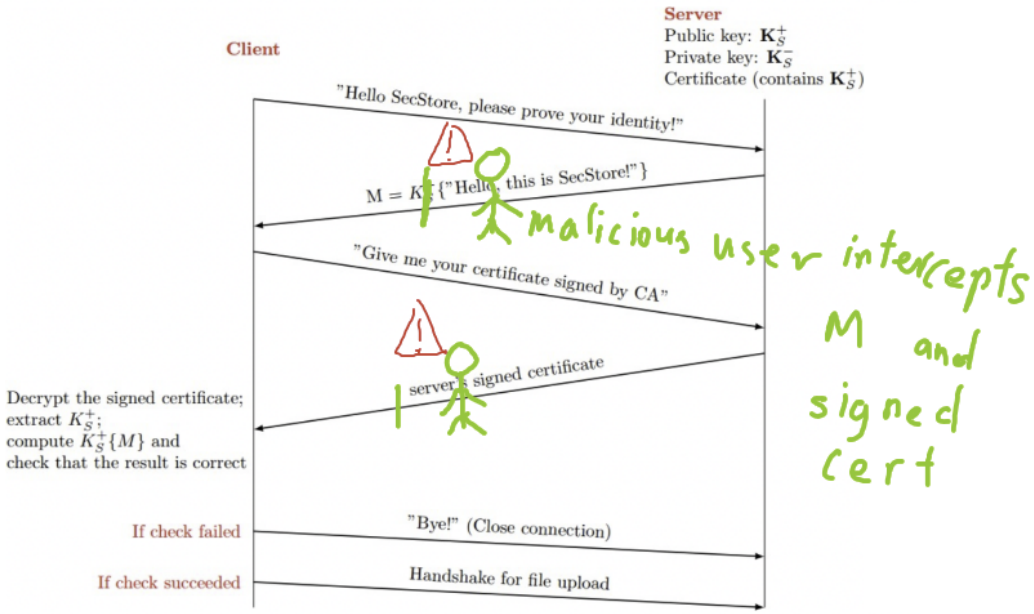


Fig. 1: Basis of Authentication Protocol

The original protocol does not prevent replay attacks. Malicious users can intercept data, as shown in the diagram above, and send it to the client while posing as the server thereby deceiving the client. The client is unable to detect whether the message is coming from the malicious user or the server.

In order to tackle this, we introduced a nonce, which is a one-time generated random number, into our AP. The nonce is generated by the client and then sent to the server in plaintext. Afterwards, the server will encrypt the nonce with its own private key and send it back to the client. The client can then use the server's public key to decrypt the nonce and check whether it is the same nonce as the one previously generated. If the nonce matches, the client has successfully verified the identity of the server!

Data

We tested CP1 and CP2 by sending one file to the server in a single connection, varying the file size and recording down the time taken for the client to upload the file, and the time taken for the server to download and decrypt the file.

Input File Information:

File Name	100.txt	200.txt	500.txt	1000.txt	5000.txt	10000.txt	50000.txt
File Size (bytes)	4600	9200	23000	46 000	230 000	460 000	2 300 000
No. of Lines	100	200	500	1000	5000	10 000	50 000

File Name	100000.txt	1M.txt	10M.txt	bakalofi.mp3	sample.bin	impmarch.wav	egg.png
File Size (bytes)	4 600 000	45999998	459999998	3783085			
No. of Lines	100 000	1 000 000	10 000 000	-	-	-	-

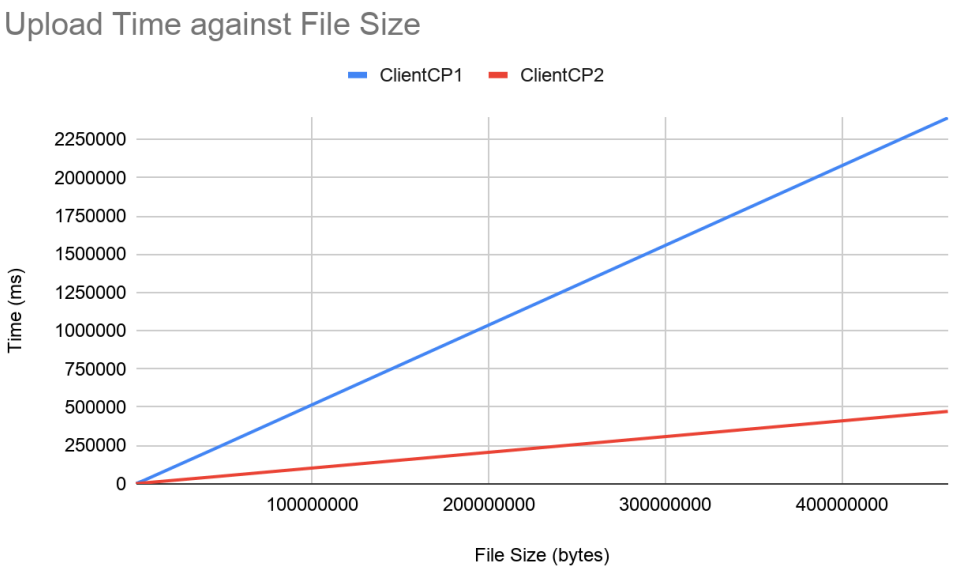
Data Collected:

