disc. 11
cs161 su23

TCP/UDP, TLS

acronyms ahoy

slides
bit.ly/cs161-disc

feedback
bit.ly/abhifeedback

hack of the day

hack of the day

- Apache Pulsar TLS vulnerability led to MITM

hack of the day

- Apache Pulsar TLS vulnerability led to MITM
  - “used by thousands of companies for…instant messaging, data integrations…managing hundreds of billions of events per day.”

hack of the day

- Apache Pulsar TLS vulnerability led to MITM
    - “used by thousands of companies for…instant messaging, data integrations…managing hundreds of billions of events per day.”
    - server certificate verification occurred AFTER authentication credentials sent

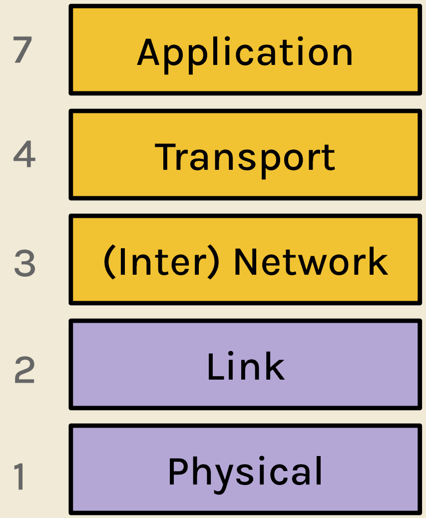
hack of the day

- Apache Pulsar TLS vulnerability led to MITM
    - “used by thousands of companies for…instant messaging, data integrations…managing hundreds of billions of events per day.”
    - server certificate verification occurred AFTER authentication credentials sent
    - attacker can impersonate client

general questions, concerns, etc.

the OSI model

- layer 1: communication of bits
- layer 2: local frame delivery
    - ethernet via 6-byte MAC addresses
- layer 3: global packet delivery
    - IP: the universal Layer 3 4/16-byte protocol
- layer 4: transport of data
    - TCP/IP
- layer 7: applications and services (the web)



types of attackers

- off-path: can’t see, modify, or drop packets
- on-path: can see packets, can’t modify or drop
- MITM: can see, modify, or drop packets

TCP

TCP

- addresses problems with IP: unreliable

TCP

- addresses problems with IP: unreliable
- **segments** large messages, sent as layer 3 packets

TCP

- addresses problems with IP: unreliable
- **segments** large messages, sent as layer 3 packets
- **ordered:** segments contain sequence numbers

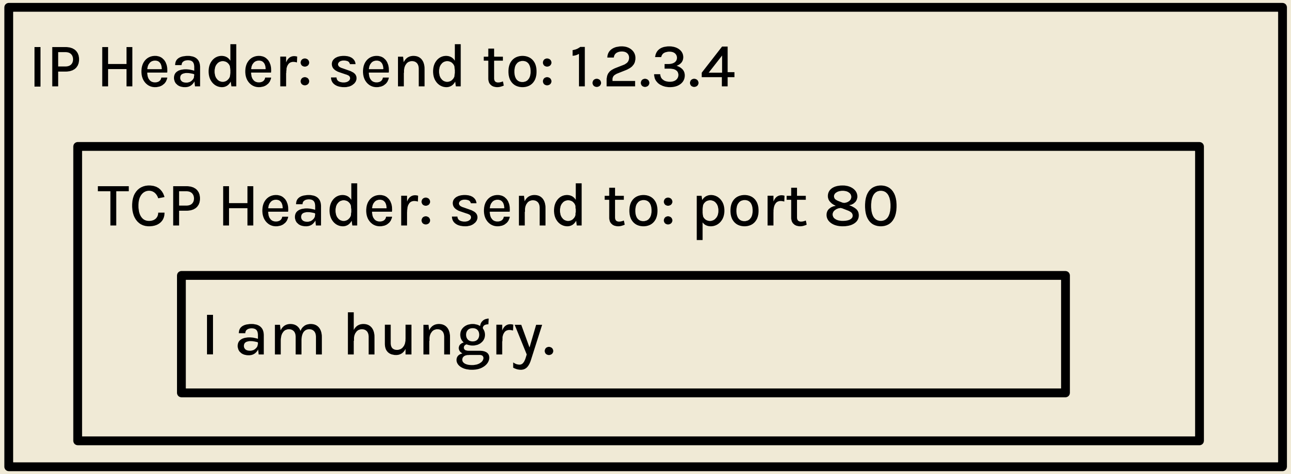
TCP

- addresses problems with IP: unreliable
- **segments** large messages, sent as layer 3 packets
- **ordered:** segments contain sequence numbers
- **reliable:** ACK for each sequence # received
    - don’t receive? send again

TCP

- addresses problems with IP: unreliable
- **segments** large messages, sent as layer 3 packets
- **ordered:** segments contain sequence numbers
- **reliable:** ACK for each sequence # received
  - don’t receive? send again
- **ports:** multiple services can share IP with ports

TCP: ports



TCP: initial sequence numbers

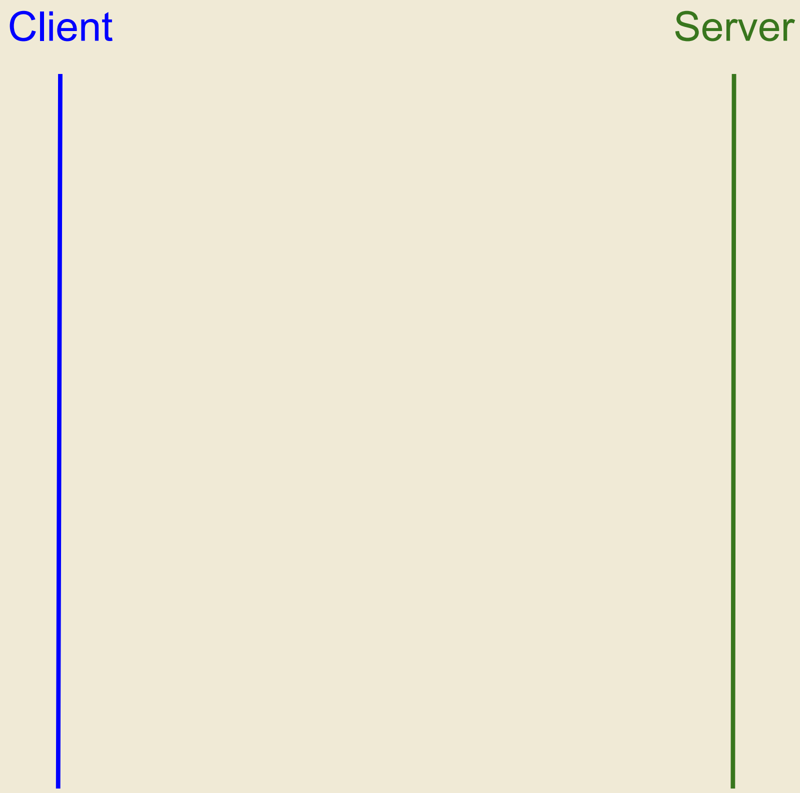
H	e	l	l	o		s	e	r	v	e	r
50	51	52	53	54	55	56	57	58	59	60	61

Messages from the client are numbered starting at 50.

H	e	l	l	o		c	l	i	e	n	t
25	26	27	28	29	30	31	32	33	34	35	36

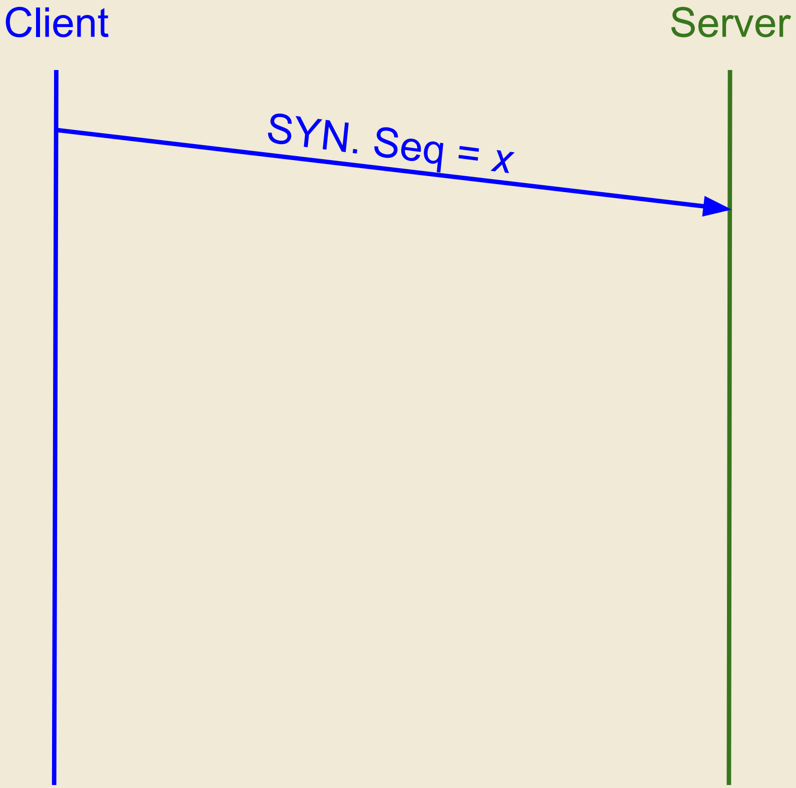
Messages from the server are numbered starting at 25.

