Putting It All Together

Securing a Communication Channel

The definition of a secure channel is one that establishes, between two programs, a data channel that has confidentiality, integrity and authenticity against a computationally-bounded network attacker.

This will involve the use of all concepts learnt so far. Let us see how we can secure a communication channel between Alice and Bob.com.

Step 1: Unilateral Authenticated Key Exchange

Alice and Bob.com will carry out unilateral authenticated key exchange using Bob's private and public key.

After authentication, both Bob and Alice know two randomly selected session keys (k, t), where k is the secret key of a symmetric-key encryption e.g. AES, and t is the secret key of a MAC.

Step 2: Authenticated Encryption

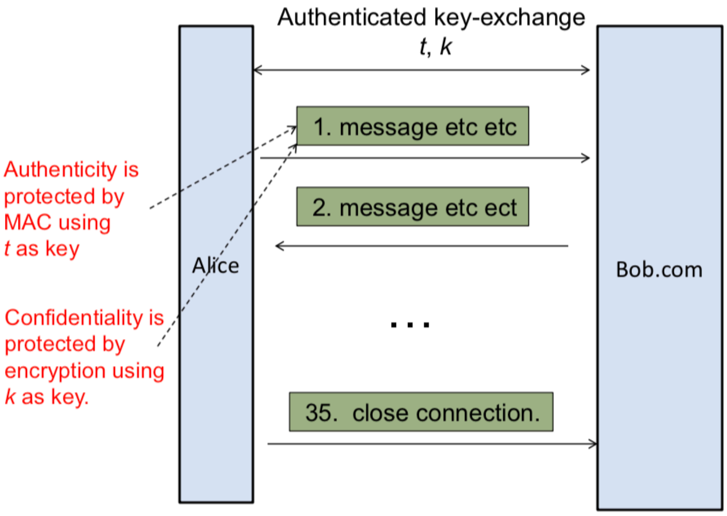
Subsequent communication between Alice and Bob.com will be protected by k, t and a sequence number i.

Suppose m_1, m_2, m_3, … are the sequence of messages exchanged, the actual data to be sent for m_i will be:

$$E_k (i \| m_i) \| MAC (E_k (i \| m_i))$$

where i is the sequence number and || refers to concatenation.

The above is known as "encrypt-then-MAC", while there are other variants of authenticated encryption such as "MAC-then-encrypt" and "MAC-and-encrypt".


Authenticated key-exchange
t, k
1. message etc etc
2. message etc ect
…
35. close connection.
Alice
Bob.com
Authenticity is protected by MAC using t as key
Confidentiality is protected by encryption using k as key.

There is a need for the sequence number as it acts as a nonce (my guess).

Use of PKI

For the above, PKI is often employed to distribute the public key. The authenticated key exchange is thus likely to involve certificates. After all, Alice needs to verify that the entity she is communicating with is indeed Bob.com.

<u>HTTPS</u>

HTTPS (Hypertext Transfer Protocol Secure) is widely used to secure Web traffic. It is built on top of SSL (Secure Sockets Layer) / TLS (Transfer Layer Security), i.e. HTTPS = HTTP + SSL. Hence HTTPS is also called HTTP over SSL or HTTP over TLS.

