**https://www.jianshu.com/p/304674bd0618**

**详解https是如何确保安全的？（转载）**

[](https://www.jianshu.com/u/35140a960a96)

[android\_hcf](https://www.jianshu.com/u/35140a960a96) 关注

2017.08.03 09:45\* 字数 4736 阅读 3495评论 0喜欢 5

**一、Https介绍**

**1. 什么是Https**

HTTPS（全称：Hypertext Transfer Protocol over Secure Socket Layer），是以安全为目标的HTTP通道，简单讲是HTTP的安全版。即HTTP下加入SSL层，HTTPS的安全基础是SSL，因此加密的详细内容就需要SSL

**2. Https的作用**

* **内容加密** 建立一个信息安全通道，来保证数据传输的安全；
* **身份验证** 确认网站的真实性；
* **数据完整性** 防止内容被第三方冒充或者篡改

**3. Https的劣势**

对数据进行加密决定了它比http慢

需要进行非对称的加解密，且需要三次握手。首次连接比较慢，当然现在也有很多的优化。  
出于安全考虑，浏览器不会在本地保存HTTPS缓存。实际上，只要在HTTP头中使用特定命令，HTTPS是可以缓存的。Firefox默认只在内存中缓存HTTPS。但是，只要头命令中有Cache-Control: Public，缓存就会被写到硬盘上。IE只要http头允许就可以缓存https内容，缓存策略与是否使用HTTPS协议无关。

**4. HTTPS和HTTP的区别**

* https协议需要到CA申请证书
* http是超文本传输协议，信息时明文传输；https则是具有安全性的ssl加密传输协议
* http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443
* http的连接很简单，是无状态的；https协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全  
  下面就是https的整个构架，现在的https基本都使用TLS了，因为更加安全，所以下图中SSL应该换成SSL/TLS。

下面就上图中的知识点进行一个大概的介绍。

**二、加解密相关知识**

**1. 对称加密**

对称加密（也叫私钥加密）指加密和解密使用相同密钥的加密算法。有时又叫传统密码算法，就是加密密钥能够从解密密钥中推算出来，同时解密密钥也可以从加密密钥中推算出来。而在大多数的对称算法中，加密秘钥和解密密钥是相同的，所以也称这种加密算法为秘密密钥算法或者单密钥算法。  
常见的对称加密有：DES（Data Encryption Standard）、AES（Advanced Encryption Standard）、RC4、IDEA

**2. 非对称加密**

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）和私有密钥（privatekey）；并且加密密钥和解密密钥是成对出现的。非对称加密算法在加密和解密过程使用了不同的密钥，非对称加密也称为公钥加密，在密钥对中，其中一个密钥是对外公开的，所有人都可以获取到，称为公钥，另一个密钥是不公开的称为私钥。

非对称加密算法对加密内容的长度有限制，不能超过公钥长度。比如现在常用的公钥长度是2048位，意味着加密内容不能超过256个字节。

**3. 摘要算法**

数字摘要是采用单项Hash函数将需要加密的明文“摘要”成一串固定长度（128位）的密文，这一串密文又称为数字指纹，它有固定的长度，而且不同的明文摘要成密文，其结果总是不同的，而同样的明文其摘要必定一致。“数字摘要”是https能确保数据完整性和防篡改的根本原因。

**4. 数字签名**

数字签名技术就和对“非对称密钥加解密”和“数字摘要”两项技术的应用，它将摘要信息用发送者的私钥加密，与原文一起传送给接收者。接收者只有用发送者的公钥才能解密被加密的摘要信息，然后用HASH函数对收到的原文产生一个摘要信息，与解密的摘要信息对比。如果相同，则说明收到的信息是完整的，在传输过程中没有被修改，否则说明信息被修改过，因此数字签名能够验证信息的完整性。