TCP: 3-way handshake



TCP: 3-way handshake

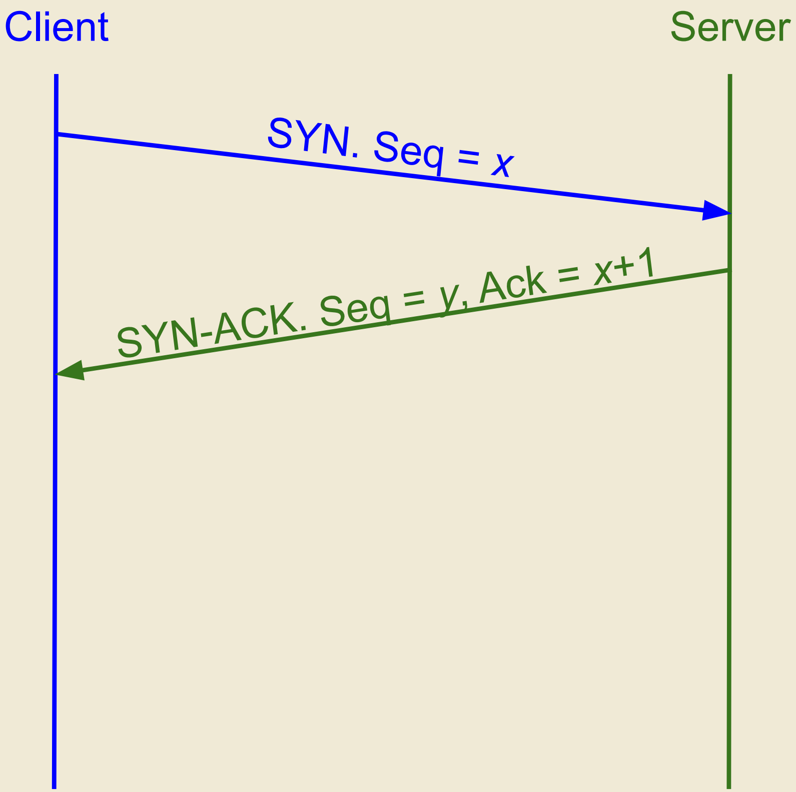
1. client chooses an initial sequence number *x* its bytes and sends a SYN (synchronize) packet to the server



TCP: 3-way handshake

1. client chooses an initial sequence number x its bytes and sends a SYN (synchronize) packet to the server

2. server chooses an initial sequence number y for its bytes and responds with a SYN-ACK packet

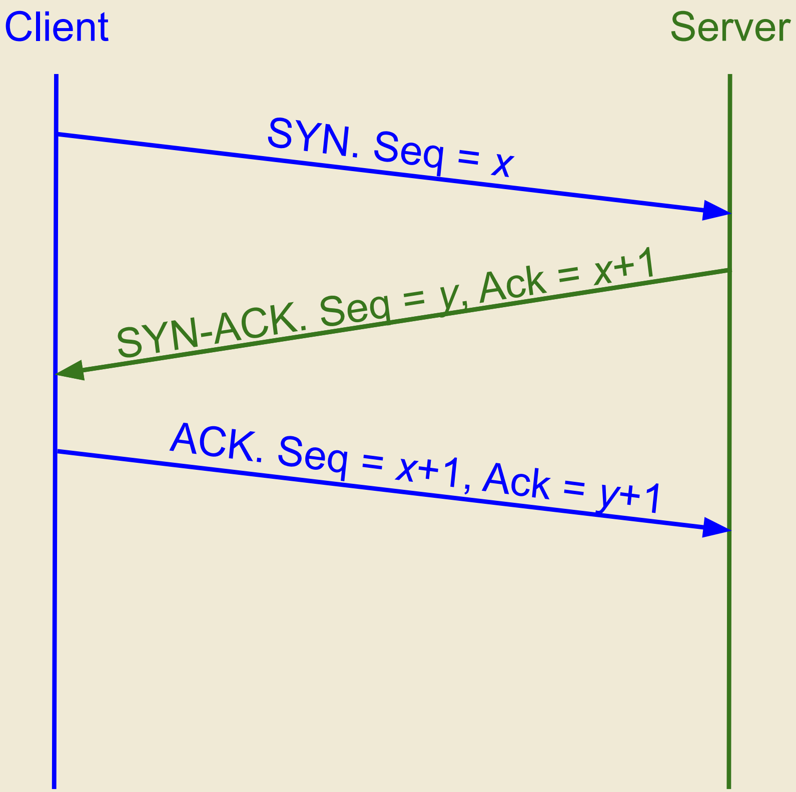


TCP: 3-way handshake

1. client chooses an initial sequence number x its bytes and sends a SYN (synchronize) packet to the server

2. server chooses an initial sequence number y for its bytes and responds with a SYN-ACK packet

3. client then returns with an ACK packet



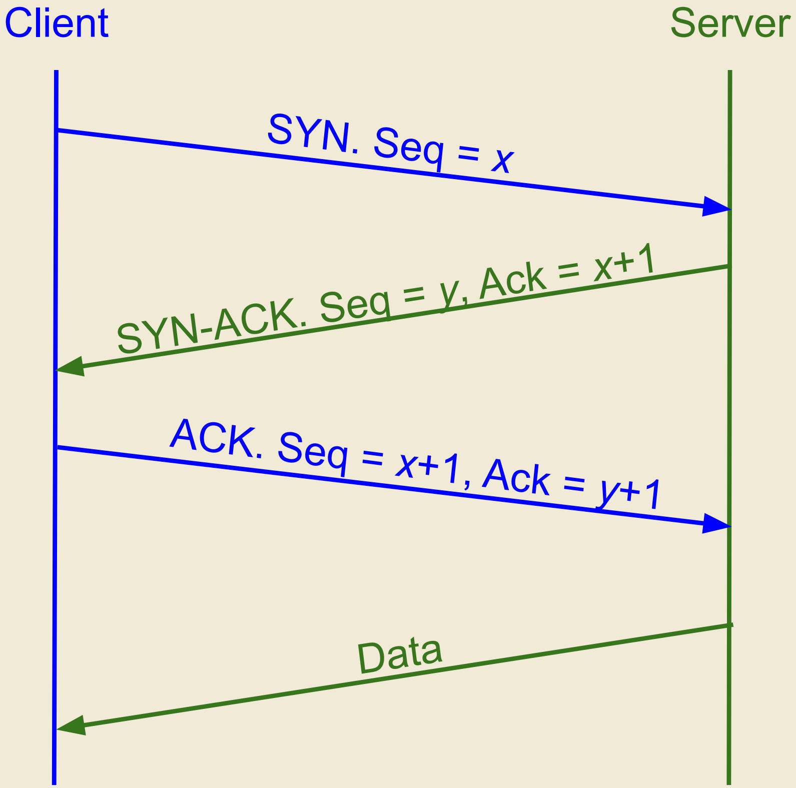
TCP: 3-way handshake

1. client chooses an initial sequence number x its bytes and sends a SYN (synchronize) packet to the server

2. server chooses an initial sequence number y for its bytes and responds with a SYN-ACK packet

3. client then returns with an ACK packet

4. once both hosts have synchronized sequence numbers, the connection is “established”



TCP

- handler tracks which TCP segments received
- TCP 5-tuple
  - source IP
  - destination IP
  - source port
  - destination port
  - protocol

TCP: sequence/ACK numbers

TCP: sequence/ACK numbers

- byte i of bytestream represented by sequence number x + i (x from initial SYN packet)

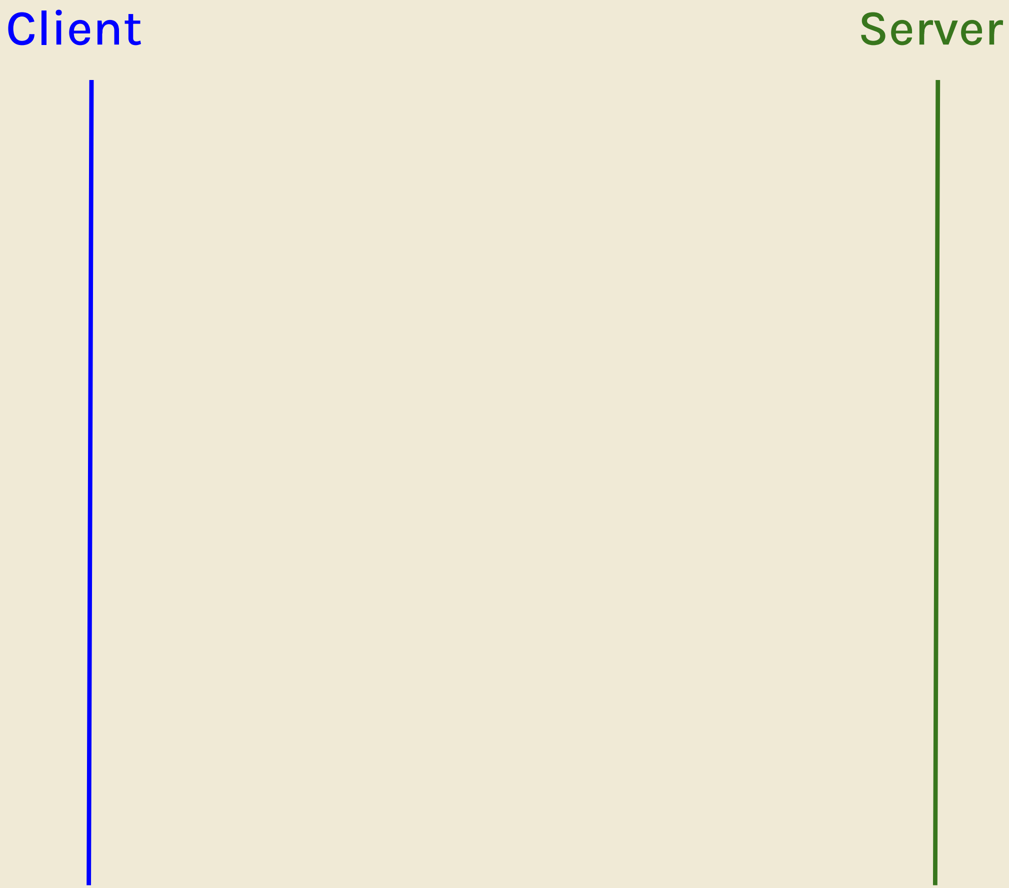
TCP: sequence/ACK numbers

- byte i of bytestream represented by sequence number x + i (x from initial SYN packet)
- sequence number of a packet is the number of the first byte of its data

