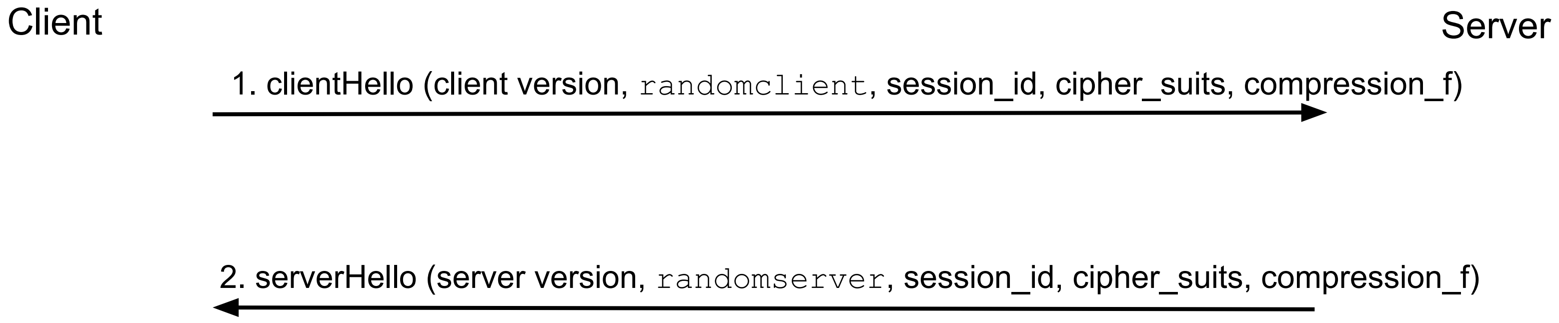
TLS and CSRF Review

Qian Long

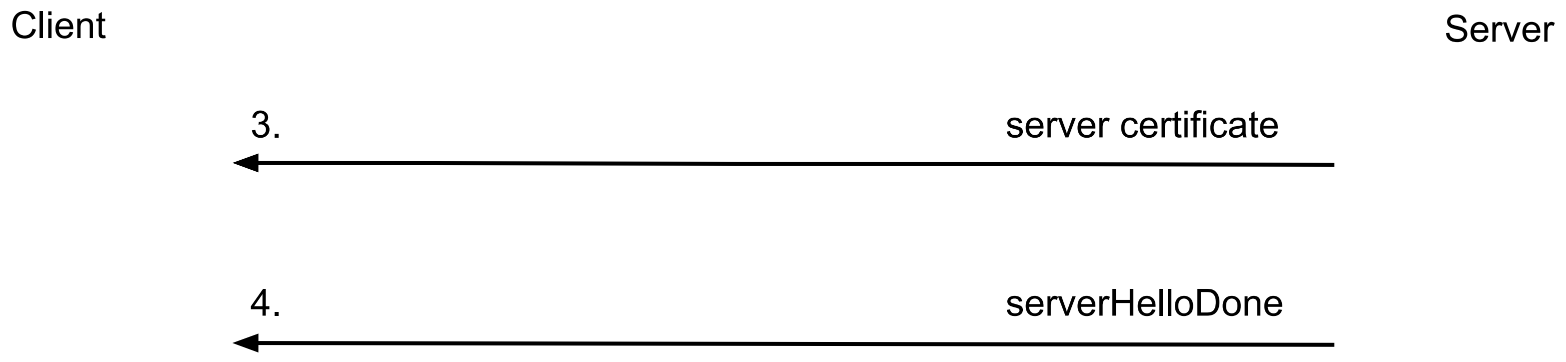
TLS Handshake Part 1

Establish TLS/SSL connection: communicate TLS/SSL version, random sequence, session id, cipher suite and compression algos


Client
Server
1. clientHello (client version, randomclient, session_id, cipher_suits, compression_f)
2. serverHello (server version, randomserver, session_id, cipher_suits, compression_f)

