Ryan Young

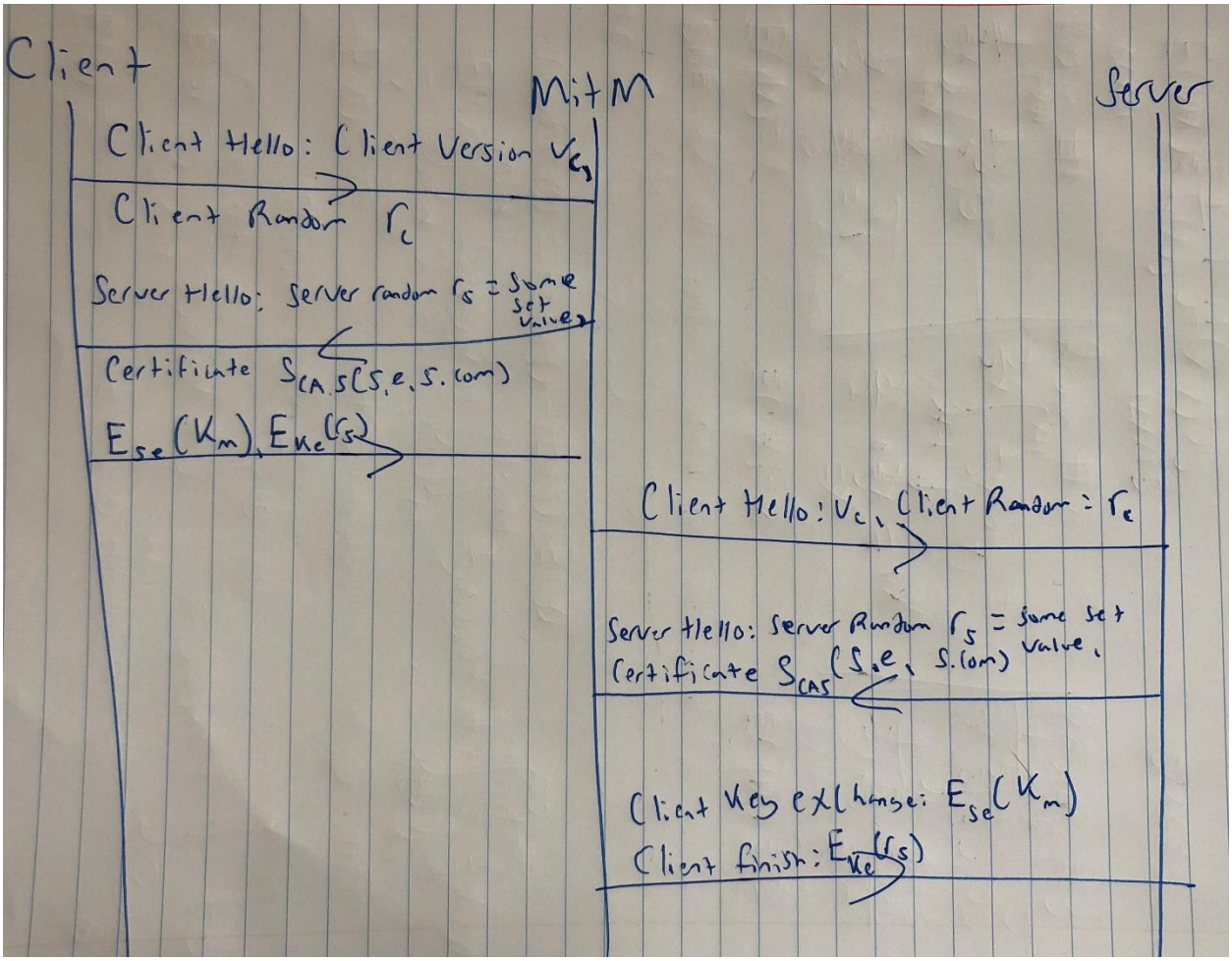
CSE 3400

4/27/2020

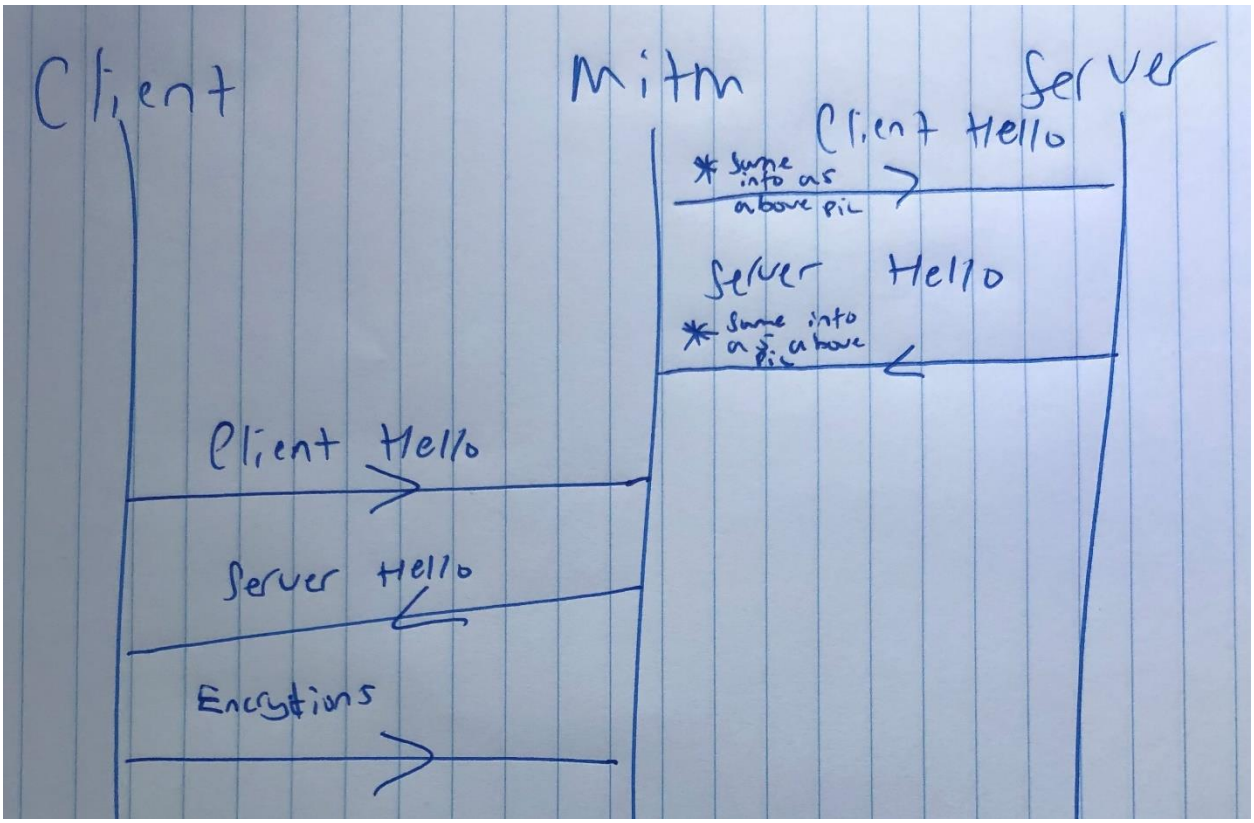
Homework 8

1.) *Consider implementations of the SSLv2 protocol, where the (1) client random or (2) server random fields are omitted (or always sent as a fixed string). Show a message sequence diagram for corresponding attacks, allowing replay of messages to the client or to the server.*

