# 13 | 如何防止数据被调包?

2020-12-23 范学雷

实用密码学 进入课程 >


**讲述：范学雷**

时长 10:38 大小 9.75M

你好，我是范学雷。

还记得我们在前面讨论过 CBC 模式补齐预言攻击吗？当时，我们谈到了一个问题，就是解密端是无法判断解密得到的数据是发送者发送的数据，还是被人掉了包的数据的。

这就好比，牛郎要送的信是，“七夕今宵看碧霄，牛郎织女渡鹊桥”。织女拆开一看，却是一堆乱码，或者是变成了“我断不思量，你莫思量我。将你从前与我心，付与他人可”。

如果这封信真的变成了一堆乱码，就意味着信息没有被有效地送达，这样会给织女愁上加愁；而如果内容是“我断不思量”，简直就是一支穿心箭，这可一点都不好玩。

那么，织女看到“我断不思量，你莫思量我。将你从前与我心，付与他人可”的信件，除了靠坚贞不渝的信念这个不靠谱的办法之外，她有没有其他的办法来判断这是一封伪造的信件？牛郎除了坚信织女有坚贞不渝的信念之外，他有没有其他的办法来预防信件的伪造？

其实，**解决问题的思路也很直观，就是要能够验证发送的信息**。牛郎发送信息时，心里要想着意外情况，要给织女提供验证信息的办法。织女收到信息时，要有办法验证信息的真伪，不能只依靠心里的信念。

那么，我们今天这一讲，就来分析一下如何防止数据被调包这个问题。

## 怎样有效地验证一段信息？

首先，我们来分析下，要想有效地验证一段信息，需要满足什么条件呢？

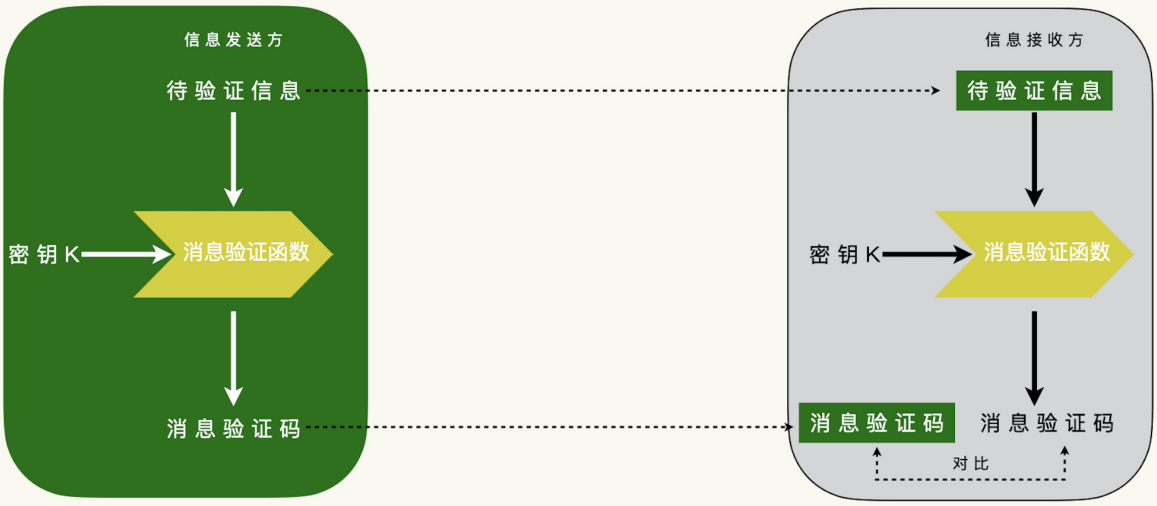
第一个条件就是，**我们要有额外的信息**。只有要验证的信息本身，是没有办法验证这个信息的。也就是说，信息本身不能自己验证自己。这个额外的信息，我们暂且称之为验证信息。

第二个条件就是，**验证信息和待验证的消息之间要有关联**。如果没有关联，也就意味着如果我们替换掉待验证的信息，验证信息并不受影响，这显然起不到验证的作用。

如果待验证的信息有变动，验证信息也要变动，而且验证信息的变动结果要不可预测。如果可以人为构造一段信息，它的验证信息和待验证的信息是一样的，也起不到验证的效果。

第三个条件，就是**验证信息的计算要快，数据要小**。不知道这一点，你能不能理解？这点说的就是，计算效率要高，要不然验证信息的实用和推广价值就会大打折扣。

那么，有同时满足上述三个条件的一个方案吗？消息验证码（Message Authentication Code, MAC）就是最常用的满足上述三个条件的一个方案。




## 消息验证码是怎么工作的？

既然消息验证码可以同时满足我们的三个条件，那么消息验证码是怎么工作的？

首先，我们来看使用消息验证码的前提，就是信息的发送方和接收方要持有相同的密钥，这和我们前面讨论的对称密钥的条件是一样的。能够使用对称密钥的场景，都能够满足这个前提。

