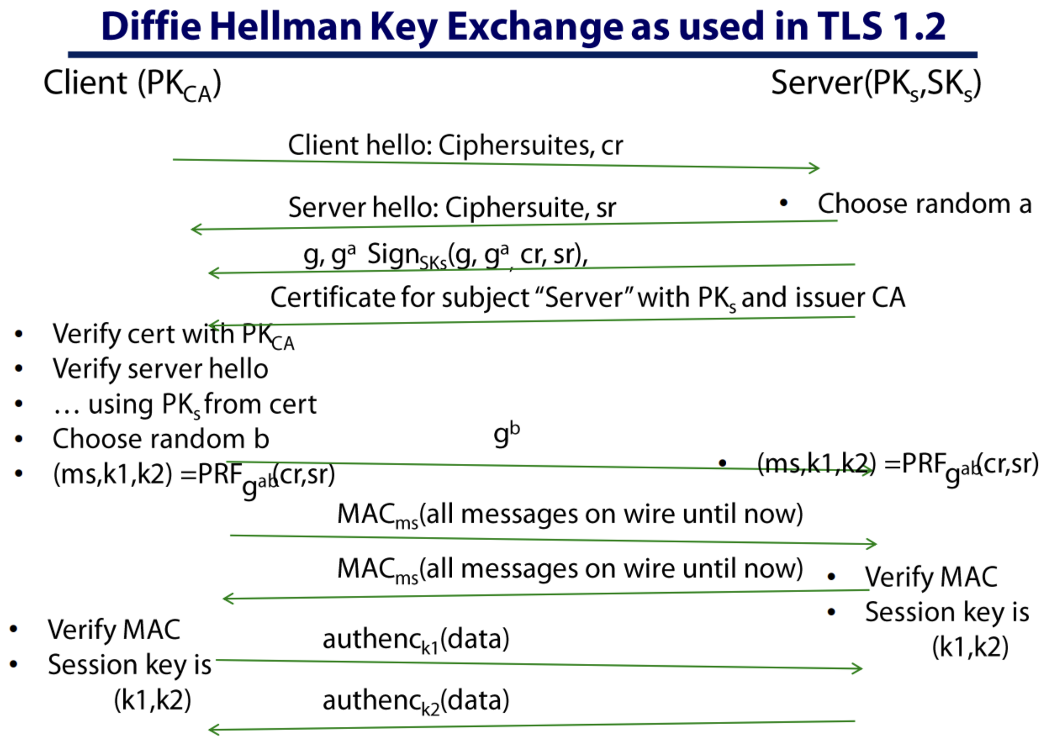
CAS CS 357

Discussion Worksheet 4

Note


Diffie Hellman Key Exchange as used in TLS 1.2
Client (PK_{CA})
Server(PK_s,SK_s)
Client hello: Ciphersuites, cr
Server hello: Ciphersuite, sr
Choose random a
g, g^a $Sign_{SKs}$(g, g^a, cr, sr),
Certificate for subject "Server" with PK_s and issuer CA
Verify cert with PK_{CA}
Verify server hello
… using PK_s from cert
Choose random b
g^b
(ms,k1,k2) =$PRF_{g^{ab}}$(cr,sr)
(ms,k1,k2) =$PRF_{g^{ab}}$(cr,sr)
MAC_{ms}(all messages on wire until now)
MAC_{ms}(all messages on wire until now)
Verify MAC
Session key is (k1,k2)
Verify MAC
Session key is (k1,k2)
$authenc_{k1}$(data)
$authenc_{k2}$(data)

