SSL fundamentally works with the following concepts:

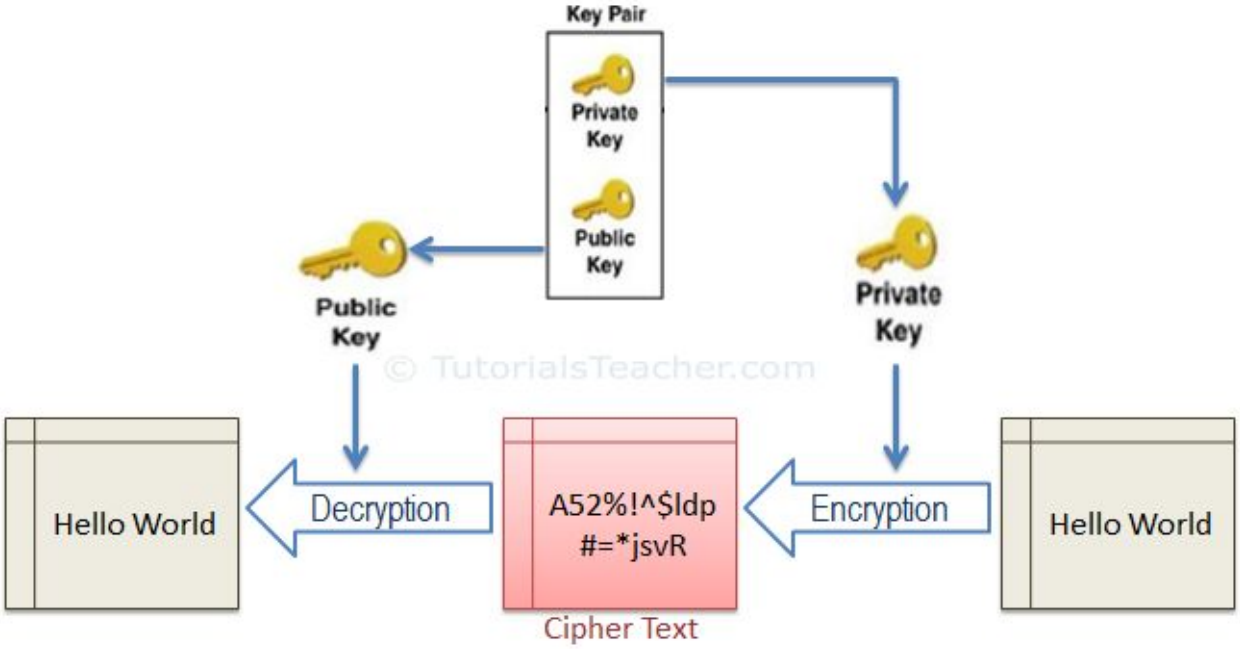
1. Asymmetric Cryptography
2. Symmetric Cryptography

Asymmetric Cryptography

Asymmetric cryptography (also known as Asymmetric Encryption or Public Key Cryptography) uses a mathematically-related key pair to encrypt and decrypt data. In a key pair, one key is shared with anyone who is interested in communication. This is called Public Key. The other key in the key pair is kept secret and is called Private Key.

Here, the keys referred to a mathematical value and were created using a mathematical algorithm which encrypts or decrypts the data.

In asymmetric cryptography, the data can be signed with a private key, which can only be decrypted using the related public key in a pair.

