今天工作的时候在配置https，一时兴起，简单记录一下自己对https的理解，以及如何在web中使用，我们主要讨论几个问题

1. 为什么要用https

2. https是如何建立链接的

3. 实际使用中的问题

**1.** 为什么要用**https**而不是**http**

• 通信使用明文（ 不加密） ， 内容可能会被窃听

• 不验证通信方的身份， 因此有可能遭遇伪装

• 无法证明报文的完整性， 所以有可能已遭篡改这些问题不仅在 HTTP 上出现，其他未加密的协议中也会存在这类问题。

**2.** 如何解决**http**协议中的安全问题

我们需要采用SSL， SSL不仅提供加密处理，而且还使用了一种被称为证书的手段，可用于确定方。证书由值得信任的第三方机构颁发，用以证明服务器和客户端是实际存在的。另外，伪造证书从技术角度来说是异常困难的一件事。所以只要能够确认通信方（服务器或客户端）持有的证书，即可判断通信方的真实意图

我们可以简单理解为 **https** 就是 **http** 套上了 **ssl** 的外壳 (•̀ᴗ•́)و ̑̑

**3.** 抛开上面的概念问题，我们自己考虑下如何保证通讯的安全 **(**ง **•̀\_•́)**ง

1. 对称加密是否可行？

◦ 该方式加解密的密钥是同一个，对称加密的安全度很高

◦ 假设 A 和 B 通信，A 发出的消息经过密钥加密发送，B 收到消息后，用密钥解密

◦ 在保证密钥不泄露的情况下，这种方式是绝对OK的

◦ 但是在网络中的情况很复杂，要实现对称加密，需要客户端和服务端同时具备密钥，在密钥的保存和传递过程中，很容易出现泄露，尤其是在客户端

◦ 所以对称加密一定可行，但是不能直接采用，我们还需要改进！(ง •̀\_•́)ง

2. 如何在对称加密的基础上改进？

◦ 密钥应该是在每次发起请求时临时生成的，如果是一个固定的密钥，就始终存在着泄露的风险，SSL就采用的协商机制来解决这个问题

◦ 我们需要保证对称加密的密钥在传递过程中的安全，这里我们就需要借助非对称加密（公钥 私钥）

3. 非对称加密的作用？

◦ 非对称加密的密钥分为 公钥 和 私钥 ， （公钥加密，私钥解密），（私钥加密，公钥解密）

◦ https中，这种加密方式主要是为了规避对称加密在密钥传递过程中的安全问题

◦ 公钥是公开的，谁都可以拿到，私钥只有服务器拥有，不会传递，所以不用担心泄露

◦ 由于公钥是公开的，所以服务器加密的数据，任何持有公钥的客户端都可以解密，所以服务器 -> 浏览器的信息，约等于明文，不具备安全性

◦ 由于私钥服务器拥有，不传递，任何以公钥加密的数据，只有服务器可以解开，所以浏览器 -> 服务器的信息发送，是安全的

◦ 非对称加密具有单向安全性

◦ 所以我们可以利用这个特点，如果对称加密的密钥由客户端（浏览器）生成，再用公钥加密发送给服务器，服务器得到对称密钥后，双方再以对称加密的方式通信，就可以保证通信的安全

4. 服务器和客户端如何协商生成对称密钥？

◦ 我们已经提到，需要让对称密钥临时生成，这里我们引入一种协商机制（后面再讨论公钥篡改问题，先忽略）

◦ 建立https链接的时候，浏览器会先向服务器发送自己支持的加密算法和摘要算法，并且附带一个随机数

◦ 服务器收到请求后，返回公钥，同时也附带一个随机数

◦ 浏览器收到公钥后，再次生成随机数，利用刚才这3个随机数生成一个对称密钥，公钥加密该对称密钥，发送给服务器，之后就可以开始对称加密通信了(ง •̀\_•́)ง

◦ 每次请求都会重复以上步骤

以上就是**https**建立链接的大体步骤（当然，我们忽略了公钥篡改问题）

**4.** 如何确保公钥没有被篡改

由于非对称加密只具有单向安全性，公钥的传递，必定是不安全的，以上的https链接创建，是我们假设得到了正确的公钥，由于公钥是公开的，所以公钥本身，有被篡改的可能，那么我们如何解决这个问题呢

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，等用户通过浏览器访问服务器的时候，再也不用满世界的找某个服务器的公钥了。当他们访问某服务器的时候，这台服务器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），可以颁发证书的CA有很多（国内外都有），但是只有少数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，我们的操作系统中会预先安装好一些证书发布机构的证书。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。

证书可以证明当前传递的公钥是合法有效的，但是CA证书也是服务器传递的，CA证书里面的公钥还是有被篡改的问题，这里会涉及到签名和根证书

• 根证书：CA机构自己也有一个证书，就叫做根证书，并且CA机构有自己的私钥和公钥，CA机构颁发的证书，都用CA机构的私钥加密过

• 签名：就是对传输的内容，通过hash算法计算出一段固定长度的串，通过CA机构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”。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

• 我们的浏览器中，内置了大多数权威CA机构的根证书，当我们收到一个CA证书的时候，浏览器会找到相应CA机构的根证书，取出公钥，对CA证书解密，并比对数字签名

以上方式就保证了证书本身的安全性

**5.** 总结一下**https**完整的建立连接步骤

1. TCP三次握手

2. Client Hello – 客户端发送所支持的 SSL/TLS 最高协议版本号、所支持的加密算法及压缩方法和随机数A等信息给服务器端。

3. Server Hello – 服务器端收到客户端信息后，选定双方都能够支持的 SSL/TLS 协议版本和加密方法及压缩方法、 随机数B和服务器证书返回给客户端。

4. 客户端根据证书类型，找到内置的根证书，对证书进行解密，并验证数字签名有效性，如果是非法证书，拒绝连接，并在页面提醒，如果你选择信任该证书，依然可以继续

5. 客户端主动再生成一个随机数C，开始生成秘钥，这个生成秘钥的算法是客户端跟服务器端共享的，因为之前协商的时候已经确定了算法了，生成秘钥后就可以加密一段内容，试着跟服务区通信了，这个内容是经过先散列，散列后将原内容和散列集一起用刚才的密钥加密；接着用服务器端证书中的公钥对随机数C加密。

6. 然后把加密过的内容和加密好的随机数一起发向服务器端。

7. 服务器用私钥解密得到随机数C，这样服务器端也同时拥有了随机数A、B、C，即刻生成密钥，再用密钥对加密的内容进行解密，然后解开后对其中的明文内容进行散列，与客户端发过来的散列值进行比较，如果相等，说明就是客户端发过来的，通信成功。

8. 用步骤3中同样的方式发一段加密过的内容给客户端。

9. 用步骤5一样的方式对服务器发来的内容进行验证。

10. 客户端确定开始通信。

11. 服务端确定开始通信。

**6. https**使用中的问题

1. 只有注册登录页，才需要HTTPS？

◦ 当然不是，如果仅在登录页使用了HTTPS，但是登录以后，其他页面就变成了HTTP。这时，我们的cookie里的session值就暴露了。

2. HTTPS太慢？

◦ 这个在工作中试过，由于一直都在用Https，和http相比，使用上没有速度快慢的感觉，https建立连接的开销，对于目前的硬件性能，不值一提

3. 免费的证书哪里搞？

◦ <https://www.startcomca.com/> 好东西，不多说

作者：感觉被掏空

链接：http://www.jianshu.com/p/dc6009efdab0

來源：简书

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。