【HTTP】HTTPS 和 SSL/TLS 协议

要说清楚 HTTPS 协议的实现原理，至少需要如下几个背景知识。

1. 大致了解几个基本术语（HTTPS、SSL、TLS）的含义
2. 大致了解 HTTP 和 TCP 的关系（尤其是“短连接”VS“长连接”）
3. 大致了解加密算法的概念（尤其是“对称加密与非对称加密”的区别）
4. 大致了解 CA 证书的用途

考虑到很多技术菜鸟可能不了解上述背景，俺先用最简短的文字描述一下。如果你自认为不是菜鸟，请略过本章节，直接去看“HTTPS 协议的需求”。

先澄清几个术语——HTTPS、SSL、TLS

1. “HTTP”是干嘛用滴？

首先，HTTP 是一个网络协议，是专门用来帮你传输 Web 内容滴。关于这个协议，就算你不了解，至少也听说过吧？比如你访问俺的博客的主页，浏览器地址栏会出现如下的网址 http://www.techug.com/俺加了粗体的部分就是指 HTTP 协议。大部分网站都是通过 HTTP 协议来传输 Web 页面、以及 Web 页面上包含的各种东东（图片、CSS 样式、JS 脚本）。

2. “SSL/TLS”是干嘛用滴？

SSL 是洋文“Secure Sockets Layer”的缩写，中文叫做“安全套接层”。它是在上世纪 90 年代中期，由网景公司设计的。（顺便插一句，网景公司不光发明了 SSL，还发明了很多 Web 的基础设施——比如“CSS 样式表”和“JS 脚本”）

为啥要发明 SSL 这个协议捏？因为原先互联网上使用的 HTTP 协议是明文的，存在很多缺点——比如传输内容会被偷窥（嗅探）和篡改。发明 SSL 协议，就是为了解决这些问题。

到了 1999 年，SSL 因为应用广泛，已经成为互联网上的事实标准。IETF 就在那年把 SSL 标准化。标准化之后的名称改为 TLS（是“Transport Layer Security”的缩写），中文叫做“传输层安全协议”。

很多相关的文章都把这两者并列称呼（SSL/TLS），因为这两者可以视作同一个东西的不同阶段。

3. “HTTPS”是啥意思？

解释完 HTTP 和 SSL/TLS，现在就可以来解释 HTTPS 啦。咱们通常所说的 HTTPS 协议，说白了就是“HTTP 协议”和“SSL/TLS 协议”的组合。你可以把 HTTPS 大致理解为——“HTTP over SSL”或“HTTP over TLS”（反正 SSL 和 TLS 差不多）。

再来说说 HTTP 协议的特点

作为背景知识介绍，还需要再稍微谈一下 HTTP 协议本身的特点。HTTP 本身有很多特点，考虑到篇幅有限，俺只谈那些和 HTTPS 相关的特点。

1. HTTP 的版本和历史

如今咱们用的 HTTP 协议，版本号是 1.1（也就是 HTTP 1.1）。这个 1.1 版本是 1995 年底开始起草的（技术文档是 RFC2068），并在 1999 年正式发布（技术文档是 RFC2616）。

在 1.1 之前，还有曾经出现过两个版本“0.9 和 1.0”，其中的 HTTP 0.9 【没有】被广泛使用，而 HTTP 1.0 被广泛使用过。

另外，据说明年（2015）IETF 就要发布 HTTP 2.0 的标准了。俺拭目以待。

2. HTTP 和 TCP 之间的关系

简单地说，TCP 协议是 HTTP 协议的基石——HTTP 协议需要依靠 TCP 协议来传输数据。

在网络分层模型中，TCP 被称为“传输层协议”，而 HTTP 被称为“应用层协议”。

有很多常见的应用层协议是以 TCP 为基础的，比如“FTP、SMTP、POP、IMAP”等。

TCP 被称为“面向连接”的传输层协议。关于它的具体细节，俺就不展开了（否则篇幅又失控了）。你只需知道：传输层主要有两个协议，分别是 TCP 和 UDP。TCP 比 UDP 更可靠。你可以把 TCP 协议想象成某个水管，发送端这头进水，接收端那头就出水。并且 TCP 协议能够确保，先发送的数据先到达（与之相反，UDP 不保证这点）。