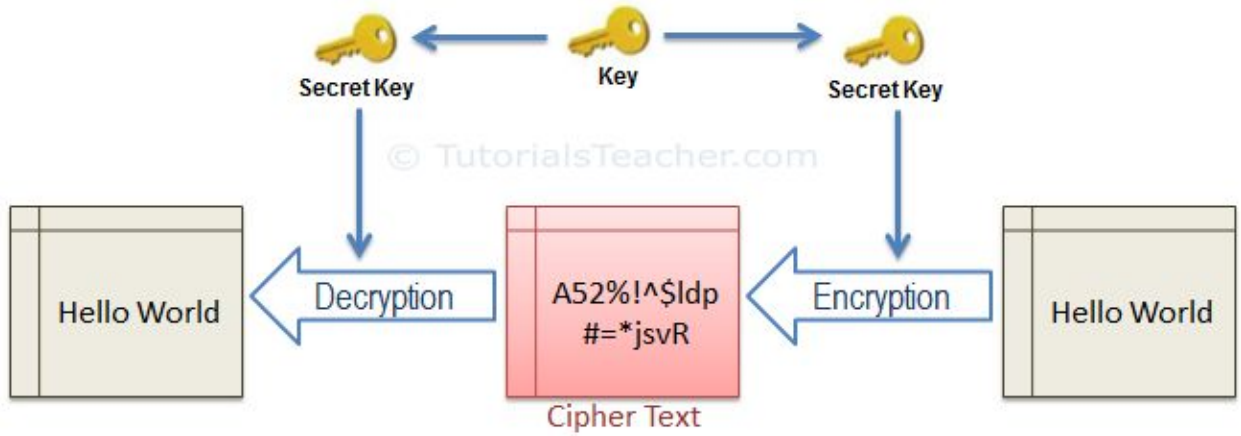


Asymmetric Cryptography

SSL uses asymmetric cryptography to initiate the communication which is known as SSL handshake. Most commonly used asymmetric key encryption algorithms include EIGamal, RSA, DSA, Elliptic curve techniques and PKCS.

Symmetric Cryptography

In the symmetric cryptography, there is only one key which encrypts and decrypts the data. Both sender and receiver should have this key, which is only known to them.

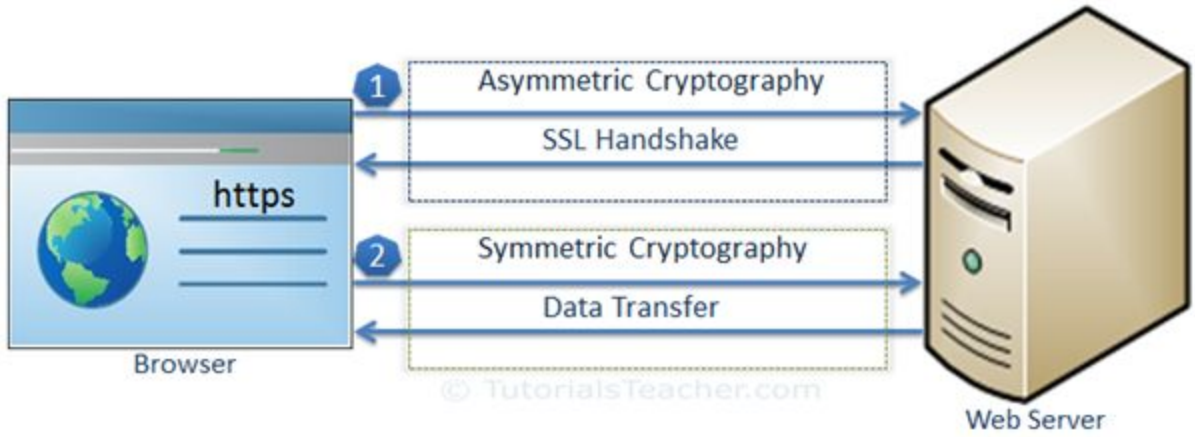


Symmetric Cryptography

SSL uses symmetric cryptography using the session key after the initial handshake is done. The most widely used symmetric algorithms are AES-128, AES-192 and AES-256.

Data Transfer over SSL

SSL protocol uses asymmetric and symmetric cryptography to transfer data securely. The following figure illustrates the steps of SSL communication:



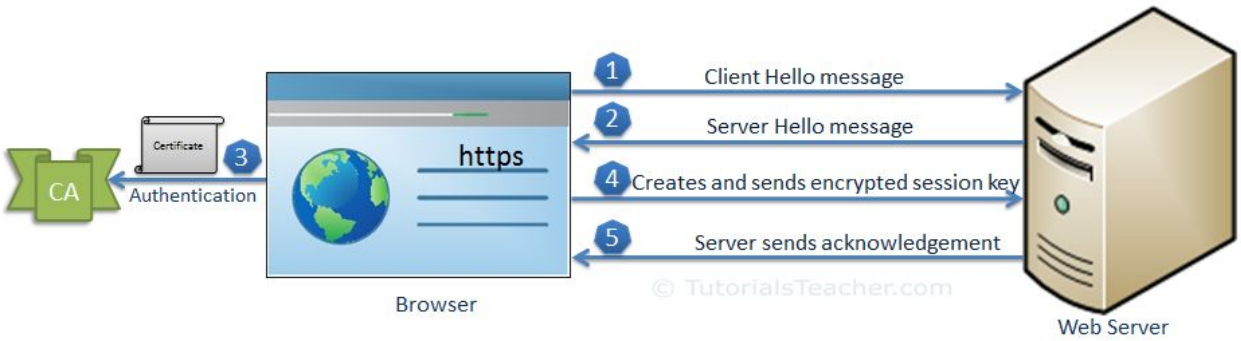
SSL Communication

As you can see in the above figure, SSL communication between the browser and the web server (or any other two systems) is mainly divided into two steps: the SSL handshake and the actual data transfer.