TLS Handshake Part 2

Server Authentication

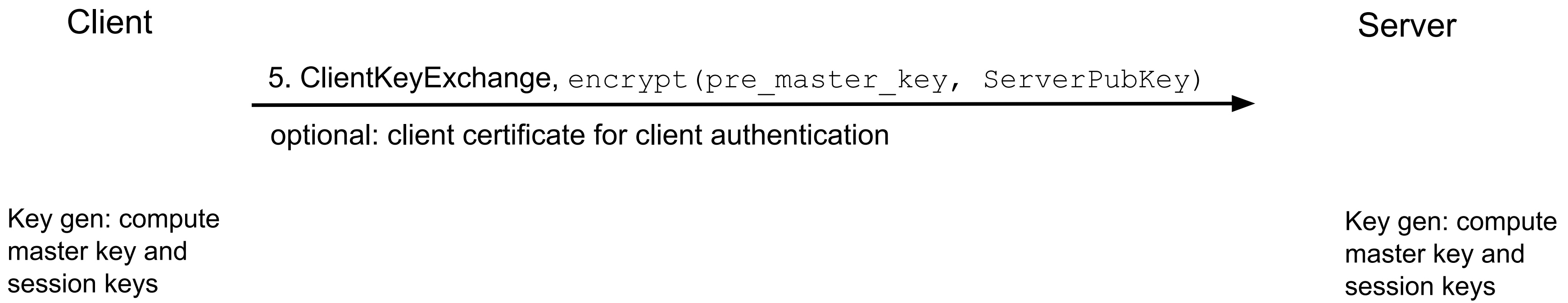


client verifies certificates
sent by server

Certificate Authority

- Trusted 3rd party to verify that a public key belongs to a particular identity
- Certificate
    - identity, identity’s pk, signature
    - signature = Sign{identity, identity’s pk} with CA’s secret key
- Verify(signature) with CA public key
- CA public key may come bundled with browser

TLS Handshake Part 3



Client

Server

5. ClientKeyExchange, `encrypt(pre_master_key, ServerPubKey)`

optional: client certificate for client authentication

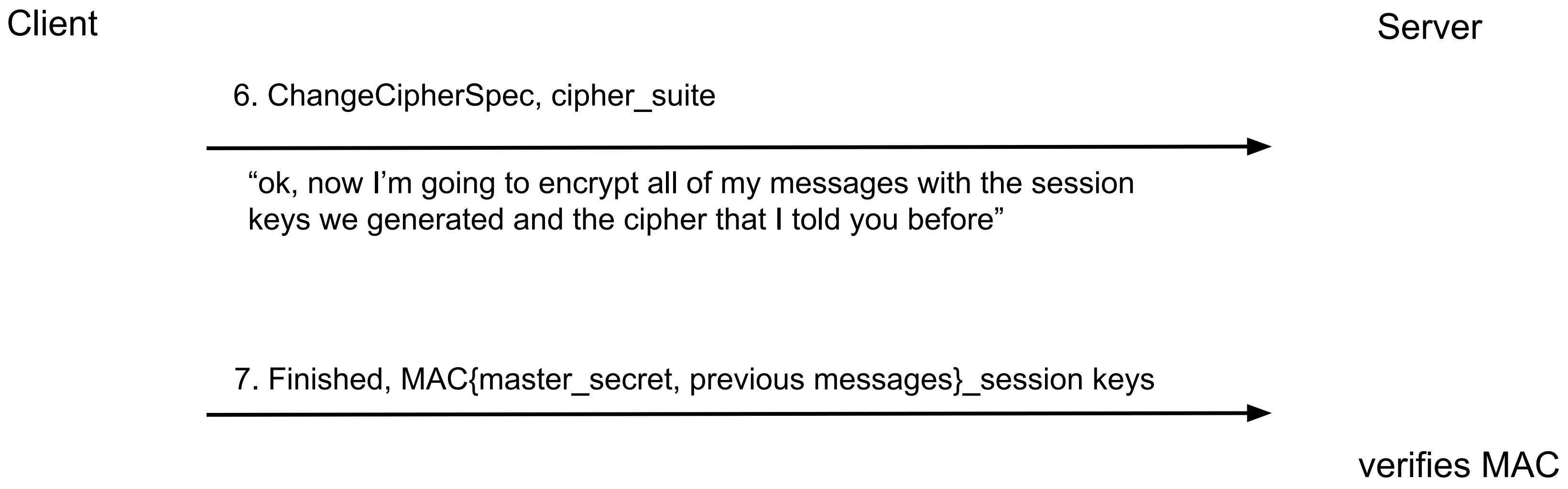
Key gen: compute master key and session keys

