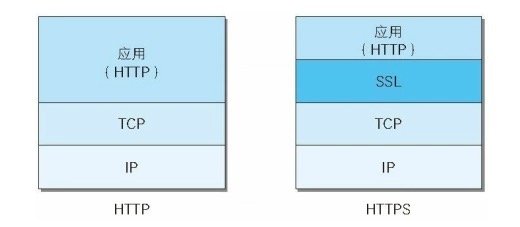
# HTTPS

## 什么是 HTTPS？

在说 HTTPS 之前先说说什么是 HTTP，HTTP 就是我们平时浏览网页时候使用的一种协议。HTTP 协议传输的数据都是未加密的，也就是明文的，因此使 用 HTTP 协议传输隐私信息非常不安全。为了保证这些隐私数据能加密传输，于是网景公司设计了 SSL（Secure Sockets Layer）协议用于对 HTTP 协议传输的数据进行加密，从而就诞生了 HTTPS。



## 对称和非对称加密

加密方式从“加密密钥和解密密钥是否相同”可以分为对称加密和非对称加密。

对加密和解密使用相同密钥的加密算法。由于其速度，对称性加密通常在消息发送方需要加密大量数据时使用。常用的对称加密有：DES、IDEA、RC2、RC4、SKIPJACK 算法等 。采用单钥密码系统的加密方法，同一个密钥可以同时用作信息的加密和解密，这种加密方法称为对称加密，也称为单密钥加密。这就有一个问题，我怎么把密钥传递给对方呢？通过网络么？那在加密之前不还是明文，那岂不是会被轻易破解么？



这个时候非对称加密就要隆重出场了！

相对于”对称加密算法”这种方法也叫做”非对称加密算法”。与对称加密算法不同，非对称加密算法需要两个密钥：公开密钥（publickey）和私有密 (privatekey)。公开密钥与私有密钥是一对，如果用公开密钥对数据进行加密，只有用对应的私有密钥才能解密；如果用私有密钥对数据进行加密，那么只有用对应的公开密钥才能解密。因为加密和解密使用的是两个不同的密钥，所以这种算法叫作非对称加密算法。如果你用过 ssh，那么我想你一定对此不陌生。

ls ~/.ssh  
config id\_rsa id\_rsa.pub known\_hosts

如上 id\_rsa 是你的密钥，id\_rsa.pub 是你的公钥。常见的非对称加密算法有 DH (Diffie-Hellman)，RSA。

因此我只需要向对方发送公钥，让对方用你的公钥加密，你自己使用私钥解密即可。由于私钥不会在网络传输，因此其是安全的。

## 哈希算法

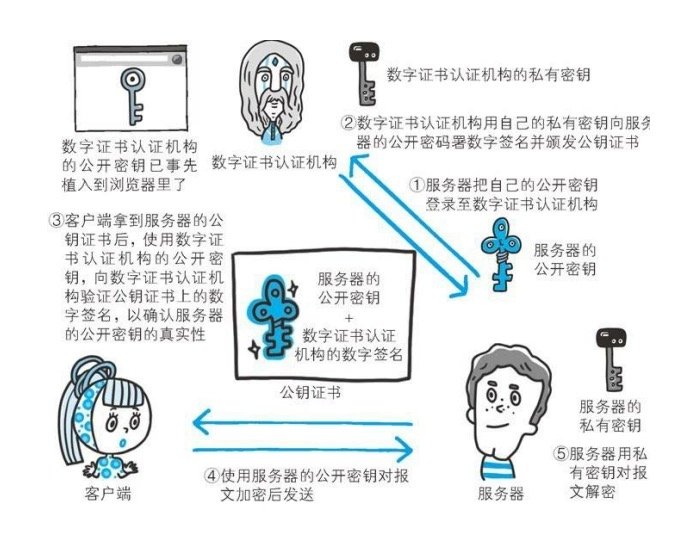
引入哈希算法，是为了防止篡改，和破解密文没有任何关系。常用的摘要算法包括 MD5，SHA1，SHA256。使用哈希算法可以生成一个唯一的固定长度的值，它由一个单向 Hash 加密函数对消息进行作用而产生。如果消息在途中改变了，则接收者通过对收到消息的新产生的摘要与原摘要比较，就可知道消息是否被改变了。因此消息摘要保证了消息的完整性。

## HTTPS 流程

上文提到，对称加密算法的复杂度低，所以在传输数据的时候，使用的是对称加密算法； 但是对称加密算法使用的是相同的秘钥，为了保证秘钥的安全性，单独对秘钥的传输采用非对称加密； 同时，数据的传输过程中时候用 HASH 算法用于验证数据的完整性。