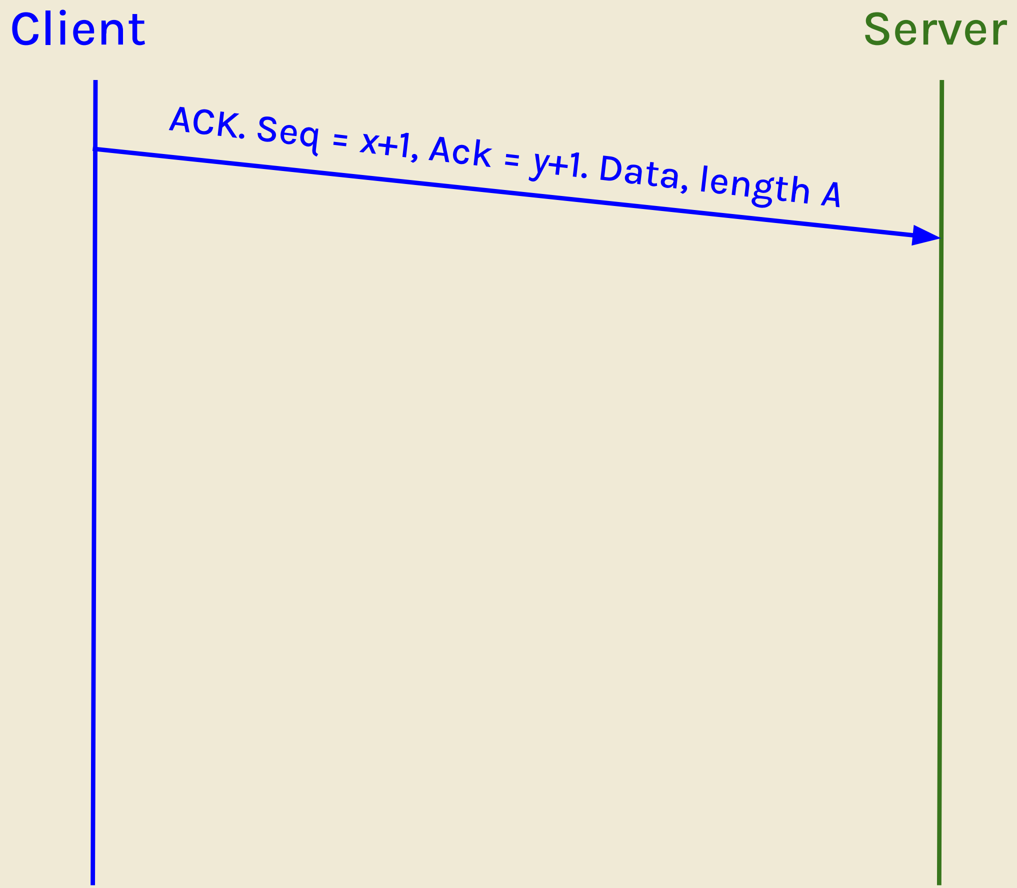
TCP: sequence/ACK numbers

- byte i of bytestream represented by sequence number x + i (x from initial SYN packet)
- sequence number of a packet is the number of the first byte of its data
- ACK number of packet is (sequence number + length of data) for last received packet

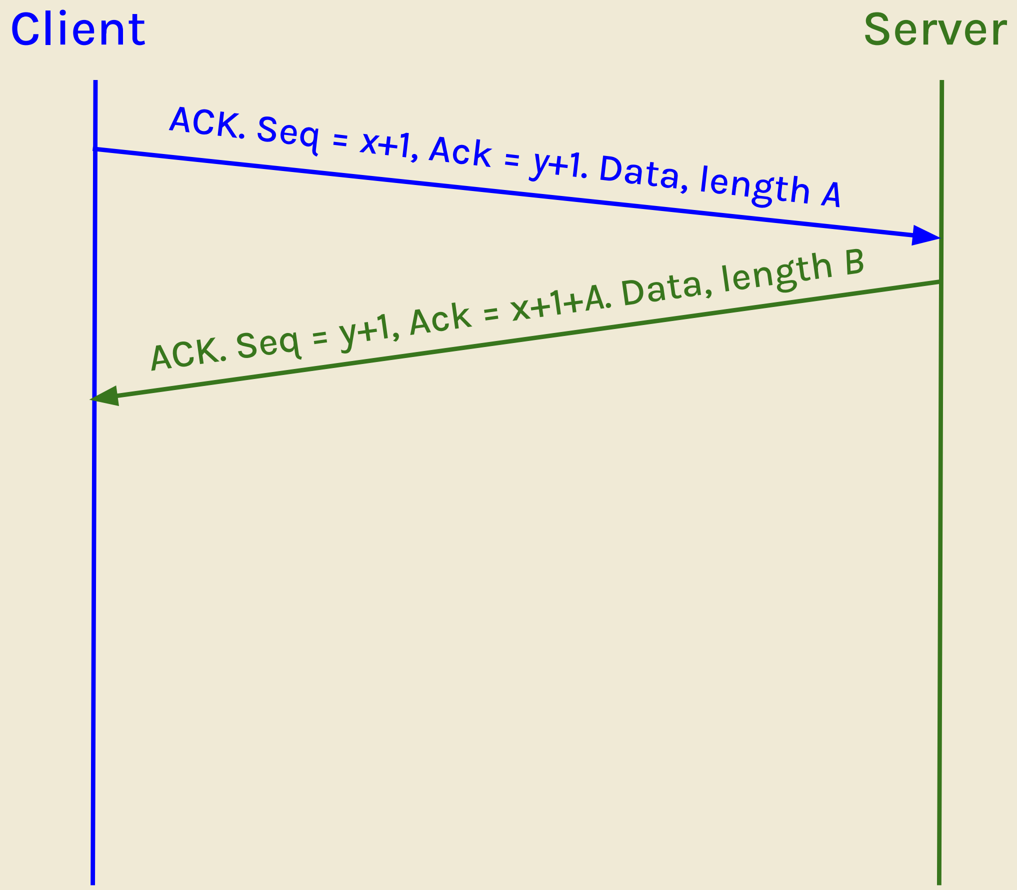
TCP



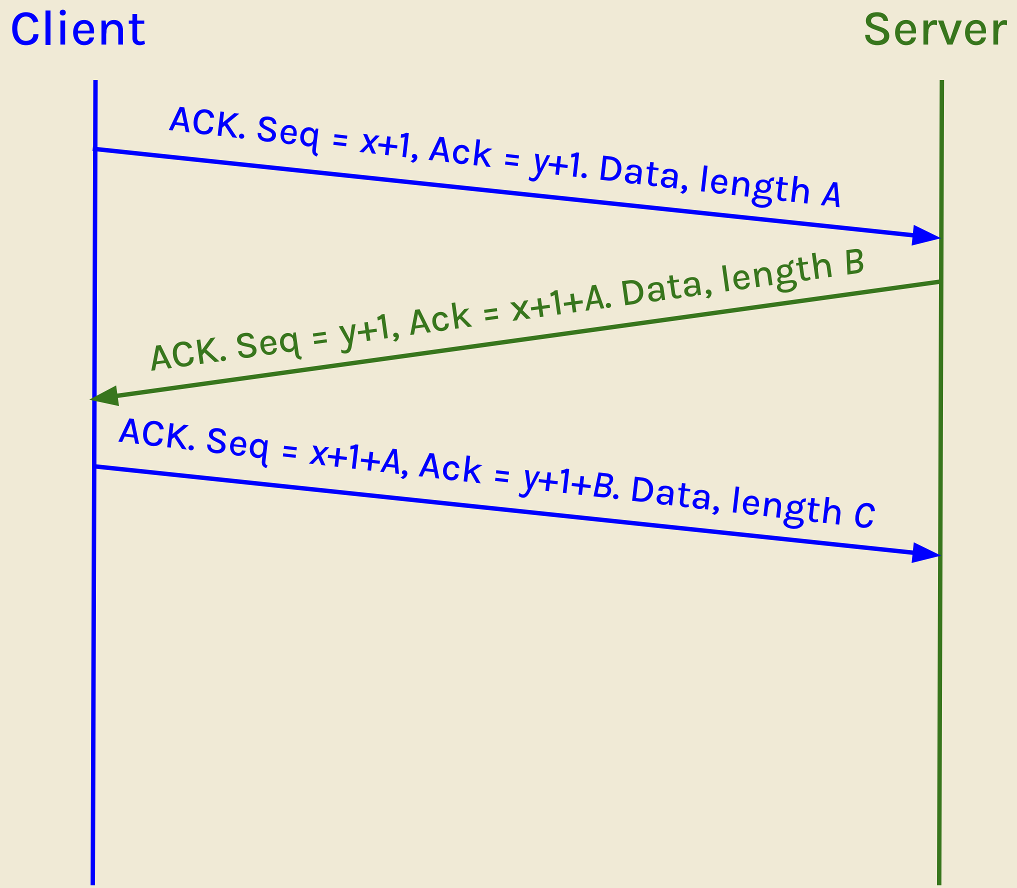
TCP



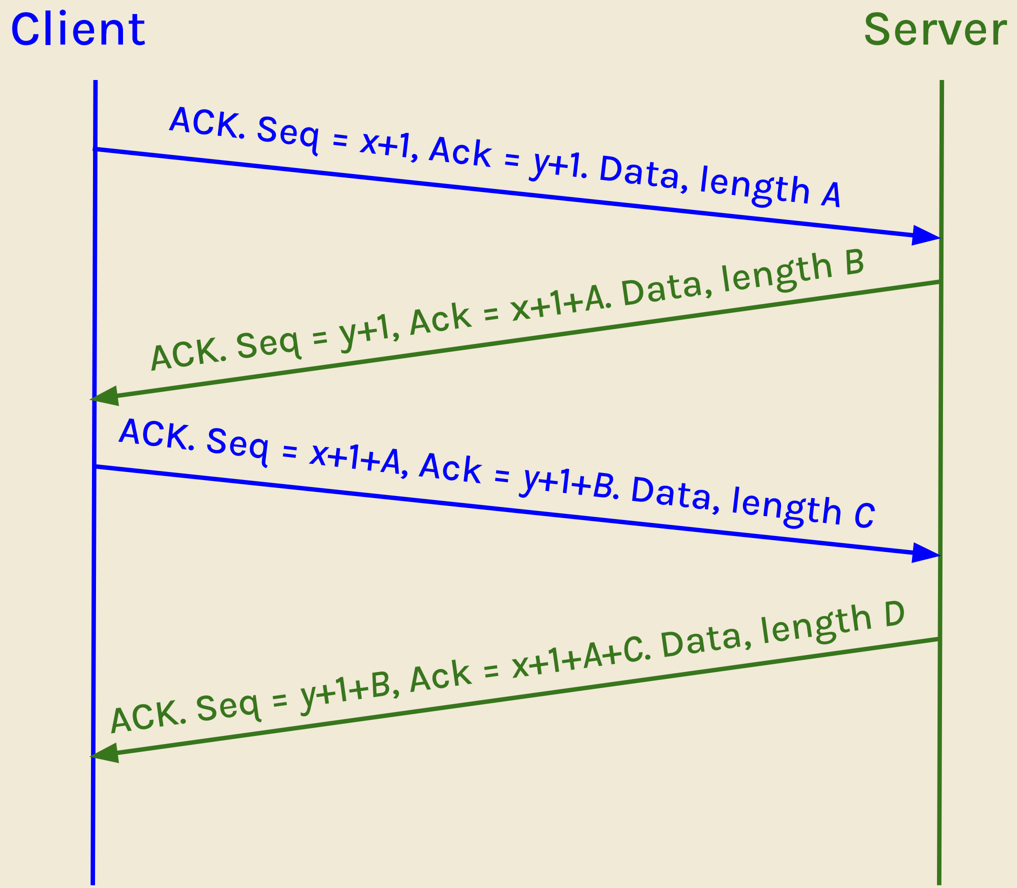
TCP


Client
Server
ACK. Seq = x+1, Ack = y+1. Data, length A
ACK. Seq = y+1, Ack = x+1+A. Data, length B

TCP


Client
Server
ACK. Seq = x+1, Ack = y+1. Data, length A
ACK. Seq = y+1, Ack = x+1+A. Data, length B
ACK. Seq = x+1+A, Ack = y+1+B. Data, length C

TCP



Client
Server
ACK. Seq = x+1, Ack = y+1. Data, length A
ACK. Seq = y+1, Ack = x+1+A. Data, length B
ACK. Seq = x+1+A, Ack = y+1+B. Data, length C
ACK. Seq = y+1+B, Ack = x+1+A+C. Data, length D

TCP: retransmission

- if packet dropped, no ACK received, so resent
- if ACK dropped, packet resent, recipient ignores data and resends ACK

TCP: ending/aborting

- **FIN flag:** I will no longer send, but I'll receive (**end** transmission)
- **RST flag:** I will no longer send or receive (**abort** transmisson)

TCP attacks

TCP attacks

- TCP hijacking: inject data into connection

TCP attacks

- TCP hijacking: inject data into connection
  - data injection: spoof packets

TCP attacks

- TCP hijacking: inject data into connection
    - data injection: spoof packets
        - need sender’s sequence number

TCP attacks

- TCP hijacking: inject data into connection
  - data injection: spoof packets
    - need sender’s sequence number
    - can MITM/on-path attackers do this?