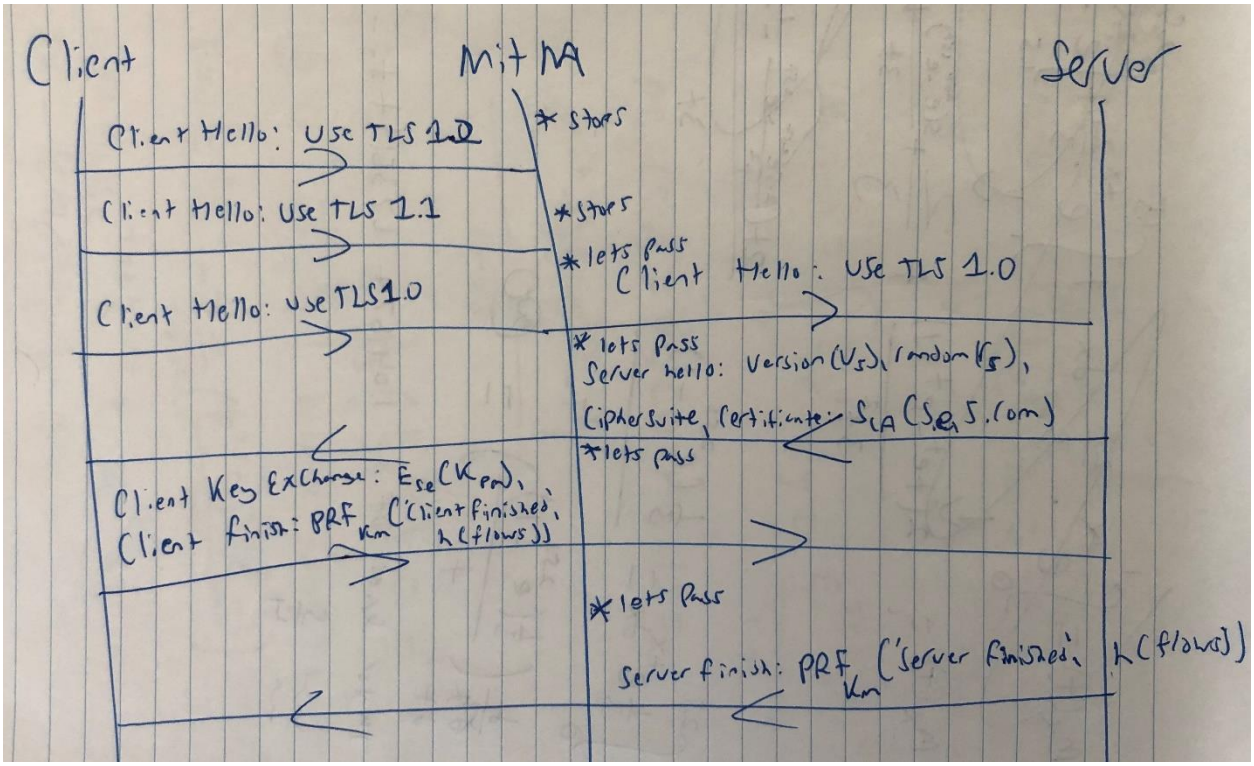
Sequence Diagram:



Other way uses same information above just with simplified syntax:



2.) *Consider a client that supports 'downgrade dance' as described above, trying to connect using TLS 1.2, if fails -using TLS 1.1, and if fails – using TLS 1.0. Present a message sequence diagram for a MitM attack, tricking this client into using TLS 1.0.*
Sequence Diagram:



3.) *Consider a client and server that use TLSv1.2 with ephemeral DH public keys, as in Fig. 7.12. Assume that the client and server run this protocol daily, at the beginning of everyday i. (Within each day, they may use session resumption to avoid additional public key operations; but this is not relevant to the question). Assume that Mal can (1) eavesdrop on communication every day, (2) perform MitM attacks (only) every even day (I s.t. I ≡ 0 (mod 2)), (3) is given all the keys known to the server on the fourth day. Note: the server erases any key once it is no longer in use (i.e., on fourth day, attacker is not given the 'session keys' established n previous days). Fill the 'Exposed on' column of day i in in Table 7.12, indicating the first day j ≥ i in which the adversary should be able to decrypt (expose) the traffic sent on day i between client and server. Write 'never' if the adversary should never be able to decrypt the traffic of day i. Briefly justify.*