Steps of TLS

1. Client sends hello to server and includes the ciphersuites of the client

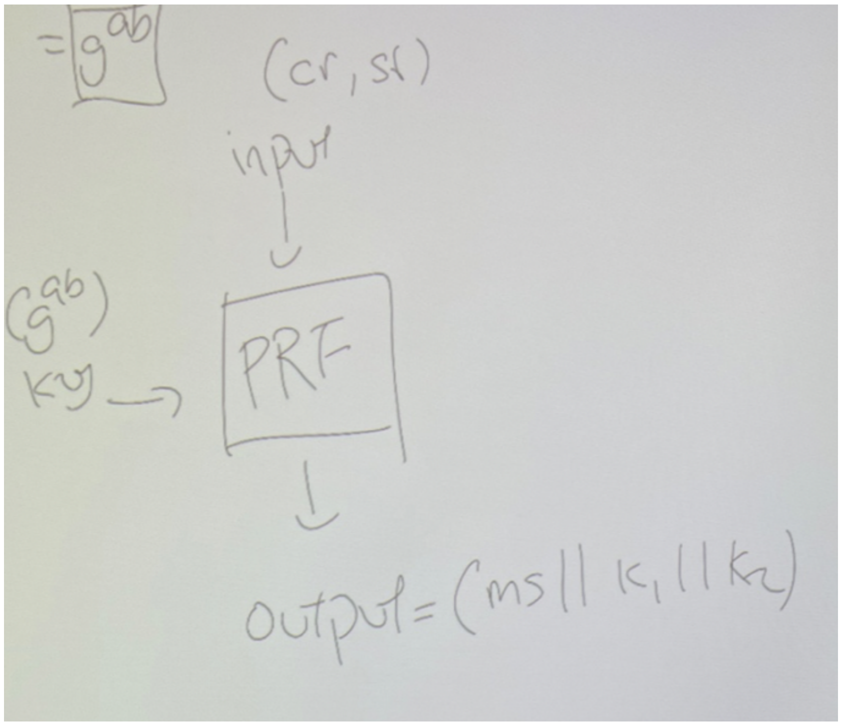
   Ciphersuites include encryption scheme, the hash, the Mac that is used in cryptography

2. Server sends back hello to client and includes the ciphersuites of the server

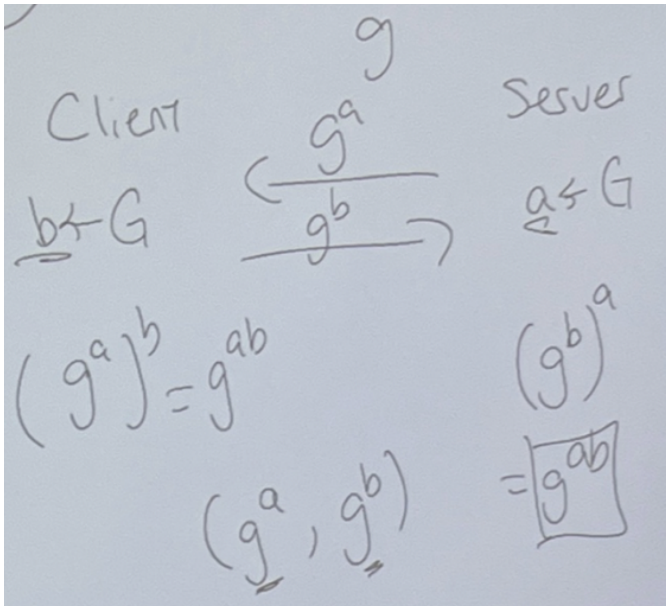
3. Server chooses a random variable "a"

4. Server sends back g, g^a, $Sign_{sks}$(g,g^a,cr,sr)

5. Server sends back certificate for subject "server" with public key of the server and the client

6. Client verifies the certificate with PK_{CA}
7. Client verifies server hello using PK_s from certificate
8. Client chooses a random variable "b"
9. Client sends back g^b to the server
10. 

11. Using a pseudo generator, both the client and server inputs key value of g^(a*b) and the ciphersuites to output a key that involves (master suite, key 1, and key 2)
12. The client and server Macs the information that was sent over between each other in the previous steps by using the master suite to protect the man-in-the-middle from changing the information (once this step is completed, the server and client will notify when the man in the middle takes place and changes the information such as b → b' since a different Mac will be created once man in the middle steps in)
13. After verification, the server and the client send messages to each other by using authentication using the symmetric key (k1 == k2) (authentication is MAC + encryption)



1. The variable “b” is the secret key of the client and the variable “a” is the secret key of the server
2. The variables are sent over to each other by g^a or g^b
3. Given g^a or g^b, the adversary finds it difficult to find the variable “a” and “b” (g is a group element)
4. The symmetric key that is shared between the client and the server is g^(a*b), which can be done by (g^a)^b or (g^b)^a, which can be easily done by the cline and the server but is difficult to the adversary who cannot find “a” and “b”

Questions

1. The protocol shows in the figure supports “one sided authentication”. That is, the client knows that she is talking to the server. However, the server does not know which client she is talking to. To make this point crystal clear, consider a man-in-the-middle attacker that wishes to impersonate the client to the server. (That is, the man-in-the-middle

attacker sits between the client and server. Write down the protocol that attacker would use to impersonate the client to server (i.e. convince the server that the server is talking to the client, instead of to the attacker).

