25 | 固若金汤的根本（下）：数字签名与证书

2019-07-24 Chrono

《透视HTTP协议》 课程介绍 >

讲述：Chrono

时长 10:58 大小 15.09M

上一讲中我们学习了对称加密和非对称加密，以及两者结合起来的混合加密，实现了机密性。

但仅有机密性，离安全还差的很远。

黑客虽然拿不到会话密钥，无法破解密文，但可以通过窃听收集到足够多的密文，再尝试着修改、重组后发给网站。因为没有完整性保证，服务器只能“照单全收”，然后他就可以通过服务器的响应获取进一步的线索，最终就会破解出明文。

另外，黑客也可以伪造身份发布公钥。如果你拿到了假的公钥，混合加密就完全失效了。你以为自己是在和“某宝”通信，实际上网线的另一端却是黑客，银行卡号、密码等敏感信息就在“安全”的通信过程中被窃取了。

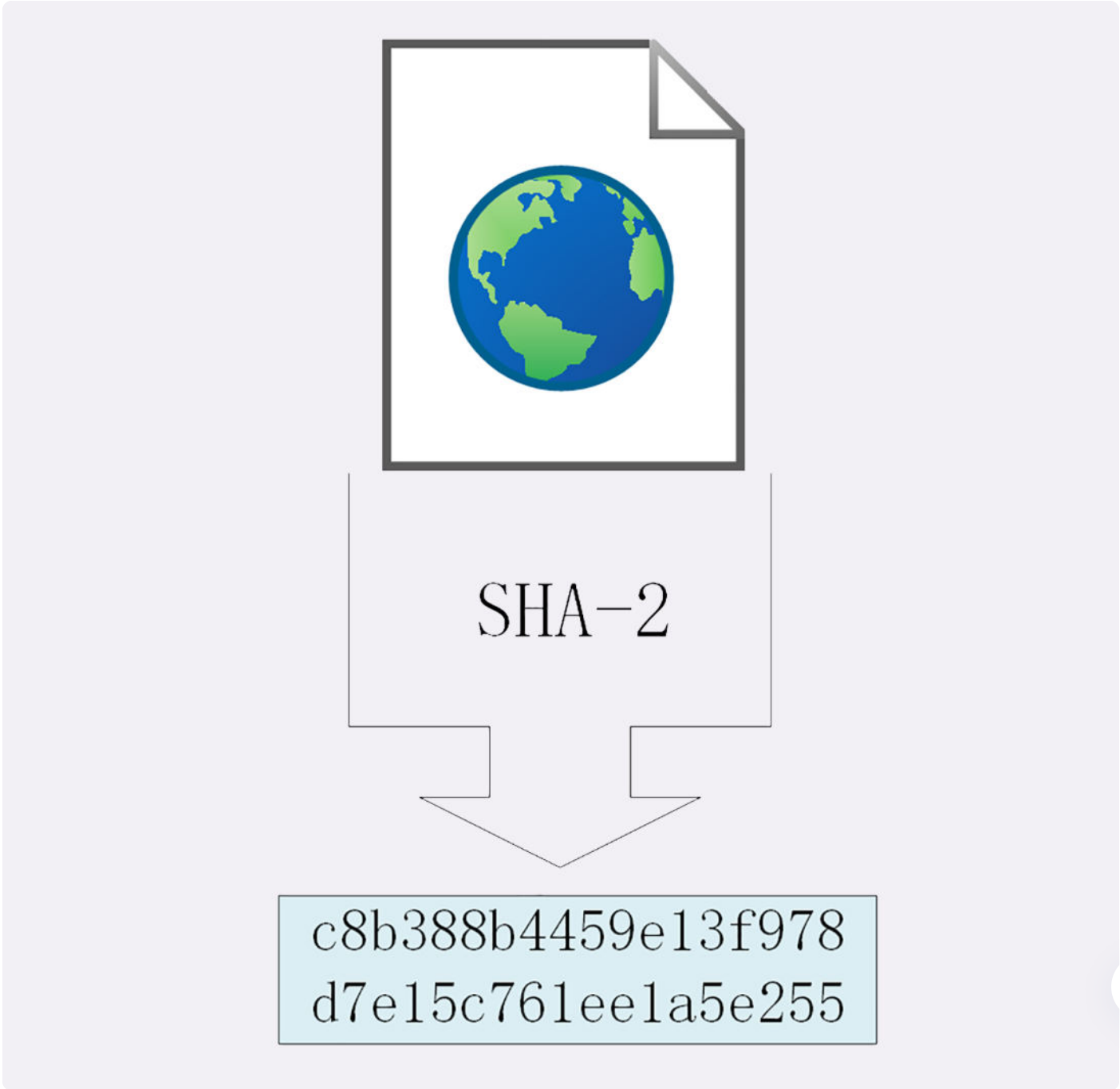
所以，在机密性的基础上还必须加上完整性、身份认证等特性，才能实现真正的安全。

摘要算法

实现完整性的手段主要是**摘要算法**（Digest Algorithm），也就是常说的散列函数、哈希函数（Hash Function）。

你可以把摘要算法近似地理解成一种特殊的压缩算法，它能够把任意长度的数据“压缩”成固定长度、而且独一无二的“摘要”字符串，就好像是给这段数据生成了一个数字“指纹”。

换一个角度，也可以把摘要算法理解成特殊的“单向”加密算法，它只有算法，没有密钥，加密后的数据无法解密，不能从摘要逆推出原文。



摘要算法实际上是把数据从一个“大空间”映射到了“小空间”，所以就存在“冲突”（collision，也叫碰撞）的可能性，就如同现实中的指纹一样，可能会有两份不同的原文对应相同的摘要。好的摘要算法必须能够“抵抗冲突”，让这种可能性尽量地小。

因为摘要算法对输入具有“单向性”和“雪崩效应”，输入的微小不同会导致输出的剧烈变化，所以也被 TLS 用来生成伪随机数（PRF，pseudo random function）。

你一定在日常工作中听过、或者用过 MD5（Message-Digest 5）、SHA-1（Secure Hash Algorithm 1），它们就是最常用的两个摘要算法，能够生成 16 字节和 20 字节长度的数字摘要。但这两个算法的安全强度比较低，不够安全，在 TLS 里已经被禁止使用了。

目前 TLS 推荐使用的是 SHA-1 的后继者：SHA-2。

SHA-2 实际上是一系列摘要算法的统称，总共有 6 种，常用的有 SHA224、SHA256、SHA384，分别能够生成 28 字节、32 字节、48 字节的摘要。