<u>SSL / TLS</u>

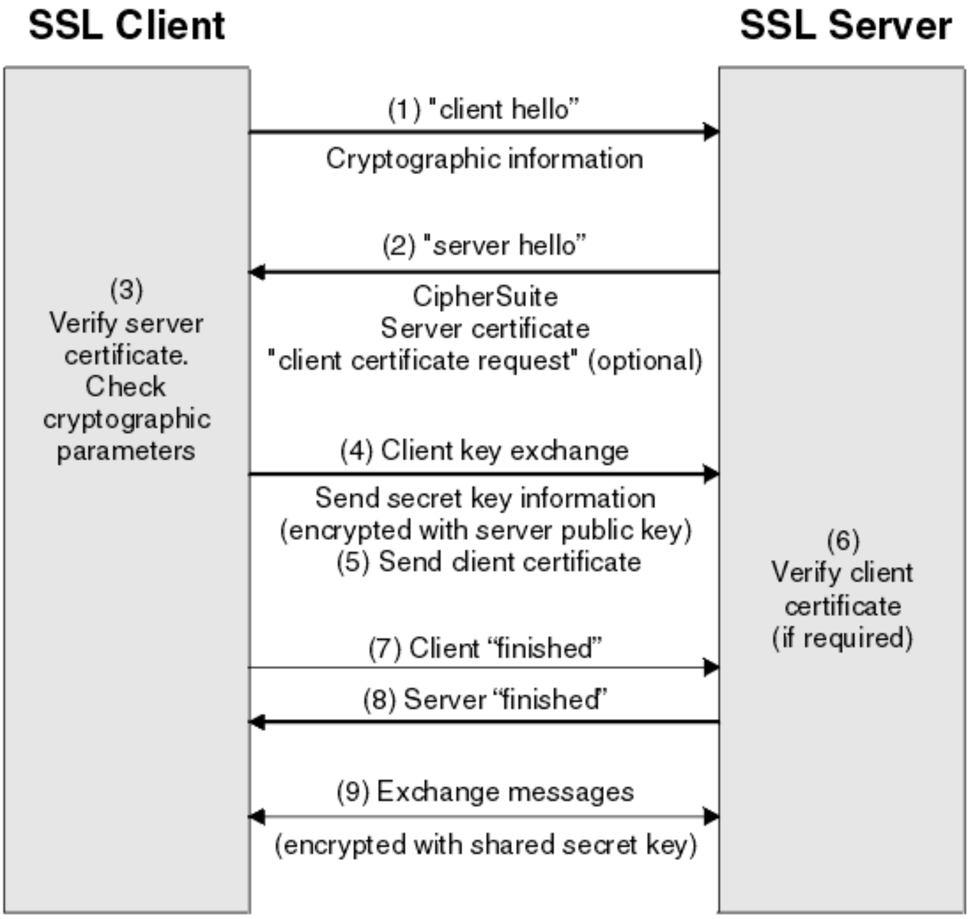
Transport Layer Security (TLS) is a protocol to secure communication using cryptographic means. SSL is the predecessor of TLS: Netscape SSL 2.0. They adopt a similar framework to secure a communication channel, and works in a similar manner to what was described above.

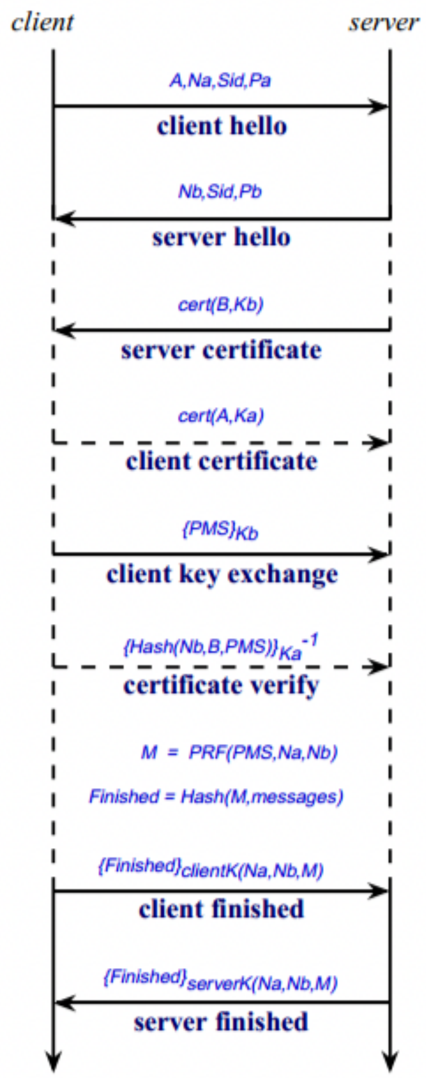
<u>HTTPS</u>

Overview of how HTTPS works:

- Ciphers Negotiation
- Authenticated Key Exchange (AKE)
    - Exchange of session key, which also authenticates the identities of parties involved
- Symmetric-key based secure communication
- Re-negotiation (if necessary)

TLS Handshake (Ciphers Negotiation & Authenticated Key Exchange)


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)



To summarise, these are the steps

1. User asks for certificate — handshake
2. Server / Site sends the certificate — handshake
3. User verifies certificate and gets the public key Ke
4. User will generate a session key pair (k, t) - k is for encryption, t is for authentication
5. User will encrypt his session key pair using public key : E(Ke, (k, t)) and send to Server

This session key pair will expire after a while, and renegotiation is needed. What happens is that steps 1-3 will be skipped, since we already have the certificate and the public key. We just need to redo steps 4 and 5.

Protocol

In computer networking, a protocol is a set of rules for exchanging information between multiple entities. A protocol is often described as steps of actions to be carried out by the entities, and the data to be transmitted.