TCP attacks

- TCP hijacking: inject data into connection
  - data injection: spoof packets
    - need sender's sequence number
    - can MITM/on-path attackers do this?
  - RST injection: spoof an RST packet
    - terminate connection

TCP attacks

- TCP hijacking: inject data into connection
  - data injection: spoof packets
    - need sender’s sequence number
    - can MITM/on-path attackers do this?
  - RST injection: spoof an RST packet
    - terminate connection
- TCP spoofing: packets appear to come from different IP

TCP attacks

- no confidentiality or integrity
- defense against off-path attackers relies on choosing random sequence numbers

UDP (user datagram protocol)

- datagram: message sent in single layer 3 packet
- no reliability (best effort), but adds ports
- faster than TCP, no 3-way handshake
  - used in high-speed applications—games, streaming, etc.
- attack: easy to spoof, no sequence numbers

worksheet

(on 161 website)

TLS (transport layer security)

- built atop TCP
- goal: confidentiality, integrity, authenticity

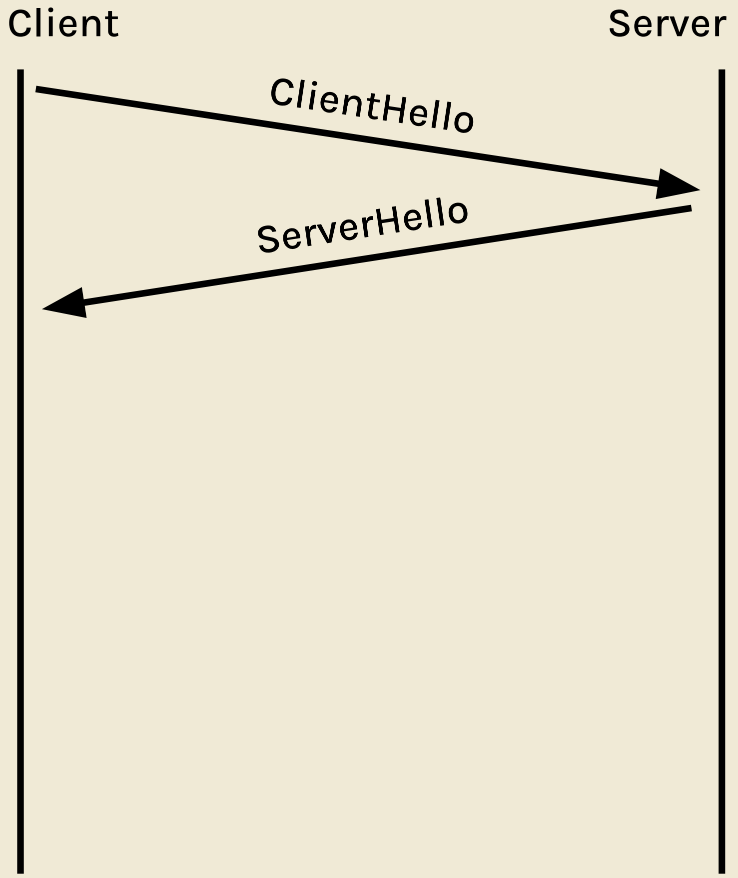
TLS (transport layer security)

- built atop TCP
- goal: confidentiality, integrity, authenticity
  - what do these mean in a TCP context?

TLS handshake

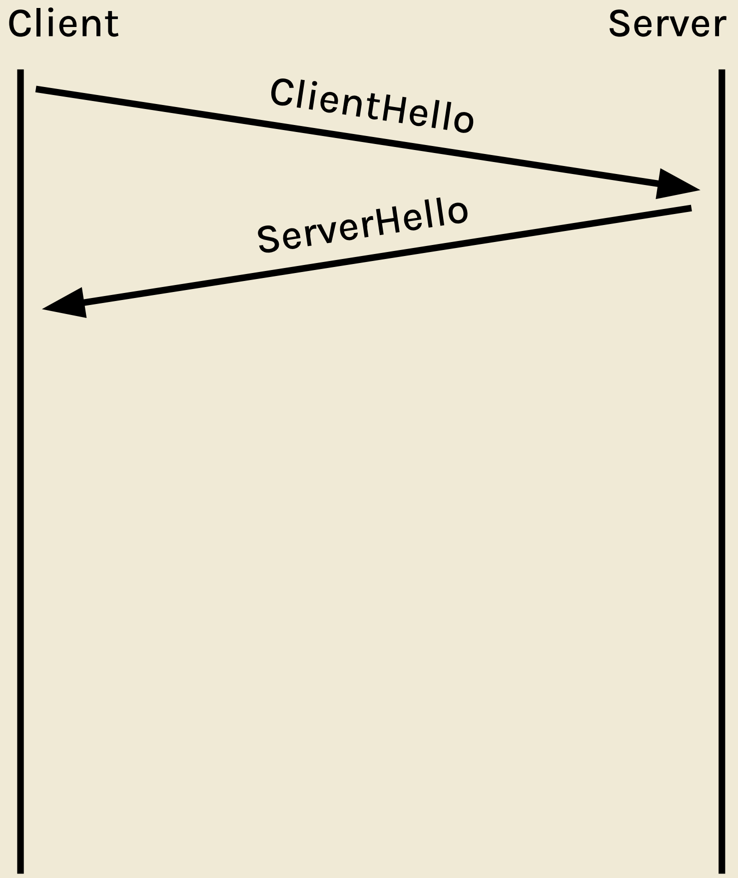
1. exchange hellos

- clientHello: R_B (256-bit “client random”)
- serverHello: R_S (256-bit “server random”)



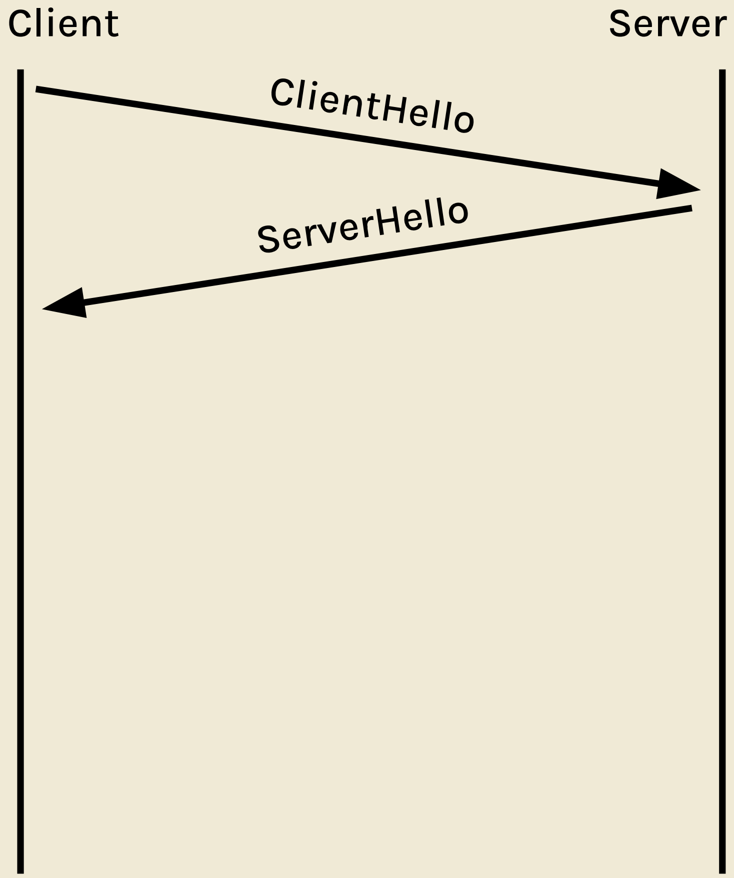
1. exchange hellos

- clientHello: *RB* (256-bit "client random")
- serverHello: *RS* (256-bit "server random")
- prevent replay attacks



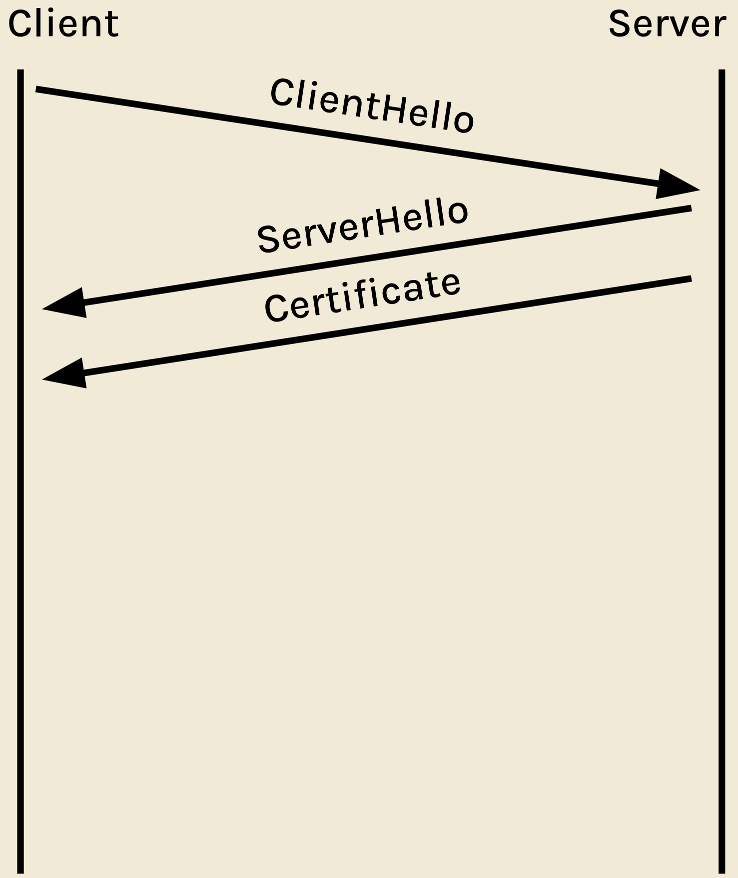
1. exchange hellos

- clientHello: R_B (256-bit "client random")
- serverHello: R_S (256-bit "server random")
- prevent replay attacks
  - two handshakes never exactly identical