你可以用实验环境的 URI“/25-1”来测试一下 TLS 里的各种摘要算法，包括 MD5、SHA-1 和 SHA-2。

```
https://www.chrono.com/25-1?algo=md5
https://www.chrono.com/25-1?algo=sha1
https://www.chrono.com/25-1?algo=sha256
```

完整性

摘要算法保证了“数字摘要”和原文是完全等价的。所以，我们只要在原文后附上它的摘要，就能够保证数据的完整性。

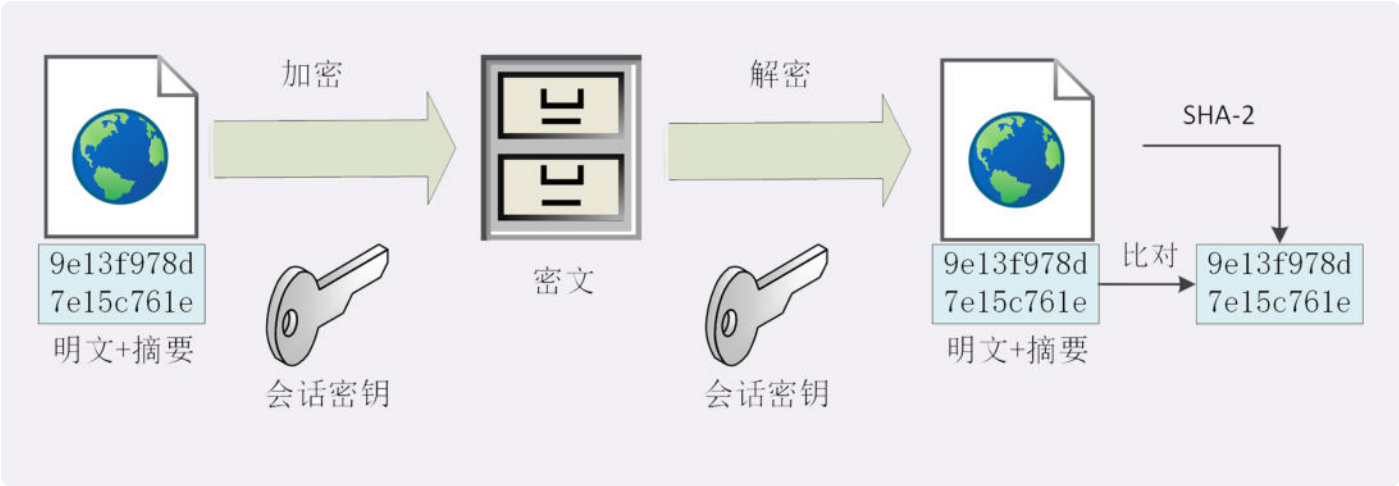
比如，你发了条消息：“转账 1000 元”，然后再加上一个 SHA-2 的摘要。网站收到后也计算一下消息的摘要，把这两份“指纹”做个对比，如果一致，就说明消息是完整可信的，没有被修改。

如果黑客在中间哪怕改动了一个标点符号，摘要也会完全不同，网站计算比对就会发现消息被窜改，是不可信的。

不过摘要算法不具有机密性，如果明文传输，那么黑客可以修改消息后把摘要也一起改了，网站还是鉴别不出完整性。

所以，真正的完整性必须要建立在机密性之上，在混合加密系统里用会话密钥加密消息和摘要，这样黑客无法得知明文，也就没有办法动手脚了。

这有个术语，叫哈希消息认证码（HMAC）。



数字签名

加密算法结合摘要算法，我们的通信过程可以说是比较安全了。但这里还有漏洞，就是通信的两个端点（endpoint）。

就像一开始所说的，黑客可以伪装成网站来窃取信息。而反过来，他也可以伪装成你，向网站发送支付、转账等消息，网站没有办法确认你的身份，钱可能就这么被偷走了。

现实生活中，解决身份认证的手段是签名和印章，只要在纸上写下签名或者盖个章，就能够证明这份文件确实是由本人而不是其他人发出的。

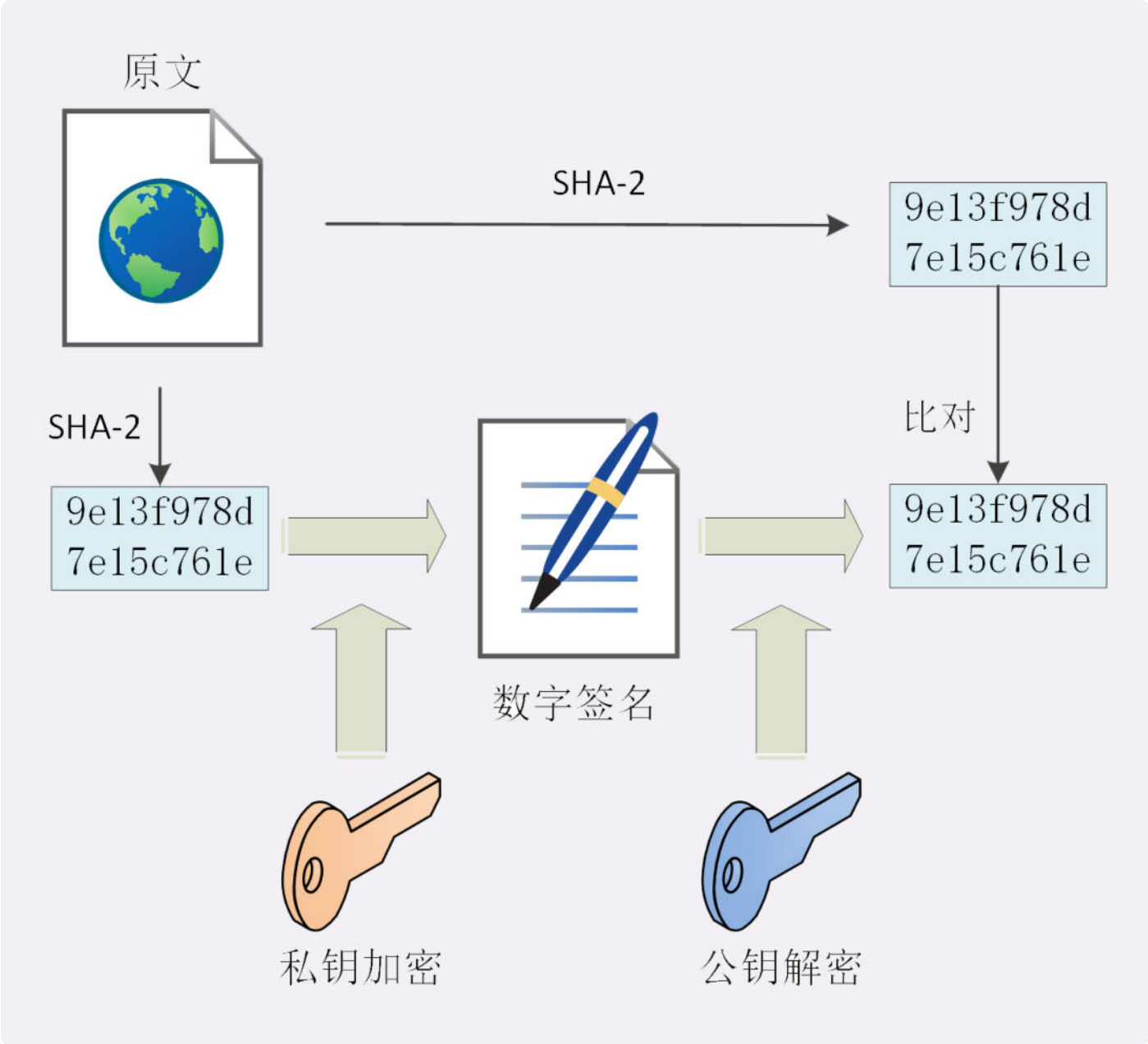
你回想一下之前的课程，在 TLS 里有什么东西和现实中的签名、印章很像，只能由本人持有，而其他任何人都不会有呢？只要用这个东西，就能够在数字世界里证明你的身份。

