50.005 Programming Assignment 2

Jisa Mariam Zachariah (1003638)
Xie Han Keong (1003876)

Part (A): Overview of Protocols

SecStore Authentication Protocol (SSAP) is an authentication protocol that enables the client to authenticate the identity of SecStore before initiating a connection.

SecStore Confidentiality Protocol (SSCP) is the confidentiality protocol that encrypts the data sent to and from SecStore to protect it from being exposed to malicious attackers.

SecStore Transfer Protocol (SSTP) is the transfer protocol which is used for the communication between the client and SecStore.

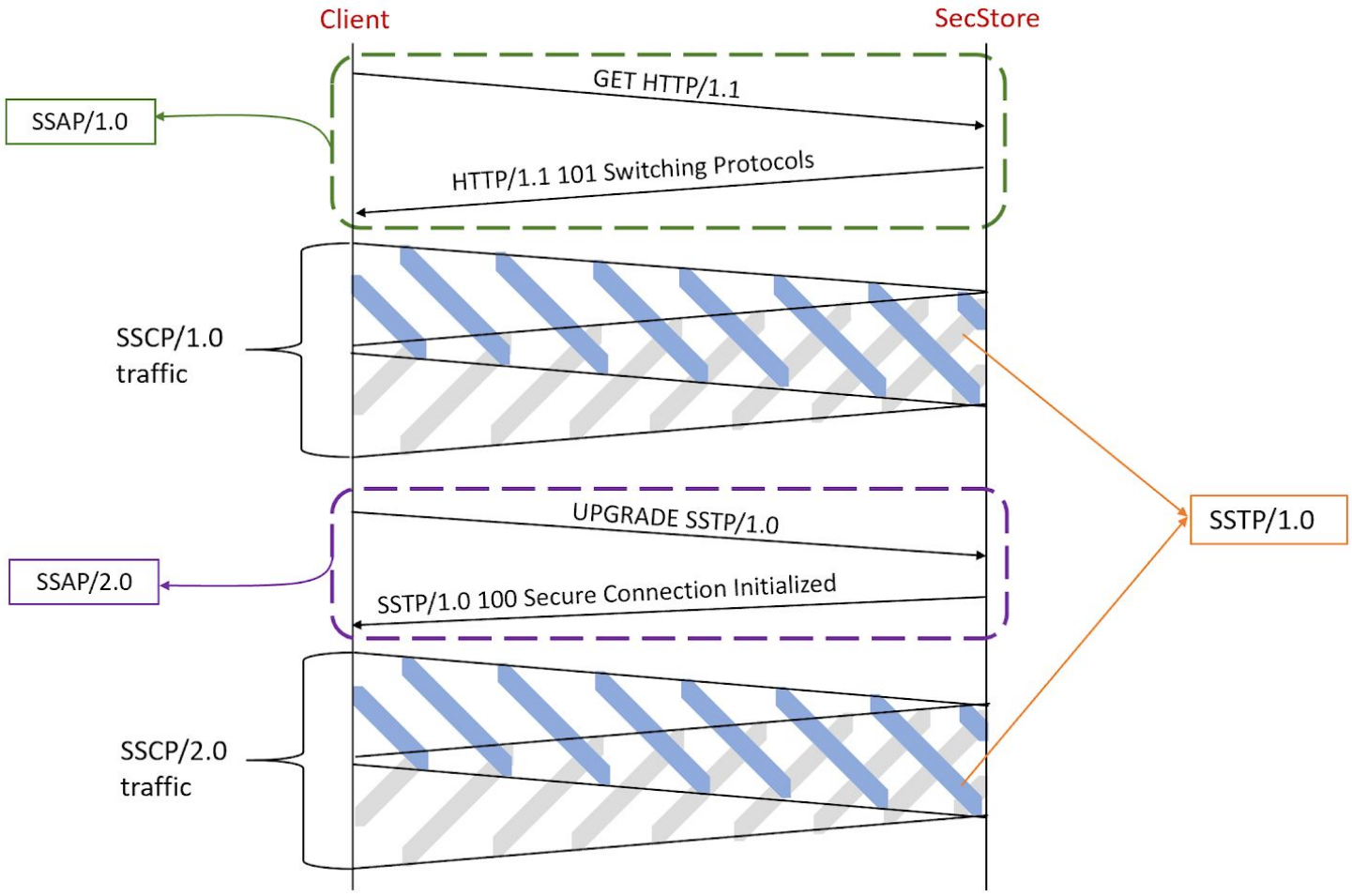


Figure 1.0: An overview of SecStore protocols

1. Specifications for SSAP/1.0 & SSAP/2.0

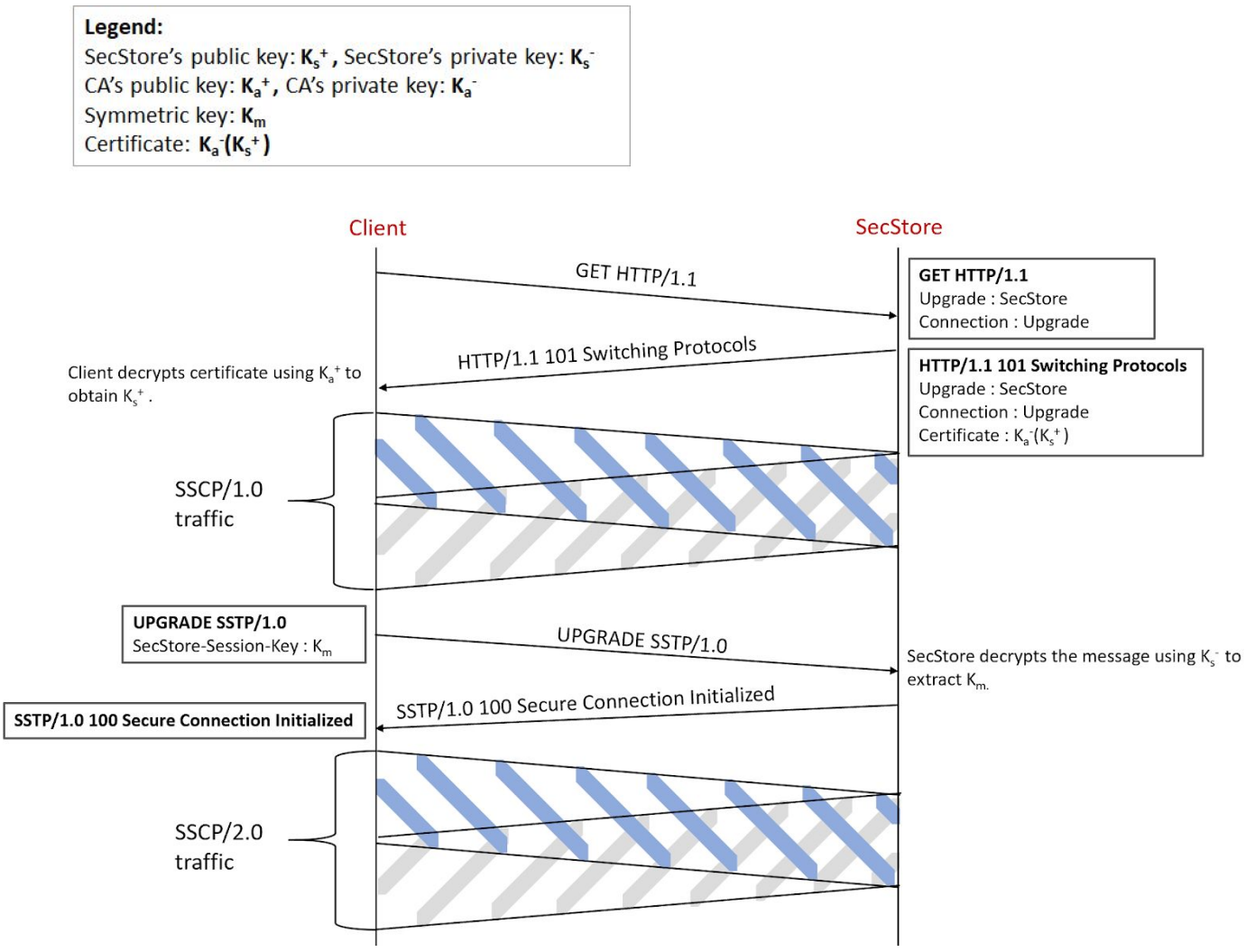


Figure 1.1: The SSAP authentication protocol

2. Specifications for SSCP/1.0 & SSCP/2.0

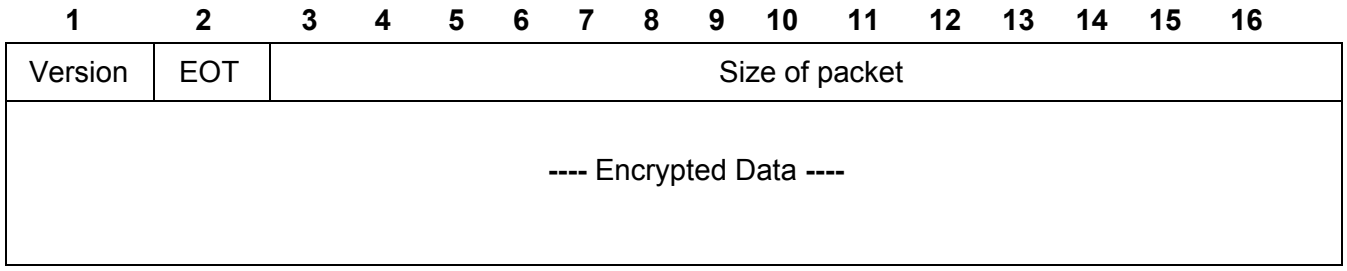


Figure 1.2: The SSCP confidentiality protocol

Bit 1 : Version
If bit 1 is 0, it means that the version is SSCP/1.0.
If bit 1 is 1, it means that the version is SSCP/2.0.

Bit 2 : End of transmission (EOT)
If bit 2 is 0, it means that it is not the last packet yet, i.e. there are more incoming packets.
If bit 2 is 1, it means that the end of transmission has been reached and there are no more incoming packets.