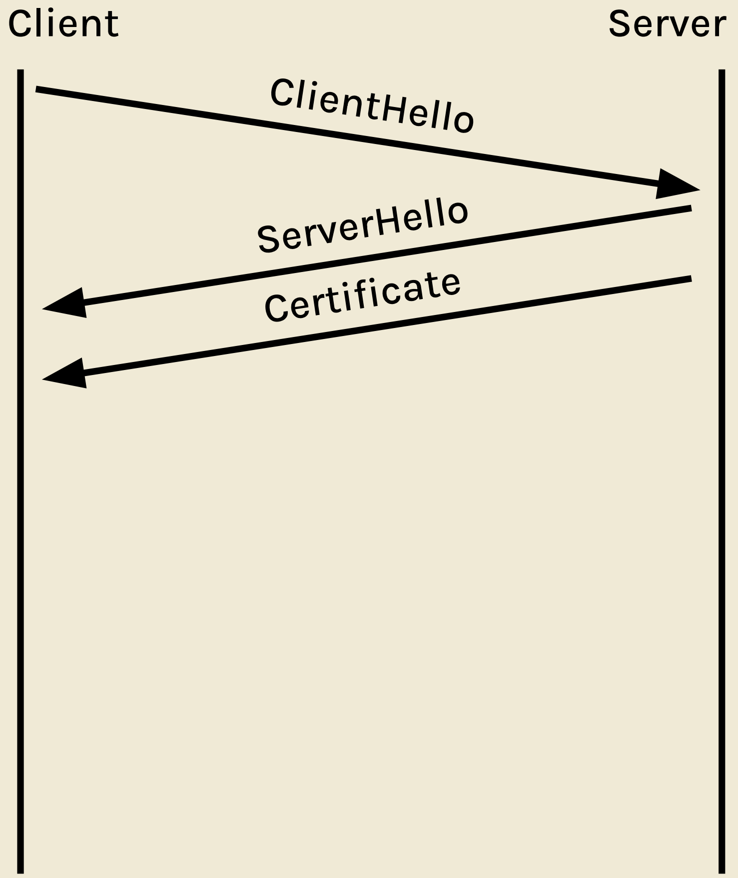
2. certificate

- server sends certificate, client validates certificate



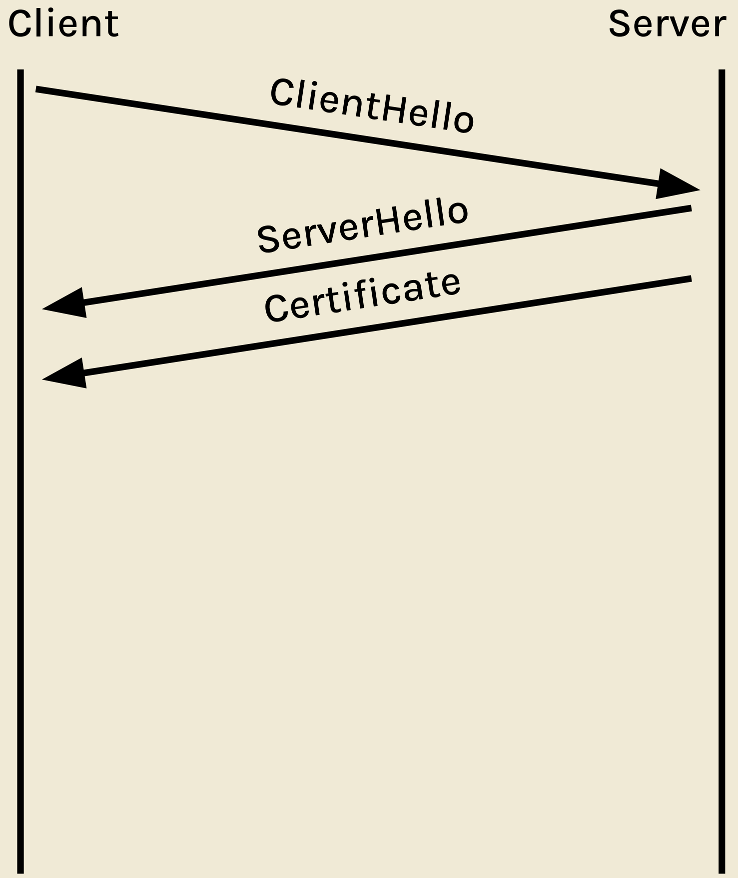
2. certificate

- server sends certificate, client validates certificate
- does the client know they’re talking to the server now?



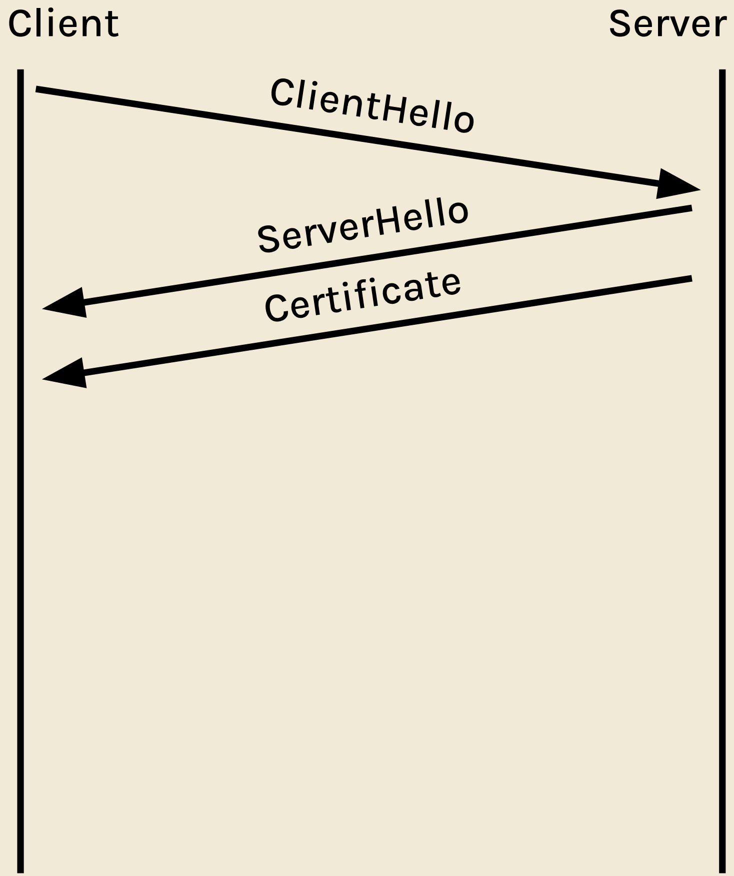
2. certificate

- server sends certificate, client validates certificate
- does the client know they're talking to the server now?
    - no, certificates are public



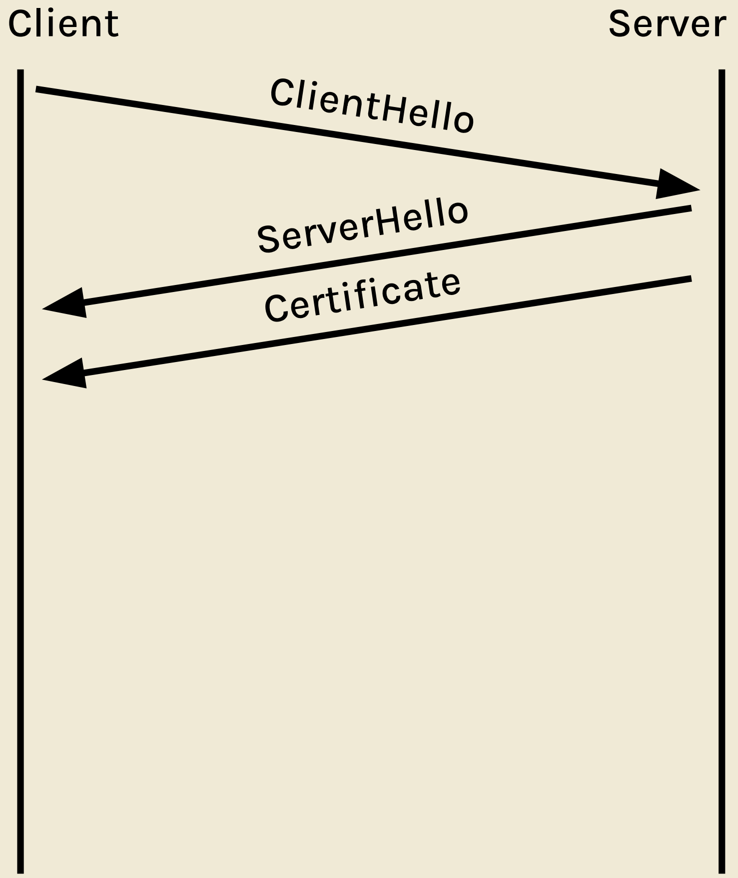
2. certificate

- server sends certificate, client validates certificate
- does the client know they’re talking to the server now?
  - no, certificates are public
- now knows server’s public key



