# HTTPS 科普扫盲帖

2016/09/12 · [IT技术](http://blog.jobbole.com/category/it-tech/) · [HTTPS](http://blog.jobbole.com/tag/https/)

分享到：**1**

* **[强力django+杀手级xadmin打造上线标准的在线教育平台](http://coding.imooc.com/class/78.html?from=jobboleblog" \o "强力django+杀手级xadmin打造上线标准的在线教育平台" \t "http://blog.jobbole.com/105496/_blank)**
* **[Android网络层架构设计实战基于okhttp3](http://coding.imooc.com/class/70.html?from=jobboleblog" \o "Android网络层架构设计实战基于okhttp3" \t "http://blog.jobbole.com/105496/_blank)**
* **[前端后台ThinkPHP开发整站](http://coding.imooc.com/class/48.html?from=jobboleblog" \o "前端后台ThinkPHP开发整站" \t "http://blog.jobbole.com/105496/_blank)**
* **[Android5.0+高级动画开发 矢量图动画 轨迹动画 路径变换](http://coding.imooc.com/class/76.html?from=jobboleblog" \o "Android5.0+高级动画开发 矢量图动画 轨迹动画 路径变换" \t "http://blog.jobbole.com/105496/_blank)**

原文出处： [程序猿小卡](http://www.cnblogs.com/chyingp/p/https-introduction.html" \t "http://blog.jobbole.com/105496/_blank)

## **为什么需要https**

HTTP是明文传输的，也就意味着，介于发送端、接收端中间的任意节点都可以知道你们传输的内容是什么。这些节点可能是路由器、代理等。

举个最常见的例子，用户登陆。用户输入账号，密码，采用HTTP的话，只要在代理服务器上做点手脚就可以拿到你的密码了。

用户登陆 –> 代理服务器（做手脚）–> 实际授权服务器

在发送端对密码进行加密？没用的，虽然别人不知道你原始密码是多少，但能够拿到加密后的账号密码，照样能登陆。

## **HTTPS是如何保障安全的**

HTTPS其实就是****secure http****的意思啦，也就是HTTP的安全升级版。稍微了解网络基础的同学都知道，HTTP是应用层协议，位于HTTP协议之下是传输协议TCP。TCP负责传输，HTTP则定义了数据如何进行包装。

HTTP –> TCP （明文传输）

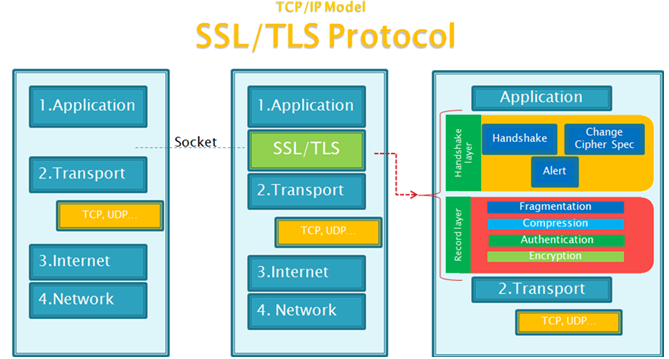
HTTPS相对于HTTP有哪些不同呢？其实就是在HTTP跟TCP中间加多了一层加密层****TLS/SSL****。

****神马是TLS/SSL？****

通俗的讲，TLS、SSL其实是类似的东西，SSL是个加密套件，负责对HTTP的数据进行加密。TLS是SSL的升级版。现在提到HTTPS，加密套件基本指的是TLS。

****传输加密的流程****

原先是应用层将数据直接给到TCP进行传输，现在改成应用层将数据给到TLS/SSL，将数据加密后，再给到TCP进行传输。

大致如图所示。  


就是这么回事。将数据加密后再传输，而不是任由数据在复杂而又充满危险的网络上明文裸奔，在很大程度上确保了数据的安全。这样的话，即使数据被中间节点截获，坏人也看不懂。

## **HTTPS是如何加密数据的**

对安全或密码学基础有了解的同学，应该知道常见的加密手段。一般来说，加密分为对称加密、非对称加密（也叫公开密钥加密）。

### **对称加密**

****对称加密****的意思就是，加密数据用的密钥，跟解密数据用的密钥是一样的。

对称加密的优点在于加密、解密效率通常比较高。缺点在于，数据发送方、数据接收方需要协商、共享同一把密钥，并确保密钥不泄露给其他人。此外，对于多个有数据交换需求的个体，两两之间需要分配并维护一把密钥，这个带来的成本基本是不可接受的。