SSL Handshake

The communication over SSL always begins with the SSL handshake. The SSL handshake is an asymmetric cryptography which allows the browser to verify the web server, get the public key and establish a secure connection before the beginning of the actual data transfer.

The following figure illustrates the steps involved in the SSL handshake:



SSL Handshake

Let's understand the above steps:

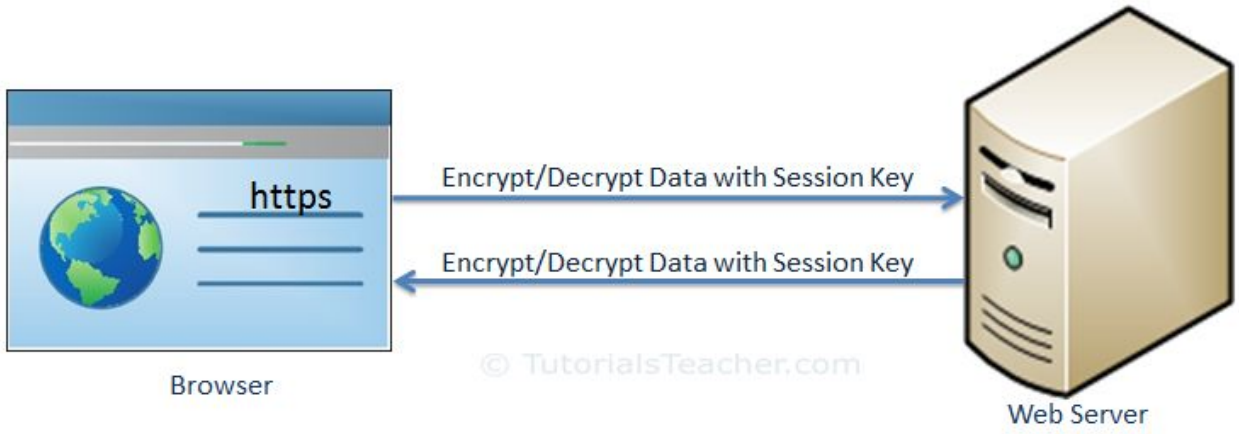
1. The client sends a "client hello" message. This includes the client's SSL version number, cipher settings, session-specific data and other information that the server needs to communicate with the client using SSL.
2. The server responds with a "server hello" message. This includes the server's SSL version number, cipher settings, session-specific data, an SSL certificate with a public key and other information that the client needs to communicate with the server over SSL.
3. The client verifies the server's SSL certificate from CA (Certificate Authority) and authenticates the server. If the authentication fails, then the client refuses the SSL connection and throws an exception. If the authentication succeeds, then proceed to step 4.
4. The client creates a session key, encrypts it with the server's public key and sends it to the server. If the server has requested client authentication (mostly in server to server communication), then the client sends his own certificate to the server.

5. The server decrypts the session key with its private key and sends the acknowledgement to the client encrypted with the session key.

Thus, at the end of the SSL handshake, both the client and the server have a valid session key which they will use to encrypt or decrypt actual data. The public key and the private key will not be used any more after this.

Actual Data Transfer

The client and the server now use a shared session key to encrypt and decrypt actual data and transfer it. This is done using the same session key at both ends and so, it is a symmetric cryptography. The actual SSL data transfer uses symmetric cryptography because it is easy and takes less CUP consumption compared with the asymmetric cryptography.



SSL Data Transfer

Thus, SSL fundamentally works using asymmetric cryptography and symmetric cryptography. There are certain infrastructures involved in achieving SSL communication in real life, which are called Public Key Infrastructure.

`-Djavax.net.debug=ssl,handshake`

There are two types of SSL handshakes described as one-way SSL and two-way SSL (Mutual SSL). Difference between those two is that in one -way SSL only the server authenticates to the client whereas, in two-way SSL, both server and client authenticate to each other. Usually, when we browse an HTTPS website, one-way SSL is being used where only our browser (client) validates the identity of the website (server). Two-way SSL is mostly used in server to server communication where both parties need to validate the identity of each other.

