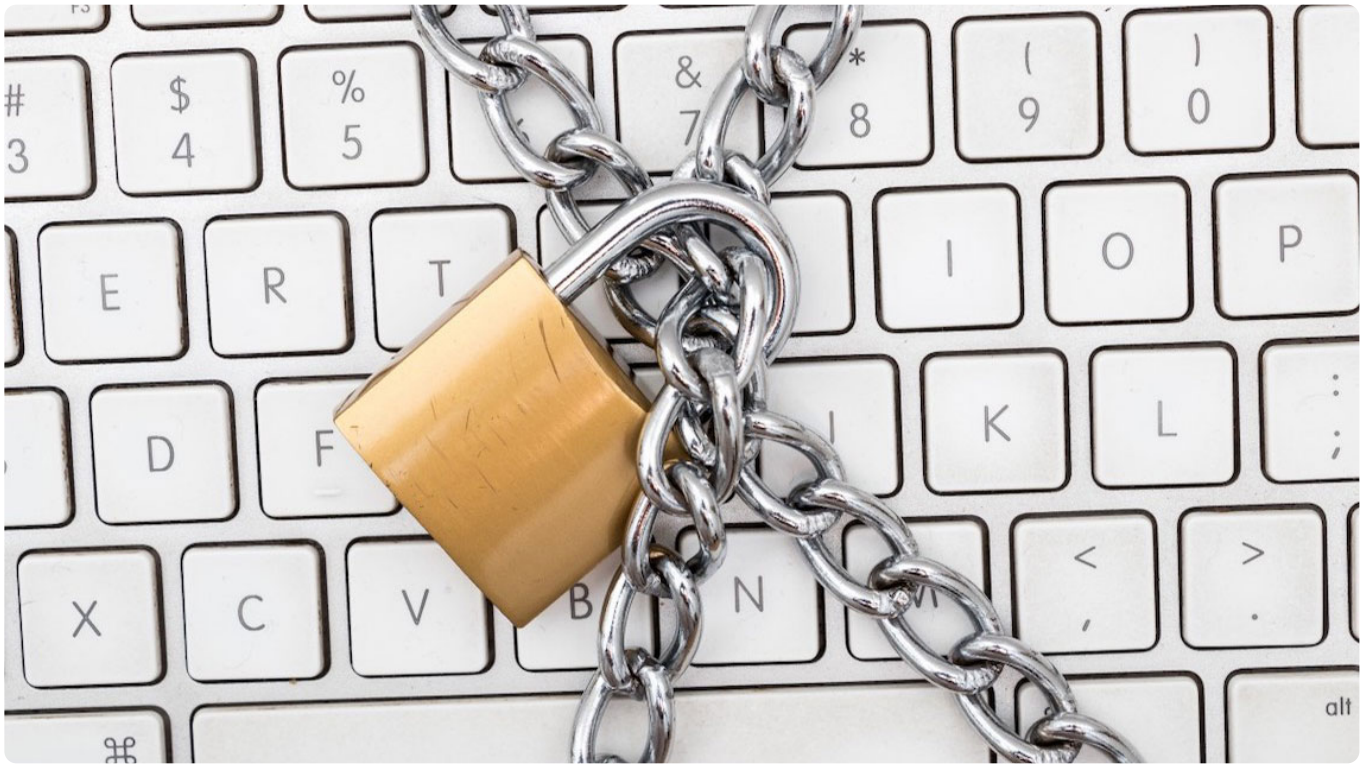
02｜单向散列函数：如何保证信息完整性？

2020-11-25 范学雷

实用密码学 进入课程 >



讲述：范学雷
时长 16:59　大小 15.56M

你好，我是范学雷。

从今天开始，我就要和你一起逐渐接触密码学的具体细节了。在这个过程中，我会讲到很多密码学相关的概念和诉求，这是我们打好基础的关键。不过，你也不用紧张，我会和你一起分析，一步步带你掌握这些知识点。

还记得上一讲，我们讨论的话题吗？我们通过牛郎织女约会送信的小例子，探讨了“密码学有什么用”这个问题，从而理解了信息安全的基本问题和基本需求。

问题出现了，我们也知道了该用密码学。现在就要来解决问题了，接下来的几讲，我们就先来解决“信息的完整性”这个问题。你还记得解决完整性的工具是什么吗？——单向散列函数。

可是，什么是单向散列函数？它是怎么解决完整性问题的？今天，我们就来讨论这两个问题。

什么是单向散列函数?

