

1、客户端或者浏览器向服务器发送https请求；

2、服务器将公钥返回给客户端

3、浏览器接收到服务器公钥

1. 浏览器收到公钥后会对公钥进行一系列的验证，如：颁发机构，过期时间等等。
2. 认证失败会有警告，比如上面说的我们自己生成的证书，不是授信组织办法。
3. 成功的话，会生成一个随机值作为对称加密的密钥稍后还会提及到。
4. 用服务器返回的公钥对这个随机值进行加密

4、客户端将加密后的随机值发送到服务器，服务器收到公钥加密后的随机值，会拿自己的私钥去解（**非对称加密**），解开之后得到的随机值

5、客户端与服务器进行加密通信，客户端和服务器端信息数据通信，就是秘钥为随机值（**对称加密**）。

**为什么要用对称加密呢，因为相对于非对称加密，对称加密对服务器的资源开销客户忽略不计。**

**需要注意的是由于SSL协议依赖于TCP连接实施，所以在SSL交互之前需要先建立TCP连接。客户端connect服务端，服务端acccept客户端，经历三次握手以后TCP连接建立。**

**SSL的交互过程**

**1、建立TCP连接**

由于SSL协议依赖于TCP连接实施，所以在SSL交互之前需要先建立TCP连接。客户端connect服务端，服务端acccept客户端，经历三次握手以后TCP连接建立。

**2、客户端发送SSL请求**

客户端（Client）向服务端（Server）发送自身使用的SSL版本（SSL一共有三个版本）、加密算法的相关配置、随机数据(Client random)以及其在在SSL协议中需要用到的信息。

**3、服务端处理SSL请求**

服务器（Server）反馈给客户端（Client）自己的SSL版本、加密算法的相关配置、随机数据(Server random)以及用自己的私有密钥加密的SERVER-HELLO信息。服务端（Server）紧接着将自己的证书（包含公共密钥）传递过去。同时有个可选项目，即服务端（Server）可以要求客服端（Client）发送自己的证书。

**4、客户端验证服务端身份**

客服端（Client）用服务端（Server）传递过来证书验证服务端（Server）的身份，如果身份未验证通过则结束本次通信。证书验证通过后利用服务端（Server）的公共密钥尝试解开被其用私有密钥加密过的SERVER-HELLO信息，如果解开失败，说明该SERVER-HELLO必然是假的，故结束本次通信。

**5、客户端发送公共密钥加密过的随机数据**

客户端端（Client）生成随机数据（sharedsecret），并且把这个随机数据用服务端（Server）发送过来的的公共密钥加密，此次加密过程产生本次握手中的premastersecret（这个步骤是有可能有服务端的参与的，具体情况由他们使用的加密算法决定），然后将它（premastersecret）送回给服务端（Server）。如果服务端（Server）要求需要验证客户端（Client），那么客服端（Client）也需要自己把自己的证书（包含公共密钥）传递过去，同时送一些自己签过名（私有密钥加密）的数据过去。

**6、服务端用私有密钥解密加密后的随机数据并协商暗号**

Server验证完client的身份之后，然后用自己的私有密钥解密得到premastersecret然后双方利用这个premastersecret来共同协商，得到mastersecret(可理解为premastersecret为双方协商的暗号，然后使用这个暗号再协商一个mastersecret用来产生真正的会话密钥用来传输数据）以此来保证数据的决对安全。

**7、服务端跟客户端利用暗号生成加密算法跟密钥key**

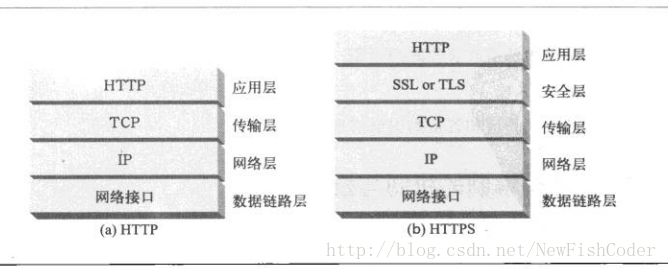
双方用mastersecret一起产生真正的sessionkey，这将是一个对称加密的key。这个key还可以用来验证数据完整性。双方再交换结束信息。握手结束。

接下来双方就可以用协商好的算法和密钥key，采用对称加密算法来通信了。

一般采用https的都是，在公网传输的web请求，像分布式的局域网数据传输是没有必要的，只会加重服务器开销降低性能。

要想提高https的性能可以用nginx做负载均衡，或者购买ssl加速卡。

相对于http，https会多出SSL/TLS握手过程多了请求过程，加密解密过程也会消耗时间。http请求80、8080端口，https请求443端口。



在https请求中，SSL/TLS层对http请求加密，通过TCP层发出https请求。服务端接收到https请求，TCP响应请求，在SSL/TLS层对https请求解密，在将解密的http请求交下一步处理。HTTPS基于传输层，HTTP基于应用层。