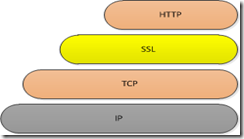
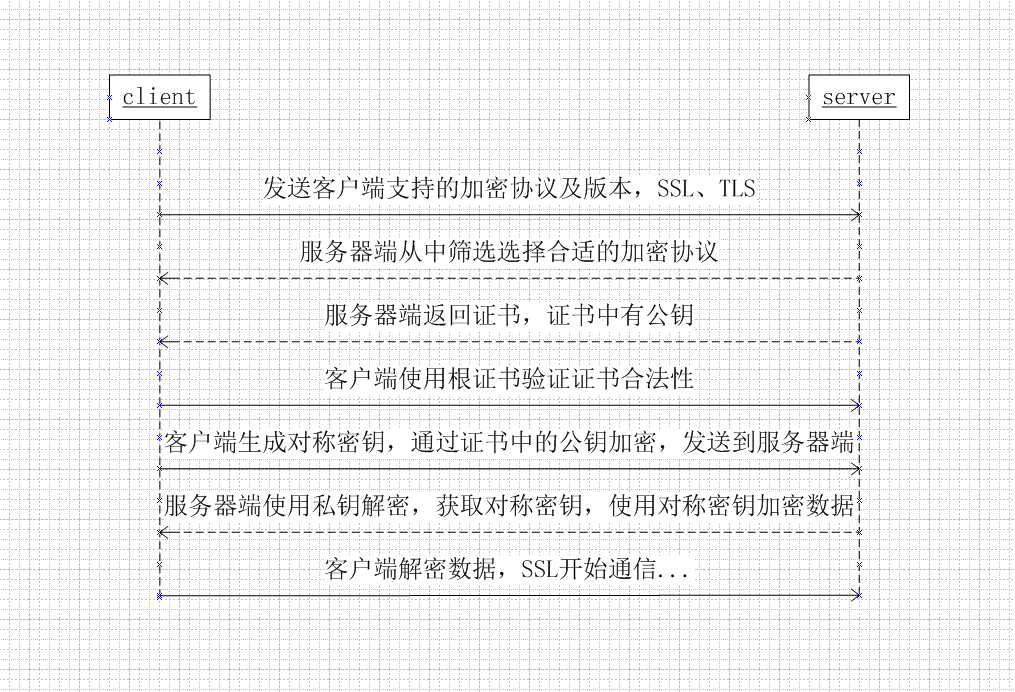
**Https协议中的ssl加密解密流程:**

HTTPS其实是有两部分组成：HTTP + SSL / TLS，也就是在HTTP上又加了一层处理加密信息的模块。服务端和客户端的信息传输都会通过TLS进行加密，所以传输的数据都是加密后的数据。

[](http://images.cnblogs.com/cnblogs_com/P_Chou/201012/201012272252072023.png)



**1. 客户端发起HTTPS请求**

用户在浏览器里输入一个https网址，然后连接到server的443端口。

**2. 服务端的配置**

采用HTTPS协议的[服务器](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)必须要有一套数字证书，可以自己制作，也可以向组织申请。区别就是自己颁发的证书需要客户端验证通过，才可以继续访问，而使用受信任的公司申请的证书则不会弹出提示页面(startssl就是个不错的选择，有1年的免费服务)。这套证书其实就是一对公钥和私钥。如果对公钥和私钥不太理解，可以想象成一把钥匙和一个锁头，只是全世界只有你一个人有这把钥匙，你可以把锁头给别人，别人可以用这个锁把重要的东西锁起来，然后发给你，因为只有你一个人有这把钥匙，所以只有你才能看到被这把锁锁起来的东西。

**3. 传送证书**

这个证书其实就是公钥，只是包含了很多信息，如证书的颁发机构，过期时间等等。

**4. 客户端解析证书**

这部分工作是有客户端的TLS来完成的，首先会验证公钥是否有效，比如颁发机构，过期时间等等，如果发现异常，则会弹出一个警告框，提示证书存在问题。如果证书没有问题，那么就生成一个随机值。然后用证书对该随机值进行加密。就好像上面说的，把随机值用锁头锁起来，这样除非有钥匙，不然看不到被锁住的内容。