没错，这个东西就是非对称加密里的“**私钥**”，使用私钥再加上摘要算法，就能够实现“**数字签名**”，同时实现“身份认证”和“不可否认”。

数字签名的原理其实很简单，就是把公钥私钥的用法反过来，之前是公钥加密、私钥解密，现在是私钥加密、公钥解密。

但又因为非对称加密效率太低，所以私钥只加密原文的摘要，这样运算量就小的多，而且得到的数字签名也很小，方便保管和传输。

签名和公钥一样完全公开，任何人都可以获取。但这个签名只有用私钥对应的公钥才能解开，拿到摘要后，再比对原文验证完整性，就可以像签署文件一样证明消息确实是你发的。



刚才的这两个行为也有专用术语，叫做“**签名**”和“**验签**”。

只要你和网站互相交换公钥，就可以用“签名”和“验签”来确认消息的真实性，因为私钥保密，黑客不能伪造签名，就能够保证通信双方的身份。

比如，你用自己的私钥签名一个消息“我是小明”。网站收到后用你的公钥验签，确认身份没问题，于是也用它的私钥签名消息“我是某宝”。你收到后再用它的公钥验一下，也没问题，这样你和网站就都知道对方不是假冒的，后面就可以用混合加密进行安全通信了。

实验环境的 URI“/25-2”演示了 TLS 里的数字签名，它使用的是 RSA1024。

数字证书和 CA

到现在，综合使用对称加密、非对称加密和摘要算法，我们已经实现了安全的四大特性，是不是已经完美了呢?

不是的，这里还有一个“**公钥的信任**”问题。因为谁都可以发布公钥，我们还缺少防止黑客伪造公钥的手段，也就是说，怎么来判断这个公钥就是你或者某宝的公钥呢?

真是“按下葫芦又起了瓢”，安全还真是个麻烦事啊，“一环套一环”的。

我们可以用类似密钥交换的方法来解决公钥认证问题，用别的私钥来给公钥签名，显然，这又会陷入“无穷递归”。

但这次实在是“没招”了，要终结这个“死循环”，就必须引入“外力”，找一个公认的可信第三方，让它作为“信任的起点，递归的终点”，构建起公钥的信任链。

这个“第三方”就是我们常说的 **CA**（Certificate Authority，证书认证机构）。它就像网络世界里的公安局、教育部、公证中心，具有极高的可信度，由它来给各个公钥签名，用自身的信誉来保证公钥无法伪造，是可信的。

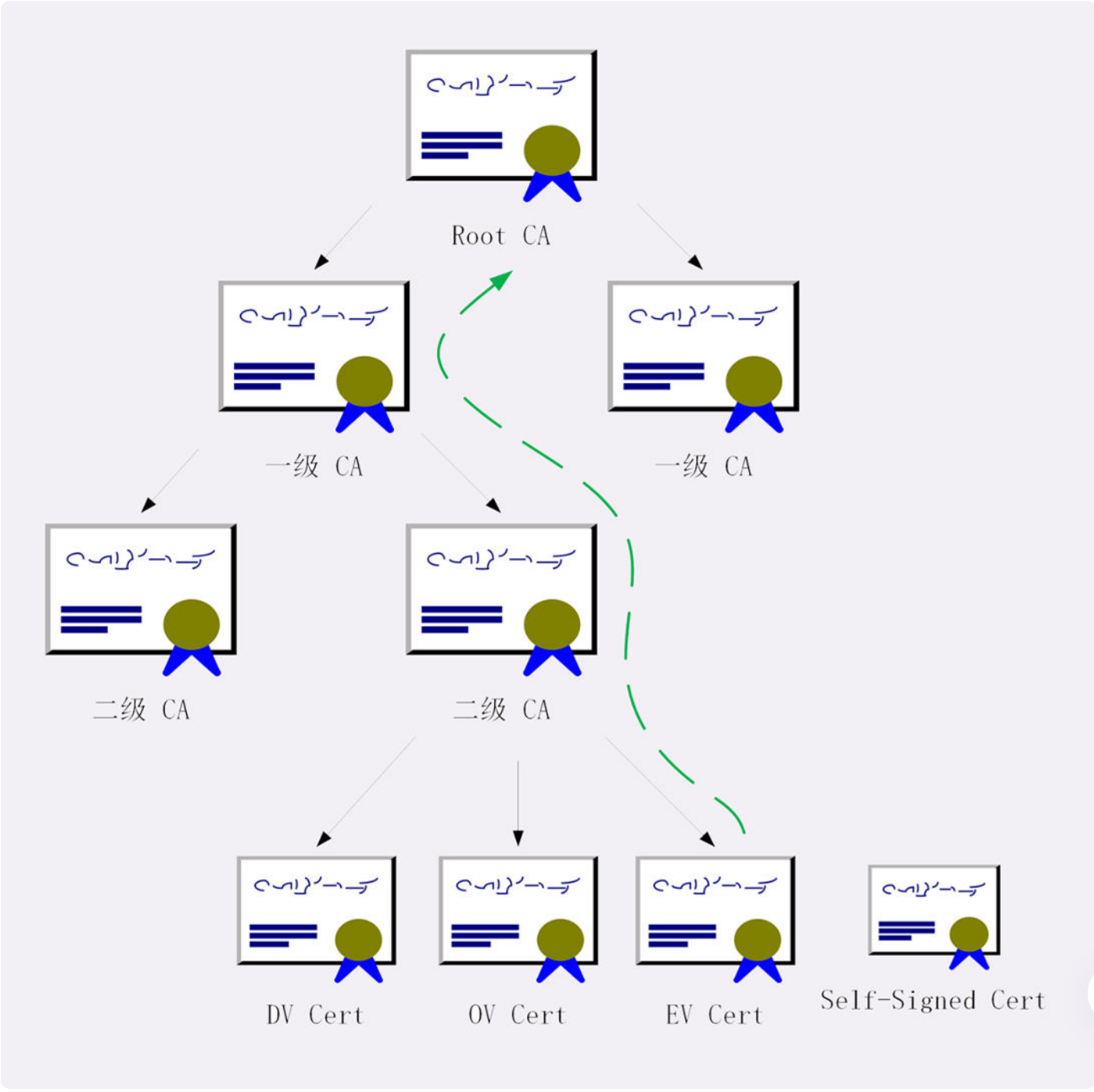
CA 对公钥的签名认证也是有格式的，不是简单地把公钥绑定在持有者身份上就完事了，还要包含序列号、用途、颁发者、有效时间等等，把这些打成一个包再签名，完整地证明公钥关联的各种信息，形成“**数字证书**”（Certificate）。

