
SSL, SSH

1 June 2011
Lecture 13

Slide Credits: Steve Zdancewic (UPenn)

Topics for Today

- SSL
- SSH

Secure Sockets Layer

- One real world application for the techniques we have discussed so far: Secure Sockets Layer (SSL)
 - Or Transport Layer Security (TLS) Protocol
- Designed by Netscape in 1996
 - Adapted by IETF
 - Now in RFC 5246 – TLS 1.2 in Aug 2008
 - Many extensions and outside applications
- Most important use is on the web (HTTP)
 - Commonly called HTTP**S**
 - SSL has **no relation** to HTTP, however

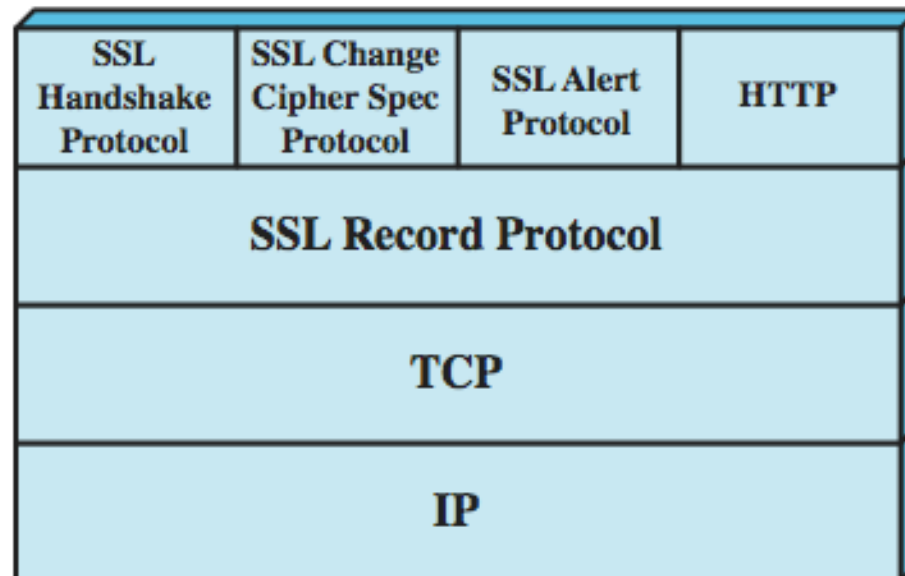
Secure Sockets Layer

Main goal: Establish a secure communication channel between two computers

- We've been talking about this the whole semester, so what's so hard?
 - Different operating systems (easy)
 - Different cryptographic services (harder)
 - Different versions (harder)
 - No Trusted Third Party (?)
 - One side may not have any authentication tokens (harder)
- Also:
 - It must be efficient
 - Must be flexible
 - It must be exportable
 - Online negotiation (!)

Secure Sockets Layer

- Solution: Add another layer in the protocol stack on top of TCP
 - Well, two layers really
 - Several sub-protocols too

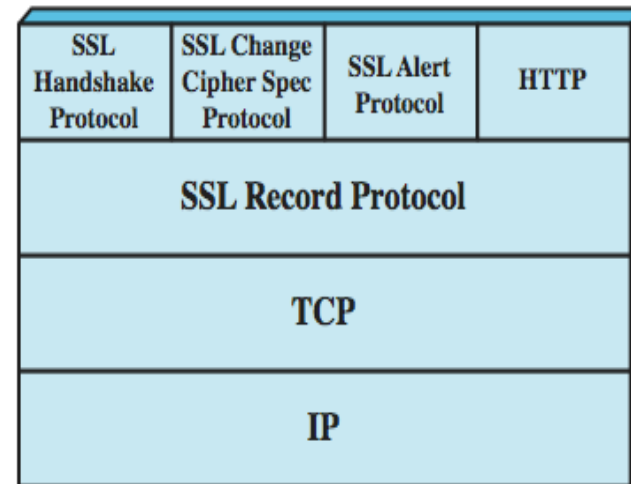


Sessions and Connections

- Setting up a secure conversation involves online negotiation
 - Expensive! 2 RTTs minimum...
- Web content is sent in a series of Requests
 - Each request (connection) gets 1 item
 - HTTP 1.1 changes this a bit
 - That shouldn't mean we negotiate for each request!
- Solution: Long running **Sessions** and short lived **Connections**
 - Do the negotiation once for the session
 - Make many connections on the same session

