# HTTPS 为什么更安全，先看这些

2017/02/27 · [IT技术](http://blog.jobbole.com/category/it-tech/) · [3 评论](http://blog.jobbole.com/110373/" \l "article-comment)· [HTTPS](http://blog.jobbole.com/tag/https/)

分享到：**11**

* **[Laravel和 AngularJS开发全栈知乎](http://coding.imooc.com/class/65.html?from=jobboleblog" \o "Laravel和 AngularJS开发全栈知乎" \t "http://blog.jobbole.com/110373/_blank)**
* **[webApp书城整站开发](http://coding.imooc.com/class/13.html?from=jobboleblog" \o "webApp书城整站开发" \t "http://blog.jobbole.com/110373/_blank)**
* **[巧用第三方快速开发Android App 热门SDK及框架](http://coding.imooc.com/class/81.html?from=jobboleblog" \o "巧用第三方快速开发Android App 热门SDK及框架" \t "http://blog.jobbole.com/110373/_blank)**
* **[React Native快速开发 厕所在哪App LBS定位 框架封装](http://coding.imooc.com/class/69.html?from=jobboleblog" \o "React Native快速开发 厕所在哪App LBS定位 框架封装" \t "http://blog.jobbole.com/110373/_blank)**

本文作者： [伯乐在线](http://blog.jobbole.com/) - [刘志军](http://www.jobbole.com/members/liuzhijun) 。未经作者许可，禁止转载！  
欢迎加入伯乐在线 [专栏作者](http://blog.jobbole.com/99322" \t "http://blog.jobbole.com/110373/_blank)。

HTTPS 是建立在密码学基础之上的一种安全通信协议，严格来说是基于 HTTP 协议和 SSL/TLS 的组合。理解 HTTPS 之前有必要弄清楚一些密码学的相关基础概念，比如：明文、密文、密码、密钥、对称加密、非对称加密、信息摘要、数字签名、数字证书。接下来我会逐个解释这些术语，文章里面提到的『数据』、『消息』都是同一个概念，表示用户之间通信的内容载体，此外文章中提到了以下几个角色：

* Alice：消息发送者
* Bob：消息接收者
* Attacker：中间攻击者
* Trent：第三方认证机构

### **密码**

密码学中的“密码”术语与网站登录时用的密码（password）是不一样的概念，password 翻译过来其实是“口令”，它是用于认证用途的一组文本字符串。

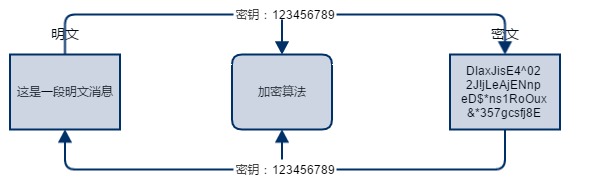
而密码学中的密码（cipher）是一套算法(algorithm)，这套算法用于对消息进行加密和解密，从明文到密文的过程称之为加密，密文反过来生成明文称之为解密，加密算法与解密算法合在一起称为****密码算法****。

### **密钥**

密钥（key）是在使用密码算法过程中输入的一段参数。同一个明文在相同的密码算法和不同的密钥计算下会产生不同的密文。很多知名的密码算法都是公开的，密钥才是决定密文是否安全的重要参数，通常密钥越长，破解的难度越大，比如一个8位的密钥最多有256种情况，使用穷举法，能非常轻易的破解。根据密钥的使用方法，密码可分为对称加密和公钥加密。

### **对称加密**

对称密钥（Symmetric-key algorithm）又称为共享密钥加密，加密和解密使用相同的密钥。常见的对称加密算法有DES、3DES、AES、RC5、RC6。对称密钥的优点是计算速度快，但是它有缺点，接收者需要发送者告知密钥才能解密，因此密钥如何安全的发送给接收者成为了一个问题。



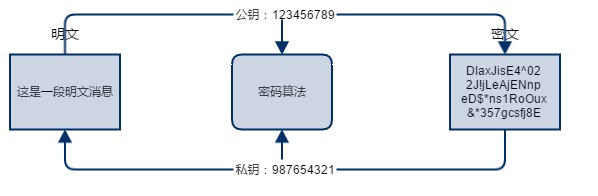
Alice 给 Bob 发送数据时，把数据用对称加密后发送给 Bob，发送过程中由于对数据进行了加密，因此即使有人窃取了数据也没法破解，因为它不知道密钥是什么。但是同样的问题是 Bob 收到数据后也一筹莫展，因为它也不知道密钥是什么，那么 Alice 是不是可以把数据和密钥一同发给 Bob 呢。当然不行，一旦把密钥和密钥一起发送的话，那就跟发送明文没什么区别了，因为一旦有人把密钥和数据同时获取了，密文就破解了。所以对称加密的密钥配是个问题。如何解决呢，公钥加密是一个办法。