数字签名的过程如下：  
明文 ——> hash算法 ——> 摘要 ——> 私钥加密 ——> 数字签名

数字签名有两种功效：  
一、能确定消息确实是由发送方签名并发出来的，因为别人假冒不了发送方的签名。  
二、数字签名能确定消息的完整性。

**注意：**  
数字签名只能验证数据的完整性，数据本身是否加密不属于数字签名的控制范围

**5. 数字证书**

**1）为什么要有数字证书？**

对于请求方来说，它怎么能确定它所得到的公钥一定是从目标主机那里发布的，而且没有篡改过呢？亦或这请求的目标主机本身就从事窃取用户信息的不正当行为呢？这时候，我们需要有一个权威的值得信赖的第三方机构（一般由政府审核并授权的机构）来统一对外发放主机机构的公钥，只要请求方这种机构获取公钥，就避免了上述问题的发生。

**2）数字证书的颁发过程**

用户首先产生自己的密钥对，并将公钥及部分个人身份信息传送给认证中心。认证中心在核实身份后，将执行一些必要的步骤，以确信请求确实由用户发送而来，然后认证中心将发给用户一个数字证书，该证书包含用户的个人信息和他的公钥信息，同时还附有认证中心的签名信息（根证书私钥签名）。用户就可以使用该证书进行相关的各种活动。数字证书由独立的证书颁发机构发布，数字证书各不相同，每种证书可提供不同级别的可信度。

**3）证书包含哪些内容**

* 证书颁发机构的名称
* 证书本身的数字签名
* 证书持有者公钥
* 证书签名用到的Hash算法（转载者理解：原文转摘要用到，参见下面验证证书的有效性第3条）

**4）验证证书的有效性**

浏览器默认都会内置CA根证书，其中根证书包含了CA的公钥

1. 证书颁发的机构时伪造的：浏览器不认识，直接认为是危险证书
2. 证书颁发的机构确实存在，于是根据CA名，找到对应内置的CA根证书、CA的公钥。用CA的公钥，对伪造的证书的摘要进行解密，发现解不了，认为是危险证书。
3. 对于篡改的证书，是哟CA的公钥对数字签名进行解密得到摘要A，然后再根据签名的HASH算法计算出证书的摘要B，对比A与B，若相等则正常，否则就是被篡改过的。
4. 证书可在其过期前被吊销，通常情况是该证书的私钥已经失密。较新的浏览器如Chrome、Firefox、Opera和Internet Explorer都实现了在线证书状态协议（OCSP）以排除这种情况：浏览器将网站提供的证书的序列号通过OCSP发送给证书颁发机构，后者会告诉浏览器证书是否还是有效的。

**三、SSL与TLS**

**1. SSL（Secure Socket Layer，安全套接字层）**

SSL为Netscape所研发，用以保障在Internet上数据传输之安全，利用数据加密技术，可确保数据在网络上传输过程中不会被截取，当前为3.0版本。

SSL协议可分为两层：SSL记录协议（SSL Recore Protocol）：它建立在可靠地传输协议（如TCP）之上，为高层协议提供数据封装、压缩、加密等基本功能的支持。SSL握手协议（SSL Handshake Protocol）：它建立在SSL记录协议之上，用于在实际的数据传输开始前，通讯双方进行身份证、协商加密算法、交换加密密钥等。

**2. TLS（Transport Layer Security，传输层安全协议）**

用于两个应用程序之间提供保密性和数据完整性。  
TLS 1.0是IETF（Internet Engineering Task Force，Internet工程任务组）制定的一种新的协议，它建立在SSL 3.0协议规范之上，是SSL 3.0的后续版本，可以理解为SSL 3.1，它是写入了RFC的。该协议由两层组成：TLS记录协议（TLS Recore）和TLS握手协议（TLS Handshake）。较低的层为TLS记录协议，位于某个可靠的传输协议（例如TCP）上面。

**3. SSL/TLS协议作用：**

* 认证用户和服务器，确保数据发送到正确的客户机和服务器；（意思是HTTP的可能会发送到不正确的客户机和服务器）
* 加密数据以防止数据中途被窃取；
* 维护数据的完整性，确保数据在传输过程中不被改变。

