## [Https 客户端与服务器交互过程](http://www.cnblogs.com/softidea/p/4949589.html)

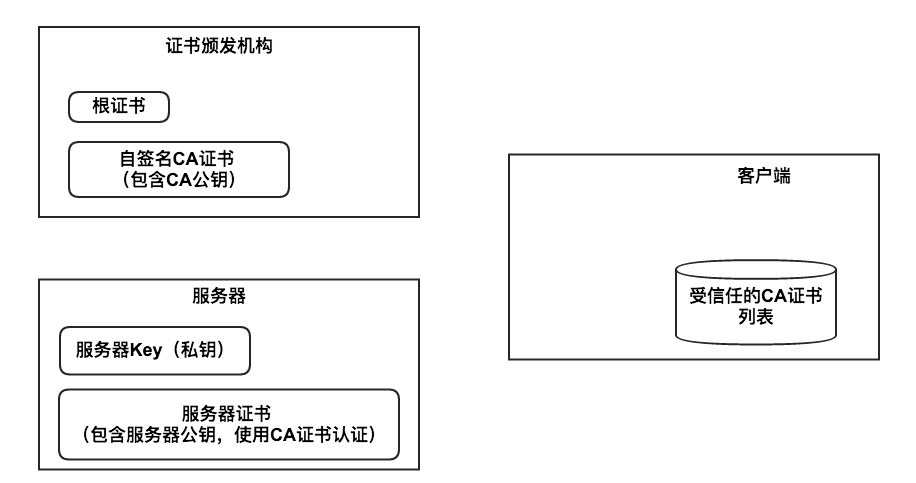
本文试图以通俗易通的方式介绍Https的工作原理，不纠结具体的术语，不考证严格的流程。我相信弄懂了原理之后，到了具体操作和实现的时候，方向就不会错，然后条条大路通罗马。阅读文本需要提前大致了解对称加密、非对称加密、信息认证等密码学知识。如果你不太了解，可以阅读Erlang发明人Joe Armstrong最近写的[Cryptography Tutorial](https://github.com/joearms/crypto_tutorial)。大牛出品，通俗易懂，强力推荐。

### **Https涉及到的主体**

1. 客户端。通常是浏览器(Chrome、IE、FireFox等)，也可以自己编写的各种语言的客户端程序。
2. 服务端。一般指支持Https的网站，比如github、支付宝。

CA(Certificate Authorities)机构。Https证书签发和管理机构，比如Symantec、Comodo、GoDaddy、GlobalSign。

