SSL

The Handshake Layer in v3.3 = TLS 1.2

Prof. Clarkson
CSci 6545, Fall 2013
Unilateral Authentication
Client Hello

- TLS version
- Timestamp
- Client random nonce $N_C$
- List of ciphersuites
- ...  \textit{MitM attack might cause selection of least secure ciphersuite!}
Server Hello

- TLS version
- Timestamp
- Server random nonce $N_s$
- Single ciphersuite
  // we'll assume RSA chosen for key exchange
- ...

Thursday, October 10, 13
Server Certificate

• $\text{cert}(S, K_S)$

  Binds server to RSA encryption key $K_S$

  Might actually be certificate chain
Server Hello Done

- (nothing)
Client Key Exchange

- Enc(PMS; $K_S$)

PMS = PreMasterSecret, 48 random bytes

Note: use SecureRandom when generating randomness for crypto yourself!
Master Secret

- MasterSecret MS computed from PMS, client & server nonces:
  \[ \text{TLSHash}(\text{PMS}, N_C, N_S) \]

- Session crypto state derived from MS (encryption keys, MAC keys, IVs)

- Underlying record layer uses authenticated encryption
Client Finished

- [with AuthEnc] TLSHash(MS, all previous messages in handshake)
Server Finished

- [with AuthEnc] TLSHash(MS, all previous messages in handshake)
Connection Finished

• All further records continue to use AuthEnc
Mutual Authentication
Client hello
Server hello
Server certificate
Certificate request
Server hello done
Client certificate
Client key exchange
Certificate verify
Client finished
connection secured
Server finished
Certificate Request

• Specification of what kinds of certificates server is willing to accept

• Acceptable certificate authorities

• Acceptable crypto algorithms
Client Certificate

Binds client to RSA verification key $K_C$

Might actually be certificate chain
Certificate Verify

- \[\text{Sign}(\text{all previous messages in handshake}; K_C)\]
“Anonymous”
Authentication
Don’t!

- Vulnerable to man-in-the-middle attacks
- *Might* be acceptable if used as part of more complex protocol that has another means to ensure authentication