Secure Sockets Layer (SSL) / Transport Layer Security (TLS)

Brad Karp
UCL Computer Science

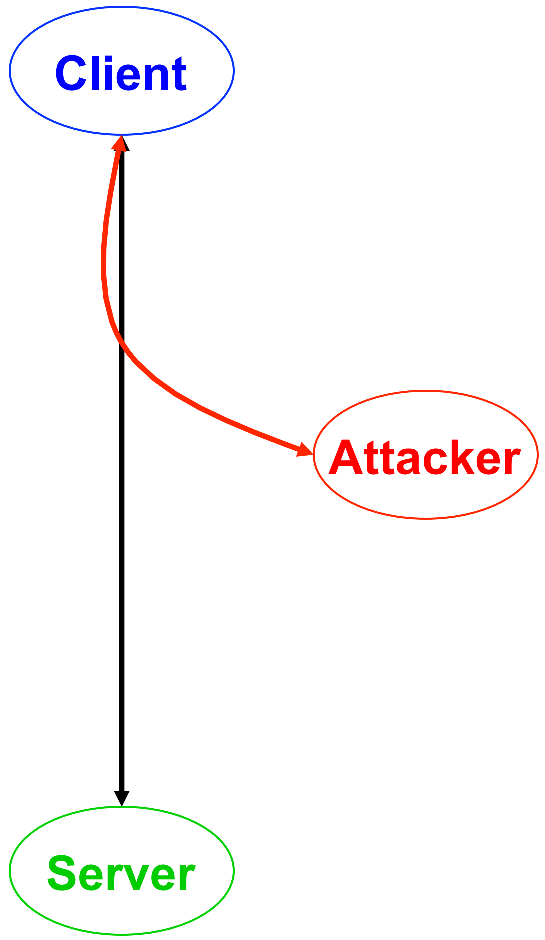
CS GZ03 / M030
2nd December, 2009

What Problems Do SSL/TLS Solve?

- Two parties, client and server, not previously known to one another
  - i.e., haven't been able to establish a shared secret in a secure room
- Want to authenticate one another
  - in today's lecture, focus on client authenticating server; e.g., "am I talking to the real amazon.com server?"
- Want secrecy and integrity of communications in both directions

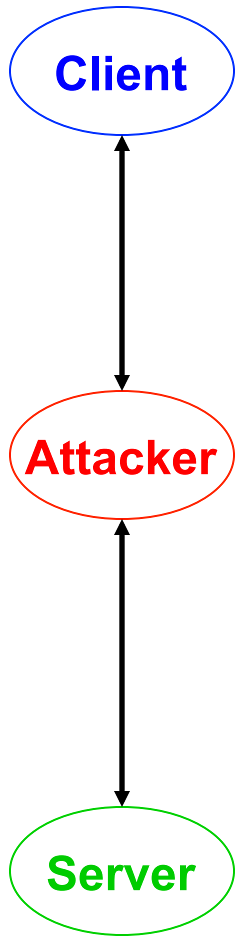
Problem: Man in the Middle Attacks

- Recall: public-key cryptography alone not enough to give robust authentication
  - Client can ask server to prove identity by signing data
  - But how does client know he has real server's public key?
- Attacker may impersonate server
  - Gives client his own public key, claiming to be server
  - Client may send sensitive data to attacker
  - Attacker may send incorrect data back to client



Man in the Middle Attacks (2)

- Attacker may not appear like server
  - e.g., might not have same content as real web server's page
- Solution: attacker acts as **man in the middle**
  - Emulates server when talking to client
  - Emulates client when talking to server
  - Passes through most messages as-is
  - Substitutes own public key for client's and server's
  - Records secret data, or modifies data to cause damage



Challenge: Key Management

- Publish public keys in a well-known broadcast medium
  - e.g., in the telephone directory, or in the pages of the New York Times
  - How do you know you have the real phone directory, or New York Times?
  - How can software use these media?
- Exchange keys with people in person
- "Web of trust": accept keys for others via friends you trust (used by PGP)

Approach to Key Management: Offline Certification Authorities (CAs)

- Idea: use digital signatures to indicate endorsement of binding between principal and public key
  - i.e., if I sign {amazon.com, pubkey}, I am stating, "I attest that amazon.com's public key is pubkey."
- Certification Authority (CA): third-party organization trusted by parties that wish to mutually authenticate
- Each CA has public/private key pair: K_{CA}, K_{CA}^{-1}
- CA creates certificate C_S for server S containing, e.g.,:
  - info = {"www.amazon.com", "Amazon, Inc.", www.amazon.com's public key, expiration date, CA's name}
  - sig = $\{H(\text{info})\}_{K_{CA}^{-1}}$
- Server S can present C_S to browser
- If browser knows K_{CA}, can validate that CA attests that S's public key is K_S

Approach to Key Management: Offline Certification Authorities (CAs)

- Idea: use digital signatures to indicate endorsement of binding between principal and public key

Key benefit: CA need not be reachable by C or S at time C wishes to authenticate S!