3. HTTP 协议如何使用 TCP 连接？

HTTP 对 TCP 连接的使用，分为两种方式：俗称“短连接”和“长连接”（“长连接”又称“持久连接”，洋文叫做“Keep-Alive”或“Persistent Connection”）

假设有一个网页，里面包含好多图片，还包含好多【外部的】CSS 文件和 JS 文件。在“短连接”的模式下，浏览器会先发起一个 TCP 连接，拿到该网页的 HTML 源代码（拿到 HTML 之后，这个 TCP 连接就关闭了）。然后，浏览器开始分析这个网页的源码，知道这个页面包含很多外部资源（图片、CSS、JS）。然后针对【每一个】外部资源，再分别发起一个个 TCP 连接，把这些文件获取到本地（同样的，每抓取一个外部资源后，相应的 TCP 就断开）

相反，如果是“长连接”的方式，浏览器也会先发起一个 TCP 连接去抓取页面。但是抓取页面之后，该 TCP 连接并不会立即关闭，而是暂时先保持着（所谓的“Keep-Alive”）。然后浏览器分析 HTML 源码之后，发现有很多外部资源，就用刚才那个 TCP 连接去抓取此页面的外部资源。

在 HTTP 1.0 版本，【默认】使用的是“短连接”（那时候是 Web 诞生初期，网页相对简单，“短连接”的问题不大）；

到了 1995 年底开始制定 HTTP 1.1 草案的时候，网页已经开始变得复杂（网页内的图片、脚本越来越多了）。这时候再用短连接的方式，效率太低下了（因为建立 TCP 连接是有“时间成本”和“CPU 成本”滴）。所以，在 HTTP 1.1 中，【默认】采用的是“Keep-Alive”的方式。

关于“Keep-Alive”的更多介绍，可以参见维基百科词条（在“这里”）

谈谈“对称加密”和“非对称加密”的概念

1. 啥是“加密”和“解密”？

通俗而言，你可以把“加密”和“解密”理解为某种【互逆的】数学运算。就好比“加法和减法”互为逆运算、“乘法和除法”互为逆运算。

“加密”的过程，就是把“明文”变成“密文”的过程；反之，“解密”的过程，就是把“密文”变为“明文”。在这两个过程中，都需要一个关键的东东——叫做“密钥”——来参与数学运算。

2. 啥是“对称加密”？

所谓的“对称加密技术”，意思就是说：“加密”和“解密”使用【相同的】密钥。这个比较好理解。就好比你用 7zip 或 WinRAR 创建一个带密码（口令）的加密压缩包。当你下次要把这个压缩文件解开的时候，你需要输入【同样的】密码。在这个例子中，密码/口令就如同刚才说的“密钥”。

3. 啥是“非对称加密”？

所谓的“非对称加密技术”，意思就是说：“加密”和“解密”使用【不同的】密钥。这玩意儿比较难理解，也比较难想到。当年“非对称加密”的发明，还被誉为“密码学”历史上的一次革命。

由于篇幅有限，对“非对称加密”这个话题，俺就不展开了。有空的话，再单独写一篇扫盲。

4. 各自有啥优缺点？

看完刚才的定义，很显然：（从功能角度而言）“非对称加密”能干的事情比“对称加密”要多。这是“非对称加密”的优点。但是“非对称加密”的实现，通常需要涉及到“复杂数学问题”。所以，“非对称加密”的性能通常要差很多（相对于“对称加密”而言）。

这两者的优缺点，也影响到了 SSL 协议的设计。

CA 证书的原理及用途

关于这方面，请看俺 4 年前写的《数字证书及 CA 的扫盲介绍》。这里就不再重复唠叨了，免得篇幅太长。

HTTPS 协议的需求是啥？

花了好多口水，终于把背景知识说完了。下面正式进入正题。先来说说当初设计 HTTPS 是为了满足哪些需求？

很多介绍 HTTPS 的文章一上来就给你讲实现细节。个人觉得：这是不好的做法。早在 2009 年开博的时候，发过一篇《学习技术的三部曲：WHAT、HOW、WHY》，其中谈到“WHY 型问题”的重要性。一上来就给你讲协议细节，你充其量只能知道 WHAT 和 HOW，无法理解 WHY。俺在前一个章节讲了“背景知识”，在这个章节讲了“需求”，这就有助于你理解：当初