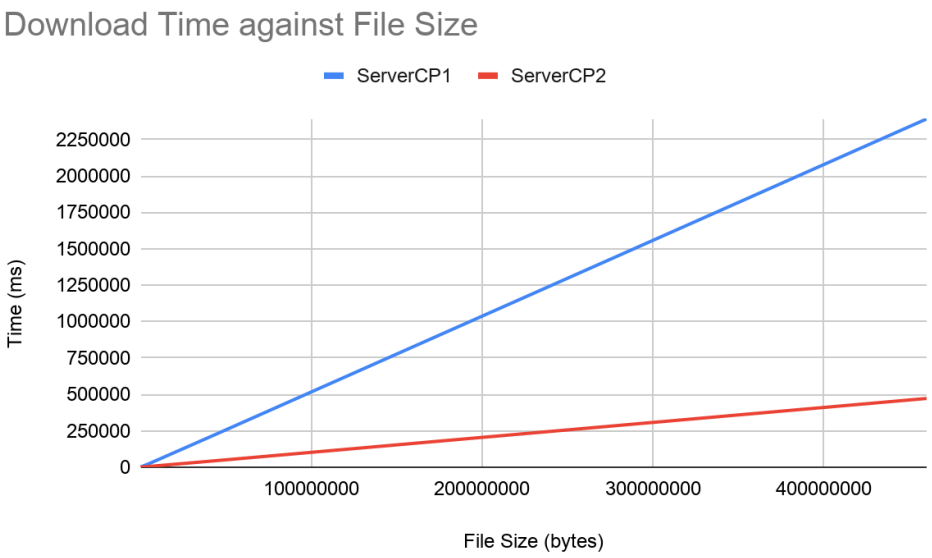
		Upload Time (ms)	
File Name	File Size (bytes)	ClientCP1	ClientCP2
100.txt	4600	79.6806	31.0337
200.txt	9200	130.0481	43.224199
500.txt	23000	218.1016	63.967099
1000.txt	46000	336.9155	99.504899
5000.txt	230000	1336.108	309.2248
10000.txt	460000	2579.752	826.1418
50000.txt	2300000	11507.12	2655.477
100000.txt	4600000	21895.59	4992.278
1M.txt	45999998	235553.5752	46794.0784
10M.txt	459999998	2391776.567	472408.8017
bakalofi.mp3	3783085	18557.7617	4054.340001
sample.bin	252544	1399.9727	313.825001

impmarch.wav	2646044	13106.5369	2758.059401
egg.png	408260	2176.456699	482.0484

		Download Time (ms)	
File Name	File Size (bytes)	ServerCP1	ServerCP2
100.txt	4600	65.6751	8.4343
200.txt	9200	114.2793	14.627401
500.txt	23000	201.8694	37.406801
1000.txt	46000	320.9118	74.369501
5000.txt	230000	1312.927	285.4581
10000.txt	460000	2563.252	802.4842
50000.txt	2300000	11490.22	2630.306
100000.txt	4600000	21874.87	4968.532
1M.txt	45999998	235532.023	46763.555
10M.txt	459999998	2391688.005	472382.48
bakalofi.mp3	3783085	18539.7549	4028.9251
sample.bin	252544	1381.7274	292.7187
impmarch.wav	2646044	13089.9113	2735.0356
egg.png	408260	2158.029101	458.4583

Graphs





Conclusions

At all file sizes, CP2 has better performance in both uploading and downloading files. This is because CP1 uses RSA to encrypt the whole file, while CP2 uses a session key based on AES. The session key makes use of a *symmetric key cryptography system,* which is much faster than RSA.

References

Github: **https://github.com/nugglet/50.005-CSE-Shell/tree/main/PA2**

*Note: The textfile 10M.txt is not included in the repo due to large file size, but it's the same as the other test textfiles, just with 10 million lines.