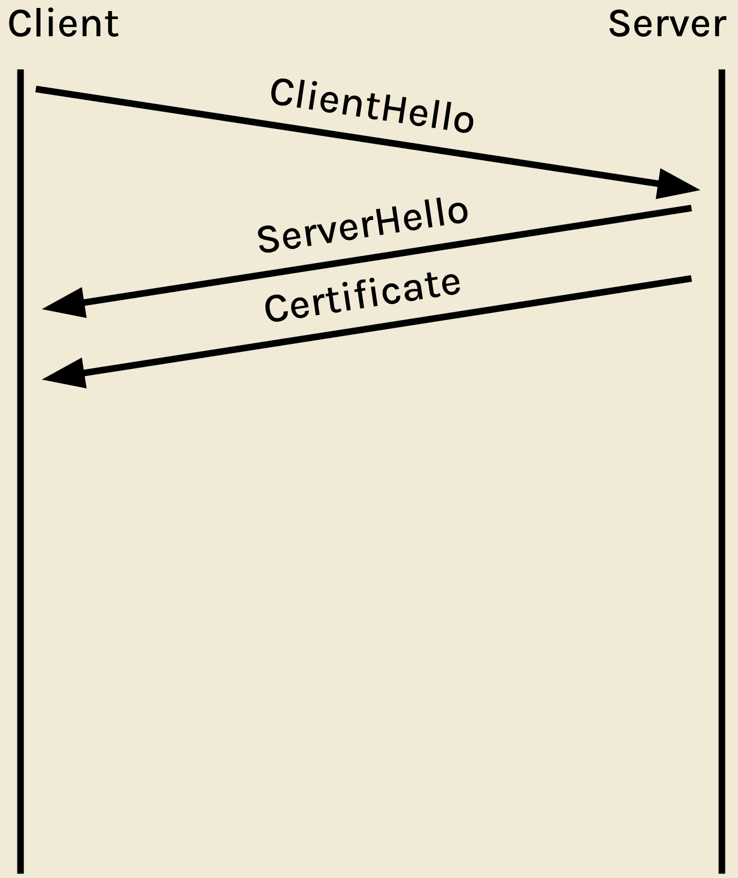
3. premaster secret

- make sure client is talking to legitimate server



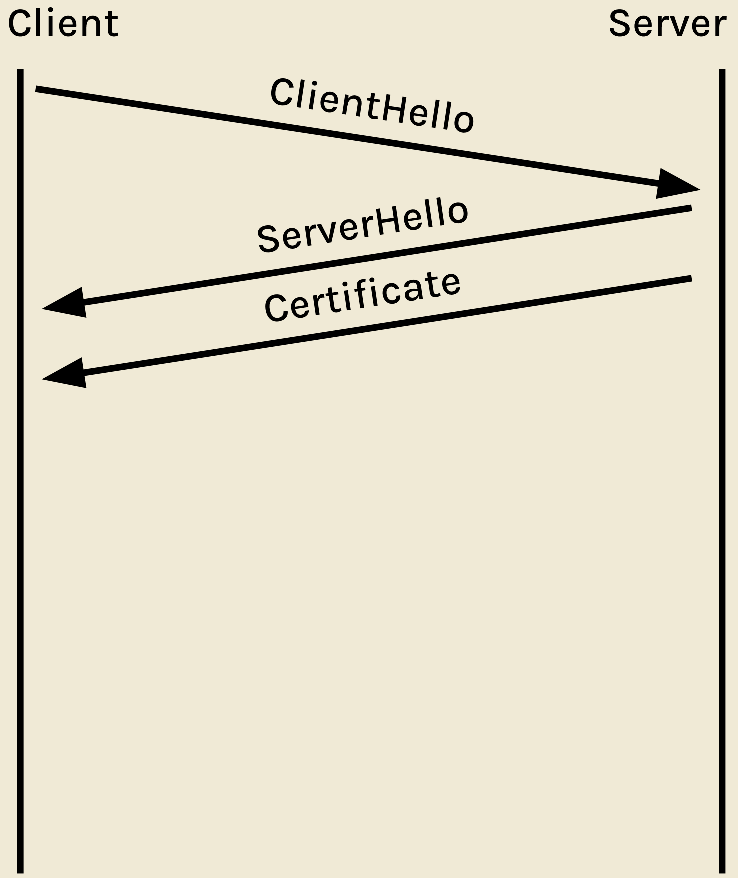
3. premaster secret

- make sure client is talking to legitimate server
- give the client and server a shared secret



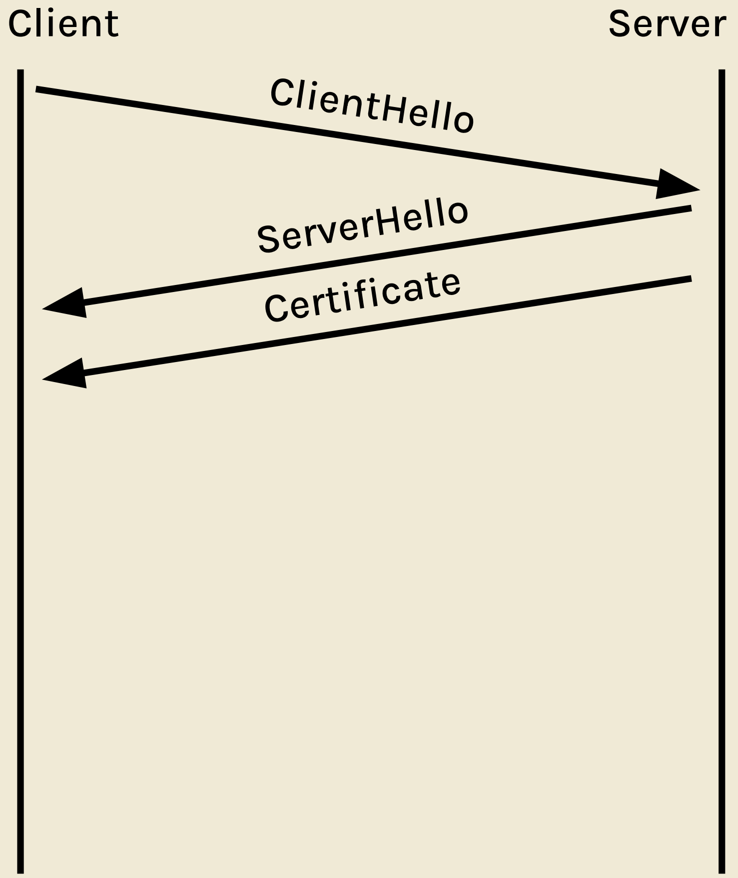
3. premaster secret

- make sure client is talking to legitimate server
- give the client and server a shared secret
- two approaches

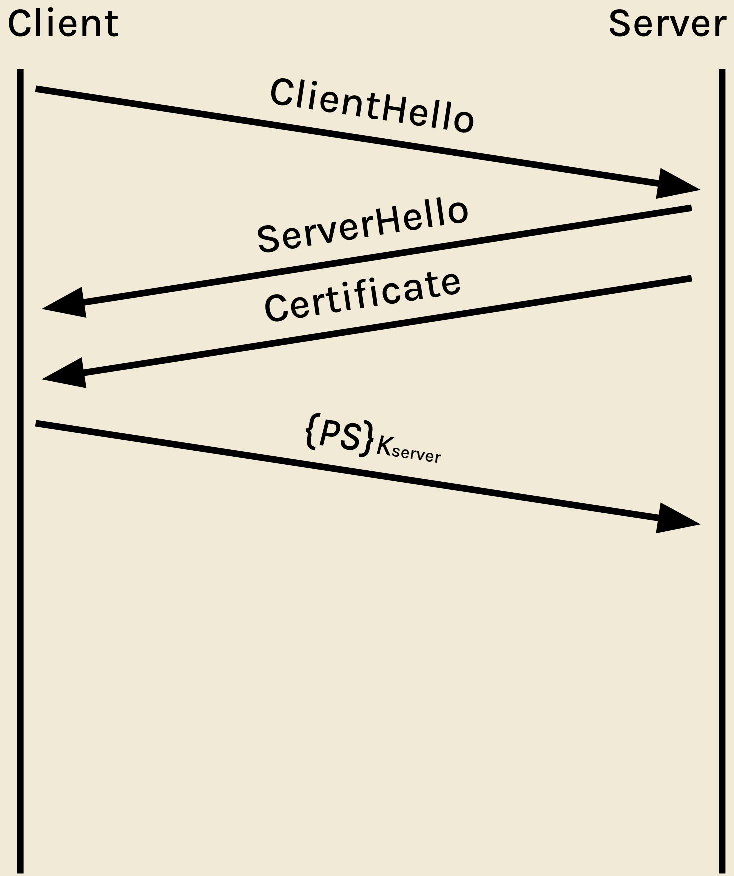


3. premaster secret

- make sure client is talking to legitimate server
- give the client and server a shared secret
- two approaches
  - RSA or DHE (diffie-hellman)

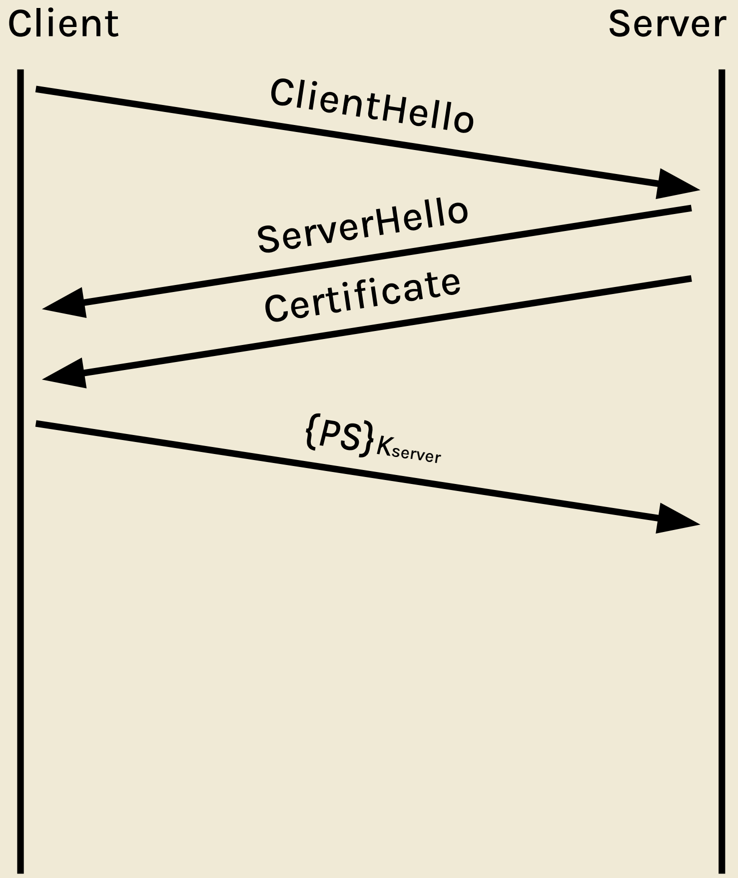


3. premaster secret (RSA)