下图里我画出了这几个角色：

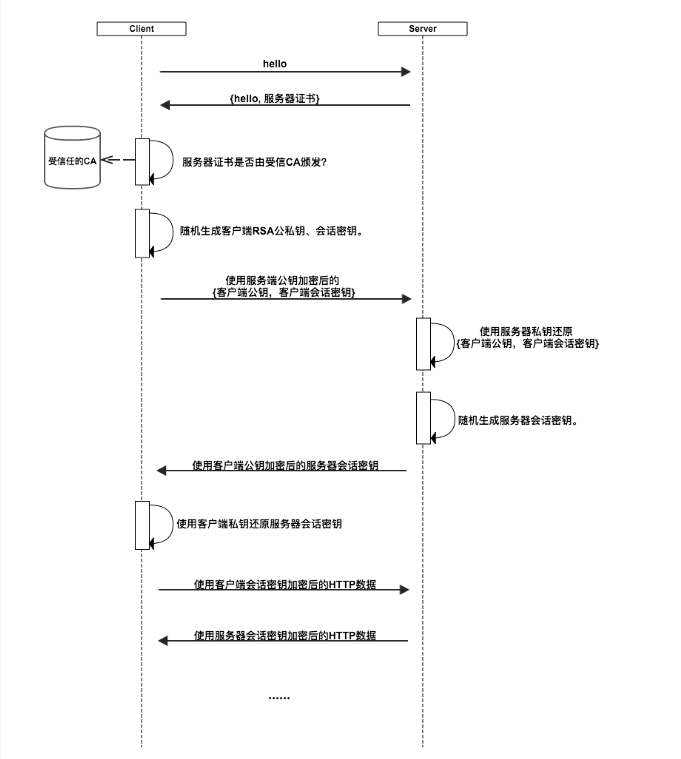


### **发明Https的动机**

1. 认证正在访问的网站。什么叫认证网站？比如你正在访问支付宝，怎样确定你正在访问的是阿里巴巴提供的支付宝而不是假冒伪劣的钓鱼网站呢？
2. 保证所传输数据的私密性和完整性。众所周知，Http是明文传输的，所以处在同一网络中的其它用户可以通过网络抓包来窃取和篡改数据包的内容，甚至运营商或者wifi提供者，有可能会篡改http报文，添加广告等信息以达到盈利的目的。

**Https的工作流程**

这一节通过介绍Https协议的工作流程，来说明Https是如何达成自己的两个目的的。下图我画出了Https的工作流程，注意，这只是原理示意图，并不是详细的协议解析。

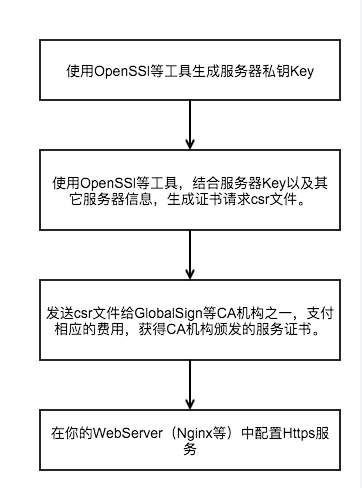


可以看到工作流程，基本分为三个阶段：

1. 认证服务器。浏览器内置一个受信任的CA机构列表，并保存了这些CA机构的证书。第一阶段服务器会提供经CA机构认证颁发的服务器证书，如果认证该服务器证书的CA机构，存在于浏览器的受信任CA机构列表中，并且服务器证书中的信息与当前正在访问的网站（域名等）一致，那么浏览器就认为服务端是可信的，并从服务器证书中取得服务器公钥，用于后续流程。否则，浏览器将提示用户，根据用户的选择，决定是否继续。当然，我们可以管理这个受信任CA机构列表，添加我们想要信任的CA机构，或者移除我们不信任的CA机构。
2. 协商会话密钥。客户端在认证完服务器，获得服务器的公钥之后，利用该公钥与服务器进行加密通信，协商出两个会话密钥，分别是用于加密客户端往服务端发送数据的客户端会话密钥，用于加密服务端往客户端发送数据的服务端会话密钥。在已有服务器公钥，可以加密通讯的前提下，还要协商两个对称密钥的原因，是因为非对称加密相对复杂度更高，在数据传输过程中，使用对称加密，可以节省计算资源。另外，会话密钥是随机生成，每次协商都会有不一样的结果，所以安全性也比较高。
3. 加密通讯。此时客户端服务器双方都有了本次通讯的会话密钥，之后传输的所有Http数据，都通过会话密钥加密。这样网路上的其它用户，将很难窃取和篡改客户端和服务端之间传输的数据，从而保证了数据的私密性和完整性。

**使用Https的流程**

如果你是一个服务器开发者，想使用Https来保护自己的服务和用户数据安全，你可以按照以下流程来操作。



**总结**

* 1. 说是讨论Https，事实上Https就是Http跑在SSl或者TLS上，所以本文讨论的原理和流程其实是SSL和TLS的流程，对于其它使用SSL或者TLS的应用层协议，本文内容一样有效。
  2. 本文只讨论了客户端验证服务端，服务端也可以给客户端颁发证书并验证客户端，做双向验证，但应用没有那么广泛，原理类似。
  3. 由于采用了加密通讯，Https无疑要比Http更耗费服务器资源，这也是很多公司明明支持Https却默认提供Http的原因。