### **非对称加密**

****非对称加密****的意思就是，加密数据用的密钥（公钥），跟解密数据用的密钥（私钥）是不一样的。

什么叫做公钥呢？其实就是字面上的意思——公开的密钥，谁都可以查到。因此非对称加密也叫做公开密钥加密。

相对应的，私钥就是非公开的密钥，一般是由网站的管理员持有。

公钥、私钥两个有什么联系呢？

简单的说就是，通过公钥加密的数据，只能通过私钥解开。通过私钥加密的数据，只能通过公钥解开。

很多同学都知道用私钥能解开公钥加密的数据，但忽略了一点，私钥加密的数据，同样可以用公钥解密出来。而这点对于理解HTTPS的整套加密、授权体系非常关键。

### **举个非对称加密的例子**

* 登陆用户：小明
* 授权网站：某知名社交网站（以下简称XX）

小明都是某知名社交网站XX的用户，XX出于安全考虑在登陆的地方用了非对称加密。小明在登陆界面敲入账号、密码，点击“登陆”。于是，浏览器利用公钥对小明的账号密码进行了加密，并向XX发送登陆请求。XX的登陆授权程序通过私钥，将账号、密码解密，并验证通过。之后，将小明的个人信息（含隐私），通过私钥加密后，传输回浏览器。浏览器通过公钥解密数据，并展示给小明。

* 步骤一： 小明输入账号密码 –> 浏览器用公钥加密 –> 请求发送给XX
* 步骤二： XX用私钥解密，验证通过 –> 获取小明社交数据，用私钥加密 –> 浏览器用公钥解密数据，并展示。

用非对称加密，就能解决数据传输安全的问题了吗？前面特意强调了一下，私钥加密的数据，公钥是可以解开的，而公钥又是加密的。也就是说，非对称加密只能保证单向数据传输的安全性。

此外，还有公钥如何分发/获取的问题。下面会对这两个问题进行进一步的探讨。

## **公开密钥加密：两个明显的问题**

前面举了小明登陆社交网站XX的例子，并提到，单纯使用公开密钥加密存在两个比较明显的问题。

1. 公钥如何获取
2. 数据传输仅单向安全

### **问题一：公钥如何获取**

浏览器是怎么获得XX的公钥的？当然，小明可以自己去网上查，XX也可以将公钥贴在自己的主页。然而，对于一个动不动就成败上千万的社交网站来说，会给用户造成极大的不便利，毕竟大部分用户都不知道“公钥”是什么东西。

### **问题二：数据传输仅单向安全**

前面提到，公钥加密的数据，只有私钥能解开，于是小明的账号、密码是安全了，半路不怕被拦截。

然后有个很大的问题：****私钥加密的数据，公钥也能解开****。加上公钥是公开的，小明的隐私数据相当于在网上换了种方式裸奔。（中间代理服务器拿到了公钥后，毫不犹豫的就可以解密小明的数据）

下面就分别针对这两个问题进行解答。

## **问题一：公钥如何获取**

这里要涉及两个非常重要的概念：证书、CA（证书颁发机构）。

****证书****

可以暂时把它理解为网站的身份证。这个身份证里包含了很多信息，其中就包含了上面提到的公钥。

也就是说，当小明、小王、小光等用户访问XX的时候，再也不用满世界的找XX的公钥了。当他们访问XX的时候，XX就会把证书发给浏览器，告诉他们说，乖，用这个里面的公钥加密数据。

这里有个问题，所谓的“证书”是哪来的？这就是下面要提到的CA负责的活了。

****CA（证书颁发机构）****

强调两点：

1. 可以颁发证书的CA有很多（国内外都有）。
2. 只有少数CA被认为是权威、公正的，这些CA颁发的证书，浏览器才认为是信得过的。比如****VeriSign****。（CA自己伪造证书的事情也不是没发生过。。。）

证书颁发的细节这里先不展开，可以先简单理解为，网站向CA提交了申请，CA审核通过后，将证书颁发给网站，用户访问网站的时候，网站将证书给到用户。

至于证书的细节，同样在后面讲到。

## **问题二：数据传输仅单向安全**

上面提到，通过私钥加密的数据，可以用公钥解密还原。那么，这是不是就意味着，网站传给用户的数据是不安全的？

答案是：是！！！（三个叹号表示强调的三次方）

看到这里，可能你心里会有这样想：用了HTTPS，数据还是裸奔，这么不靠谱，还不如直接用HTTP来的省事。

但是，为什么业界对网站HTTPS化的呼声越来越高呢？这明显跟我们的感性认识相违背啊。

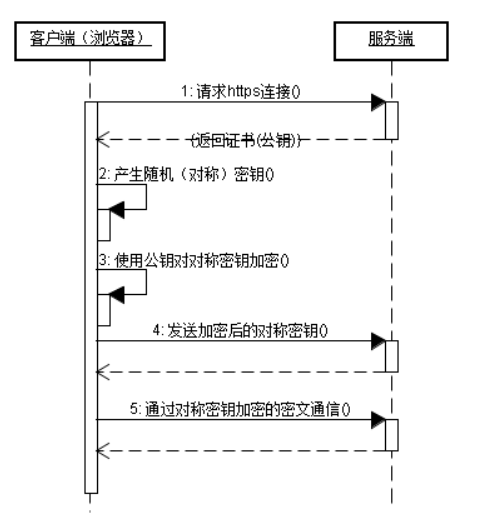
因为：HTTPS虽然用到了公开密钥加密，但同时也结合了其他手段，如对称加密，来确保授权、加密传输的效率、安全性。