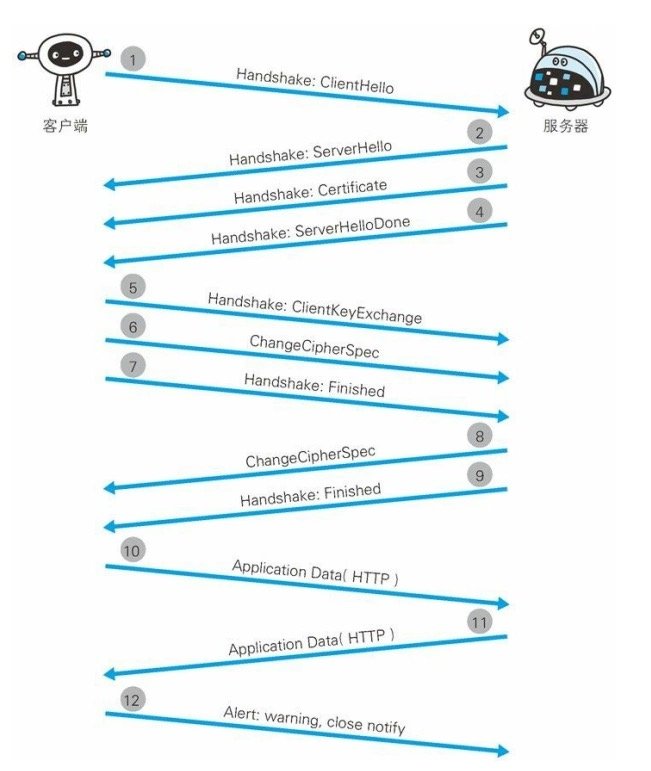
非对称加密算法用于在握手过程中加密生成的密码 对称加密算法用于对真正传输的数据进行加密 而 HASH 算法用于验证数据的完整性。

客户端有公钥，服务器有私钥，客户端用公钥对对称密钥进行加密，将加密后的对称密钥发送给服务器，服务器用私钥对其进行解密，所以客户端和服务器可用对称密钥来进行通信。公钥和私钥是用来加密密钥，而对称密钥是用来加密数据，分别利用了两者的优点。



## HTTPS 是怎么握手的？

1. 浏览器将自己支持的一套加密规则发送给网站，如 RSA 加密算法，DES 对称加密算法，SHA1 摘要算法
2. 网站从中选出一组加密算法与 HASH 算法，并将自己的身份信息以证书的形式发回给浏览器。证书里面包含了网站地址，加密公钥，以及证书的颁发机构等信息（证书中的私钥只能用于服务器端进行解密，在握手的整个过程中，都用到了证书中的公钥和浏览器发送给服务器的随机密码以及对称加密算法）
3. 获得网站证书之后浏览器要做以下工作：
4. 验证证书的合法性（颁发证书的机构是否合法，证书中包含的网站地址是否与正在访问的地址一致等），如果证书受信任，则浏览器栏里面会显示一个小锁头，否则会给出证书不受信的提示。
5. 如果证书受信任，或者是用户接受了不受信的证书，浏览器会生成一串随机数的密码（这其实就是用于之后数据通信的对称加密算法的秘钥），并用证书中提供的公钥加密（对秘钥的加密采用非对称的方法）。
6. 使用约定好的 HASH 算法计算握手消息，并使用生成的随机数（对称算法的秘钥）对消息进行加密，最后将之前生成的被公钥加密的随机数密码，HASH 摘要值（已经被对称算法加密）一起发送给服务器
7. 网站接收浏览器发来的数据之后要做以下的操作：
8. 使用自己的私钥将信息解密并取出浏览器发送给服务器的随机密码（得到了对称加密算法的秘钥），使用密码解密浏览器发来的握手消息（用对称算法的秘钥解 HASH 摘要值），并验证 HASH 是否与浏览器发来的一致。
9. 使用随机密码加密一段握手消息，发送给浏览器。
10. 浏览器解密并计算握手消息的 HASH，如果与服务端发来的 HASH 一致，此时握手过程结束，之后所有的通信数据将由之前浏览器生成的随机密码并利用对称加密算法进行加密。



从上面可以看到，握手的整个过程使用到了数字证书、对称加密、HASH 摘要，非对称加密算法。

此时客户端和服务端都有三个随机数，客户端和服务端用商定的算法利用 3 个随机数生成一个对话秘钥（session key），随后的通信就用这个秘钥进行加密解密，保证通信过程不被别人监听或者窜改。所以唯一的风险点在于这个对话秘钥是否会被窃取，窃取的唯一方式也就是身份冒充，在上述四次握手阶段中窃取到对话秘钥

## 中间人攻击

理论上 HTTPS 可以保证你的通道是安全，不被窃听和篡改的。假如 A 和 B 通讯，我同时对 A 和 B 建立 HTTPS 链接，这样我不就可以窃听了么？实际上是可以的，这种情况证书的作用就来了。我们知道 HTTPS 是有证书的，但是我们使用 SSH 登录是没有证书这回事的，那我们是不是也会被中间人攻击？答案是肯定的，感兴趣的可以查阅相关资料，这里不展开。

## 参考

* [https 的加密方式 介绍 + 常见的加密技术](https://blog.csdn.net/guizaijianchic/article/details/77961418)
* [白话 HTTPS & SSL/TLS](https://www.jianshu.com/p/992bad24412e)