### **公钥加密**

公开密钥加密（public-key cryptography）简称公钥加密，这套密码算法包含配对的密钥对，分为加密密钥和解密密钥。发送者用加密密钥进行加密，接收者用解密密钥进行解密。加密密钥是公开的，任何人都可以获取，因此加密密钥又称为公钥（public key），解密密钥不能公开，只能自己使用，因此它又称为私钥（private key）。常见的公钥加密算法有 RSA。

还是以Alice 给 Bob 发送数据为例，公钥加密算法由接收者 Bob 发起

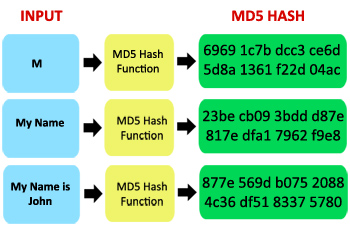
1. Bob 生成公钥和私钥对，私钥自己保存，不能透露给任何人。
2. Bob 把公钥发送给 Alice，发送过程中即使被人窃取也没关系
3. Alice 用公钥把数据进行加密，并发送给 Bob，发送过程中被人窃取了同样没关系，因为没有配对的私钥进行解密是没法破解的
4. Bob 用配对的私钥解密。



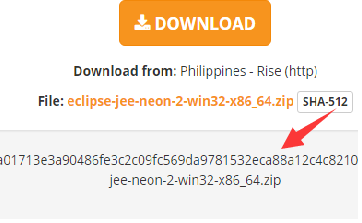
虽然公钥加密解决了密钥配送的问题，但是你没法确认公钥是不是合法的，Bob 发送的公钥你不能肯定真的是 Bob 发的，因为也有可能在 Bob 把公钥发送给 Alice 的过程中出现中间人攻击，把真实的公钥掉包替换。而对于 Alice 来说完全不知。还有一个缺点是它的运行速度比对称加密慢很多。

### **消息摘要**

消息摘要（message digest）函数是一种用于判断数据完整性的算法，也称为散列函数或哈希函数，函数返回的值叫散列值，散列值又称为消息摘要或者指纹（fingerprint）。这种算法是一个不可逆的算法，因此你没法通过消息摘要反向推倒出消息是什么。所以它也称为单向散列函数。下载软件时如何确定是官方提供的完整版呢，如果有中间人在软件里面嵌入了病毒，你也不得而知。所以我们可以使用散列函数对消息进行运算，生成散列值，通常软件提供方会同时提供软件的下载地址和软件的散列值，用户把软件下载后在本地用相同的散列算法计算出散列值，与官方提供的散列值对比，如果相同，说明该软件是完成的，否则就是被人修改过了。常用的散列算法有MD5、SHA。