Bits 3 - 16 : Size of packet
The 14 least significant bits of the first two bytes – a number from 0 to $(2^{14} - 1) = 16383$, represents the size of the packet.

Byte 3 onwards : Encrypted Data
If the version is SSCP/1.0, the encrypted data will be decrypted using SecStore's public key, K_s^+.
If the version is SSCP/2.0, the encrypted data will be decrypted using the symmetric key, K_m.
For both versions, the encrypted data will terminate when the size of the packet has been reached.

3. Specifications for SSTP/1.0

Download Request	**DOWNLOAD SSTP/1.0** File-Name : File Name
	DOWNLOAD_PACKET SSTP/1.0 File-Name : File Name Packet-Number : Packet Number
Download Response	**SSTP/1.0 202 File Not Found** File-Name : File Name
	SSTP/1.0 200 Download Starting File-Name : File Name File-Size : File Size File-Format : File Format Total-Packets : Total Packets
	SSTP/1.0 201 Downloading Packet Packet-Number : Packet Number Packet-Hash : Packet Hash Packet-Size : Packet Size Packet Data
Upload Request	**UPLOAD SSTP/1.0** File-Name : File Name File-Size : File Size File-Format : File Format Total-Packets : Total Packets
	UPLOAD_PACKET SSTP/1.0 File-Name : File Name Packet-Number : Packet Number Packet-Hash : Packet Hash Packet-Size : Packet Size Packet Data