为什么

要设计成这样？——这就是 WHY 型的问题。

兼容性

因为是先有 HTTP 再有 HTTPS。所以，HTTPS 的设计者肯定要考虑到对原有 HTTP 的兼容性。

这里所说的兼容性包括很多方面。比如已有的 Web 应用要尽可能无缝地迁移到 HTTPS；比如对浏览器厂商而言，改动要尽可能小；……

基于“兼容性”方面的考虑，很容易得出如下几个结论：

1. HTTPS 还是要基于 TCP 来传输

（如果改为 UDP 作传输层，无论是 Web 服务端还是浏览器客户端，都要大改，动静

太大了）

2. 单独使用一个新的协议，把 HTTP 协议包裹起来

（所谓的“HTTP over SSL”，实际上是在原有的 HTTP 数据外面加了一层 SSL 的封装。HTTP 协议原有的 GET、POST 之类的机制，基本上原封不动）

打个比方：如果原来的 HTTP 是塑料水管，容易被戳破；那么如今新设计的 HTTPS 就像是在原有的塑料水管之外，再包一层金属水管。一来，原有的塑料水管照样运行；二来，用金属加固了之后，不容易被戳破。

可扩展性

前面说了，HTTPS 相当于是“HTTP over SSL”。

如果 SSL 这个协议在“可扩展性”方面的设计足够牛逼，那么它除了能跟 HTTP 搭配，还能够跟其它的应用层协议搭配。岂不美哉？

现在看来，当初设计 SSL 的人确实比较牛。如今的 SSL/TLS 可以跟很多常用的应用层协议（比如：FTP、SMTP、POP、Telnet）搭配，来强化这些应用层协议的安全性。

接着刚才打的比方：如果把 SSL/TLS 视作一根用来加固的金属管，它不仅可以用来加固输水的管道，还可以用来加固输煤气的管道。

保密性（防泄密）

HTTPS 需要做到足够好的保密性。

说到保密性，首先要能够对抗嗅探（行话叫 Sniffer）。所谓的“嗅探”，通俗而言就是监视你的网络传输流量。如果你使用明文的 HTTP 上网，那么监视者通过嗅探，就知道你在访问哪些网站的哪些页面。

嗅探是最低级的攻击手法。除了嗅探，HTTPS 还需要能对抗其它一些稍微高级的攻击手法——比如“重放攻击”（后面讲协议原理的时候，会再聊）。

完整性（防篡改）

除了“保密性”，还有一个同样重要的目标是“确保完整性”。关于“完整性”这个概念，在之前的博文《扫盲文件完整性校验——关于散列值和数字签名》中大致提过。健忘的同学再去温习一下。

在发明 HTTPS 之前，由于 HTTP 是明文的，不但容易被嗅探，还容易被篡改。

举个例子：

比如咱们天朝的网络运营商（ISP）都比较流氓，经常有网友抱怨说访问某网站（本来是没有广告的），竟然会跳出很多中国电信的广告。为啥会这样捏？因为你的网络流量需要经过 ISP 的线路才能到达公网。如果你使用的是明文的 HTTP，ISP 很容易就可以在你访问的页面中植入广告。

所以，当初设计 HTTPS 的时候，还有一个需求是“确保 HTTP 协议的内容不被篡改”。

真实性（防假冒）

在谈到 HTTPS 的需求时，“真实性”经常被忽略。其实“真实性”的重要程度不亚于前面的“保密性”和“完整性”。

举个例子：

你因为使用网银，需要访问该网银的 Web 站点。那么，你如何确保你访问的网站确实是你想访问的网站？（这话有点绕口令）

有些天真的同学会说：通过看网址里面的域名，来确保。为啥说这样的同学是“天真的”？因为 DNS 系统本身是不可靠的（尤其是在设计 SSL 的那个年代，连 DNSSEC 都还没发明）。