**5. 传送加密信息**

这部分传送的是用证书加密后的随机值，目的就是让服务端得到这个随机值，以后客户端和服务端的通信就可以通过这个随机值来进行加密解密了。

**6. 服务段解密信息**

服务端用私钥解密后，得到了客户端传过来的随机值(私钥)，然后把内容通过该值进行对称加密。所谓对称加密就是，将信息和私钥通过某种算法混合在一起，这样除非知道私钥，不然无法获取内容，而正好客户端和服务端都知道这个私钥，所以只要加密算法够彪悍，私钥够复杂，数据就够安全。

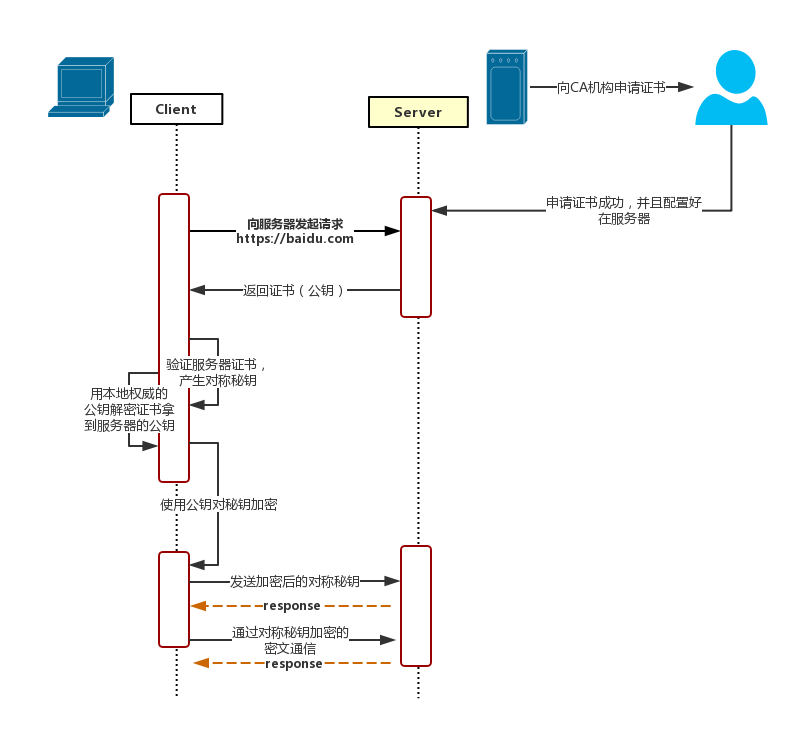
**7. 传输加密后的信息**

这部分信息是服务段用私钥加密后的信息，可以在客户端被还原。

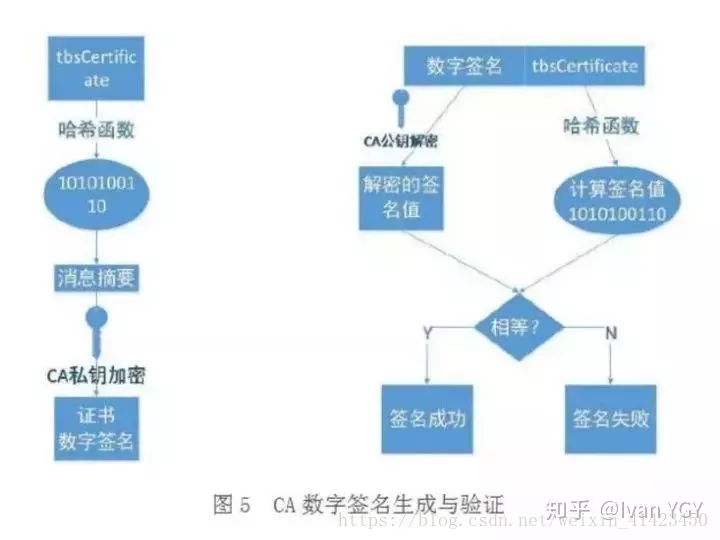
**8. 客户端解密信息**

客户端用之前生成的私钥解密服务段传过来的信息，于是获取了解密后的内容。整个过程第三方即使监听到了数据，也[束手无策](https://www.baidu.com/s?wd=%E6%9D%9F%E6%89%8B%E6%97%A0%E7%AD%96&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "_blank)。