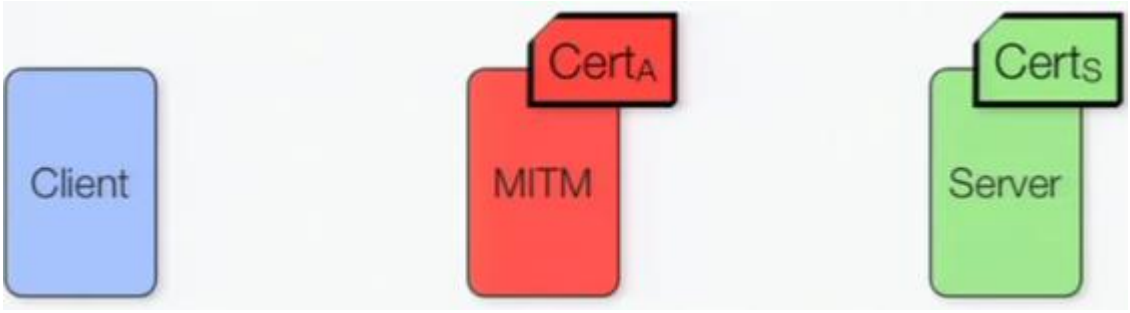
Day	Eavesdrop?	MitM	Given Keys?	Exposed On…	Justify
1	Yes	No	No	Never	TLS 1.2 has PFS and the keys for this day are not exposed
2	Yes	Yes	No	Never	TLS 1.2 has PFS and the keys for this day are not exposed
3	Yes	No	No	Never	TLS 1.2 has PFS and the keys for this day are not exposed
4	Yes	Yes	Yes	Day 4	With the MitM and the Keys the attacker will be able to decrypt the data
5	Yes	No	No	Never	Keys for this day are never exposed and this version of TLS 1.2 will not get exposed to MitM in this scenario because it cannot be downgraded
6	Yes	Yes	No	Never	Keys for this day are never exposed and this version of TLS 1.2

					will not get exposed to MitM in this scenario because it cannot be downgraded
7	Yes	No	No	Never	Keys for this day are never exposed and this version of TLS 1.2 will not get exposed to MitM in this scenario because it cannot be downgraded
8	Yes	Yes	No	Never	Keys for this day are never exposed and this version of TLS 1.2 will not get exposed to MitM in this scenario because it cannot be downgraded

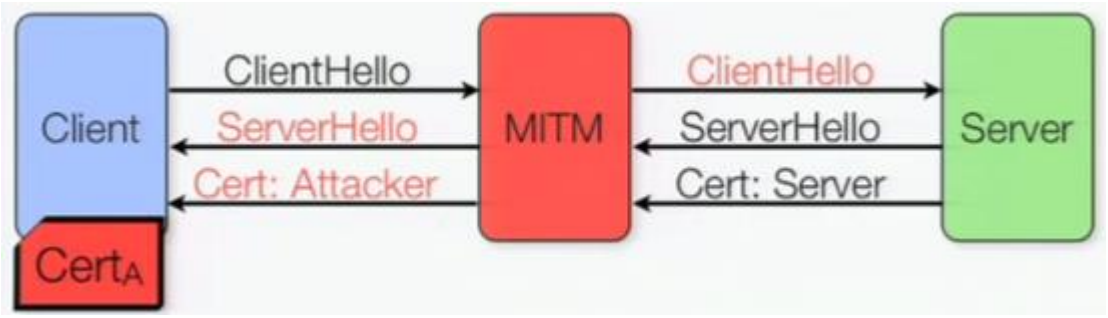
The only day that information will get exposed on will be the 4th day because on that day the keys are exposed, and the attacker has man in the middle capabilities. Prior to that the keys are not exposed and since TLS 1.2 with ephemeral DH public keys, as in Fig. 7.12 has perfect forward secrecy they won't be after the keys are exposed on day 4. After day 4 nothing else will be exposed because the next master keys cannot be exposed. This is due to the master key being derived using a PRF and a pre-master key computed each day using a secure version of ephemeral Diffie-Hellman!

4.) *(IE failure to validate basic constraint). Old versions of the IE browser failed to validate the basic constraint field. Show a sequence diagram for an attack exploiting this vulnerability, allowing a MitM attacker to collect the user's password to trusted sites which authenticate the user using user-id and password, protected using SSL/TLS.*

Since we know that in this scenario we are protected by SSL/TLS and we know that the basic constraints field was not validated by the old version of internet explorer the attacker who has man in the middle capabilities can exploit the fact that he or she can become a CA. This is roughly represented in diagram below:



With $Cert_A$ being a fake but valid certificate for the identity of the sever, then:



5.) *Consider the certificate-hash-tree variant of OCSP, described above and illustrated in subsection 4.7.1.*