由于 DNS 的不可靠（存在“域名欺骗”和“域名劫持”），你看到的网址里面的域名【未必】是真实滴！

（不了解“域名欺骗”和“域名劫持”的同学，可以参见俺之前写的《扫盲 DNS 原理，兼谈“域名劫持”和“域名欺骗/域名污染”》）

所以，HTTPS 协议必须有某种机制来确保“真实性”的需求（至于如何确保，后面会细聊）。

性能

再来说最后一个需求——性能。

引入 HTTPS 之后，【不能】导致性能变得太差。否则的话，谁还愿意用？

为了确保性能，SSL 的设计者至少要考虑如下几点：

1. 如何选择加密算法（“对称”or“非对称”）？
2. 如何兼顾 HTTP 采用的“短连接”TCP 方式？

（SSL 是在 1995 年之前开始设计的，那时候的 HTTP 版本还是 1.0，默认使用的是“短连接”的 TCP 方式——默认不启用 Keep-Alive）

前言

HTTPS（全称：HyperText Transfer Protocol over Secure Socket Layer），其实 HTTPS 并不是一个新鲜协议，Google 很早就开始启用了，初衷是为了保证数据安全。 近两年，Google、Baidu、Facebook 等这样的互联网巨头，不谋而合地开始大力推行 HTTPS， 国内外的大型互联网公司很多也都已经启用了全站 HTTPS，这也是未来互联网发展的趋势。

为鼓励全球网站的 HTTPS 实现，一些互联网公司都提出了自己的要求：

1）Google 已调整搜索引擎算法，让采用 HTTPS 的网站在搜索中排名更靠前；

2）从 2017 年开始，Chrome 浏览器已把采用 HTTP 协议的网站标记为不安全网站；

3）苹果要求 2017 年 App Store 中的所有应用都必须使用 HTTPS 加密连接；

4）当前国内炒的很火热的微信小程序也要求必须使用 HTTPS 协议；

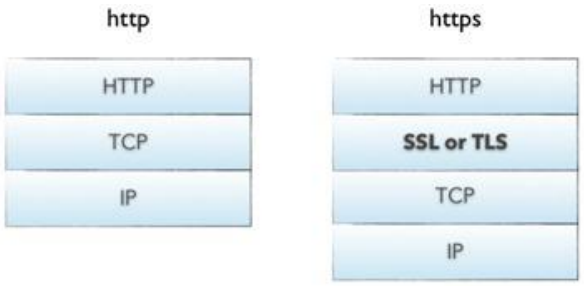
5）新一代的 HTTP/2 协议的支持需以 HTTPS 为基础。

等等，因此想必在不久的将来，全网 HTTPS 势在必行。

概念、协议

1、HTTP 协议（HyperText Transfer Protocol，超文本传输协议）：是客户端浏览器或其他程序与 Web 服务器之间的应用层通信协议 。

2、HTTPS 协议（HyperText Transfer Protocol over Secure Socket Layer）：可以理解为 HTTP+SSL/TLS， 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，用于安全的 HTTP 数据传输。



如上图所示 HTTPS 相比 HTTP 多了一层 SSL/TLS

SSL（Secure Socket Layer，安全套接字层）：1994 年为 Netscape 所研发，SSL 协议位于 TCP/IP 协议与各种应用层协议之间，为数据通讯提供安全支持。

TLS（Transport Layer Security，传输层安全）：其前身是 SSL，它最初的几个版本（SSL 1.0、SSL 2.0、SSL 3.0）由网景公司开发，1999 年从 3.1 开始被 IETF 标准化并改名，发展至今已经有 TLS 1.0、TLS 1.1、TLS 1.2 三个版本。SSL3.0 和 TLS1.0 由于存在安全漏洞，已经很少被使用到。TLS 1.3 改动会比较大，目前还在草案阶段，目前使用最广泛的是 TLS 1.1、TLS 1.2。