Upload Response	**SSTP/1.0 303 File Already Exists**
	SSTP/1.0 300 Proceed With Upload
	SSTP/1.0 301 Packet Upload OK File-Name : File Name Packet-Number : Packet Number
	SSTP/1.0 302 Packet Upload Failed File-Name : File Name Packet-Number : Packet Number

Delete Request	**DELETE SSTP/1.0** File-Name : File Name
Delete Response	**SSTP/1.0 202 File Not Found**
	SSTP/1.0 400 File Deleted File-Name : File Name

Part (B): Question and Answer

Fig. 1 below gives the basis of a possible protocol. However, there's one problem with the. What is the problem? Explain it in your handout for submission, and give a fix for the problem.

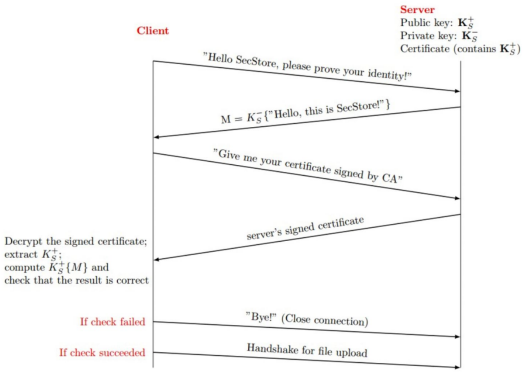


Fig. 1: Basis of Authentication Protocol

One problem with this authentication protocol is that there may be a replay attack. Since the closing handshake of this protocol is just a fixed message, it is possible for malicious users to replay that same message back to any client attempting to connect to the server. This allows hackers to pose as the server and complete the authentication process with a client while being undetected, since the client will not be able to differentiate whether the message is coming from the server or the attacker. As such, authentication is compromised.