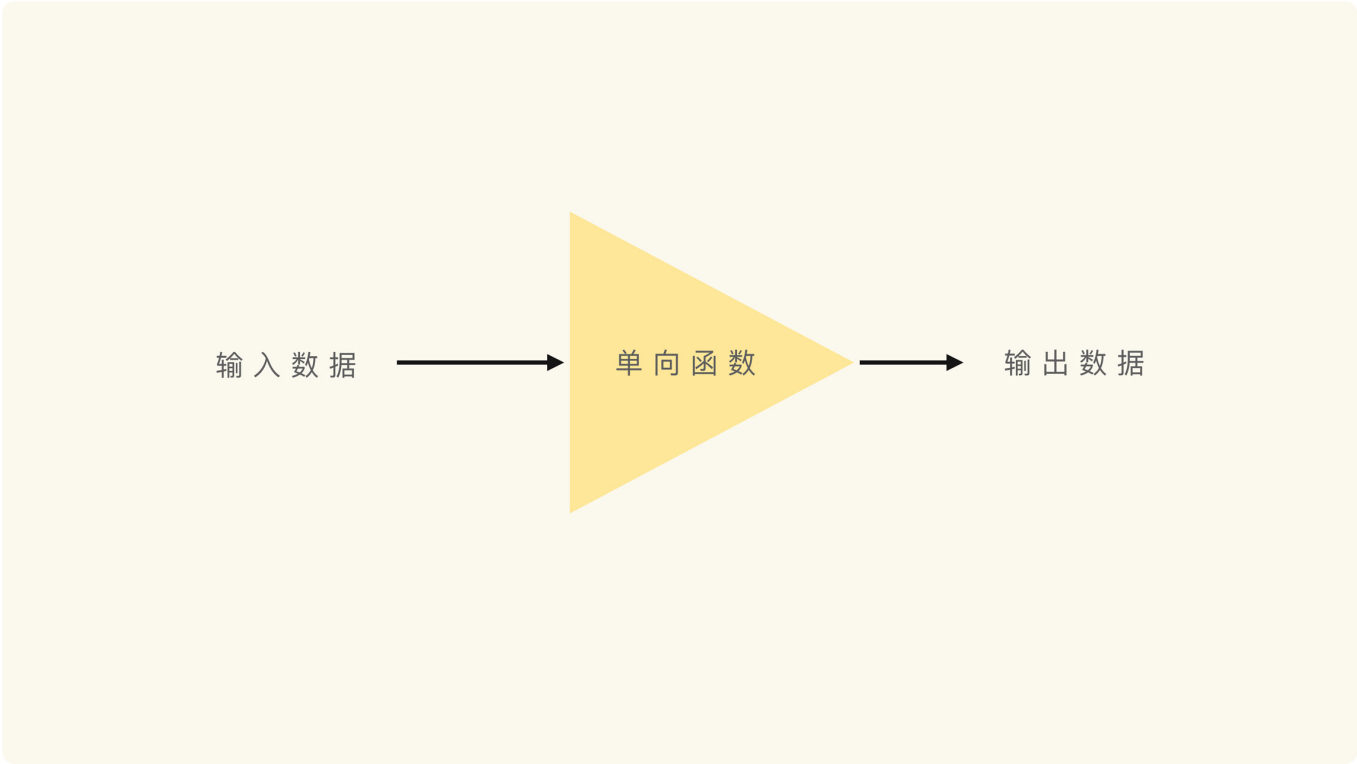
首先，我们从名字上看，一眼就能看出来单向散列函数有两个关键修饰词，“单向”和“散列”。

其实，在数学上，单向函数和散列函数是两个不同类型的函数。所以，我们要想理解单向散列函数，我们就要先知道什么是单向函数，什么又是散列函数。

什么是单向函数?

如果你没有了解过什么是单向函数，你可以先猜一下，为什么它叫单向函数?

单向函数（One-way Function）是正向计算容易，逆向运算困难的函数。也就是说，给定你一个输入，你很容易计算出输出；但是给定你一个输出，你却很难计算出输入是什么。



还有这样的函数？是不是感觉有点神奇?

《应用密码学》有一个很生动的例子来解释单向函数。把盘子打碎是一件很简单的事情，但是把这些碎片再拼接成一个完整的盘子，就是一件非常困难的事情。

也许，你会想，虽然把盘子碎片再拼接起来非常困难，但是仅仅就是非常困难而已，无论是手工还是计算机辅助，碎盘子还是可以拼接起来的。是的，这就是这个例子巧妙的地方。

单向函数就是这样的一个盘子。虽然我们强调，单向函数只能正向计算，不能逆向运算。但其实，这只是一个美好的愿望。为什么我这么说？

因为，我们能找到的、谈到的所谓的单向函数，都是正向计算容易，逆运算困难的函数。是的，我用的词语是“困难”，而不是“不能”，可能性只是很小，但不是没有。

在我们的日常生活里，泼出去的水再也收不回，说过的话、做过的事也没地方买后悔药，单向似乎才是生活的常态。但在数学领域，有很多函数看起来像是严格的单向函数，我们既证明不了它是单向函数，也暂时找不到逆向运算的办法。到底有没有逆向运算的办法，我们现在还不知道。