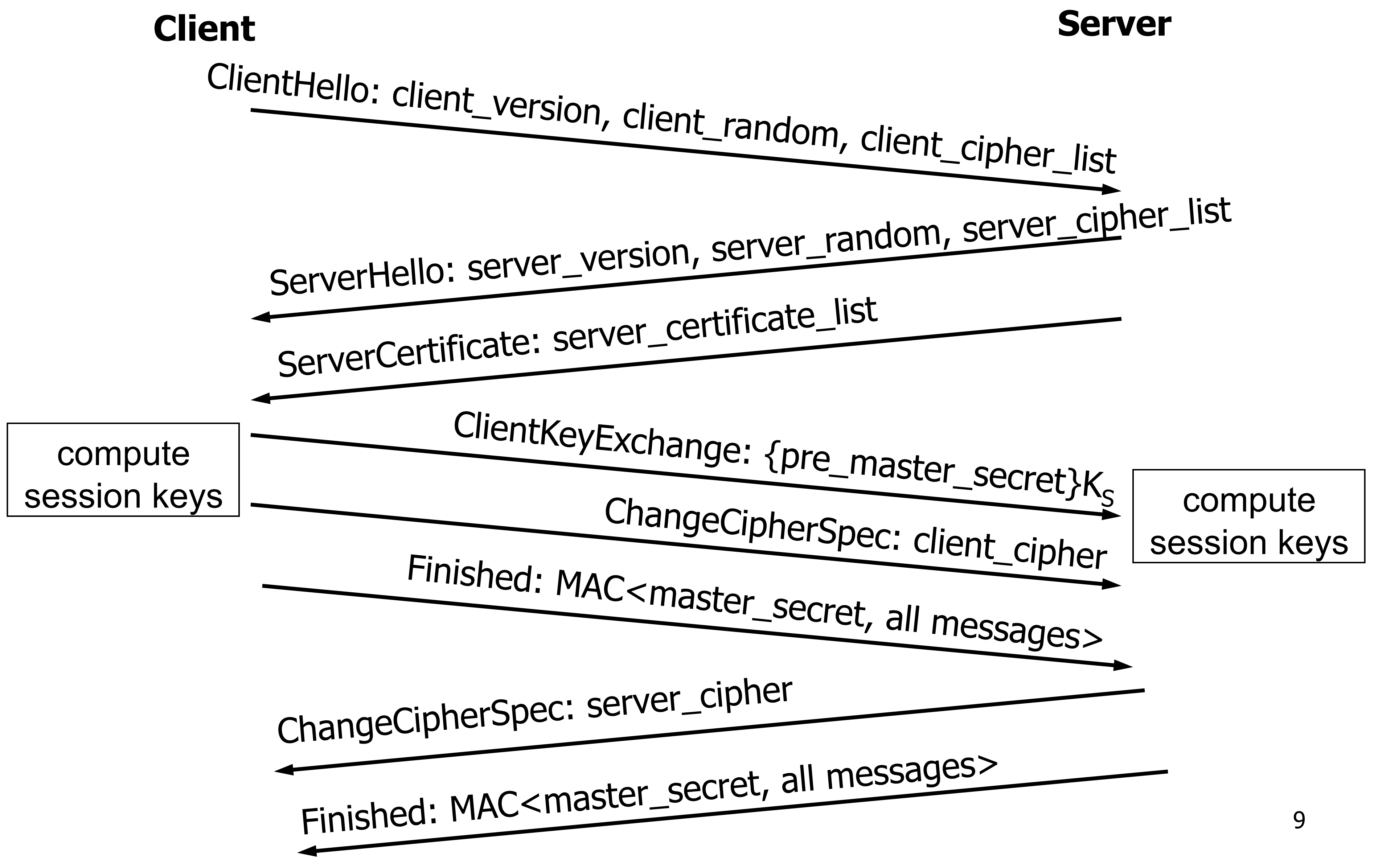
CAs and certificates are the heart of SSL's authentication mechanism