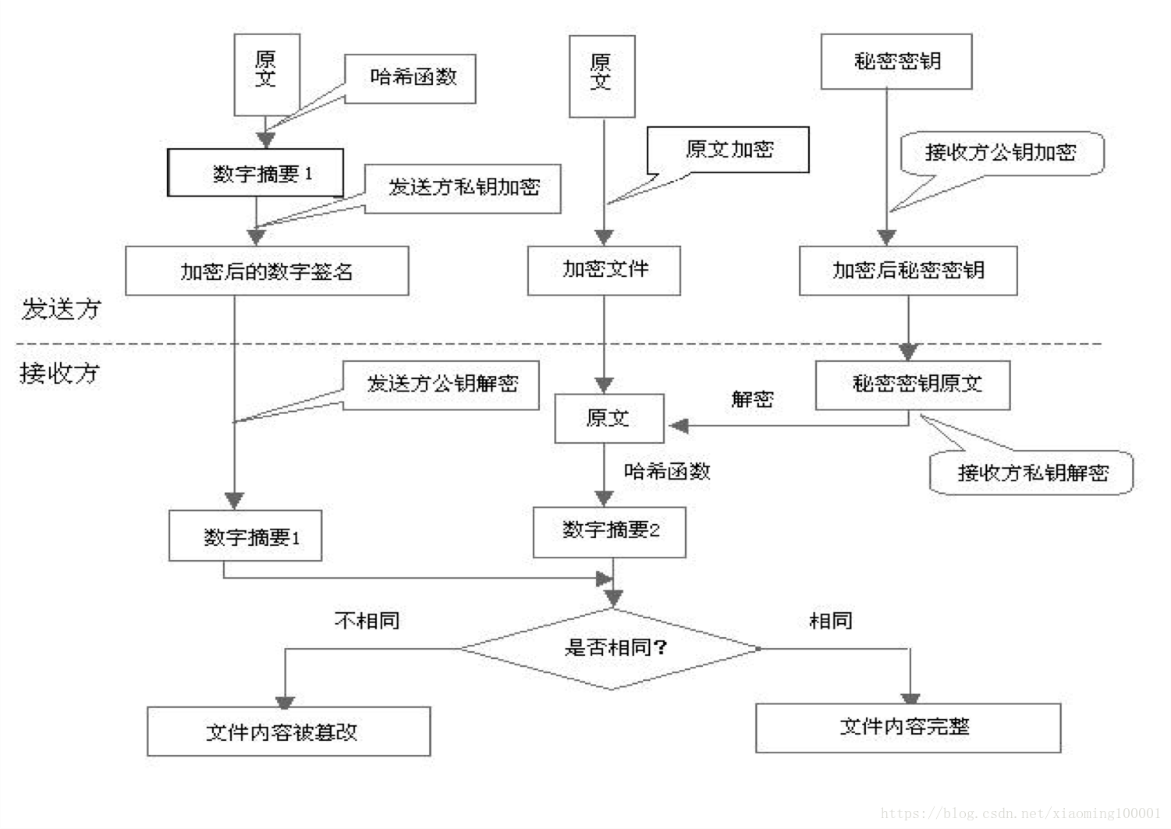
**HTTPS实现过程图解:**

**数字签名：**

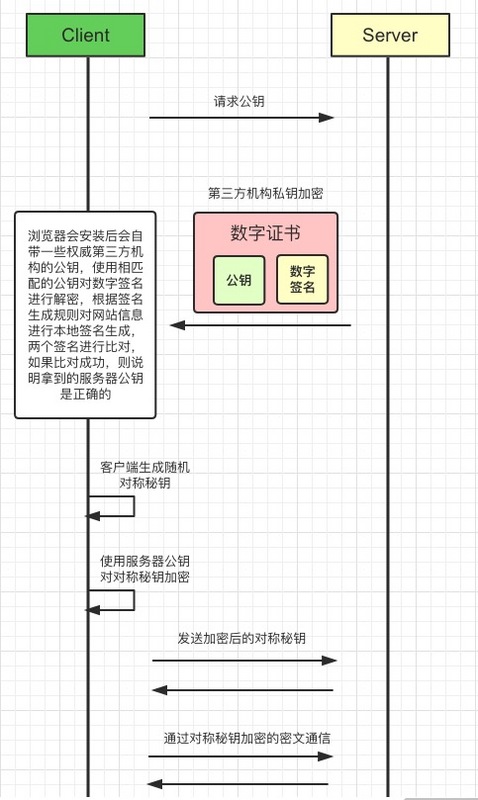
网站在使用HTTPS前，需要向“CA机构”申请颁发一份数字证书，数字证书里有证书持有者、证书持有者的公钥等信息，证书内容生成一份“签名”，比对证书内容和签名是否一致就能察觉是否被篡改。这种技术就叫数字签名