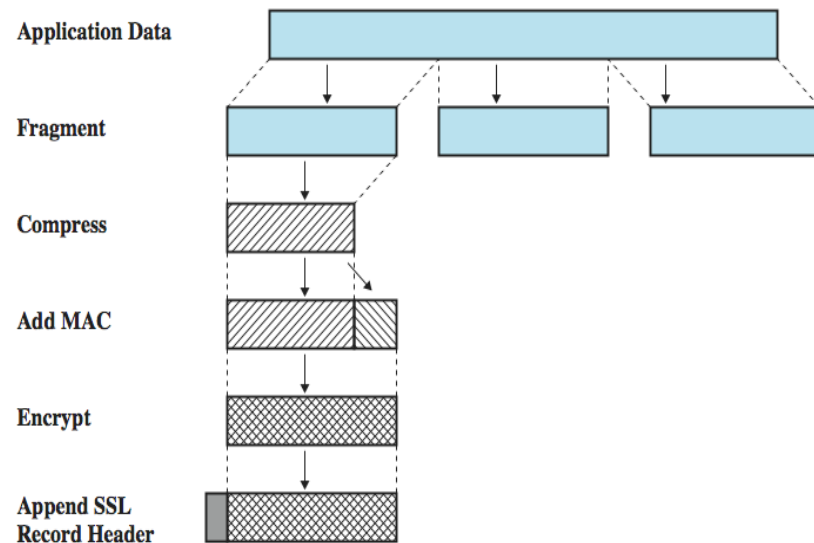
The SSL Protocols

- SSL Record Protocol
 - Move data
- SSL Handshake Protocol
 - Negotiate security decisions
- SSL Change Cipher Spec
 - Activate the negotiated security decisions
- SSL Alert Protocol
 - !



SSL Record Protocol

1. Fragment packets into 2^{14} bytes or less (16,384)
2. Compress (if you want)
3. Message Authentication Code
 - (Keyed) Hash
4. Encrypt
5. Append Header
 - Content Type (Protocol)
 - Change Cipher Spec
 - Alert
 - Handshake
 - Application_Data
 - Major Version
 - Minor Version
 - Compressed length