CLIENT HELLO

RandomCookie: * ClientHello, TLSv1.2**

RandomCookie: GMT: -1892413556 bytes = { GMT: -351008774 bytes = { 169, 131, 204, 213, 154, 96, 7, 136, 43, 142, 232, 138, 148, 171, 52, 226, 155, 202, 145, 57, 210, 132, 227, 182, 67, 222, 161, 28, 20 }

Session ID: 239, 10, 92, 143, 185, {}

93, Cipher Suites: [Unknown 0x8a:0x8a,
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, Unknown 0xcc:0xa9, Unknown 0xcc:0xa8,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_256_GCM_SHA384,
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA,
SSL_RSA_WITH_3DES_EDE_CBC_SHA]

In the above log, we can see that the client hello with TLS v1.2. By this, the client notifies the server that it has the support for TLS[1] versions 1.2 and below. List of ciphers[2] that are supported by the client can also be seen from the above log. Out of this list, the server will select a cipher suite that it supports. If the list contains cipher suites that server does not recognize, support, or wish to use, the server will ignore those ciphers. If no supported cipher suites were found the server will send a failure alert and close the connection.

2. Server Hello

*** ServerHello, TLSv1.2

```
*** ServerHello, TLSv1.2
```

Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

The server will respond back with the configuration it selected from the Client Hello along with its information to proceed with the handshake. Server Hello will be as follows.

Sever will select the TLS version according to the lower of that suggested by the client in the Client Hello message and the highest supported by the server. The server will also send back the cipher suite it selected from the list of ciphers presented by the client.

Along with the Server Hello, the server will also send the certificate[3] of the server with the certificate chain. The certificate chain will be validated against the certificates in the client trust store[4].

** Certificate chain

chain [0] = [

[

Version: V3

Subject: CN=server, OU=ID, O=IBM, L=Hursley, ST=Hants, C=GB

Signature Algorithm: SHA256withRSA, OID = 1.2.840.113549.1.1.11

Key: Sun RSA public key, 2048 bits

3. Server Key Exchange Message

This message will be sent by the server to the client carrying the required details for the client to generate the per-master secret. This message will not be sent if RSA key exchange algorithm (more in this later) or any other key exchange

algorithms are used that do not require the information from the server to generate a pre-master secret.

For Elliptic Curve Diffie Hellman(ECDH)[5] key exchange, the following details on the server ECDH public key are being sent to the client.

*** ECDH ServerKeyExchange

Signature Algorithm SHA256withRSA

Server key: Sun EC public key, 256 bits

4. Certificate Request

This is the place where one-way SSL defers from two-way SSL. In one-way SSL, the authenticity of the client is not being validated. Hence, this step is omitted in one-way SSL handshake.

During this step, the server will send a certificate request from the client with the certificate type, certificate signature algorithms and certificate authorities [6] supported by the server. There can be situations where the certificate authorities list can be empty. In such scenarios, the client may choose whether to send or avoid sending of the client certificate (depends on the client implementation)

Finally, the server sends the Server Hello Done message indicating the end of Server Hello. After sending this message, the server will wait for a client response.

5. Client Certificate

The client presents its certificate chain to the server. The certificate needs to be appropriate for the negotiated cipher suite's key exchange algorithm, and any negotiated extensions.

6. Client Key Exchange Message

This message needs to be sent by the client following the Client Certificate message. If the client certificate is not being presented (in one-way SSL), the client key exchange message should be sent after the client receives the ServerHelloDone message.

As we all know the data transferred between the server and the client in an HTTPS connection will be encrypted. Symmetric encryption[7] is being used for this purpose as the computational cost is much lower than Asymmetric encryption. In order to use symmetric encryption, there needs to be a common key between the two ends. The purpose of this message is to generate that common key between that client and the server without exposing to an outsider.

There are two client key exchange methods described in the TLS v1.2 spec. They are RSA[8] and Diffie-Hellman.

If RSA is used as the key exchange algorithm, the client generates a 48-byte pre-master secret. The client encrypts the pre-master secret by the public key of the certificate and sends to the server. Only the server will have the corresponding private key to decrypt and get the client generated pre-master secret.

If Diffie-Hellman is used, the Diffie-Hellman parameters are transmitted to allow both client and server to generate the same pre-master secret.

After that, both sides will generate a master secret using the pre-master secret and the master secret will be used to generate the symmetric key for encrypting the session data

7. Finished

After successful authentication and generating the pre-master secrets/master secrets, a change cipher spec

message will be sent by both client and server indicating that the rest of the communication will be encrypted.

The Finished message will immediately follow the change cipher spec message. This will be the first message encrypted with the negotiated algorithms. Application data will be transferred only after both parties send the finished message and verifying the content of the message.