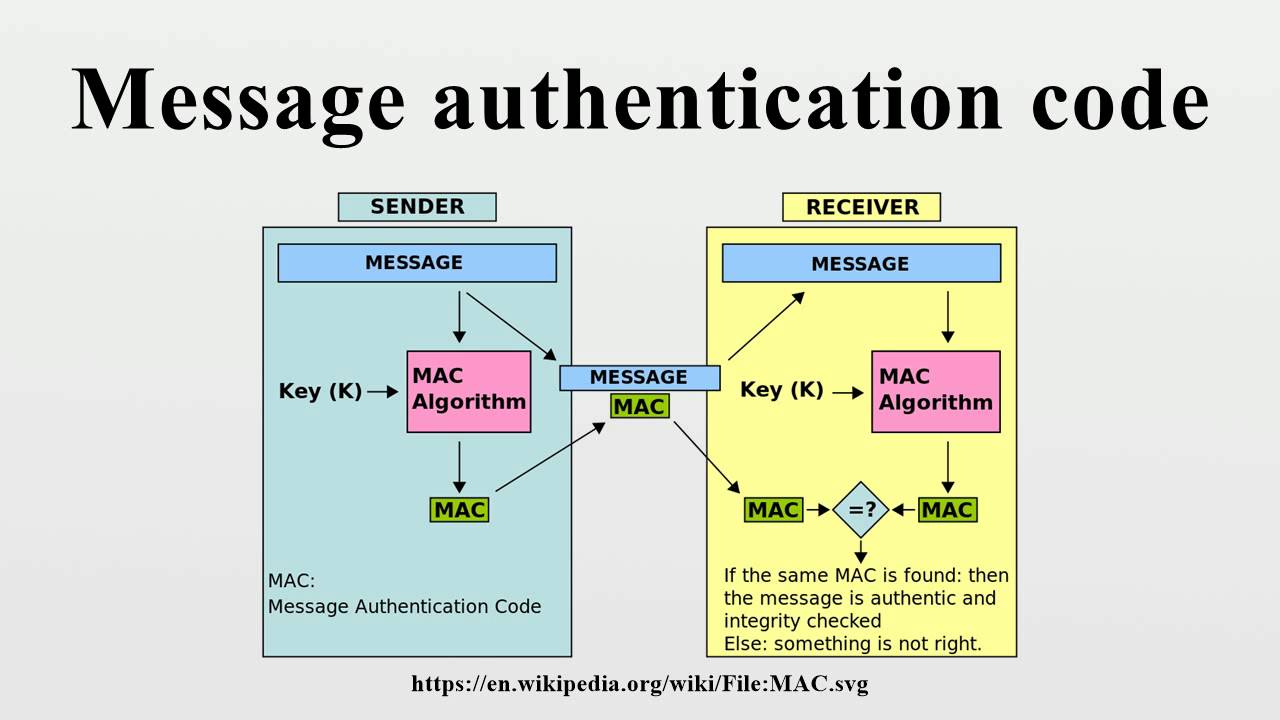
下载 Eclipse 时，官方网站同时提供了软件地址和消息摘要



散列函数可以保证数据的完整性，识别出数据是否被篡改，但它并不能识别出数据是不是伪装的，因为中间人可以把数据和消息摘要同时替换，数据虽然是完整的，但真实数据被掉包了，接收者收到的并不是发送者发的，而是中间人的。消息认证是解决数据真实性的办法。认证使用的技术有消息认证码和数字签名。

### **消息认证码**

消息认证码（message authentication code）是一种可以确认消息完整性并进行认证（消息认证是指确认消息来自正确的发送者）的技术，简称 MAC。消息认证码可以简单理解为一种与密钥相关的单向散列函数。



Alice 给 Bob 发送消息前，先把共享密钥（key）发送给 Bob，Alice 把消息计算出 MAC 值，连同消息一起发送给 Bob，Bob 接收到消息和 MAC 值后，与本地计算得到 MAC 值对比，如果两者相同，就说明消息是完整的，而且可以确定是 Alice 发送的，没有中间人伪造。不过，消息认证码同样会遇到对称加密的密钥配送问题，因此解决密钥配送问题还是要采用公钥加密的方式。

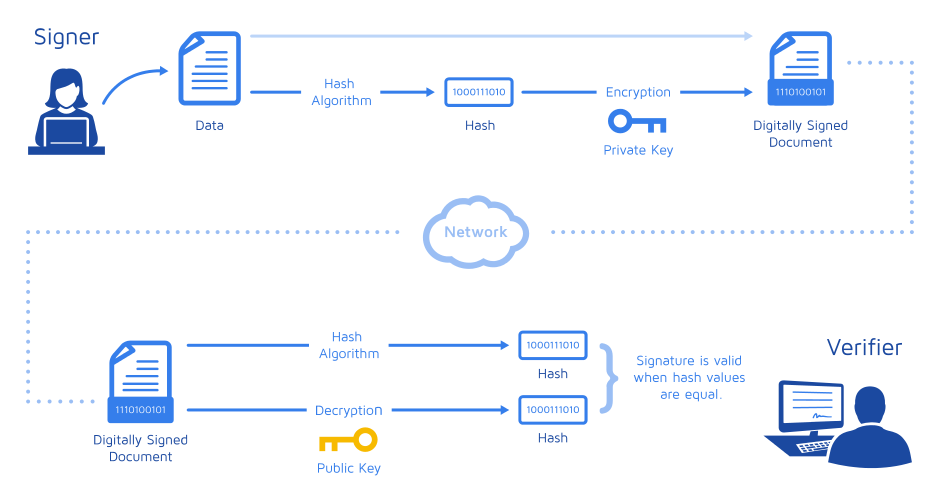
此外，消息认证码还有一个无法解决的问题，Bob 虽然可以识别出消息的篡改和伪装，但是 Alice 可以否认说：“我没发消息，应该是 Bob 的密钥被 Attacker 盗取了，这是 Attacker 发的吧”。Alice 这么说你还真没什么可以反驳的，那么如何防止 Alice 不承认呢，数字签名可以实现。