另外，信息的发送方和接收方要使用相同的消息验证函数。这个函数的输入数据就是对称密钥和待验证信息。信息的发送方使用消息验证函数，可以生成消息验证码。

接下来，信息发送方把待验证信息和消息验证码都发送给信息接收方。信息接收方使用相同的消息验证函数和对称密钥，以及接收到的待验证信息，生成消息验证码。

然后，信息接收方对比接收到的消息验证码和自己生成的消息验证码。如果两个消息验证码是一样的，就表明待验证信息不是伪造的信息。否则，待验证信息就是被篡改过的信息。

这听起来是一个不错的方案。不过，消息验证函数需要使用对称密钥，输入任意大小的数据，输出为一小段数据。什么样的消息验证函数能够承担起这样的任务呢？

通常的对称密钥算法，密文数据不会小于明文数据，这样的话，就不能满足验证数据小的要求。因此，通常的对称密钥算法，我们并不能当做消息验证函数使用。

那么，我们到底该怎么选择消息验证函数？

## 该怎么选择消息验证函数？

我刚才提到，消息验证函数的输出应该是一小段数据，这一点有没有让你想起，我们前面提到的单向散列函数？我们再来回顾一下单向散列函数的三个特点：

正向计算容易，逆向运算困难；

运算结果均匀分布，构造碰撞困难；

给定数据的散列值是确定的，长度固定。

如果你再回头看看，我们上面讨论的消息验证的三个条件，单向散列函数就能够完美地满足这三个条件。那么，对称密钥怎么和单向散列函数结合起来，构造出消息验证函数呢？**最常见的方案就是基于单向散列函数的消息验证码。**

基于单向散列函数的消息验证码（Hash-based Message Authentication Code，HMAC）这个名字是不是听起来太长了，所以我们通常使用它的简称 HMAC。

在 HMAC 的算法里，单向散列函数的输入数据是由对称密钥和待验证消息构造出来的。到目前为止，这种构造方法还没有明显的安全问题，我们不再讨论构造细节。如果你感兴趣，可以查阅 1997 年发表的 RFC 2104。

一个密码学算法经历了二十多年还没有明显的安全漏洞，这真的是很难得！

不过，你需要注意的是，HMAC 算法并不使用我们前面讨论过的链接模式。所以，对称密钥链接模式的各种安全问题并不会影响 HMAC 算法的安全性。

## 为什么需要对称密钥？

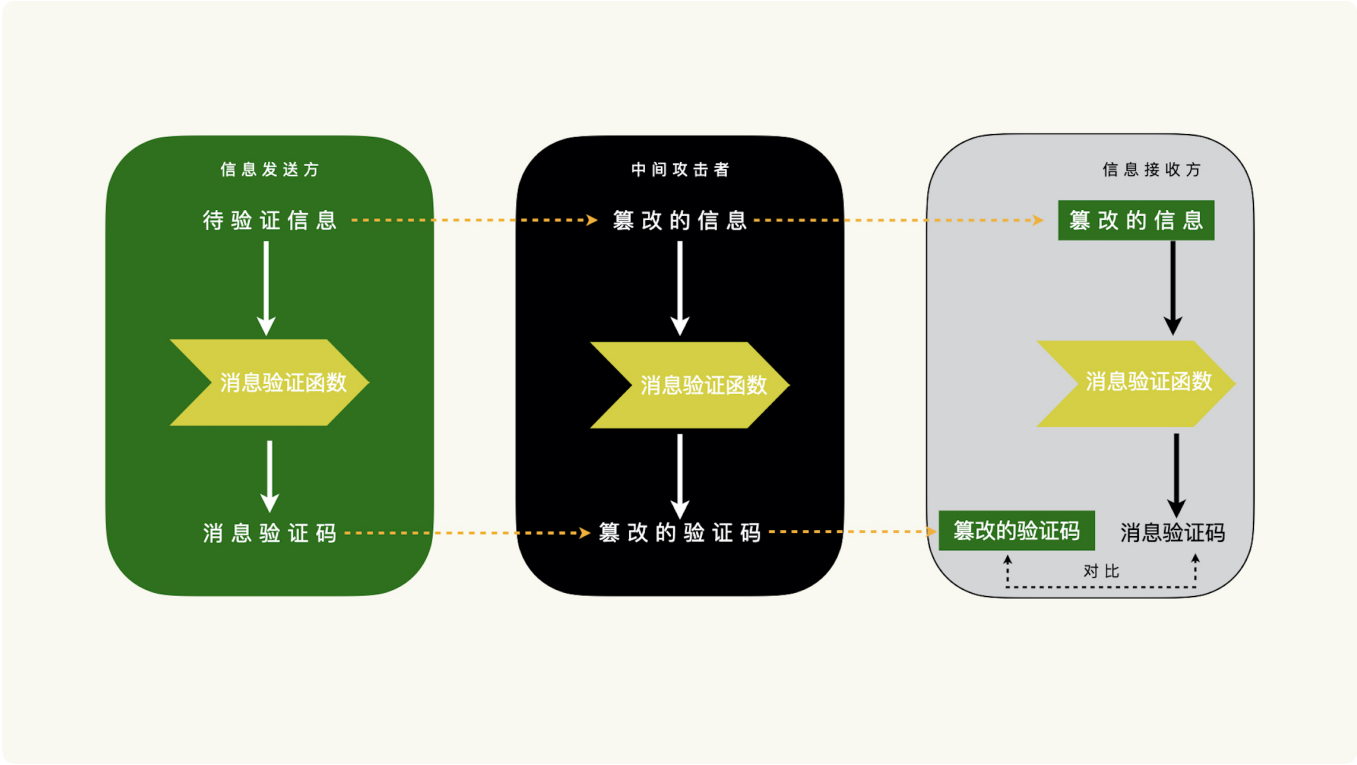
到这里，你是不是早就有了一个疑问，消息验证函数为什么还需要对称密钥呢？我们前面提到，单向散列函数也可以验证数据的完整性。为什么它不直接使用单向散列函数呢？