A fix for this problem is to use a nonce, which is a 1-time generated number. The client will initiate the communication with the server by sending a nonce to the server. It is in plaintext. Upon receiving it, the server will send back the nonce which will be encrypted with its private key, together with its certificate. When the client decrypts the certificate with CA's public key, he will get the server's public key. Using the server's public key, the client will then decrypt the encrypted Nonce and if the Nonce is the same as the one the client had sent to the server previously, the identity of the server can be trusted. Since the nonce changes every time a new connection is established, the attacker will no longer be able to imitate the server by resending old Nonces, preventing any replay attack.

However, without encrypting the plaintext, the connection is still subject to eavesdropping. Attackers may listen for the packets being exchanged and exploit any sensitive information within it. As a result, confidentiality is breached. Eavesdropping can be circumvented by encrypting the plaintext. A symmetric key, also known as a session key, should be used instead of the server's public key as it is more efficient at encrypting long messages. One way to do it is as follows. After authenticating the server's identity, the client generates a symmetric key. The client will then encrypt the symmetric key using the server's public key and send it to the server. Upon receiving the message, the server will decrypt it using its private key to extract the symmetric key. The server will send back an acknowledgement message which is encrypted by

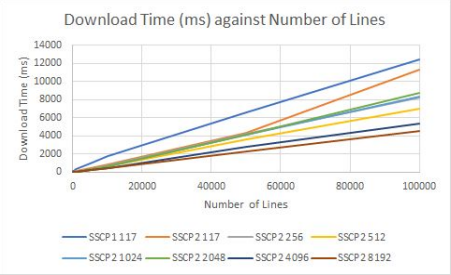
the symmetric key. If the transfer of the acknowledgement message is successful, the client and server will continue their transmission of messages using the symmetric key.

Since only the server knows its private key, no third party will be able to intercept the transfer of message and obtain the symmetric key shared by the client. As a result, the symmetric key will be known only to the client and the server and this will ensure confidentiality and solve the problem of eavesdropping.

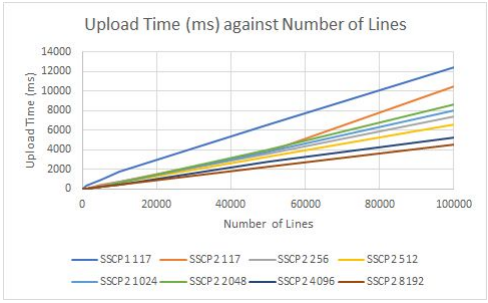
Part (C): Data Throughput

We varied the lengths of block sizes for SSCP2 (which uses AES encryption) to see the effects of block size on upload and download efficiency. We observed that in general, the larger the block size, the faster it is to send larger files. Also, we measured how SSCP2 would perform with the same block length that SSCP1 uses (i.e. 117 bytes), and observed that SSCP2 still outperforms SSCP1 at all file sizes.

			Number of lines							
SSCP	Block Size		100	200	500	1000	5000	10000	50000	100000
SSCP1	117	Time taken to DOWNLOAD file(ms) - client	38	58	109	294	970	1735	6602	12402
SSCP2	117		12	17	42	60	411	772	4268	11349
SSCP2	256		7	16	23	37	317	756	4218	8185
SSCP2	512		4	5	14	25	192	629	3557	7001
SSCP2	1024		4	10	10	20	245	600	4075	8288
SSCP2	2048		4	6	10	17	266	651	4143	8721
SSCP2	4096		4	7	9	17	216	449	2738	5334
SSCP2	8192		4	6	8	13	185	375	2309	4566



			Number of lines							
SSCP	Block Size		100	200	500	1000	5000	10000	50000	100000
SSCP1	117	Time taken to UPLOAD file(ms) - server	33	58	106	296	976	1738	6600	12406
SSCP2	117		6	12	33	61	380	685	3759	10498
SSCP2	256		5	9	22	36	311	696	3616	7367
SSCP2	512		3	5	11	24	194	629	3324	6532
SSCP2	1024		3	5	10	18	241	600	3817	7974
SSCP2	2048		3	4	7	12	265	652	4007	8583
SSCP2	4096		2	4	8	14	218	383	2739	5261
SSCP2	8192		2	5	7	14	184	376	2310	4567



Appendix

Github link: https://github.com/han-keong/50005PA2