知名的 CA 全世界就那么几家，比如 DigiCert、VeriSign、Entrust、Let’s Encrypt 等，它们签发的证书分 DV、OV、EV 三种，区别在于可信程度。

DV 是最低的，只是域名级别的可信，背后是谁不知道。EV 是最高的，经过了法律和审计的严格核查，可以证明网站拥有者的身份（在浏览器地址栏会显示出公司的名字，例如 Apple、GitHub 的网站）。

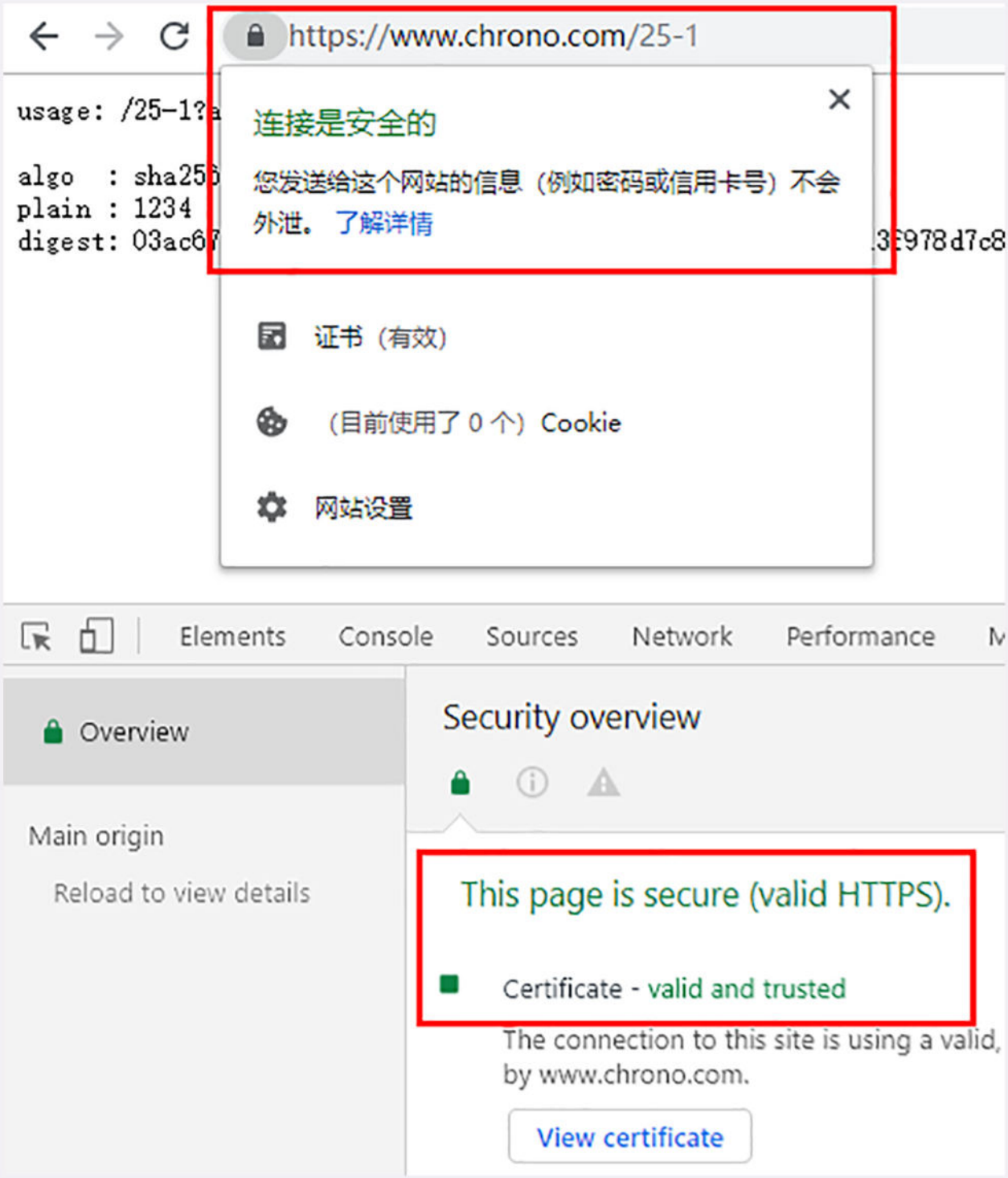
不过，CA 怎么证明自己呢?

这还是信任链的问题。小一点的 CA 可以让大 CA 签名认证，但链条的最后，也就是 **Root CA**，就只能自己证明自己了，这个就叫“**自签名证书**”（Self-Signed Certificate）或者“**根证书**”（Root Certificate）。你必须相信，否则整个证书信任链就走不下去了。



有了这个证书体系，操作系统和浏览器都内置了各大 CA 的根证书，上网的时候只要服务器发过来它的证书，就可以验证证书里的签名，顺着证书链（Certificate Chain）一层层地验证，直到找到根证书，就能够确定证书是可信的，从而里面的公钥也是可信的。

我们的实验环境里使用的证书是“野路子”的自签名证书（在 Linux 上用 OpenSSL 命令行签发），肯定是不会被浏览器所信任的，所以用 Chrome 访问时就会显示成红色，标记为不安全。但你只要把它安装进系统的根证书存储区里，让它作为信任链的根，就不会再有危险警告。



证书体系的弱点

证书体系（PKI，Public Key Infrastructure）虽然是目前整个网络世界的安全基础设施，但绝对的安全是不存在的，它也有弱点，还是关键的“**信任**”二字。

