## HTTPS

### 原理

**对称加密**

密钥需要由服务器产生，并发送给客户端，使用明文传输。



缺点:

* 中间人拿到密钥，相当于拿到了万能钥匙；
* 如果每个客户端使用不同密钥，服务端管理会很麻烦。

**非对称加密**

公钥加密，私钥解密

用对方的公钥来加密传输的报文。

公钥可以分发给别人，私钥永远不会。



缺点:

* 性能

非对称加密 + 对称加密:

非对称加密传输共享密钥，使用该密钥进行对称加密;

1. 服务端传输分发明文公钥
2. 客户端用该公钥加密客户端的密钥，发送到服务端
3. 服务端私钥解密，得到该客户端的密钥。

* 安全

中间人劫持服务端发送的明文公钥。



**数字签名**

防篡改;

验证身份;

与报文传输加解密不同的是： 私钥加密，公钥解密;

1. 服务端私钥加密摘要信息digest1;
2. 客户端用公钥来解密签名, 只要能成功，就说明该信息来源正确。
3. 再用相同的hash来计算摘要digest2, 如果 digest2 == digest1, 则信息没有被篡改过。

**数字证书**

用来确定来源是否是真实的。 验证签名是否就是那个人签的.

权威机构给某网站颁发的一种认可凭证。

证书里需要包含真实的服务器公钥和公司/个人信息:



客户端进行比对:



**HTTPS**

性能与安全平衡:

用数字证书保证服务器端的公匙可以安全无误的到达客户端;

用非对称加密安全的传递共享密匙;

用共享密匙(对称加密)安全的交换数据;

1. 浏览器发送client\_hello, 包含随机数random1
2. 服务端回复server\_hello，发送随机数random2, 发送数字证书(包含公钥)
3. 浏览器根据random1 random2 生成premaster\_secret, master\_secret. 使用服务端公钥加密premaster\_secret, 发送给服务端;
4. 服务端解密得到premaster\_secret, 再根据random1, random2 生成和客户端一致的master\_secret。

详细见参考源4.

**参考源**

1. Hollis/2019/04/一篇文章读懂HTTPS及其背后的加密原理.html
2. Hollis/2018/11/【漫画】https%20加密那点事.html
3. 收藏\_20191021/2017/02/HTTPS的中那些加密算法.html
4. 收藏\_20191021/2015/05/【大型网站的HTTPS实践】%20HTTPS协议和原理.html