还记得吗？我们前面在讨论单向散列函数解决数据完整性问题的时候，还有一个遗留的问题，就是怎么获得原始数据的散列值。对称密钥就是用来解决这个问题的。

我们先来看看，如果没有对称密钥的加入，消息验证码还能不能工作。

信息发送方把待验证信息和消息验证码都发送给信息接收方。假设存在一个中间攻击者，能够解开待验证信息和消息验证码。由于单向散列函数是公开的算法，中间攻击者就可以篡改待验证信息，重新生成消息验证码。

然后，中间攻击者把篡改的信息和篡改的验证码发给信息接收方。篡改的信息和篡改的验证码能够通过信息接收方的信息验证。也就是说，这样的话，信息接收方就没有办法识别出这个信息是不是原始的、没有篡改的信息。这样，信息验证就失效了。




可是，如果对称密钥参与了消息验证码的运算，由于中间攻击者并不知道对称密钥的数据，攻击者就很难伪造出一个能够通过验证的消息验证码。换一个说法，**对称密钥的参与，是为了确保散列值来源于原始数据，而不是篡改的数据。**

有了对称密钥这个私有数据的参与，消息验证码的算法是不是就没有安全漏洞了呢？

## 怎么计算 HMAC 算法的强度？

HMAC 算法与对称密钥和单向散列函数息息相关，所以，对称密钥的安全强度和单向散列函数的安全强度，都会影响 HMAC 算法的安全强度。该怎么计算 HMAC 算法的安全强度呢？

严格的来说，HMAC 算法的安全强度，是由对称密钥的安全强度和两倍的散列值长度之间较小的那个数值决定的。比如，如果我们选择 256 位的对称密钥，以及散列值长度是 160 位的 SHA-1。

两倍的散列值长度就是 320 位。那么，在 256 位和 320 位两个数值之间，256 位是较小的数值。那么，这个 HMAC 运算的安全强度就是 256 位。

一般来说，两倍的散列值长度通常大于流行对称密钥强度。所以，HMAC 算法的强度，通常也是由对称密钥决定。简单起见，**对于流行的 HMAC 算法，我们只需要考虑对称密钥的安全强度**。

## 有哪些常见的 HMAC 算法？

HMAC 算法是由单向散列函数的算法确定的。下面的表格，我列出了一些常见的算法。同样的，我们把 HMAC 算法也按照退役的、遗留的以及现行的算法来分类。

| 单向散列函数 | | 现在能用吗？ | 发布日期 | 散列值长度（位） | 处理能力（位） |
|---|---|---|---|---|---|
| HmacMD5 | | 退役 | 1992.04 | 128 | 2^64 |
| HmacSHA1 | | 遗留 | 1995.04 | 160 | 2^64 |
| SHA-2 | HmacSHA224 | 现行 | 2004.02 | 224 | 2^64 |
| | HmacSHA256 | 现行 | 2002.08 | 256 | 2^64 |
| | HmacSHA384 | 现行 | 2002.08 | 384 | 2^128 |
| | HmacSHA512 | 现行 | 2002.08 | 512 | 2^128 |

| 单向散列函数 | | 现在能用吗? | 发布日期 | 散列值长度（位） | 处理能力（位） |
|---|---|---|---|---|---|
| SHA-2 | HmacSHA512/224 | 现行 | 2012.03 | 224 | 2^128 |
| | HmacSHA512/256 | 现行 | 2012.03 | 256 | 2^128 |
| SHA-3 | HmacSHA3-224 | 现行 | 2015.08 | 224 | – |
| | HmacSHA3-256 | 现行 | 2015.08 | 256 | – |
| | HmacSHA3-384 | 现行 | 2015.08 | 384 | – |
| | HmacSHA3-512 | 现行 | 2015.08 | 512 | – |