- CA creates certificate C_S for server S containing, e.g.,:
  - info = {"www.amazon.com", "Amazon, Inc.", www.amazon.com's public key, expiration date, CA's name}
  - sig = $\{H(\text{info})\}_{K_{CA}^{-1}}$
- Server S can present C_S to browser
- If browser knows K_{CA}, can validate that CA attests that S's public key is K_S

Offline Certification Authorities (2)

- Key benefit: CA need not be reachable by C or S at time C wishes to authenticate S!
  - Hence offline certification authority
- SSL/TLS model for browsers authenticating web servers:
  - Everybody trusts CA
  - Everybody knows CA’s public key (i.e., pre-configured into web browser)

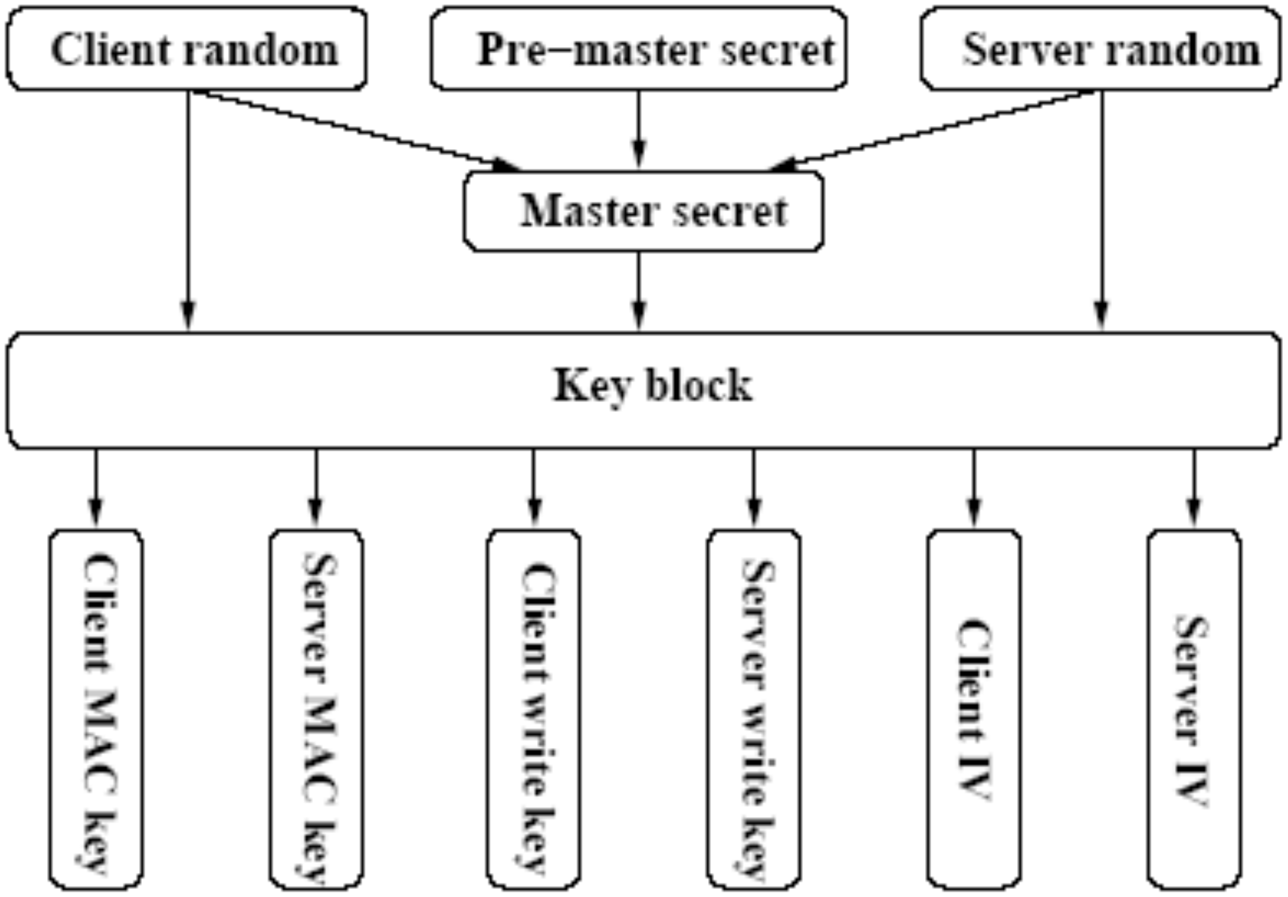
SSL 3.0 Handshake Overview


Client
Server
ClientHello: client_version, client_random, client_cipher_list
ServerHello: server_version, server_random, server_cipher_list
ServerCertificate: server_certificate_list
compute session keys
ClientKeyExchange: {pre_master_secret}K_S
compute session keys
ChangeCipherSpec: client_cipher
Finished: MAC<master_secret, all messages>
ChangeCipherSpec: server_cipher
Finished: MAC<master_secret, all messages>