加密算法：

据记载，公元前 400 年，古希腊人就发明了置换密码；在第二次世界大战期间，德国军方启用了“恩尼格玛”密码机，所以密码学在社会发展中有着广泛的用途。

1、对称加密

有流式、分组两种，加密和解密都是使用的同一个密钥。

例如：DES、AES-GCM、ChaCha20-Poly1305 等

2、非对称加密

加密使用的密钥和解密使用的密钥是不相同的，分别称为：公钥、私钥，公钥和算法都是公开的，私钥是保密的。非对称加密算法性能较低，但是安全性超强，由于其加密特性，非对称加密算法能加密的数据长度也是有限的。

例如：RSA、DSA、ECDSA、 DH、ECDHE

3、哈希算法

将任意长度的信息转换为较短的固定长度的值，通常其长度要比信息小得多，且算法不可逆。

例如：MD5、SHA-1、SHA-2、SHA-256 等

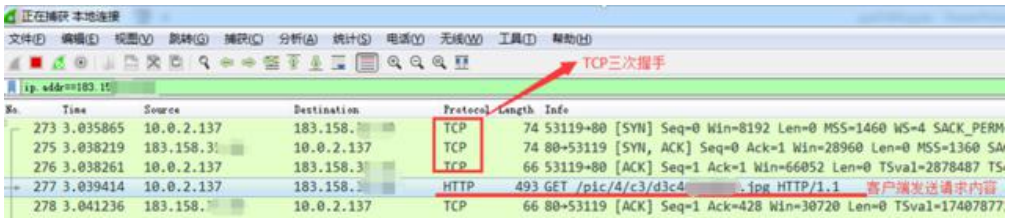
4、数字签名

签名就是在信息的后面再加上一段内容（信息经过 hash 后的值），可以证明信息没有被修改过。hash 值一般都会加密后（也就是签名）再和信息一起发送，以保证这个 hash 值不被修改。