概括来说，整个简化的加密通信的流程就是：

1. 小明访问XX，XX将自己的证书给到小明（其实是给到浏览器，小明不会有感知）
2. 浏览器从证书中拿到XX的公钥A
3. 浏览器生成一个只有自己自己的对称密钥B，用公钥A加密，并传给XX（其实是有协商的过程，这里为了便于理解先简化）
4. XX通过私钥解密，拿到对称密钥B
5. 浏览器、XX 之后的数据通信，都用密钥B进行加密

注意：对于每个访问XX的用户，生成的对称密钥B理论上来说都是不一样的。比如小明、小王、小光，可能生成的就是B1、B2、B3.

参考下图：（附上[原图出处](http://www.cnblogs.com/ttltry-air/archive/2012/08/20/2647898.html)）



## **证书可能存在哪些问题**

了解了HTTPS加密通信的流程后，对于数据裸奔的疑虑应该基本打消了。然而，细心的观众可能又有疑问了：怎么样确保证书有合法有效的？

证书非法可能有两种情况：

1. 证书是伪造的：压根不是CA颁发的
2. 证书被篡改过：比如将XX网站的公钥给替换了

****举个例子：****

我们知道，这个世界上存在一种东西叫做代理，于是，上面小明登陆XX网站有可能是这样的，小明的登陆请求先到了代理服务器，代理服务器再将请求转发到的授权服务器。

小明 –> 邪恶的代理服务器 –> 登陆授权服务器  
小明

然后，这个世界坏人太多了，某一天，代理服务器动了坏心思（也有可能是被入侵），将小明的请求拦截了。同时，返回了一个非法的证书。

小明 –> 邪恶的代理服务器 –x–> 登陆授权服务器  
小明 登陆授权服务器

如果善良的小明相信了这个证书，那他就再次裸奔了。当然不能这样，那么，是通过什么机制来防止这种事情的发生的呢。

下面，我们先来看看”证书”有哪些内容，然后就可以大致猜到是如何进行预防的了。

## **证书简介**

在正式介绍证书的格式前，先插播个小广告，科普下数字签名和摘要，然后再对证书进行非深入的介绍。

为什么呢？因为数字签名、摘要是证书防伪非常关键的武器。

### **数字签名与摘要**

简单的来说，“摘要”就是对传输的内容，通过hash算法计算出一段固定长度的串（是不是联想到了文章摘要）。然后，在通过CA的私钥对这段摘要进行加密，加密后得到的结果就是“数字签名”。（这里提到CA的私钥，后面再进行介绍）

明文 –> hash运算 –> 摘要 –> 私钥加密 –> 数字签名

结合上面内容，我们知道，这段数字签名只有CA的公钥才能够解密。

接下来，我们再来看看神秘的“证书”究竟包含了什么内容，然后就大致猜到是如何对非法证书进行预防的了。

数字签名、摘要进一步了解可参考 [这篇文章](http://blog.csdn.net/bluishglc/article/details/7585965)。

### **证书格式**

先无耻的贴上一大段内容，证书格式来自这篇不错的文章《[OpenSSL 与 SSL 数字证书概念贴](http://seanlook.com/2015/01/15/openssl-certificate-encryption/)》

内容非常多，这里我们需要关注的有几个点：

1. 证书包含了颁发证书的机构的名字 — CA
2. 证书内容本身的数字签名（用CA私钥加密）
3. 证书持有者的公钥
4. 证书签名用到的hash算法

此外，有一点需要补充下，就是：

1. CA本身有自己的证书，江湖人称“根证书”。这个“根证书”是用来证明CA的身份的，本质是一份普通的数字证书。
2. 浏览器通常会内置大多数主流权威CA的根证书。

****证书格式****

1. 证书版本号(Version)  
版本号指明X.509证书的格式版本，现在的值可以为:  
1) 0: v1  
2) 1: v2  
3) 2: v3  
也为将来的版本进行了预定义

2. 证书序列号(Serial Number)  
序列号指定由CA分配给证书的唯一的”数字型标识符”。当证书被取消时，实际上是将此证书的序列号放入由CA签发的CRL中，  
这也是序列号唯一的原因。