### **数字签名**

Alice 发邮件找 Bob 借1万钱，因为邮件可以被人篡改（改成10万），也可以被伪造（Alice 根本就没发邮件，而是 Attacker 伪造 Alice 在发邮件），Alice 借了钱之后还可以不承认（不是我借的，我没有签名啊）。

消息认证码可以解决篡改和伪造的问题，Alice 不承认自己借了钱时，Bob 去找第三方机构做公正，即使这样，公正方也没法判断 Alice 有没有真的借钱，因为他们俩共享了密钥，也就是说两个都可以计算出正确的 MAC 值，Bob 说：“明明你发的消息和 MAC 值和我自己生成的 MAC 值一样，肯定是你发的消息”，Alice 说：“你把密钥透露给了其他人，是他发的邮件，你找他去吧”。Alice 矢口否认。

数字签名（Digital Signature）就可以解决否认的问题，发送消息时，Alice 和 Bob 使用不同的密钥，把公钥加密算法反过来使用，发送者 Alice 使用私钥对消息进行签名，而且只能是拥有私钥的 Alice 可以对消息签名，Bob 用配对的公钥去验证签名，第三方机构也可以用公钥验证签名，如果验证通过，说明消息一定是 Alice 发送的，抵赖也不行，因为你只有 Alice 可以生成签名。这就防止了否认的问题。



它的流程是:

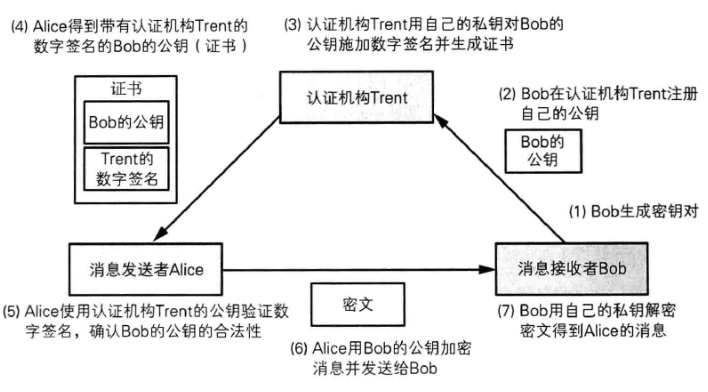
****第一步****：发送者 Alice 把消息哈希函数处理生成消息摘要，摘要信息使用私钥加密之后生成签名，连同消息一起发送给接收者 Bob。

****第二步****：数据经过网络传输，Bob收到数据后，把签名和消息分别提取出来。

****第三步****：对签名进行验证，验证的过程是先把消息提取出来做同样的Hash处理，得到消息摘要，再与 Alice 传过来的签名用公钥解密，如果两者相等，就表示签名验证成功，否则验证失败，表示不是 Alice发的。

### **公钥证书**

公钥密码在数字签名技术里面扮演举足轻重的角色，但是如何保证公钥是合法的呢，如果是遭到中间人攻击，掉包怎么办？这个时候公钥就应该交给一个第三方权威机构来管理，这个机构就是认证机构（Certification Authority）CA，CA 把用户的姓名、组织、邮箱地址等个人信息收集起来，还有此人的公钥，并由 CA 提供数字签名生成公钥证书（Public-Key Certificate）PKC，简称证书。



Alice 向 Bob 发送消息时，是通过 Bob 提供的公钥加密后的数据，而 Alice 获取的公钥并不是由 Bob 直接给的，而是由委托一个受信任的第三方机构给的。

1. Bob 生成密钥对，私钥自己保管，公钥交给认证机构 Trent。
2. Trent 经过一系列严格的检查确认公钥是 Bob 本人的
3. Trent 事先也生成自己的一套密钥对，用自己的私钥对 Bob 的公钥进行数字签名并生成数字证书。证书中包含了 Bob 的公钥。公钥在这里是不需要加密的，因为任何人获取 Bob 的公钥都没事，只要确定是 Bob 的公钥就行。
4. Alice 获取 Trent 提供的证书。
5. Alice 用 Trent 提供的公钥对证书进行签名验证，签名验证成功就表示证书中的公钥是 Bob 的。
6. 于是 Alice 就可以用 Bob 提供的公钥对消息加密后发送给 Bob。
7. Bob 收到密文后，用与之配对的私钥进行解密。

至此，一套比较完善的数据传输方案就完成了。HTTPS（SSL/TLS）就是在这样一套流程基础之上建立起来的。