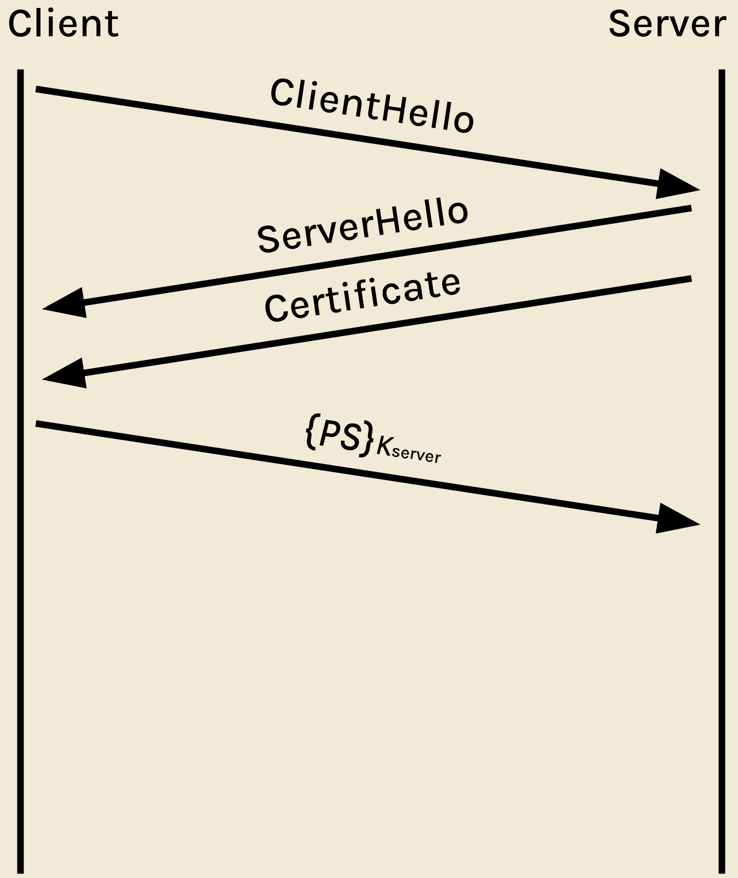
3. premaster secret (RSA)

- client randomly generates premaster secret (PS) and encrypts with server's public key

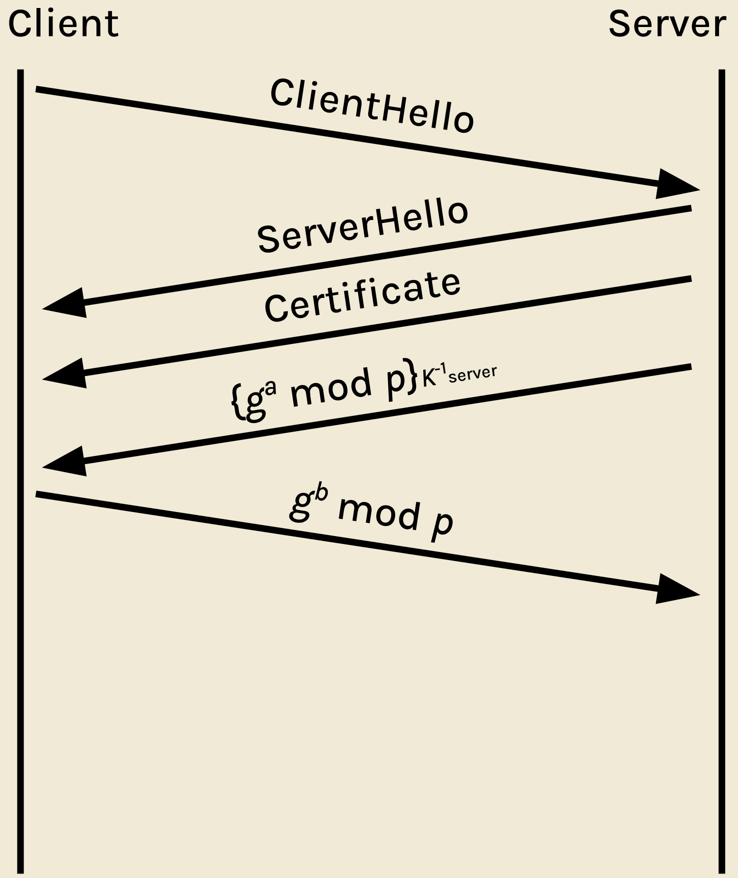


3. premaster secret (RSA)

- client randomly generates premaster secret (PS) and encrypts with server's public key
- server decrypts PS with RSA private key

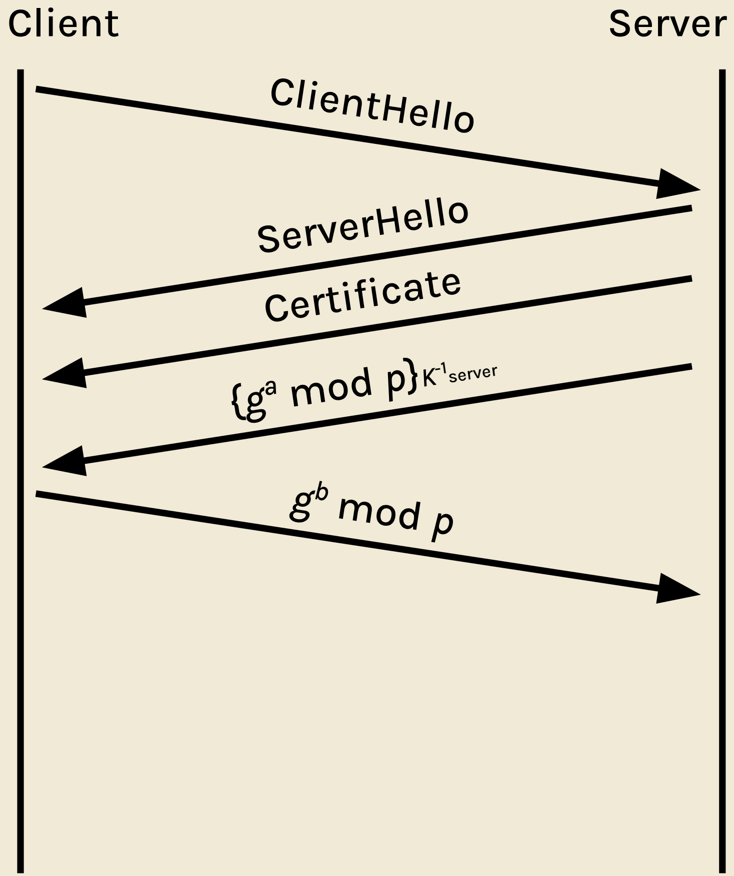


3. premaster secret (DHE)



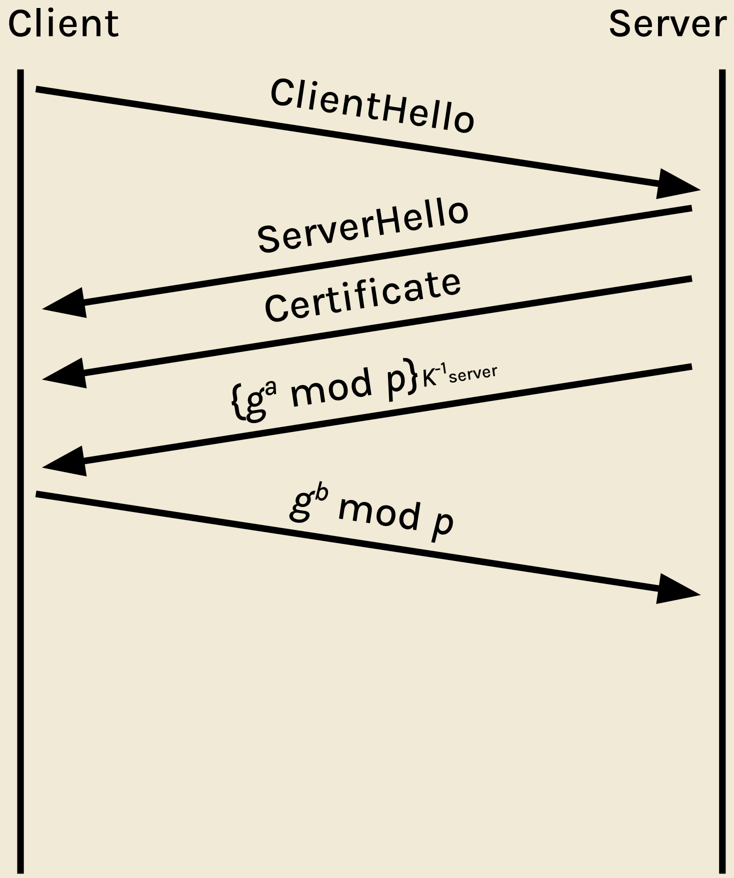
3. premaster secret (DHE)

- a, b random



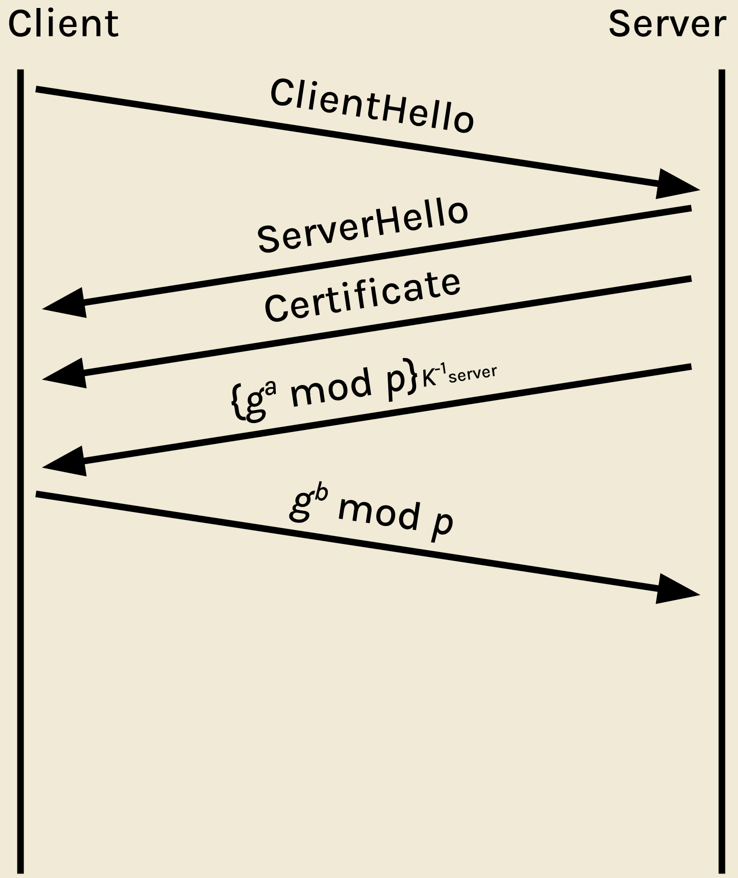
3. premaster secret (DHE)