如果 CA 失误或者被欺骗，签发了错误的证书，虽然证书是真的，可它代表的网站却是假的。

还有一种更危险的情况，CA 被黑客攻陷，或者 CA 有恶意，因为它（即根证书）是信任的源头，整个信任链里的所有证书也就都不可信了。

这两种事情并不是“耸人听闻”，都曾经实际出现过。所以，需要再给证书体系打上一些补丁。

针对第一种，开发出了 CRL（证书吊销列表，Certificate revocation list）和 OCSP（在线证书状态协议，Online Certificate Status Protocol），及时废止有问题的证书。

对于第二种，因为涉及的证书太多，就只能操作系统或者浏览器从根上“下狠手”了，撤销对 CA 的信任，列入“黑名单”，这样它颁发的所有证书就都会被认为是不安全的。

小结

今天我们学习了数字签名和证书、CA，是不是有种“盗梦空间”一层套一层的感觉？你可以在课后再去各大网站，结合它们“小锁头”里的信息来加深理解。

今天的内容可以简单概括为四点：

1. 摘要算法用来实现完整性，能够为数据生成独一无二的“指纹”，常用的算法是 SHA-2；
2. 数字签名是私钥对摘要的加密，可以由公钥解密后验证，实现身份认证和不可否认；
3. 公钥的分发需要使用数字证书，必须由 CA 的信任链来验证，否则就是不可信的；
4. 作为信任链的源头 CA 有时也会不可信，解决办法有 CRL、OCSP，还有终止信任。

课下作业

1. 为什么公钥能够建立信任链，用对称加密算法里的对称密钥行不行呢？
2. 假设有一个三级的证书体系（Root CA=> 一级 CA=> 二级 CA），你能详细解释一下证书信任链的验证过程吗？

欢迎你把自己的学习体会写在留言区，与我和其他同学一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。

课外小贴士

01 摘要算法除了用于 TLS 安全通信，还有很多其他的用途，比如散列表、数据校验、大文件比较等。

02 虽然 SHA-2 很安全，但出于“未雨绸缪”的考虑，又出现了 SHA-3，它也有 6 种算法，名字与 SHA-2 差不多，比如 SHA3-224、SHA3-256，目前还未纳入 TLS。

03 “账号 + 密码”也能够实现简单的的身份认证，但在安全通信未建立前使用很容易就会被窃取，所以在 TLS 里不能用。

04 Let's Encrypt 是著名的免费 CA，它只颁发 DV 证书，而且出于安全目的有效期只有 90 天，但可以用 Certbot 工具自动续订。

05 证书的格式遵循 X509 v3 标准，有两种编码方式，一种是二进制的 DER，另一种是 ASCII 码的 PEM，实验环境使用的是 PEM。

分享给需要的人，Ta订阅超级会员，你将得 50 元

Ta单独购买本课程，你将得 20 元

生成海报并分享

赞 17　提建议

上一篇　24 | 固若金汤的根本（上）：对称加密与非对称加密

下一篇　26 | 信任始于握手：TLS1.2连接过程解析

精选留言 (100)

写留言

Geek_steven_wang

2019-08-24

保密性：靠混合加密解决，非对称加密实现对称加密秘钥传递，对称加密实现内容加密。

完整性：靠摘要算法解决。
身份认证：靠数字证书解决，数字证书因为CA机构的信任变成一个完整信任链条，从而实现通过数字证书证明了对方真实身份，但注意身份真实也可能是挂羊头卖狗肉，是一个坏人，所以，有了CRL、OCSP，还有终止信任。
不可否认：靠数字签名解决，内容摘要算法得到摘要，私钥加密摘要，对方使用对应公钥解密，得到摘要，再和自己得到的服务器提供的原文摘要对比，一致说明这个内容就是原服务器提供的，由证书说明了服务器的身份。

关于证书验证：
服务器返回的是证书链（不包括根证书，根证书预置在浏览器中），然后浏览器就可以使用信任的根证书（根公钥）解析证书链的根证书得到一级证书的公钥+摘要验签，然后拿一级证书的公钥解密一级证书拿到二级证书的公钥和摘要验签，再然后拿二级证书的公钥解密二级证书得到服务器的公钥和摘要验签，验证过程就结束了。

作者回复: 说的非常好。

共 5 条评论> 82

放开那个猴子
2019-07-24

看完老师的文章有点迷惑，主要是没搞清完整的流程，又去找资料看了一下，说下自己的理解，老师看看对不。

数字签名和数字证书只用于TSL/SSL的握手阶段，主要是保证服务器的公钥能够正确地传给浏览器（不被中间人伪装发送假的公钥）

具体流程大概是：
1、服务器去CA机构申请证书，证书中包含了要发给客户端的公钥、签发者、到期时间等等信息。如果这样简单地把证书发给浏览器，中间人可以轻松地修改成自己的公钥，之后的通信就是不安全的了。于是需要一定的加密手段，这里的做法就是使用数字签名：将证书的信息利用摘要算法计算出摘要之后，用CA的秘钥进行加密，生成数字签名。
2、服务器将数字证书和数字签名一起发给浏览器，因为有数字签名，所以数字证书无法被中间人做修改（修改之后生成的数字签名无法被CA公钥解密），浏览器拿到数字证书之后，去本地的信任机构中查询到对应的机构，利用其公钥解密数字签名，验证证书是否有被修改过。这一步就保证了浏览器获取到的公钥一定是正确的。
3、公钥正确地传给浏览器之后，接着就是协商对称加密的密钥，然后通信等等..

参考：
http://www.ruanyifeng.com/blog/2011/08/what_is_a_digital_signature.html
https://www.zhihu.com/question/52493697

作者回复: 态度很认真，值得表扬。

有一点小错误，数字签名的防窜改不是因为“修改之后生成的数字签名无法被CA公钥解密”，而是修改后的摘要变动了，与签名里解密出的原始摘要不匹配，所以能够发现原文被窜改。

另外，你说的这些是目前流行的PKI体系，但公钥私钥本身并不一定要用证书，它们本身属于密码学。

共 2 条评论>

25

Leon📷

2019-07-24

重放和篡改的问题没有提，黑客是解不开秘文，但是可以重复发送，需要时间戳和随机数再合起来做一个不可逆的签名，服务端收到重复的就丢弃

作者回复: 感谢补充，这个就是nonce了。

共 2 条评论>

24

郭凯强

2019-07-24

问题1. 非加密算法需要公开公钥从而让客户端能解密。如果用对称加密，加密秘钥公开，就达不到加密效果了
问题2. 客户端发现当前网站的证书是二级CA，在可信任签发机构中找不到，就会去拿二级CA的数字证书的签发机构去做检查，发现它是一级CA，也不在可信任签发机构中，再找一级CA的数字证书的签发机构，发现是受信任的ROOT CA，至此完成验证。如果到最后一层CA都不受信任，就会警告用户

