



Secure Socket Layer (SSL)



SSL

- Originally from Netscape
- De facto standard security protocol used for web security
 - All good web servers support it
 - All popular browsers support it
 - Ex. - SSL is invoked when you open a secure site with `https://...`
- Can be used for other applications as well
- IETF has proposed TLS (Transport Layer Security, RFC 2246) which is based on and similar to SSL



- Layer on top of TCP
 - TCP – reliable, end-to-end stream
 - TCP + SSL – authenticated, reliable, end-to-end, encrypted stream
- Port depends on application layer using SSL (ex. 443 for HTTP over SSL, 465 for SMTP over SSL)
- Main objectives:
 - Authentication using digital certificates
 - Integrity using MAC (hash + symmetric encryption)
 - Confidentiality using symmetric key encryption/decryption
 - Secure key exchange using public key cryptography (RSA, Diffie-Hellman etc.)



- Actually 2 layers on TCP
 - SSL Record Protocol on top of TCP, carries all SSL data, provides basic security services like hashing, encryption/decryption etc.
 - Three protocols on top of this
 - SSL Handshake – negotiate cryptographic parameters at the beginning, authenticate server and client
 - SSL Change Cipher Spec – make negotiated changes come into effect
 - SSL Alert – notify errors to other party



Connection and Session

- Connection
 - a peer-to-peer transport connection
 - Each connection associated with a session
- Session
 - Defines a set of cryptographic security parameters
 - Created by the Handshake protocol
 - Can be shared by multiple connection, avoids negotiation for each connection



Session State

- Session identifier
- Compression method
- Cipher Spec – specifies different cryptographic parameters such as encryption algo, hash algo, hash size etc. that are associated with this session
- Master Secret – a 48 byte secret shared between client and server
- Is-resumable – flag to indicate if this session can be used to initiate new connections



Connection State

- Server generated random no. and client generated random no.
- Server write MAC secret and Client write MAC secret – MAC secret key used for data written by server and client respectively (different)
- Server write key and client write key – secret key used for encryption for data written by server and client respectively (different)
- Some other info...



SSL Record Protocol

- Provides two types of services
 - Confidentiality
 - Uses symmetric cryptosystem, with secret key defined by handshake protocol
 - Integrity
 - Using MAC, with shared secret key defined again by handshake protocol



Steps

- Breaks messages into fragments of size 16KB or less
- May compress the message
- Computes MAC over the compressed data (MD5 or SHA-1 hash + symmetric encryption)
- Encrypts message + MAC using symmetric key encryption (with different secret key than the one used for MAC)
- Adds a header and transmits it
 - Content-type: identifies higher layer protocol (handshake, alert etc.)
 - Major and minor versions – SSL version info
 - Compressed length – length of plaintext (or compressed plaintext) fragment



SSL Change-Cipher-Spec Protocol

- Simplest of the 4
- Just one message, with a single byte with value 1
- Causes the pending (being negotiated) cipher specs between client and server to take effect on this connection



SSL Handshake Protocol

- Most complex among the four
- Allows server and client to authenticate themselves to each other
- Negotiates encryption and MAC algorithms to be used (note: both use symmetric cryptosystem)
- Negotiates and exchanges secret key to be used for encryption and MAC
- Handshake is the first protocol to be used before any application data exchange
- Has four phases



Phase 1: Establish security capabilities

- Negotiates protocol version, session id, cipher suite, compression method etc.
- Client sends “client-hello” with following fields:
 - Version – SSL version of client
 - Random – random number
 - Session id
 - non-zero : update parameters of an existing connection or create a new connection on this session
 - Zero : open a new connection on a new session
 - Cipher Suite – list of cryptographic algo.s supported by client, in decreasing order of preference
 - Compression methods: list of compression methods supported by client



- Server replies with “server-hello” with same fields
 - Version – lower of client’s and server’s supported version
 - Random – another random number
 - Session-id – copied from client if non-zero, else generate and send a new session id
 - Cipher suite – single cipher suite selected by server from those sent by client
 - Compression – compression method selected by server
- Client and server now agrees on methods, but still need to authenticate and securely exchange keys for encryption/MAC



Phase 2: Server Authentication

- Server sends its digital certificate (“certificate” message)
- Server may request client’s certificate (“certificate request” message)
- Sends a “server hello done” message to indicate end of this phase



Phase 3: Client Authentication

- Client first verifies server's certificate
 - Check expiry date
 - Verify that CA is trusted
 - Verify the digital signature of CA
 - Verify if domain name of server in certificate matches domain of actual server from where the certificate came from



- If server has requested certificate from client, client sends its certificate in a “certificate” message
- Sends a “client key exchange message”
 - Contents depends on key exchange method negotiated
 - Ex. – if RSA is selected, client sends a 48 bit “pre-master secret” encrypted with the public key of server from server’s certificate



Phase 4: Finish

- Client uses change-cipher-spec protocol to send a message and copies the negotiated cipher specs to current cipher spec to be used
- Client sends “finish” message
- The server then sends the same two messages to client
- Application data can now be exchanged



Secret key generations

- Note that we exchanged only one key using “client key exchange” message – called “pre_master_secret”
- Master key created at both sides from the pre_master and the random integers sent earlier
- Master key and the same random numbers are then used again to create the secret keys for MAC and encryption/decryption (40 bit or 128 bit)
- Total of 4 keys generated at each end (server write MAC, client write MAC, server write, client write)
- Since both sides use same pre-master secret, same random numbers, and same algorithm, keys generated are the same



SSL Alert Protocol

- Used to convey errors
- 2 byte messages
 - Byte 1 – warning or fatal. For fatal errors, connection terminated immediately; other connections on same session may continue
 - Byte 2 – code for specific alert
 - Ex. – bad_record_mac, no_certificate, bad_certificate, handshake_failure etc.