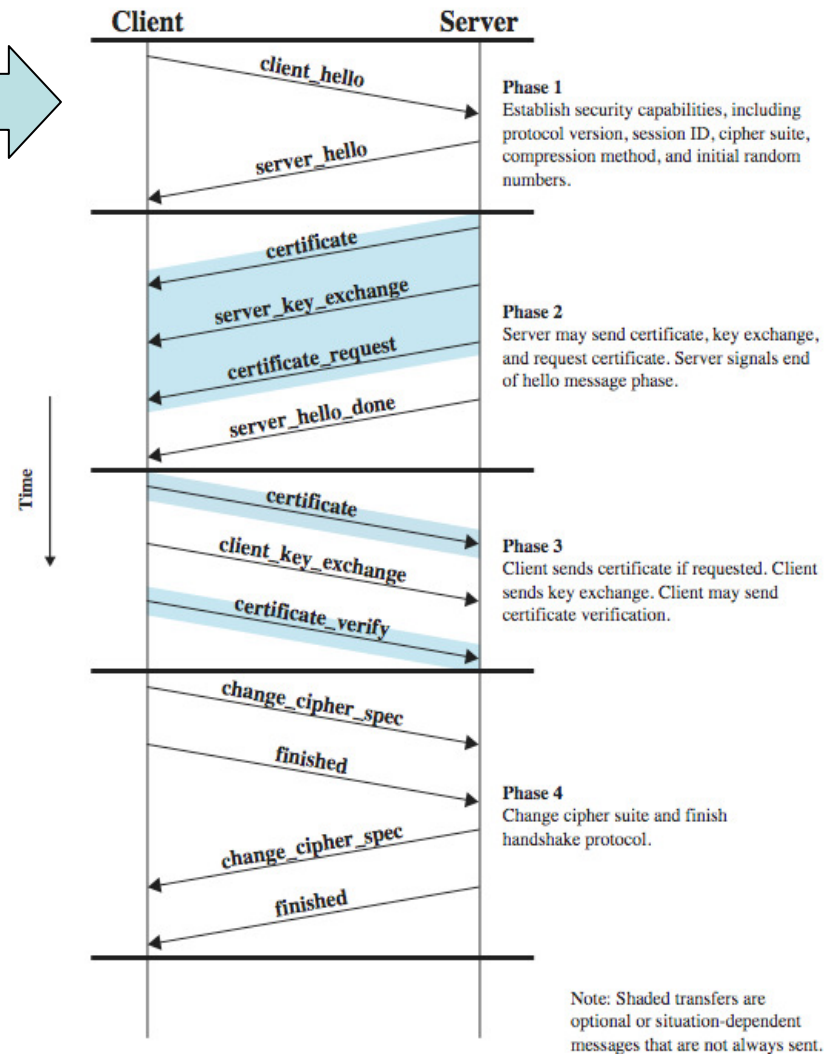


SSL Handshake Protocol

- Does the negotiation
- Four phases:
 1. Establish **client** security capabilities
 2. Establish **server** security tokens
 3. Establish **client** security tokens
 4. Implement the negotiated decisions
 - Change Cipher Spec

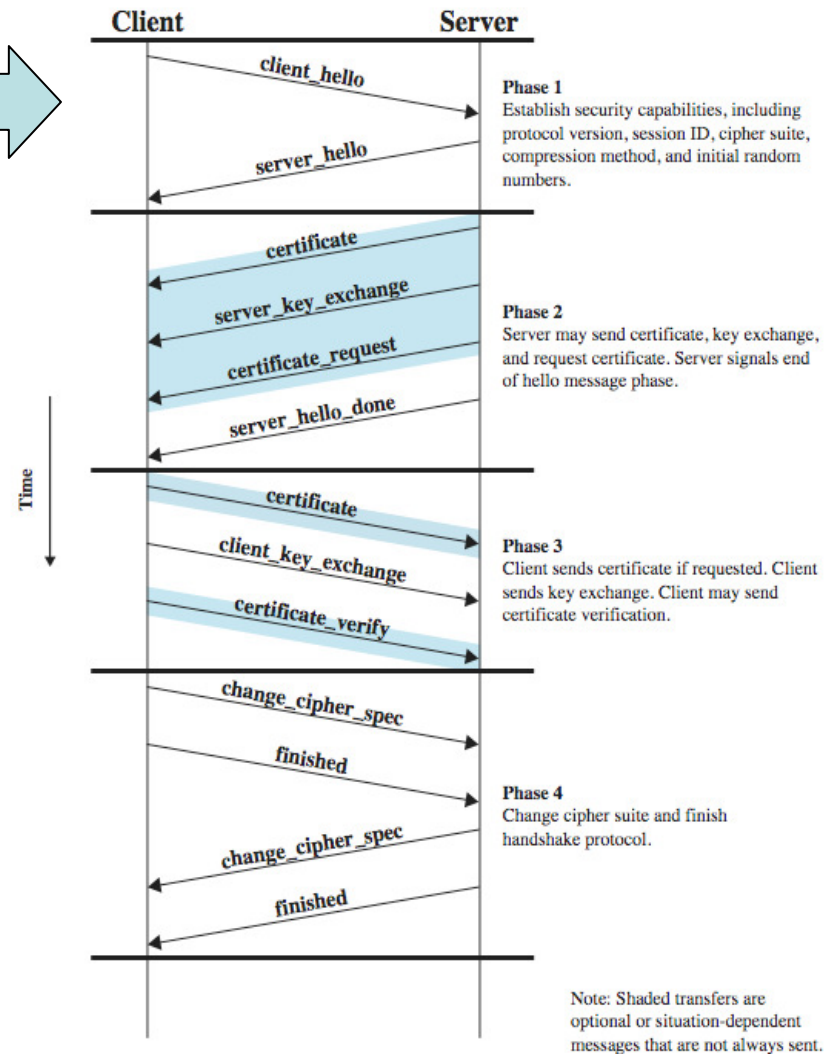
SSL Handshake Protocol

- Phase 1: Client Starts
 - (Highest) SSL Version
 - Client Nonce: n_c
 - Session Id
 - If it's 0 – a new session
 - If it's not – continue a session
 - Cipher Suite
 - List of crypto algorithms supported
 - In order of preference
 - Compression Method
 - List of supported methods
- Client waits...