**混合加密过程如下图：**



https整体过程如图：



**HTTPS有如下特点：**

内容加密：采用混合加密技术，中间者无法直接查看明文内容

验证身份：通过证书认证客户端访问的是自己的服务器

保护数据完整性：防止传输的内容被中间人冒充或者篡改

**名词解释：**

混合加密：结合非对称加密和对称加密技术。客户端使用对称加密生成密钥对传输数据进行加密，然后使用非对称加密的公钥再对秘钥进行加密，所以网络上传输的数据是被秘钥加密的密文和用公钥加密后的秘密秘钥，因此即使被黑客截取，由于没有私钥，无法获取到加密明文的秘钥，便无法获取到明文数据。

数字摘要：通过单向hash函数对原文进行哈希，将需加密的明文“摘要”成一串固定长度(如128bit)的密文，不同的明文摘要成的密文其结果总是不相同，同样的明文其摘要必定一致，并且即使知道了摘要也不能反推出明文。

数字签名技术：数字签名建立在公钥加密体制基础上，是公钥加密技术的另一类应用。它把公钥加密技术和数字摘要结合起来，形成了实用的数字签名技术。

收方能够证实发送方的真实身份；

发送方事后不能否认所发送过的报文；

收方或非法者不能伪造、篡改报文。

**常用的加密算法**

对称密码算法：是指加密和解密使用相同的密钥，典型的有DES、RC5、IDEA（分组加密），RC4（序列加密）；

非对称密码算法：会有一对秘钥，加密和解密使用不同的密钥：公钥和私钥。公钥加密的内容，只有私钥可以解开，私钥加密的内容，所有的公钥都可以解开（当然是指和秘钥是一对的公钥）。

比如A发送，B接收，A想确保消息只有B看到，需要B生成一对公私钥，并拿到B的公钥。于是A用这个公钥加密消息，B收到密文后用自己的与之匹配的私钥解密即可。反过来也可以用私钥加密公钥解密。也就是说对于给定的公钥有且只有与之匹配的私钥可以解密，对于给定的私钥，有且只有与之匹配的公钥可以解密。典型的算法有RSA，DSA，DH；

散列算法：散列变换是指把文件内容通过某种公开的算法，变成固定长度的值（散列值），这个过程可以使用密钥也可以不使用。这种散列变换是不可逆的，也就是说不能从散列值变成原文。因此，散列变换通常用于验证原文是否被篡改。典型的算法有：MD5，SHA，Base64，CRC等。

**安全性考虑：**

HTTPS协议的加密范围也比较有限，在服务器劫持等方面几乎起不到什么作用

SSL证书的信用链体系并不安全，特别是在某些国家可以控制CA根证书的情况下，中间人攻击一样可行

中间人攻击（MITM攻击）是指，黑客拦截并篡改网络中的通信数据。又分为被动MITM和主动MITM，被动MITM只窃取通信数据而不修改，而主动MITM不但能窃取数据，还会篡改通信数据。最常见的中间人攻击常常发生在公共wifi或者公共路由上。

成本考虑：

SSL证书需要购买申请，功能越强大的证书费用越高

SSL证书通常需要绑定IP，不能在同一IP上绑定多个域名，IPv4资源不可能支撑这个消耗（SSL有扩展可以部分解决这个问题，但是比较麻烦，而且要求浏览器、操作系统支持，Windows XP就不支持这个扩展，考虑到XP的装机量，这个特性几乎没用）。

根据ACM CoNEXT数据显示，使用HTTPS协议会使页面的加载时间延长近50%，增加10%到20%的耗电。

HTTPS连接缓存不如HTTP高效，流量成本高。

HTTPS连接服务器端资源占用高很多，支持访客多的网站需要投入更大的成本。

HTTPS协议握手阶段比较费时，对网站的响应速度有影响，影响用户体验。比较好的方式是采用分而治之，类似12306网站的主页使用HTTP协议，有关于用户信息等方面使用[[1]](#footnote-1)HTTPS。

HTTPS前端加密

优点：

前端加密，防止本地明文cookie被黑客发现[[2]](#footnote-2)

缺点:

无法应对服务器劫持

HTTPS后端加密

优点

降低数据库泄露的损失

缺点:

无法应对服务器劫持

1. ---------------------

   https://blog.csdn.net/xiaoming100001/article/details/81109617 [↑](#footnote-ref-1)
2. https://blog.csdn.net/ahilll/article/details/82459718 [↑](#footnote-ref-2)