为什么我要和你强调“逆向运算困难”这件事？因为密码理论领域里很多棘手的问题，密码应用领域里的很多错误，都是来源于单向函数的这种不确定性。

比方说吧，每一个被破解的单向散列函数的密码学算法，在它被发明的时候，人们都没有找到逆向运算的办法，可是被破解的时候，人们就发现原来还是有办法去逆向运算的。

今天还是安全的算法，明天就可能被破解。**这虽然使得密码学充满了挑战，但同时也使密码学充满了乐趣。**

不过，需要注意的是，我们要对这种不确定性保持足够的警惕，采取足够的防范措施。比如说，一个应用程序，至少要支持两种单向函数，当一种出现问题时，另外一种可以替补。

现在你知道了，单向函数是一个正向计算容易，逆向运算困难的函数。那我要是问你，对于我们来说，什么样的单向函数会更实用呢？我想，你应该可以回答出来：

一个更实用的单向函数，正向计算会更容易，容易程度就是这个函数的**计算性能；**

一个更实用的单向函数，逆向运算会更困难，困难程度就是这个函数的**破解强度。**

同样，我还是要强调一下，**一个实用的单向函数，计算强度和破解强度要均衡考量，不可偏废。**

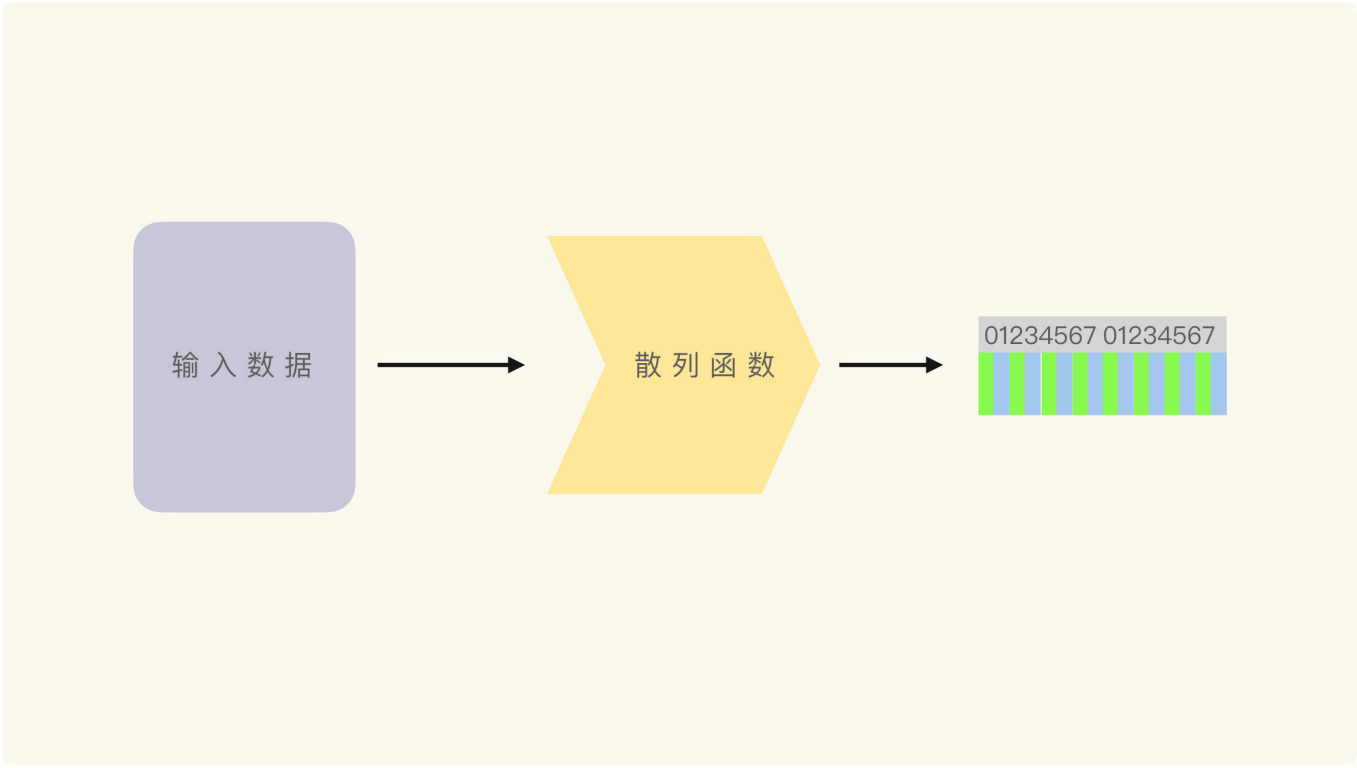
以后我们谈到单向函数，指的都是正向计算容易，逆向运算困难的函数，除非特别声明。

什么是散列函数?

