Transport Layer Security (TLS)

Practical 8

UCD School of Computer Science

November 2018

Transport Layer Security (TLS) - Introduction

Figure: TSL / SSL

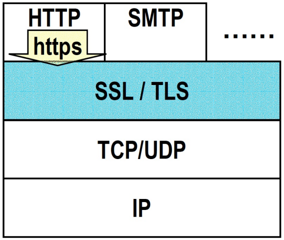
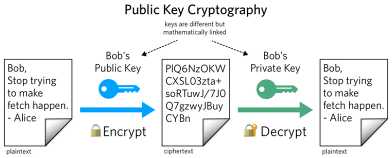


Figure: Transport Layer Security

- SSL/TLS: operates on top of TCP, but below application layer (can be considered as top sublayer for L4)
- IETF made SSL Version 3 an open standard in 1999 and called it TLS (Transport Layer Security) Version 1 (as described in RFC 2246)
- SSL/TLS plays a central role in the security and privacy needed for web commerce to work. This protocol is widely used to protect email servers (SMTP/POP/IMAP), chat servers (XMPP), remote login security (through SSH servers), instant messaging (IM) and some virtual private networks (SSL VPNs).
- Fundamental to the security that is established with the SSL/TLS protocol are the certificates issued by the Certificate Authorities (CA).

Certificate Authority



Public Key Cryptography
keys are different but mathematically linked
Bob's Public Key
Bob's Private Key
Bob, Stop trying to make fetch happen. - Alice
plaintext
Encrypt
PIQ6NzOKW CXSL03zta+ soRTuwJ/7J0 Q7gzwyJBuy CYBn
ciphertext
Decrypt
Bob, Stop trying to make fetch happen. - Alice
plaintext

Figure: Public Key Cryptography

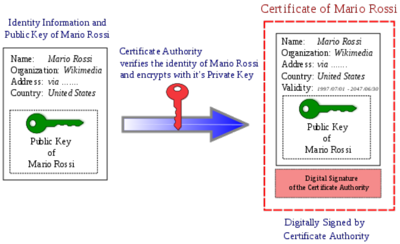
Public Key Cryptography

1. Alice uses bob's public to encrypt messages to him
2. Bob decrypt Alice's message using its private key

Problem:
How to ensure Bob's public key really belong to Bob?

Solution: Certificate authority

- Trusted entity that validate Bob identity



Identity Information and Public Key of Mario Rossi
Name: Mario Rossi
Organization: Wikimedia
Address: via ....
Country: United States
Public Key of Mario Rossi
Certificate Authority verifies the identity of Mario Rossi and encrypts with it's Private Key
Certificate of Mario Rossi
Name: Mario Rossi
Organization: Wikimedia
Address: via ....
Country: United States
Validity: 1997/07/01 - 2047/06/30
Public Key of Mario Rossi
Digital Signature of the Certificate Authority
Digitally Signed by Certificate Authority

Certificate Authority

- **Certificate** is an electronic document used to identify an individual, a server, a company, or some other entity and to associate that identity with a public key (like a driver's license, a passport, or other commonly used personal IDs, a certificate provides generally recognized proof of a person's identity).
- **Certificate authorities (CAs)** are entities that validate **identities** and issue certificates. CA can be either a public commercial entity (for example, Symantec, Lets Encrypt, etc.) or organizations running their own certificate-issuing server software (such as Red Hat Certificate System).
- Methods used to validate an **identity** depends on the policies of a given CA. Basically, the certificate issued by the CA binds a particular public key to the name of the entity that the certificate identifies (for example, it can be name of a server). Certificates help to prevent the use of "fake" public keys for impersonation. Only the public key certified by the certificate will work with the corresponding private key possessed by the entity identified by the certificate.

SSL/TLS Handshake

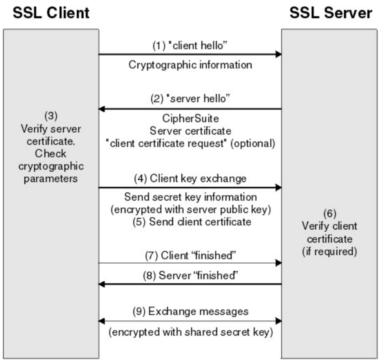

SSL Client
SSL Server
(1) "client hello"
Cryptographic information
(2) "server hello"
CipherSuite
Server certificate
"client certificate request" (optional)
(3) Verify server certificate. Check cryptographic parameters
(4) Client key exchange
Send secret key information (encrypted with server public key)
(5) Send client certificate
(6) Verify client certificate (if required)
(7) Client "finished"
(8) Server "finished"
(9) Exchange messages
(encrypted with shared secret key)

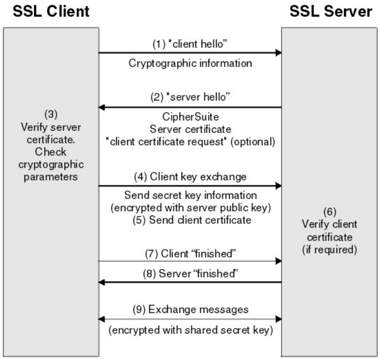
Figure: Overview of the SSL or TLS handshake

The SSL/TLS handshake ⇒ enables SSL/TLS client and server to establish **secret keys** with which they use to communicate.