**4. TLS比SSL的优势**

1. 对于消息认证使用密钥散列法：TLS使用“消息认证代码的密钥散列法”（HMAC），当记录在开放的网络（如因特网）上传送时，该代码确保记录不会被变更。SSLv3.0使用的是键控消息认证（MAC），但HMAC比MAC功能更安全。
2. 增强的伪随机功能（PRF）：PRF生成密钥数据。在TLS中，HMAC定义PRF。PRF使用两种散列算法保证其安全性。如果任一算法暴露了，只要第二种算法未暴露，则数据仍然是安全的。
3. 改进的已完成消息验证：TLS和SSLv3.0都对两个端点提供已完成的消息，该消息认证交换的消息没有被变更。然而，TLS将此已完成消息基于PRF和HMAC值之上，这也比SSLv3.0更安全。
4. 一致证书处理：与SSLv3.0不同，TLS试图指定必须在TLS之间实现交换的证书类型。
5. 特定警报消息：TLS提供更多的特定和附加警报，以指示任一会话端点检测到的问题。TLS还对何时应该发送某些警报进行记录。

**5. SSL、TLS的握手过程**

SSL与TLS握手整个过程如下图所示，下面会详细介绍每一步的具体内容：

**1）客户端首次发出请求**

由于客户端（如浏览器）对一些加密算法的支持程度不一样，但在TLS协议传输过程中必须使用同一套加解密算法才能保证数据能够正常的加解密。在TLS握手阶段，客户端首先要告知服务器端，自己支持哪些加密算法，所以客户端需要将本地支持的加密套件（Cipher Suite）的列表传送给服务端。除此之外，客户端还要产生一个随机数，这个随机数一方面需要在客户端保存，另一方面需要传送给服务端，客户端的随机数需要跟服务端产生的随机数结合起来产生后边要讲的Master Secret。

客户端需要提供如下信息：

* 支持的协议版本，比如TLS 1.0版本
* 一个客户端生成的随机数，稍后用于生成“对话密钥”
* 支持的加密方法，比如RSA公钥加密
* 支持的压缩方法

**2）服务端首次回应**

服务端在接收到客户端的Client Hello之后，服务端需要确定加密协议的版本，以及加密的算法，然后也生成一个随机数，以及将自己的证书发送给客户端，这里的随机数是整个过程的第二个随机数。

服务端需要提供的信息：

* 协议的版本
* 加密的算法
* 随机数
* 服务器证书

**3）客户端再次回应**

客户端首先会对服务器下发的证书进行验证，验证通过后，则会继续下面的操作，客户端再次产生一个随机数（第三个随机数），然后使用服务器证书中的公钥加密，以及放一个ChangeCipherSpec消息即编码改变的消息，还有整个前面所有消息的hash值，进行服务器验证，然后用新密钥（转载者的理解：使用第二次握手后服务器端返回的支持的加密算法随机生成的对称密钥）加密一段数据一并发送到服务器，确保正式通信前无误。  
客户端使用前面的两个随机数以及刚刚生成的新随机数，使用与服务器确定的加密算法，生成一个Session Secret。

ChangeCipherSpec是一个独立的协议，体现在数据包中就是一个字节的数据，用于告知服务端，客户端已经切换到之前协商好的加密套件（Cipher Suite）的状态，准备使用之前协商好的加密套件加密数据并传输了。

**4）服务器再次响应**

服务端在接收到客户端传过来的第三个随机数的加密数据后，使用私钥对解密这段加密数据，并对数据进行验证，也会使用和客户端同样的方式生成密钥。一切准备好后，也会给客户端发送一个ChangeCipherSpec，告知客户端已经切换到协商过的加密套件状态，准备使用加密套件和Session Secret加密数据了。之后，服务端也会使用Session Secret加密一段Finish消息发送给客户端，以验证之前通过握手建立起来的加解密通道是否成功。

转载者疑问：这里的Session Secret是不是就是对称密钥？

会话密钥：https://baike.baidu.com/item/%E4%BC%9A%E8%AF%9D%E5%AF%86%E9%92%A5/8831495?fr=aladdin