3. 签名算法标识符(Signature Algorithm)  
签名算法标识用来指定由CA签发证书时所使用的”签名算法”。算法标识符用来指定CA签发证书时所使用的:  
1) 公开密钥算法  
2) hash算法  
example: sha256WithRSAEncryption  
须向国际知名标准组织(如ISO)注册

4. 签发机构名(Issuer)  
此域用来标识签发证书的CA的X.500 DN(DN-Distinguished Name)名字。包括:  
1) 国家(C)  
2) 省市(ST)  
3) 地区(L)  
4) 组织机构(O)  
5) 单位部门(OU)  
6) 通用名(CN)  
7) 邮箱地址

5. 有效期(Validity)  
指定证书的有效期，包括:  
1) 证书开始生效的日期时间  
2) 证书失效的日期和时间  
每次使用证书时，需要检查证书是否在有效期内。

6. 证书用户名(Subject)  
指定证书持有者的X.500唯一名字。包括:  
1) 国家(C)  
2) 省市(ST)  
3) 地区(L)  
4) 组织机构(O)  
5) 单位部门(OU)  
6) 通用名(CN)  
7) 邮箱地址

7. 证书持有者公开密钥信息(Subject Public Key Info)  
证书持有者公开密钥信息域包含两个重要信息:  
1) 证书持有者的公开密钥的值  
2) 公开密钥使用的算法标识符。此标识符包含公开密钥算法和hash算法。  
8. 扩展项(extension)  
X.509 V3证书是在v2的基础上一标准形式或普通形式增加了扩展项，以使证书能够附带额外信息。标准扩展是指  
由X.509 V3版本定义的对V2版本增加的具有广泛应用前景的扩展项，任何人都可以向一些权威机构，如ISO，来  
注册一些其他扩展，如果这些扩展项应用广泛，也许以后会成为标准扩展项。

9. 签发者唯一标识符(Issuer Unique Identifier)  
签发者唯一标识符在第2版加入证书定义中。此域用在当同一个X.500名字用于多个认证机构时，用一比特字符串  
来唯一标识签发者的X.500名字。可选。

10. 证书持有者唯一标识符(Subject Unique Identifier)  
持有证书者唯一标识符在第2版的标准中加入X.509证书定义。此域用在当同一个X.500名字用于多个证书持有者时，  
用一比特字符串来唯一标识证书持有者的X.500名字。可选。

11. 签名算法(Signature Algorithm)  
证书签发机构对证书上述内容的签名算法  
example: sha256WithRSAEncryption

12. 签名值(Issuer’s Signature)  
证书签发机构对证书上述内容的签名值

### **如何辨别非法证书**

上面提到，XX证书包含了如下内容：

1. 证书包含了颁发证书的机构的名字 — CA
2. 证书内容本身的数字签名（用CA私钥加密）
3. 证书持有者的公钥
4. 证书签名用到的hash算法

浏览器内置的CA的根证书包含了如下关键内容：

1. CA的公钥（非常重要！！！）

好了，接下来针对之前提到的两种非法证书的场景，讲解下怎么识别

### **完全伪造的证书**

这种情况比较简单，对证书进行检查：

1. 证书颁发的机构是伪造的：浏览器不认识，直接认为是危险证书
2. 证书颁发的机构是确实存在的，于是根据CA名，找到对应内置的CA根证书、CA的公钥。
3. 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

### **篡改过的证书**

假设代理通过某种途径，拿到XX的证书，然后将证书的公钥偷偷修改成自己的，然后喜滋滋的认为用户要上钩了。然而太单纯了：

1. 检查证书，根据CA名，找到对应的CA根证书，以及CA的公钥。
2. 用CA的公钥，对证书的数字签名进行解密，得到对应的证书摘要AA
3. 根据证书签名使用的hash算法，计算出当前证书的摘要BB
4. 对比AA跟BB，发现不一致–> 判定是危险证书

## **HTTPS握手流程**

上面啰啰嗦嗦讲了一大通，HTTPS如何确保数据加密传输的安全的机制基本都覆盖到了，太过技术细节的就直接跳过了。

最后还有最后两个问题：

1. 网站是怎么把证书给到用户（浏览器）的
2. 上面提到的对称密钥是怎么协商出来的

上面两个问题，其实就是HTTPS握手阶段要干的事情。HTTPS的数据传输流程整体上跟HTTP是类似的，同样包含两个阶段：握手、数据传输。

1. 握手：证书下发，密钥协商（这个阶段都是明文的）
2. 数据传输：这个阶段才是加密的，用的就是握手阶段协商出来的对称密钥

阮老师的文章写的非常不错，通俗易懂，感兴趣的同学可以看下。

附：《[SSL/TLS 协议运行机制的概述](http://blog.jobbole.com/105405/" \t "http://blog.jobbole.com/105496/_blank)》

## **写在后面**

科普性文章，部分内容不够严谨，如有错漏请指出 :)