作者回复: √

共 2 条评论>

21

好好好

2020-04-29

看了几遍大概了解为什么要这样加密的过程了
↓ 对称加密（有密钥交换的问题）
↓ 非对称加密（基于复杂的数学难题，运行速度很慢）
↓ 混合加密（怎么保证完整性？不被修改？）
↓ 摘要算法（无法保证是用户自己）

↓ 数字签名（公钥怎么保证安全正确的？）
↓ 数字证书、CA

作者回复: 总结的非常好！

17

蓝配鸡
2019-10-30

为什么公钥能够建立信任链，用对称加密算法里的对称密钥行不行呢?
所谓建立信任链， 是指发送方相信公钥确实是接收方的。因为有CA的信任链和最后根CA的背书。
用对称密钥行不行呢?
用对称密钥来确认公钥的真实性，就好像是街头对暗号。 A:枯藤老树昏鸭 B:穿条秋裤回家。这样做确实可以确认“你就是你”，可问题是如何交换密钥呢？ 问题就又绕回到了非对称密钥了。

假设有一个三级的证书体系（Root CA=> 一级 CA=> 二级 CA），你能详细解释一下证书信任链的验证过程吗?
没有具体实战过，我猜测如下：
二级CA交给了浏览器，CA说：“我是某宝，这是我的公钥，这个一级CA给我背书了， 你要相信我！”
浏览器再去确认这个一级CA可不可信，一级CA说：“我是公安局，这是我的公钥，这个根CA给我背书了， 你要相信我！”
浏览器再去确认这个根CA可不可信，根CA说：“我是上帝我说了算，你爱信不信”。浏览器也很无奈啊。。。只能信了。

作者回复: 学习进度很快啊，回答的也很形象生动。

16

彩色的沙漠
2019-07-24

对于第二个问题证书链验证的过程，有些不理解的地方，请老师指教，您在文章说“操作系统和浏览器都内置了各大 CA 的根证书，上网的时候只要服务器发过来它的证书，就可以验证证书里的签名，顺着证书链（Certificate Chain）一层层地验，直到找到根证书”，服务器只返回了他的证书（假如返回的是二级证书），浏览器内置的是根证书（根公钥）使用根公钥只能解密根机构签名的证书，无法解密二级证书，使用一级证书（公钥）才能解密二级证。那么浏览器是怎么自下向上层层解析到根证书？我的理解的是服务器返回的是证书链，然后浏览器就可以使用信任的根证书（根公钥）解析证书链的根证书得到一级证书的公钥+摘要验签，然后

拿一级证书的公钥解密一级证书拿到二级证书的公钥和摘要验签，再然后拿二级证书的公钥解密二级证书得到服务器的公钥和摘要验签，验证过程就结束了。谢谢！

作者回复: 你理解的很对，服务器会在握手的时候返回整个证书链，但通常为了节约数据量，不会包含最终的根证书，因为根证书通常已经在浏览器或者操作系统里内置了。

共 3 条评论> 12

乘风破浪
2021-02-18

假设有一个三级的证书体系（Root CA=> 一级 CA=> 二级 CA），你能详细解释一下证书信任链的验证过程吗？---注服务器的证书由二级CA签发。
修订的第三版，这个解答相对更完善。。。
TLS协商阶段，在交换完Client Hello/Server Hello消息后，发送方【通常是服务器】，发送Certificate消息，把证书链，包括自己的证书，二级CA证书，一级CA证书，一次性发送给接收方【通常是浏览器】。

注：每个传递过来的证书包括4部分
signedCertificate签名的证书，即浏览器点击小锁头直观可以看到的证书
algorithmIdentifier算法标记，包括了签名证书用到的摘要和签名算法，如sha256WithRSAEncryption
Padding填充字符
encrpted加密摘要，注：加密摘要不包含在signedCertificate中，所以浏览器中点击小锁头看不到加密摘要。

当前接收方只有内置的Root Ca根证书，无法直接信任接收方的证书。接收方将通过证书链中包含的签发者信息，逐层向上查找直到Root Ca根证书，并从根证书开始，逐级向下做验签。首先，用根证书对一级证书做验签。具体过程是，对一级CA证书【signedCertificate】用传递过来的摘要算法【algorithmIdentifier】做摘要得到摘要1；用Root Ca根证书的公钥解密一级CA证书的数字签名【encrpted】，得到发送过来的摘要2，二者比较，如一致，则认为一级CA证书是真实有效的。类似的，继续用一级CA证书对二级CA证书做验签，二级CA证书对发送方证书做验签，如果发送方证书验证通过，则随之TLS协商进入Server key exchange阶段。

作者回复: 写的很详细，nice。

共 2 条评论> 7

极客时间
2019-07-24

到这里完全爆炸了

作者回复: 哪里不明白可以随时问。

共 3 条评论> 7

the sword the god

2020-09-10

HTTPS四大特性机密性、完整性、身份认证、不可否认，我不是能够很好地说服我自己：
如何实现机密性?
对称加密加密解密都用同一个秘钥，但是问题是这个秘钥的交换传输问题，我们无法保证在传输过程中有没有人修改我们的秘钥。
非对称加密分为公钥和私钥，我们可以随便散播我们的公钥，用私钥加密，但是问题是基于复杂的数学难题，所以速度比较慢。
所以我们可以用混合加密，用非对称加密的私钥加密对称加密的秘钥（即会话秘钥），这样客户端可以用服务端散播的公钥解密得到会话秘钥，以后传输过程中就用这个会话秘钥来传输。
但是这个时候，我们依旧无法保证我们拿到的会话秘钥是完整的，未经篡改的，黑客完全可以劫持我们传递私钥的这个报文，乱改一通，所以我们需要进一步实现完整性。
如何实现完整性?
我们用摘要算法对原文(这里也就是会话秘钥)生成摘要指纹，然后用私钥加密（会话秘钥+秘钥的摘要指纹）发出去，此时客户端收到报文后，用公钥解密报文，理想情况下，还是原文+摘要的形式，这时我们用相同的摘要算法去计算原文，进行摘要对比，只要相同，那就可以保证完整性了。如果黑客在中间修改了报文，那么按照摘要算法的雪崩效应，最终是可以确定原文被修改，或者干脆就不符合格式，也就不安全了。
然后又出现了两端安全的问题？就是如何确定服务器就是那个服务器。
因为非对称加密的私钥是独有的，可以以此作为基准。我们用私钥把我们之前得到的原文（会话秘钥）的摘要再进行加密，秘钥的加密只能用公钥解开，所以客户端收到内容后，先用公钥解开当前的内容，得到的东西是原文+秘钥加密的摘要，我们再用摘要算法去算原文的摘要，用公钥再解密出服务器给我们的摘要，两相对比，如果相同，就验证了两个端点安全且内容完整未经篡改。
但是我这里有一个疑惑，这一步到底意义是什么呢？我觉得这样做只能说明公钥和私钥的对应性，充其量只能证明确实是这两个端点在通信，中间没有被篡改。那么这相当于上一步到底多出来的实际意义是什么呢？我们就算不用私钥去加密原文的摘要，实际上客户端收到报文后既然能用公钥解密出原文和摘要，并直接进行认证，也能证明两个端点公钥和私钥的一一对应吧？我甚至觉得这一步是多余的。这里希望老师解惑。
其实最关键的还是证明公钥的来源是正确的，举个[illegible]，也就是你必须确认你拿到的是淘宝的公钥。
这里我假设上面进行身份认证的签名和验签是能够说服我的，好，解决公钥来源正确的方法是引入CA。
那么CA到底是什么呢？我感觉有点理解困难。按照之前的说法，签名是指对某个摘要使用私钥进行加密的过程，验签则是用与这个私钥对应的公钥解密出摘要并进行摘要对比的过程。
那么CA对什么进行签名呢？我们把公匙、有效时间、序列号等相关信息交给CA，CA会用它