Summary of steps:

- An agreement of the protocol version to use.
- Select cryptography algorithm to use.
- Authenticate each other by exchanging and validating digital certificates.
- Use **asymmetric** encryption technique to generate **shared secret keys**.
- Use the **shared key** for the symmetric encryption of messages.

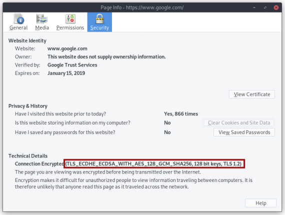
SSL/TLS Handshake - Cont



SSL Client
SSL Server
(1) "client hello"
Cryptographic information
(2) "server hello"
CipherSuite
Server certificate
"client certificate request" (optional)
(3) Verify server certificate. Check cryptographic parameters
(4) Client key exchange
Send secret key information (encrypted with server public key)
(5) Send client certificate
(6) Verify client certificate (if required)
(7) Client "finished"
(8) Server "finished"
(9) Exchange messages (encrypted with shared secret key)

Figure: Overview of the SSL or TLS handshake

1. Client sends a **client hello** message that includes some cryptographic information:
   - SSL/TLS version to use.
   - CipherSuites supported by the client.
   - A random byte string used in subsequent computations.
   - Data compression methods supported by the client.



Website Identity

Website: www.google.com
Owner: This website does not supply ownership information.
Verified by: Google Trust Services
Expires on: January 15, 2019

Privacy & History

Have I visited this website prior to today? Yes, 866 times
Is this website storing information on my computer? No
Have I saved any passwords for this website? No

Technical Details

Connection Encrypted (TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, 128 bit keys, TLS 1.2)
The page you are viewing was encrypted before being transmitted over the Internet.
Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.

Figure: SSL Ciphersuites

SSL/TLS Handshake - Cont

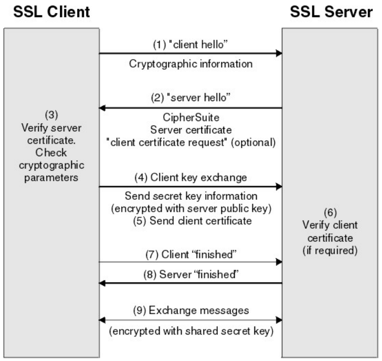
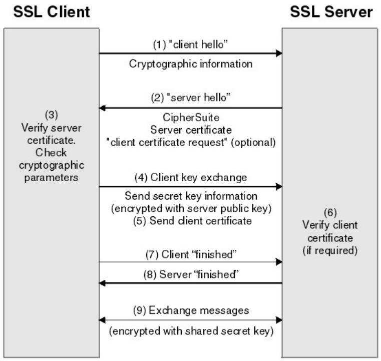

SSL Client
SSL Server
(1) "client hello"
Cryptographic information
(2) "server hello"
CipherSuite
Server certificate
"client certificate request" (optional)
(3) Verify server certificate. Check cryptographic parameters
(4) Client key exchange
Send secret key information (encrypted with server public key)
(5) Send client certificate
(6) Verify client certificate (if required)
(7) Client "finished"
(8) Server "finished"
(9) Exchange messages
(encrypted with shared secret key)

Figure: Overview of the SSL or TLS handshake

2. Server responds with a **server hello** message with following information:
    - CipherSuites chosen by the server from the list provided by the client.
    - Session ID.
    - Another random byte string.
    - Server's **digital certificate**
    - If server requires a digital certificate from client authentication, then it sends a **client certificate request** that includes a list of the types of certificates supported and acceptable CAs.
3. Client verifies the server's digital certificate.
4. Client then sends the random byte string that enables both client and server to compute the **secret key** used for encrypting subsequent message data. Random byte string is encrypted with provided server's public key.

SSL/TLS Handshake - Cont



SSL Client
SSL Server
(1) "client hello"
Cryptographic information
(2) "server hello"
CipherSuite
Server certificate
"client certificate request" (optional)
(3) Verify server certificate. Check cryptographic parameters
(4) Client key exchange
Send secret key information (encrypted with server public key)
(5) Send client certificate
(6) Verify client certificate (if required)
(7) Client "finished"
(8) Server "finished"
(9) Exchange messages
(encrypted with shared secret key)

Figure: Overview of the SSL or TLS handshake

5. If server sent a **client certificate request** $\rightarrow$ client then responds with a random byte string encrypted with client's private key, together with client's own digital certificate.
6. Server verifies the client's certificate (if required).
7. Client sends a **finished** message (that is encrypted with the secret key), to notify that the handshake at client side is complete.
8. Server sends a **finished** message (that is encrypted with the secret key), to notify that the handshake at client side is complete.
9. Server and client now can exchange messages that are symmetrically encrypted with the **shared secret key**.

ENJOY !!!