其中，**HmacSHA256 和 HmacSHA384 是目前最流行的两个 HMAC 算法**。和以前的讨论一样，为了最大限度的互操作性和兼容性，我们应该选择当前最流行的算法。

如果单独的加密并不能解决信息的有效传递问题，有没有加密算法，能够综合考虑信息的机密性和完整性？如果存在这样的算法，我们就不需要额外设计消息验证码了。下一次，我们来讨论这个问题。

## Take Away（今日收获）

今天，我们讨论了防止数据被调包的技术，也就是消息验证码。我们讨论了消息验证码要解决的问题，以及消息验证码的工作原理。我们还谈到如何选择消息验证函数，最常见的方案就是选择基于单向散列函数的消息验证码，也就是 HMAC。

为什么我们需要对称密钥？其实是为了解决单向散列函数的遗留问题，因为对称密钥的参与，可以确保散列值来源于原始数据，而不是篡改的数据。

最后，我们还研究了怎么计算 HMAC 算法的强度，还列出了目前常见的 HMAC 算法。我们应该选择当前最流行的算法，而对于流行的 HMAC 算法，我们只需要考虑对称密钥的安全强度。

通过今天的讨论，我们要：

了解消息验证码要解决的问题；

尽量选用现行的、流行的算法：HmacSHA256 和 HmacSHA384。

## 思考题

今天的思考题，是一个复习题，也是一个改进的题目。

我们前面讨论过的牛郎织女的约会问题。我们再来看看现在，我们有没有更好的办法解决这个问题。如果牛郎要给织女发信息，七夕相约鹊桥会。

> 织女：
> 七月初七晚七点，鹊桥相会。不见不散。
> 牛郎

你能够帮助牛郎想想吗？该怎么使用单向散列函数和消息验证码，来防范约会信息被恶意修改？你建议的办法还有没有其他的问题？

欢迎在留言区留言，记录、讨论你的想法。

好的，今天就这样，我们下次再聊。

### 附：表格中算法参考文献链接

HmacMD5：RFC 1321

HmacSHA1：FIPS 180-1

HmacSHA224：FIPS 180-3

HmacSHA256、HmacSHA384、HmacSHA512：FIPS 180-2

HmacSHA512/224、HmacSHA512/256：FIPS 180-4

HmacSHA3-224、HmacSHA3-256、HmacSHA3-384、HmacSHA3-512：FIPAS 202

提建议



上一篇 12 | 怎么利用加密端攻击?

下一篇 14 | 加密数据能够自我验证吗?

## 精选留言 (5)

写留言

**Litt1eQ**
2020-12-23

双方可以用之前约定好的密钥 然后牛郎用这个密钥给消息生成一个HMAC 然后消息和HMAC一同发给织女 织女收到消息 验证HMAC和消息是否匹配 则可确定消息有无篡改 这样做的前提是牛郎和织女能够安全的交换密钥 消息内容也最好加上具体的时间（防止重放攻击）

展开

作者回复: 嗯，要想办法先交换密钥。

1

**zzzz**
2021-01-19

老师，HMAC的传递方式是没有涉及上一节讲的明文密文对吗?
就是，加密解密和这里的消息验证是两块不相关的内容吗
我先把问题记在这里，等看完了全部再来看看嘿嘿

展开

作者回复: 是的，加解密和消息验证是两个领域的问题。

**孜孜**
2020-12-25

双向验证软件(two factor），比如Google身份验证器等，都会把epoch分段，然后在加上一点容错。这样即使客户端在提示的30s快到的时候输入验证码，服务端也会认证通过。 如果手动改变手机时间，双向认证软件会工作不正常。

展开

作者回复: 嗯，了解这个思路了。谢谢！

**范**

2020-12-24

谢谢老师的分享。
密码学相关的内容首先是发展变化的，没有一套算法或者密钥是无懈可击的；其次攻守双方是互相促进的；最后，密码学还是回归到人的问题，计算机只是工具。

作者回复: 同意前两条，不同意最后一条。密码学是用来解决人的问题的，回不去。认为人好了就好了，就太理想了。

**孜孜**

2020-12-23

我们用的 生成6位数字的双向验证软件，其实就是对当前时间进行hmac。因为时间这个原数据客户端和服务端都是一样的，所以hmac也应该是一样的。

展开

作者回复: 有点好奇，客户端和服务端的时间没有偏差吗？