的绝对保密的私钥对这些信息进行打包后加密，即签名（这里是不是就是用摘要算法做了一个指纹然后用私钥加密原文信息+摘要？？），形成淘宝独有的数字证书。
好，那我第二个问题来了，CA的私钥既然是绝对保密的，那我就把CA的私钥理解成是在服务器外的一个地方，请问我们如何把我们的公钥、序列号这些信息安全地送到CA处，并且安全地收回来？
还有，数字证书是相当于被送回到了服务器了吗？也就是说，之前我们都是散播我们的公钥，现在变成了我们四处散播我们的数字证书？
我们每次登陆网站，比如登陆淘宝网，是不是相当于我们拿到了这个网站发给我们的数字证书，然后我们用内置保存在操作系统和浏览器中的CA的公钥去解密数字证书，还原成（公匙+序列号等）原文信息+摘要，然后要摘要算法计算原文信息进行比对，只要对上了，我们就能根据序列号一些信息，保证我们访问的就是淘宝网？
然后这个CA的证书体系树到底是什么？是不是其实CA机构散播的也是他们的数字证书，也就是说我们保存在操作系统和浏览器中的也是他们的数字证书？我们证明淘宝网的数字证书需要上一层CA的数字证书，但是CA的数字证书也需要证明，所以需要更上一层的CA的数字证书，所以往上找回找到根证书，我们必须相信，不然信任链走不下去了。
那么我有问题，上面文章中讲了，我们浏览器和操作系统只内置了各大CA的根证书，那么如果淘宝网的数字证书是由一个三级证书认证的，那请问这个三级证书那里来？我们去哪里找？？
好了，假设上面的信任链走通了，我们确定了我们拿到的就是淘宝网的公钥，我们解密出了原文，在现在的分析中，也就是对称加密的会话秘钥，也验证了完整性，我们要开始用这个会话秘钥开始通信了，那么我想知道现在用会话秘钥加密传输这个过程中就没有完整性、身份认证、不可否认的问题了吗？？

作者回复: 可以看出学习的非常认真，态度端正，先点赞。

学习密码学有困惑，很正常，很多步骤都感觉是绕来绕去的，我当初也是这么觉得。但你要知道，ssl/tls的制定者都是密码学的顶级专家，而且经过了很多年的考验，几乎可以肯定地说是没有低级漏洞的，所以我们可以不用怀疑算法，只需要去理解它。

1.私钥签名、公钥验签，就是用公钥验证了私钥持有者的身份，实现了身份认证和不可否认。你说的“多余”其实在ssl/tls里也是只做一次，确实做多了没有意义。

2.公钥是公开的，所以不存在安全问题，不用担心被破坏窜改，因为一旦改了就和私钥不对应了。

3.证书里含有公钥和其他信息，也是公开的，由ca签名保证完整性，窜改即失效，所以可以随便发布。

4.证书把网站的身份和公钥绑定在一起，所以验证了公钥对应的私钥，也就相当于验证了网站。用的不是证书序列号。

5.一些大ca的根证书都是内置在操作系统和浏览器的，网站也可以把整个证书链（根->一级->二级-

>网站）在握手的时候全发给客户端，客户端拿全了验证。

6.ssl/tls握手成功后，就建立了一个加密的安全信道，之前已经认证过了身份，所以通信是安全的。而且会话时用的AES-GCM等算法也有完整性保证。

共 2 条评论>

6

钱
2020-04-01

小结如下：
1：摘要算法——具有单向性和雪崩性，单向性强调只能加密不能解码，雪崩性强调明文稍微变化一丁点那摘要就会产生巨大的变化。摘要算法怎么使用，把明文和摘要一起发送过去就可以验证明文是否被篡改了。但是防止不了明文修改后把摘要也篡改的情况，所以，后面有了数字签名。

2：数字签名——私钥加密＋摘要算法，用私钥加密摘要然后用公钥验证，就可以实现身份认证以及不可否认的特性。

3：CA——也即证书认证机制防止黑客伪造公钥，如果用的是黑客伪造的公钥，那自然黑客就能用其私有加密摘要伪造某宝的身份了。所以，公钥只能有可信度比较高的三方来发布，防止黑客伪造。

4：后面也存在一些问题，比如：颁发证书的机构被攻陷或者证书颁布有误，这时就出现了CRL/OCSP/终止信任。做到万无一失的安全确实非常困难，涉及机密或金钱的也没得选吧！只能这么搞了！

作者回复: great!

5

爱看书的蜗牛
2020-12-02

既然签名是公开的，公钥也是公开的，那黑客只要先伪装成网站获得我的公钥，再拿到我的签名，不就可以伪装成我了吗?