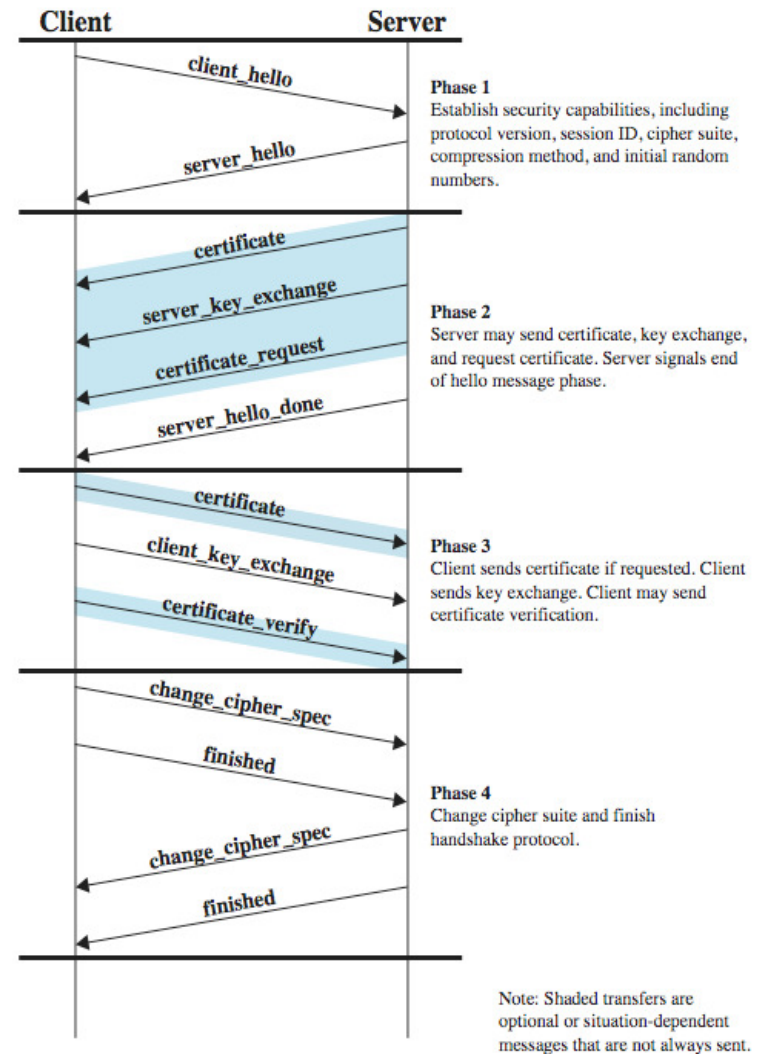
SSL Handshake Protocol

- Phase 1: Server Responds
 - Chosen SSL Version
 - Server nonce: n_s
 - Session Id
 - Old one if continuing
 - Chosen Cipher Suite
 - Chosen Compression Method
- Phase 2: Server tokens
 - Server Certificate
 - (Optional) Request Client Certificate
 - Server_Hello_Done



SSL Handshake Protocol

- Phase 3: Client tokens
 - Client verifies certificate
 - Client sends security tokens
- Certificate (Optional)
 - Signs previous messages with Certificate private key (Client Verify)
- If no certificate: Pre-master secret (48 bits)
 - Encrypted with Server Key