- a, b random
- server generates $\{g^a \bmod p\}_{K^{-1}\text{server}}$



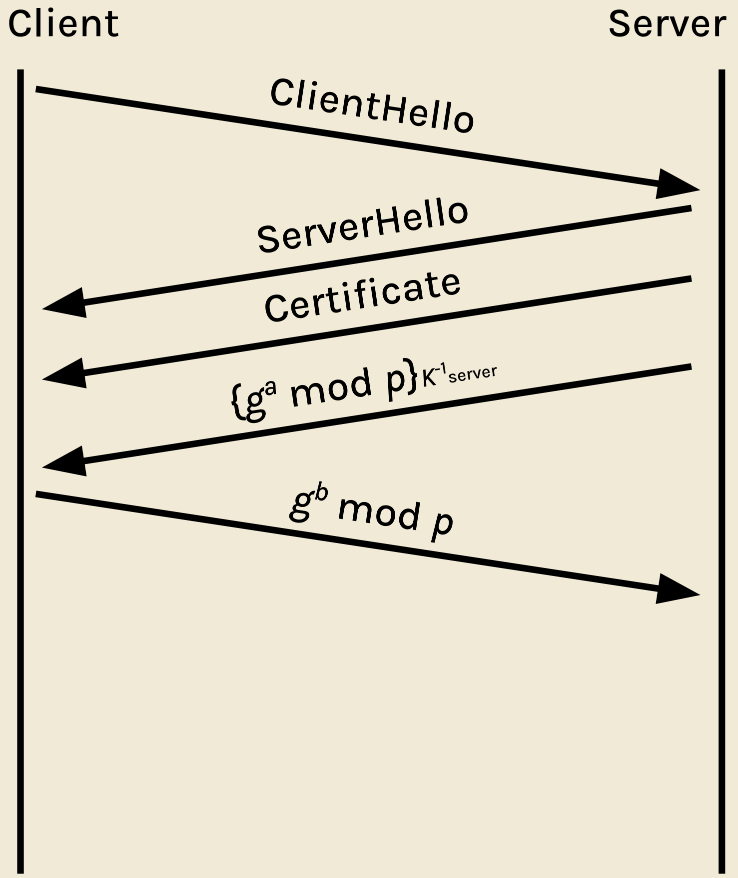
3. premaster secret (DHE)

- a, b random
- server generates $\{g^a \bmod p\}_{K^{-1}server}$
- client verifies signature, sends $g^b \bmod p$



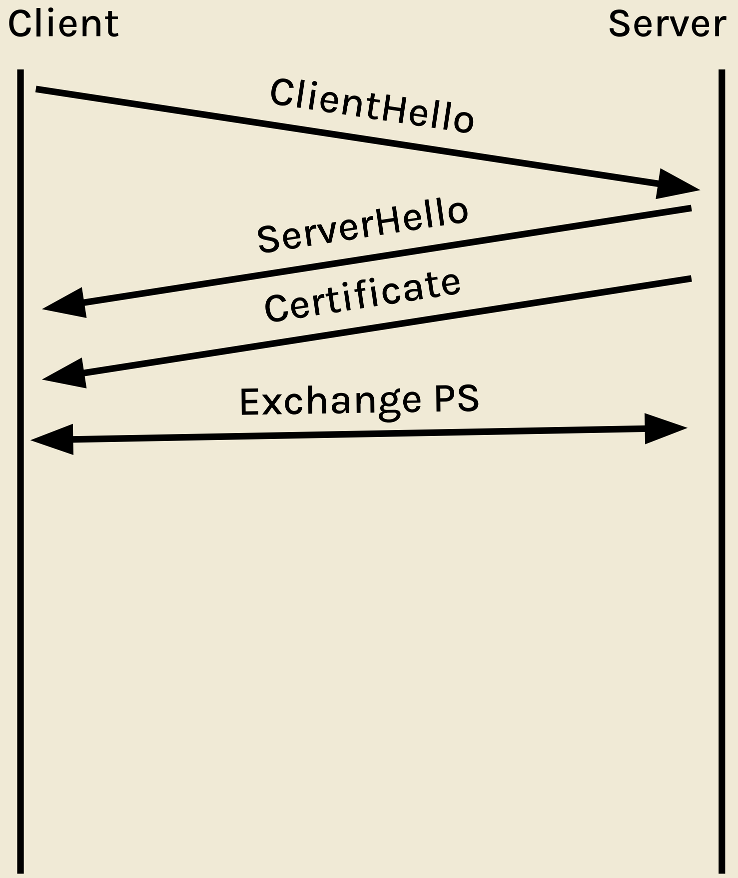
3. premaster secret (DHE)

- a, b random
- server generates $\{g^a \bmod p\}_{K^{-1}\text{server}}$
- client verifies signature, sends $g^b \bmod p$
- premaster secret: g^{ab} mod p



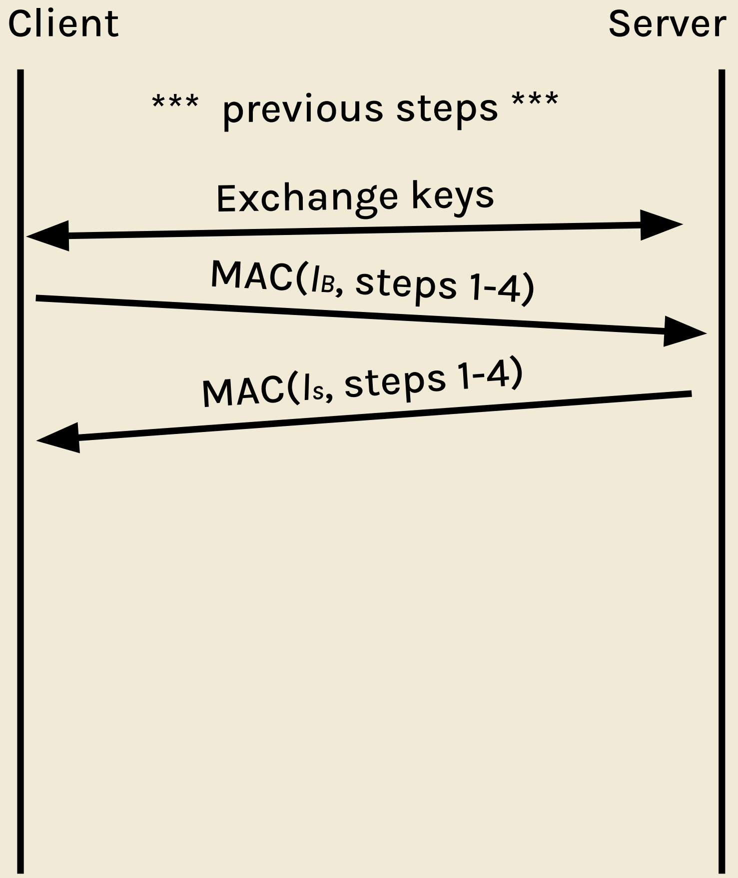
4. derive keys

- derive keys from R_B, R_S, and PS
- derive 4 symmetric keys
  - C_B: encrypt client-to-server
  - C_S: encrypt server-to-client
  - I_B: MAC client-to-server
  - I_S: MAC server-to-client
  - <u>client and server know all 4 keys</u>



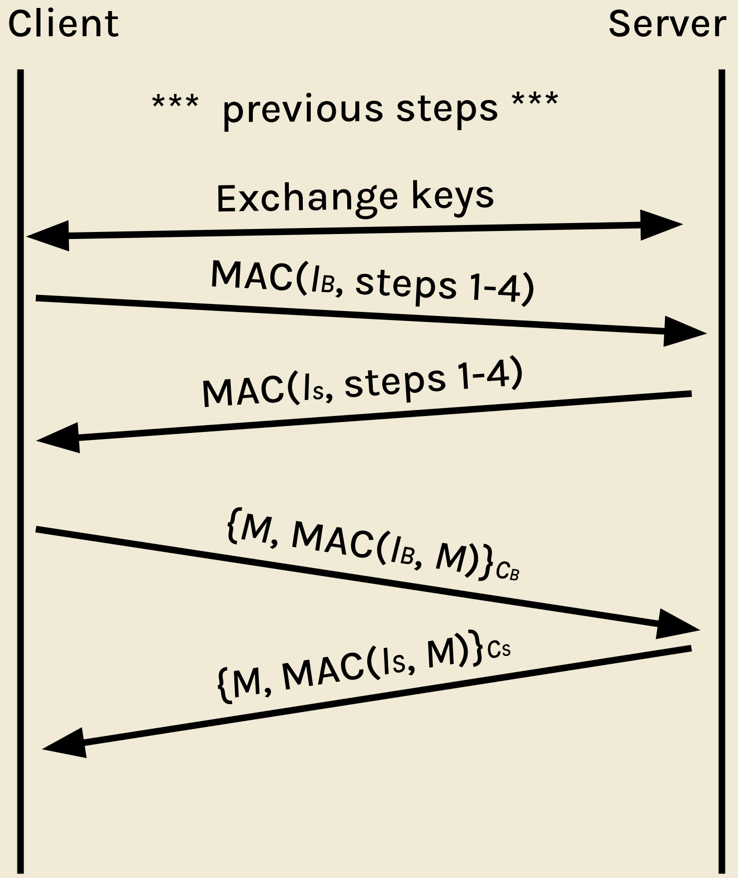
5. exchange MACs

- exchange MACs to ensure integrity of previous steps



6. send messages

- messages MACed then encrypted
  - not the best, but TLS uses legacy method
- confidential and ensures integrity!



hack of the day

- [Apache Pulsar TLS vulnerability led to MITM]
  - “used by thousands of companies for…instant messaging, data integrations…managing hundreds of billions of events per day.”
  - server certificate verification occurred AFTER authentication credentials sent
  - attacker can impersonate client

forward secrecy

forward secrecy

- if attacker records a connection and compromises secret values later, cannot compromise recorded connection

forward secrecy

- if attacker records a connection and compromises secret values later, cannot compromise recorded connection
- RSA TLS: no forward secrecy
  - can decrypt PS if server’s private key compromised

forward secrecy

- if attacker records a connection and compromises secret values later, cannot compromise recorded connection
- RSA TLS: no forward secrecy
    - can decrypt PS if server’s private key compromised
- DHE TLS: forward secrecy!
    - PS deleted after session over, keys can’t be learned

TLS applications

TLS applications

- provides end-to-end security

TLS applications

- provides end-to-end security
- cannot provide anonymity (TCP shows IPs) or availability (attacker can inject RST packets)

TLS applications

- provides end-to-end security
- cannot provide anonymity (TCP shows IPs) or availability (attacker can inject RST packets)
- HTTPS: HTTP over TLS

end to end principle

- ensuring reliability or security of a system at low levels may not be worth it
- you can provide these guarantees end to end instead

Original paper: https://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf

end to end principle

- examples
  - TCP provides reliability, even though your router or your ISP’s cables could fail
  - TLS provides security, despite attacks like ARP spoofing or packet injection existing at the lower level

worksheet

(on 161 website)

feedback

bit.ly/abhifeedback

slides: **bit.ly/cs161-disc**