详解

一、HTTP 访问过程

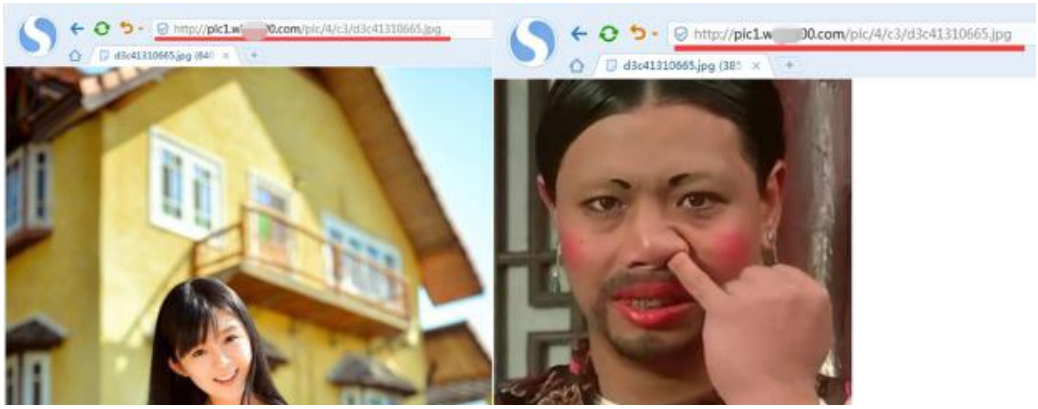
抓包如下：



如上图所示，HTTP 请求过程中，客户端与服务器之间没有任何身份确认的过程，数据全部明文传输，“裸奔”在互联网上，所以很容易遭到黑客的攻击，如下：

可以看到，客户端发出的请求很容易被黑客截获，如果此时黑客冒充服务器，则其可返回任意信息给客户端，而不被客户端察觉，所以我们经常会听到一词“劫持”，现象如下：

下面两图中，浏览器中填入的是相同的 URL，左边是正确响应，而右边则是被劫持后的响应



所以 HTTP 传输面临的风险有：

（1） 窃听风险：黑客可以获知通信内容。

（2） 篡改风险：黑客可以修改通信内容。

（3） 冒充风险：黑客可以冒充他人身份参与通信。

二、HTTP 向 HTTPS 演化的过程

第一步：为了防止上述现象的发生，人们想到一个办法：对传输的信息加密（即使黑客截获，也无法破解）

如上图所示，此种方式属于对称加密，双方拥有相同的密钥，信息得到安全传输，但此种方式的缺点是：

（1）不同的客户端、服务器数量庞大，所以双方都需要维护大量的密钥，维护成本很高

（2）因每个客户端、服务器的安全级别不同，密钥极易泄露

第二步：既然使用对称加密时，密钥维护这么繁琐，那我们就用非对称加密试试

如上图所示，客户端用公钥对请求内容加密，服务器使用私钥对内容解密，反之亦然，但上述过程也存在缺点：

（1）公钥是公开的（也就是黑客也会有公钥），所以第 ④ 步私钥加密的信息，如果被黑客截获，其可以使用公钥进行解密，获取其中的内容

第三步：非对称加密既然也有缺陷，那我们就将对称加密，非对称加密两者结合起来，取其精华、去其糟粕，发挥两者的各自的优势

如上图所示

（1）第 ③ 步时，客户端说：（咱们后续回话采用对称加密吧，这是对称加密的算法和对称密钥）这段话用公钥进行加密，然后传给服务器

（2）服务器收到信息后，用私钥解密，提取出对称加密算法和对称密钥后，服务器说：（好的）对称密钥加密

（3）后续两者之间信息的传输就可以使用对称加密的方式了

遇到的问题：

（1）客户端如何获得公钥

（2）如何确认服务器是真实的而不是黑客

第四步：获取公钥与确认服务器身份

1、获取公钥

（1）提供一个下载公钥的地址，回话前让客户端去下载。（缺点：下载地址有可能是假的；客户端每次在回话前都先去下载公钥也很麻烦）

（2）回话开始时，服务器把公钥发给客户端（缺点：黑客冒充服务器，发送给客户端假的公钥）

2、那有木有一种方式既可以安全的获取公钥，又能防止黑客冒充呢？ 那就需要用到终极武器了：SSL 证书（申购）

如上图所示，在第 ② 步时服务器发送了一个 SSL 证书给客户端，SSL 证书中包含的具体内容有：

（1）证书的发布机构 CA

（2）证书的有效期

（3）公钥

（4）证书所有者

（5）签名

……

3、客户端在接受到服务端发来的 SSL 证书时，会对证书的真伪进行校验，以浏览器为例说明如下：

（1）首先浏览器读取证书中的证书所有者、有效期等信息进行一一校验

（2）浏览器开始查找操作系统中已内置的受信任的证书发布机构 CA，与服务器发来的证书中的颁发者 CA 比对，用于校验证书是否为合法机构颁发

（3）如果找不到，浏览器就会报错，说明服务器发来的证书是不可信任的。

（4）如果找到，那么浏览器就会从操作系统中取出 颁发者 CA 的公钥，然后对服务器发来的证书里面的签名进行解密

（5）浏览器使用相同的 hash 算法计算出服务器发来的证书的 hash 值，将这个计算的 hash 值与证书中签名做对比

（6）对比结果一致，则证明服务器发来的证书合法，没有被冒充

（7）此时浏览器就可以读取证书中的公钥，用于后续加密了

4、所以通过发送 SSL 证书的形式，既解决了公钥获取问题，又解决了黑客冒充问题，一箭双雕，HTTPS 加密过程也就此形成

所以相比 HTTP，HTTPS 传输更加安全

（1） 所有信息都是加密传播，黑客无法窃听。

（2） 具有校验机制，一旦被篡改，通信双方会立刻发现。

（3） 配备身份证书，防止身份被冒充。

总结

综上所述，相比 HTTP 协议，HTTPS 协议增加了很多握手、加密解密等流程，虽然过程很复杂，但其可以保证数据传输的安全。所以在这个互联网膨胀的时代，其中隐藏着各种看不见的危机，为了保证数据的安全，维护网络稳定，建议大家多多推广 HTTPS。

HTTPS 缺点：

（1）SSL 证书费用很高，以及其在服务器上的部署、更新维护非常繁琐

（2）HTTPS 降低用户访问速度（多次握手）

（3）网站改用 HTTPS 以后，由 HTTP 跳转到 HTTPS 的方式增加了用户访问耗时（多数网站采用 302 跳转）

（4）HTTPS 涉及到的安全算法会消耗 CPU 资源，需要增加大量机器（https 访问过程需要加解密）