作者回复: 这里不对，只有私钥才是身份认证的关键，私钥严格保密，其他人不能获取。所以黑客拿到公钥没有用，因为他做不出签名，只能验证签名，而签名只能是持有私钥的人才能发出的。

这里面有点绕，也是非对称加密的关键，需要再仔细理解一下。

3

锦

2019-07-24

使用非对称加密算法RSA交换对称算法AES的密钥部分有疑问：

1，这个交换动作是谁发起的？

2，需要双方都要确认吗？

3，其中非对称加密算法RSA的私钥保存部分没看懂，网站的私钥保存在服务器端，安全；那么用户的私钥呢？是保存在ca证书颁发机构吗？还是保存在浏览器端？

另外，如果网站用户使用黑客手段修改客户端内存数据，然后提交服务器，这种情况https有招吗？

作者回复: 1/2可以看接下来的tls握手，如何交换如何确认就要使用一种双方都认可的协议。

3.私钥需要自己保管，方法有很多，比如u盾（特殊的usb设备），或者直接就是一个文本文件，想怎么存就怎么存。

4.https只保证通信链路的安全，在这之外它是无能为力的。

3

rongyefeng

2020-06-02

老师，数字签名中使用私钥加密摘要，HMAC中用会话秘钥加密摘要，到底使用哪一个加密摘要？

作者回复: 用途不同，数字签名用在握手阶段，hmac用在会话通信阶段。

2

Joker

2020-04-14

感觉，这些安全的特性是需要根据一些实际的攻击手段结合起来，才能说的明白。

比如完整性这个，为的保证内容没有被篡改，这个是建立在中间人手段的基础上的，因为黑客把你的内容篡改了，然后再通过服务器的响应获取数据，通过不断的测试，计算，从而推断出网站的私钥，导致你的信息泄漏。增加完整性的验证之后，服务器一旦知道这些内容被篡改，那么就不进行响应，中间人没有获取数据，推断也就无从谈起。

至于“验签”，比如黑客通过某种手段知道了你已经通过了服务器的身份验证，然后再根据抓包制知道了服务器的消息格式，然后突然用中间人攻击，模拟了你这个客户端，直接向服务器发送转账操作，那这个时候要咋办？此时，公钥是已知的，至于消息，模拟的十分逼真，服务

器无法判断这是你发的，还是别人发的。这个时候就需要一个判断机制：
1 客户端同样也生成一个私钥和一个公钥， 把公钥提前在身份验证之前就发给服务器，这样服务器就有了一份来自客户端的公钥。
2 利用私钥来加密摘要；不加密内容，消息发送。
3 服务器接到内容，开始对内容进行摘要计算，得到摘要；并且利用公钥对摘要进行解密，得到解密后的真正的摘要。比较摘要，如果一致，就表示的这是真的客户端发的。

身份认证：你如何保证你是真的安全，无害的？比如你想制药，就需要制药的资质，你就要想官方机构申请，官方认证了，你是安全无害，有技术手段来制药的，就会发给你证书，有了这个，你的东西别人才敢买，别人才敢吃。同样这里的也是，不过这里的官方机构不是某个郭嘉的部门，而是一些公开的机构。通过他们的认证之后，你就可以向别人发送公钥了，同样别人也可以放心的把自己的公钥发给你。

我的粗浅的理解就是以上，希望老师多多指导，谢谢~~

作者回复: 你说的很对，所以密码学只是安全的基础，要真正安全还需要有协议来保证，所以就出现了后面的TLS。

2

刘政伟
2019-08-17

老师，在nginx中一般都会配置由ca机构签发的证书（ssl_certificate）和私钥（ssl_certificate_key）；
问题一：那个证书是由ca机构的私钥进行加密过的，然后在客户端访问的时候，通过浏览器中内置的ca的公钥进行解密，这样理解对吗?
问题二：nginx中配置的私钥(ssl_certificate_key)又是什么用途呢?

作者回复:
1.基本正确，浏览器用的实际上是证书里的公钥，不是裸公钥。

2.Nginx中配置的私钥就是用来证明服务器身份的私钥，也就是服务器证书对应的私钥，用来在tls握手时验证身份。

3.可参考后面两讲，看看证书和私钥是怎么起作用的。

2

极客时间
2019-07-27

有点疑惑，如果有中间人，截获了证书，将证书替换成了自己申请的证书，这里假设中间人申

请的证书和网站申请的证书是同一家的，确保用的都是相同的第三方公钥，那么这里是不是就会泄密了呢？有没有这种可能呢，我在阅读网上大部分文章的时候，大家都在考虑中间人修改证书上的公钥，但是因为数字签名的原因，修改后就出问题了，但是中间人直接将证书替换了呢？这种可能有没有，没有可能的话为什么，有的话如何做防护呢？ 或者说我提的这个问题是我哪里理解有问题吗？万望老师解惑。

作者回复: 证书体系中的中间人攻击是可行的，需要预先在客户端信任中间人的根证书，这样中间人就可以使用这个根证书来“伪造”证书，冒充原网站，像fiddler就是这么做的。

简单修改证书是不行的，因为证书被ca签名，能够防窜改。而中间人没有ca的私钥，所以也无法伪造。

2

唐锋

2021-01-08

有些同学回答的是先由一级验证二级，再由根验证一级；有些同学回答的是先由根验证一级，再由一级验证二级；有些同学回答的是先由二级通过链找到根，然后再由根验证一级，再由一级验证二级。老师说的都是理解正确。那么哪一个才是真的理解正确呢。

作者回复: 这个顺序其实并不重要，只要最后根证书验证成功，那么整个证书链就是可信的。

比如，我先用一级验证了二级，那么只要一级可信后面的也就都可信，所以要再验证一级。

或者是先用根验证了一级，那么再用一级验证其他的也就都可信。

关键是最后的根证书，它是信任的起点，必须要验证。

1

欢乐的小马驹

2020-03-20

老师，你好。我又来问问题了，希望能理解~
1、签名和验签，只是在服务器传给客户端证书的时候需要用到，是吧?
2、摘要是传递任何数据都会用到吗？如果不是这样的话，传输数据的过程中被篡改了不就没法保证完整了吗?

作者回复:
1.签名和验签必须要有证书才能执行，所以你说的只对了一半，如果客户端发证书，那么服务器也要验签。

2.现在的加密套件比如AES128-GCM会在加密的同时做mac验证，保证数据的完整性，不一定非要用摘要。其实最简单的crc也是可以保证数据不被破坏的，只是强度没有摘要那么高。

1

欢乐的小马驹

2020-03-19

老师，你好。你文章提到“黑客虽然拿不到会话密钥，无法破解密文，但可以通过窃听收集到足够多的密文，再尝试着修改、重组后发给网站。因为没有完整性保证，服务器只能“照单全收”，然后他就可以通过服务器的响应获取进一步的线索，最终就会破解出明文”
黑客怎么破解出明文的过程是什么样的，能不能简单收一下，我这不是很理解~

作者回复: 这个有个专门的分支叫密码分析学，专门研究怎么破解密钥的。

举个例子，我发一些精心构造的明文，比如全0全1全a等等，然后服务器会返回密文，只要足够多，就能够分析出规律来。

感兴趣可以搜一下二战时候的enigma破解过程。

1