Attacker does the same thing as the client (and passes back the messages)

2. Now consider a man-in-the-middle attacker that sits between the client and server, and wishes to impersonate the server to the client. Assuming that the client knows the correct public key for the CA. Why does the attacker fail to impersonate the server to the client?

Because the attacker cannot forge the server's signature, it has to use g^a as chosen by the server. But by the hardness of discrete logs, the attacker cannot obtain a from g^a, and thus cannot compute g^(ab) and thus cannot learn the master suite ms and thus cannot provide a correct MAC. so the client will learn that something is wrong.

3. Consider a flawed TLS implementation where the client "forgets" to check the signature on the server's certificate. Write down exactly how a man-in-the-middle attacker that intercepts the communications between client and server can establish one pair of session keys (k1, k2) between itself and the client, and another pair of session keys (k1 ′ , k2 ′) between itself and the server. Explain why, by doing this, the attacker can silently intercept, read, and pass on any data sent from client to server, and vice versa, without the client or server ever realizing that their communications have been read.

The attacker can change g^a with g^a’, and sign (g,g^a’, cr, sr) using the attacker’s own secret key SK_a (3rd message). Then the attacker can pass a “fake” certificate for PK_a to the client (4th message). This will allow the attacker to have a valid connection with the client. The attacker can also pass back g^b’ to the server and act as the client. Now the attacker has two shared keds (g^(a’b) with the client, and g^(ab’)).

4. Consider a man-in-the-middle attacker that steals the secret key of the CA, SK_{CA}. Explain why, by doing this, the attacker can silently intercept, read, and pass on any data sent from client to server, and vice versa, without the client or server ever realizing that their communications have been read. (Hint, once again the attacker establishes one pair of session keys (k1, k2) between itself and the client, and another pair of session keys (k1 ′ , k2 ′) between itself and the server. Write down exactly how it does this.)

The attacker does the same thing as question 3, but now passes g^a’ and the signature (g, g^a’, cr, sr) signed with SK_s, the attacker then forwards the server’s certificate to the client. When the client sends g^b, the attacker can once again pass g^b’ to the server. The resulting shared keys is the same as question 3.

5. If this attacker becomes a man-in-the-middle for another client and the same server Server, can it carry out the same attack?

Yes

6. If this attacker becomes a man-in-the-middle for another client and a different server Server2, can it carry out the same attack?

   Yes

7. Now consider a passive attacker that has collected all the communication sent between the client and server that have been done in the past. Suppose this passive attacker has now stolen the secret key of the CA, SK_{CA}. Can it use this secret key to decrypt the past communications that it has collected?

No, because knowledge of SK_{CA} does not allow the attacker to determine a or b that were used in the previous sessions

8. Now consider a passive attacker that has collected all the communication sent between the client and server in the past. Suppose this passive attacker has now stolen the secret key of the server, SK_S. Can it use this secret key to decrypt the past communications that it has collected?

No, because knowledge of SK_{CA} does not allow the attacker to determine a or b that were used in the previous sessions

9. If you were an attacker, which key would you most want to steal? Your choices are SK_{CA}, SK_S, ms, k1, k2. Justify your response.

SK_{CA} because I could use it issue forged certificates and then impersonate any server on the internet

10. Consider the Diginotar incident that was in the news in 2012. (Look on the Internet!) Explain which of the above keys were stolen in the attack on Diginotar. Also explain how these keys were used, by the attacker, to impersonate Google to users in Iran.

11. The SuperFish malvertising software was shipped as part of Lenovo laptops in 2015. Researchers discovered that the Superfish software modified the laptop's browser to so that the SuperFish Public Key was installed as trusted CA public key. The SuperFish software also made itself a man-in-the-middle between the browser and the laptop's network connection. Explain exactly how this allowed SuperFish to decrypt any communication sent from the user to any webserver, without the user knowing, and modify this communication to inject SuperFish ads. (Hint, consider how SuperFish might forge a certificate for the Server.)

Since SuperFish's Public key is installed as a trusted CA public key, they have the corresponding secret key and can do the same attacker as question 3, where instead of a "fake" certificate, they pass in their own certificate which is pre-installed.