Establishing Session Keys

- Client randomly generates pre-master secret, sends to server encrypted with server's public key
- Server also contributes randomness in server_random
- Using both pre-master secret and server_random, server and client independently compute symmetric session keys:
  - Client MAC key
  - Server MAC key
  - Client Write key
  - Server Write key
  - Client IV
  - Server IV

Establishing Session Keys (2)



[*SSL and TLS,* Eric Rescorla]

Using Session Keys to Send Data

- Data encrypted by client and server using each’s own write key
- Data MAC’ed by client and server using each’s own MAC key
- Each SSL record (block) includes a sequence number for that sender, and a MAC over:
  - Sequence number
  - Data plaintext
  - Data length

Why MAC Data Length?

- Plaintext padded to fit symmetric cipher block length
- Length of data (without padding) must be sent to receiver
- SSL 2.0 didn’t MAC data length; only MAC’ed padded data itself
  - Active adversary could change plaintext data length field
  - MAC over data would still verify
  - **Attacker could truncate plaintext as desired!**

Why MAC Data Length?

- Plaintext padded to fit symmetric cipher

Lesson:

Always MAC "what you mean," including all context used to interpret message at receiver

- SSL 2.0 didn't MAC data length; only MAC'ed padded data itself
  - Active adversary could change plaintext data length field
  - MAC over data would still verify
  - **Attacker could truncate plaintext as desired!**

Properties Provided by SSL (1)

- Secrecy: passive eavesdropper can't decrypt data; pre-master secret encrypted with server's public key, and server's private key secret
- Authentication of server by client: can trust each data record came from server that holds private key matching public key in certificate
- Authentication of client by server? Not without client certificates…or client can send username/ password over encrypted SSL channel
- Key exchange can't be replayed; new random nonce from each side each time

Properties Provided by SSL (2)

- Data from earlier in session can't be replayed
  - Caught by MAC
- Fake server can't impersonate real one using real certificate and public key
  - Doesn't know real server's private key, so can't decrypt pre-master secret from client
- Fake server obtains own certificate for own domain name from valid CA, supplies to client
  - If domain name differs from one in https:// URL, client detects mismatch when validating certificate

Forward Secrecy

- Suppose attacker records entire communication between client and server
- At later time, attacker obtains server's private key
- If attacker cannot decrypt data from recorded session, scheme provides forward secrecy
- Does SSL 3.0 provide forward secrecy?
  - No.

Cipher Roll-Back

- SSL supports various ciphers of various key lengths and strengths
- Suppose attacker modifies cipher selection messages, to force client and server into using weak ciphers
- Each direction of handshake ends with MAC of all messages
- Can attacker adjust this MAC so it verifies?
  - No. Doesn't know master_secret!

What Is CA Actually Certifying?

- That a public key belongs to someone authorized to represent a hostname?
- That a public key belongs to someone who is associated in some way with a hostname?
- That a public key belongs to someone who has many paper trails associated with a company related to a hostname?
- That the CA has **no liability?**
- >100-page Certification Practice Statement (CPS)!

How to Get a VeriSign Certificate

- Pay VeriSign ($300)
- Get DBA license from city hall ($20)
  - No on-line check for name conflicts; can I do business as Microsoft?
- Letterhead from company (free)
- Notarize document (need driver's license) (free)
- Easy to get fradulent certificate
  - Maybe hard to avoid being prosecuted afterwards…
- But this is just VeriSign's policy
  - **many** other CAs…

CA Security

- How trustworthy is a VeriSign certificate?

  In mid-March 2001, VeriSign, Inc., advised Microsoft that on January 29 and 30, 2001, it issued two. . . [fraudulent] certificates. … The common name assigned to both certificates is “Microsoft Corporation.”

  VeriSign has revoked the certificates. . . . However. . . it is not possible for any browser’s CRL-checking mechanism to locate and use the VeriSign CRL.

  – Microsoft Security Bulletin MS01-017