Key gen: compute master key and session keys

Key Generation

- Client generates `pre_master_secret`
  - encrypt with server public key obtained from certificate
  - send to server
  - server decrypts with its secret key
- Client and server both generate `master_secret` (shared secret)
  - both have `pre_master_secret`, `randomclient`, `randomserver`
  - `master_secret` <- PRF(`pre_master_secret`, “master secret”, `randomclient + randomserver`)
  - random numbers prevent reply, so each TLS connection has unique master_secret
- Client and server both generate session keys from master_secret, randomclient, randomserver
  - `client_write_MAC_secret, server_write_MAC_secret, client_write_key, server_write_key, client_write_IV, server_write_IV`

TLS Handshake Part 4


Client

Server

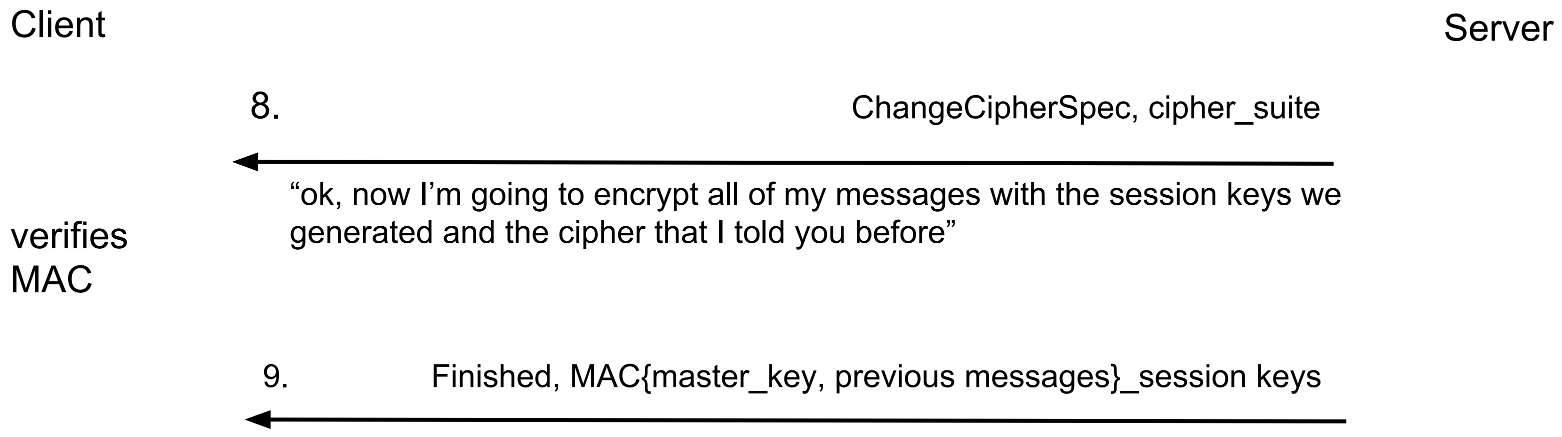
6. ChangeCipherSpec, cipher_suite

“ok, now I’m going to encrypt all of my messages with the session keys we generated and the cipher that I told you before”

7. Finished, MAC{master_secret, previous messages}_session keys

verifies MAC

TLS Handshake Part 5



8. ChangeCipherSpec, cipher_suite

“ok, now I’m going to encrypt all of my messages with the session keys we generated and the cipher that I told you before”