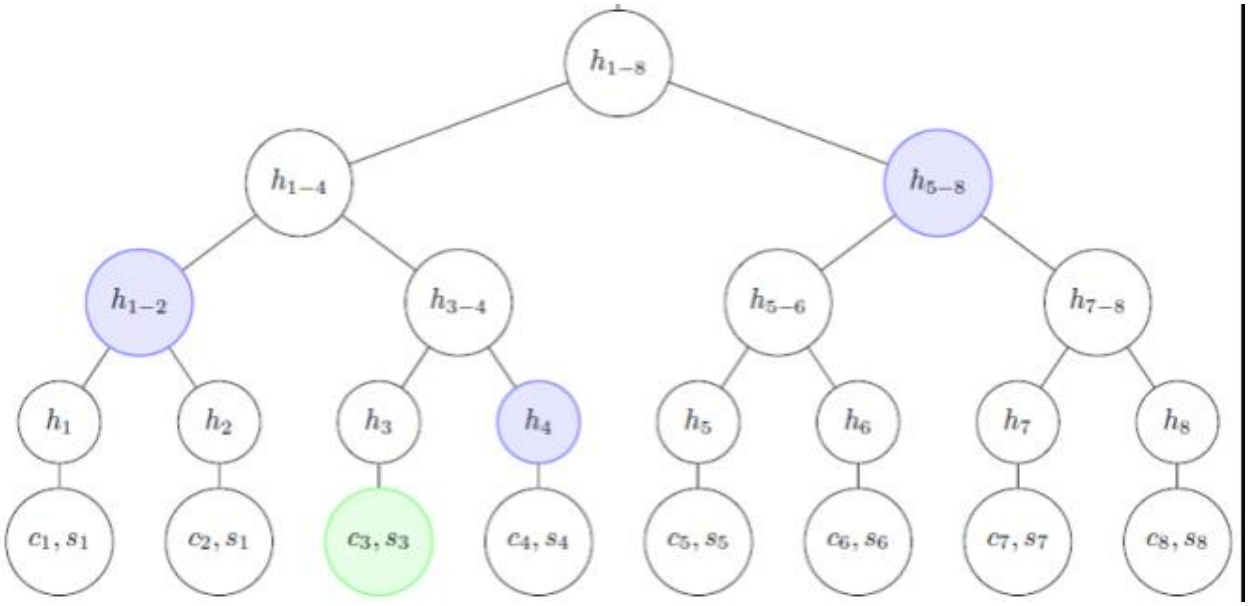
a. *Present pseudo-code for the OCSP client, including validation of the OCSP responses (including the proofs of inclusion).*

Client receives a list of hashes for the level in the tree, with n hashes. The goal is to hash(c_i) and $h_{i-(i+1)} = h(h_i||h_{i+1})$

For C_i in n:

Hash (C_i || C_i+1)

Will ultimately have a tree structure similar to:



Aka the client will hash the next hash concatenated with the previous one

b. *Let n be the number of certificates issued by a CA, r < n be the number of revoked certificates, and i < r be the number of certificate-identifiers sent in a given OCSP request. What is the number of (1) signature operations, (2) signature-validation operations, (3) hash operations, required to (a) produce and send a CRL, (b) produce and send an OCSP response,(c) produce a certificate-hash-tree OCSP response, (d) validate a CRL,(e) valid an OCSP response, (f) validate an certificate-hash-tree response.*

1. I
2. n
3. A. 1
   B. 0
   C. 0
   D. $n \log_2 n$
   E. $\log_2 n$
   F. $\log_2 n$

c. *This variant uses an (unkeyed) collision-resistant hash function (CRHF)h. Explain why it may be desirable to avoid this assumption.*
Since there are no CRHF that are unkeyed we would have to either use a weaker hash function or a keyed CRHF and this would make it a not strong assumption.

d. *Would it be Ok to use in the design a Second-Preimage Resistant (SPR) hash function, instead of the keyless CRHF? Present a convincing justification, preferably, with a*

reduction to prove security or with a counterexample showing insecurity (for the use of SPR hash function in this construction).

I do not think it would be ok to use in the design a Second-Preimage Resistant hash function, because the SPR hash function will allow collisions to occur if we assume that an attacker is able to find a collision. And they start at a leaf node they will be able to trace back and find another collision because of the tree structure it will become the hash of the previous hash concatenation with the other node. This means that SPR hashing within the tree structure will be considerably more insecure in comparison to a CRFH tree scenario.

e. *Present an alternative way to replace the keyless CRHF with a different function which is about as efficient (as the original design using CRHF) yet is secure under a more acceptable assumption.*

An alternative way to replace the keyless CRHF would be to use a Keyed CRHF. Based on the security specifications of a keyed CRFH the attacker will still not be able to find collisions even if they are given the key.