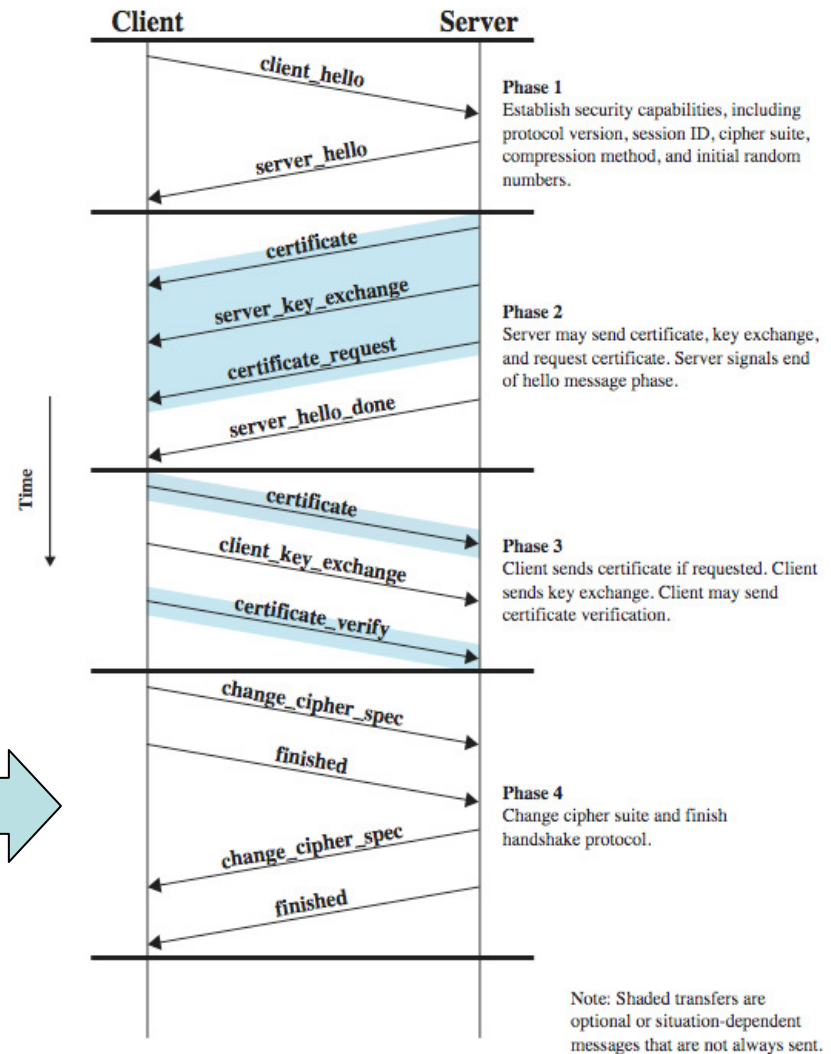


Pre-master Secret

- Using Pre-master Secret (PMS)
 - 48 bit random number
 - Combined with n_c and n_s to make a full secret
- Old Algorithm: master_secret =
 - MD5(PMS + SHA('A' + PMS + n_c + n_s)) +
 - MD5(PMS + SHA('BB' + PMS + n_c + n_s)) +
 - MD5(PMS + SHA('CCC' + PMS + n_c + n_s));
- New Algorithm: master secret is defined per cipher suite
 - Varying length supported by iterated (and concatenated) hashes
 - Based on SHA_256

SSL Handshake Protocol

- Phase 4: Implement
 - Client sends: Change Cipher Spec
 - Server sends: Change Cipher Spec
- Both indicate they are ready to use what has been negotiated



SSL Change Cipher Spec

- Simple protocol: 1 message with 1 byte of data
 - Byte set to 1
- Tells the other side to implement the agreed upon cipher suite

SSL Alert Protocol

- Two bytes of data
- Byte 1: Severity of alert
 - =1: Warning
 - =2: Fatal (terminates connection)
- Byte 2: Alert Codes
 - Examples: Close notify, Decompression failure, Bad certificate, Certificate revoked, Illegal parameter, Decode error, Insufficient security

Reflection: SSL

- Enables secure communication over the internet
- Works even if only one side has a certificate
 - Client authentication must be done some other way
- Main application for certificates and PKI
 - Has helped sell many certificates
- Secures the communication channel
 - But not the data stored on the other side
 - A thief can still steal your credit card information from the server
 - Has made it harder for governments to spy on web traffic

So Far

- SSL
- SSH

Secure Shell (SSH)

- Secure Shell (SSH) is a program to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another.
- It provides **strong authentication and secure communications over unsecure channels**.
- It is intended as a replacement for telnet, rlogin, rsh, and rcp.

SSH protocol (overview)

- See: <http://www.snailbook.com/protocols.html>
 - RFC's: 4250,4251,4252,4253,4254
- Connection Setup / Version number exchange
- Session key exchange / Server Authentication
 - Each side sends a list of preferred algorithms (e.g. Diffie-Hellman with certain parameters)
 - Guess which algorithm is used by other side
 - Optimistically send first message of key exchange (if guess is wrong the recipient will ignore it)
 - Key exchange produces a shared key K and exchange hash H (used as a session identifier)
 - Server authenticates by signing the hash H

Conclusion

- SSL
- SSH