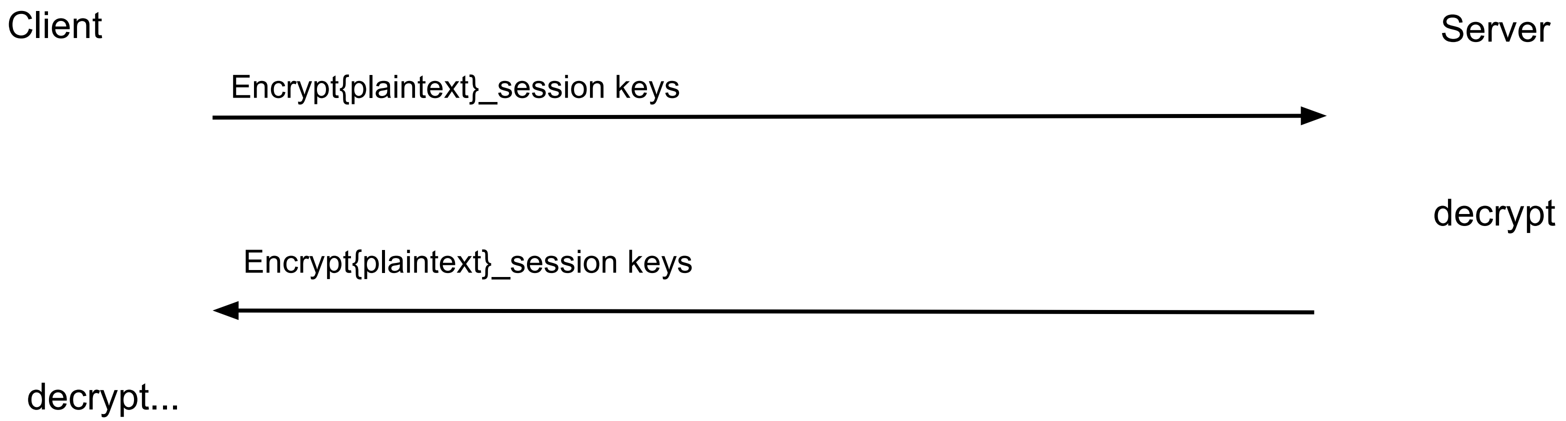
9. Finished, MAC{master_key, previous messages}_session keys

Message Authentication Code

- check for data integrity
- black box hash function
- In TLS, client and server both send each other MAC {master_secret, previous messages}_session keys
  - checks that they both indeed computed the same master_secret and the previous messages sent were not modified

TLS Handshake Done!

Communicate securely using shared session keys!


Client
Server
Encrypt{plaintext}_session keys
decrypt
Encrypt{plaintext}_session keys
decrypt...

Client and Server can each use different ciphers when encrypting data.
They both have the necessary keys to decrypt.

Same Origin Policy

- allows scripts originating from the same site (host, port) to access each other’s DOM, prevents access to DOM on other sites

Cross Site Scripting

- insert malicious code as part of the content of the website
  - example) if the website has a form, attacker might get you to input javascript (or more likely to click on a url with javascript inputted as part of a parameter), and that malicious code will be run in your browser
- Solution: sanitize user input

Cross Site Forgery (CSRF)

- exploits site’s trust in user’s browser
  - user’s browser stores session cookies
- typical setup:
  - victim is already logged into a bank site, so bank session cookies are stored in browser
  - attack gets victim to click on a link to withdraw money from victim’s account (in another site)
  - basically send request from site B to site A while taking advantage of user being logged into site A

Cross Site Forgery (CSRF)

- login CSRF
  - attacker gets victim to log in as attacker, maybe by secretly sending a cross site request to a bank site with attacker’s username and password
  - attacker can track victim activity

Cross Site Forgery (CSRF)

- CSRF tokens
  - generate a token for a session to send with the request each time, server checks if token matches
  - cross site request would have a hard time generating token
  - problematic for login when the session has not been established
- Referer Header
  - check where the request came from, if the referrer doesn’t match the hostname/port in the request, reject
  - problem: leaks privacy info b/c full url is contained, often not set
- Origin Header (custom header, proposed solution)
  - basically the same as referer header but only use hostname and port instead of the full url

Questions?

Quiz is Wednesday at 1:30pm

Good Luck!!!