**5）后续客户端与服务期间通信**

确定密钥后，服务器与客户端之间就会通过商定的密钥加密数据，进行通讯了。这个握手过程也就基本完成了。

**6. 其他补充**

对于非常重要的保密数据，服务端还需要对客户端进行验证，以保证数据传送给了安全的合法客户端。服务端可以向客户端发出Cerficate Request消息，要求客户端发送证书对客户端的合法性进行验证。比如，金融机构往往只允许认证客户连入自己的网络，就会向正式客户提供USB密钥，里面就包含了一张客户端证书。

PreMaster Secret前两个字节是TLS的版本号，这是一个比较重要的用来核对握手数据的版本号，因为在Client Hello阶段，客户端会发送一份加密套件列表和当前支持的SSL/TLS的版本号给服务端，而且是使用明文传送的，如果握手的数据包被破解之后，攻击者很有可能篡改数据包，选择一个安全性较低的加密套件和版本给服务端，从而对数据进行破解。所以，服务端需要对密文中解密出来的PreMaster版本号跟之前Client Hello阶段的版本号进行对比，如果版本号变低，则说明被篡改，则立即停止发送任何消息。

**session的恢复**

有两种方法可以恢复原来的session：一种叫做session ID，另一种叫做session ticket。

**1）session ID**

session ID的思想很简单，就是每一次对话都有一个编号（session ID）。如果对话中断，下次重连的时候，只要客户端给出这个编号，且服务器有这个编号的记录，双方就可以重新使用已有的“对话密钥”，而不必重新生成一把。

session ID是目前所有浏览器都支持的方法，但是它的缺点在于session ID往往只保留在一台服务器上。所以，如果客户端的请求发送到另一台服务器，就无法恢复对话。

**2）session ticket**

客户端发送一个服务器在上一次对话中发送过来的session ticket。这个session ticket是加密的，只有服务器才能解密，其中包括本次对话的主要信息，比如对话密钥和加密方法。当服务器收到session ticket以后，解密后就不必重新生成对话密钥了。

目前只有Firefox和Chrome浏览器支持。

**总结**

https实际上就是在TCP层与http层之间加入了SSL/TLS来为上层的安全保驾护航，主要用到对称加密、非对称加密、证书等技术进行客户端与服务器的数据加密传输，最终达到保证整个通信的安全性。

参考文章  
[数字证书的基础知识](https://link.jianshu.com/?t=http://www.enkichen.com/2016/02/26/digital-certificate-based/)  
[HTTPS科普扫盲贴](https://link.jianshu.com/?t=https://segmentfault.com/a/1190000004523659)  
[和安全有关的那些事](https://link.jianshu.com/?t=http://blog.csdn.net/bluishglc/article/details/7585965)  
[OpenSSL与SSL数字证书概念贴](https://link.jianshu.com/?t=http://seanlook.com/2015/01/15/openssl-certificate-encryption/)  
[基于OpenSSL自建CA和颁发SSL证书](https://link.jianshu.com/?t=http://seanlook.com/2015/01/18/openssl-self-sign-ca/)  
[聊聊HTTPS和SSL/TLS协议](https://link.jianshu.com/?t=http://www.techug.com/post/https-ssl-tls.html)  
[SSL/TLS协议运行机制的概述](https://link.jianshu.com/?t=http://www.ruanyifeng.com/blog/2014/02/ssl_tls.html)  
[图解SSL/TLS协议](https://link.jianshu.com/?t=http://www.ruanyifeng.com/blog/2014/09/illustration-ssl.html)  
[大型网站的HTTPS实践](https://link.jianshu.com/?t=http://developer.baidu.com/resources/online/doc/security/https-pratice-1.html)  
[SSL/TLS原理详解](https://link.jianshu.com/?t=https://segmentfault.com/a/1190000002554673)  
[扒一扒HTTPS网站的内幕](https://link.jianshu.com/?t=https://segmentfault.com/a/1190000003801450)  
[白话解释OSI模型，TLS/SSL及HTTPS](https://link.jianshu.com/?t=https://segmentfault.com/a/1190000004467714)  
[OpenSSL HearBleed漏洞原理漫画图解](https://link.jianshu.com/?t=http://c.colabug.com/forum.php?mod=viewthread&ordertype=2&tid=1053791)