讲完了单向函数，我们再来看什么是散列函数。

散列函数（Hash Function）是一个可以把任意大小的数据，转行成固定长度的数据的函数。比如说，无论输入数据是一个字节，或者一万个字节，输出数据都是 16 个字节。

我们把转换后的数据，叫做**散列值。**因为散列函数经常被人们直译为哈希函数，所以我们也可以称散列值为哈希值。通常的，对于给定的输入数据和散列函数，散列值是确定不变的。



你可能会说，我懂了，不就是输入数据任意长度，输出数据固定长度吗?

是的，可问题也来了，既然输入数据的大小没有限制，而输出结果的数据长度固定，那么你觉得，会不会存在散列值相同的两个或者多个数据呢？——是确定存在的。

通常，我们把这种情况称为**散列值碰撞**。对于散列函数，散列值碰撞可不是一件好事情。

如果你学过 Java 语言或者数据结构和算法，应该对哈希值这个概念不陌生。Java 语言里的 hashCode() 方法，或者数据结构和算法里的哈希值，就是一个散列函数的运用。

如果 hashCode() 的实现出现散列值碰撞，就会影响应用程序的性能，比如 HashMap 的检索时间会显著加长。再比如说，如果我们使用 hashCode 作为键值或者索引，散列值碰撞会导致检索错误，从而带来数据安全问题。

我在第一季《代码精进之路》专栏里，也讨论过散列值碰撞的性能基准测试。如果 10,000 个对象，只有 10 个不同的散列值，它的集合运算的性能是令人担忧的。因为这样和使用了没有散列值碰撞的实现相比，在性能方面，会有几百倍的差异。

现在，你应该意识到这个问题的重要性了，那么，我们应该怎样避免散列值碰撞呢？其实，因为输入数据的大小没有限制，输出数据的长度固定，理论上，我们是无法避免散列值碰撞的。

我们只能在降低散列值碰撞的可能性上想办法。也就是，我们要思考，如果我们不能避免散列值碰撞，我们会有什么办法可以降低散列值碰撞的风险呢？你可以先想一想。

最直观的办法，就是在**输出数据的长度**上想办法。虽然散列值长度固定，但是，我们可以让数据变得更长，**散列值越长，存在相同散列值的概率就越小，发生碰撞的可能性就越小**。

比如说，32 位固定长度的散列值就要比 16 位固定长度的散列值发生碰撞的可能性更小。

是不是觉得我们可以解决问题了？问题是解决了，但从另一个角度来说，散列值越长，通常也就意味着计算越困难，计算性能越差。而且，你想一想，为什么当初我们要使用固定长度的散列值？不就是为了减少计算本身的性能损耗，从而获得性能优化吗？

所以，散列值也不是越长越好。那么，我们到底该如何选择散列值的长度呢？

其实，散列值的长度选择，应该是**权衡性能**后的结果。比如 Java 语言里，hashCode() 的返回值是 32 位的整数，也就意味着散列值的长度是 32 位。由于 hashCode() 的返回值主要是用来检索，32 位的整数已经足够大了，所以这是一个合适的选择。

除了散列值长度之外，想要降低散列值碰撞的可能性，我们还要考虑散列值的质量。**一个好的散列函数，它的散列值应该是均匀分布的**。也就是说，每一个散列值出现的概率都是一样的。

如果不这样的话，一部分散列值出现的概率就会较高，另一部分散列值出现的概率会较低，别人就更容易构造出两个或者多个数据，使得它们具有相同的散列值。这种行为，叫做**碰撞攻击**。

如果你要实现在 Java 的 hashCode() 方法，就需要考虑散列值的均匀分布问题。你可以看看《Effective Java》这本书，里面有专门的文章介绍如何实现 hashCode() 方法，降低散列值碰撞的风险。

什么是单向散列函数?

我们说完了什么是单向函数和什么是散列函数，现在我们可以探讨什么是单向散列函数了。

单向散列函数既是一个单向函数，也是一个散列函数。它不仅要满足单向函数的要求，还要满足散列函数的要求。你还记得这两种函数的要求吗？其中，最要紧的就是：

逆向运算困难；

构造碰撞困难。

大部分的 hashCode() 方法的实现，都满足不了逆向运算困难的要求，所以它们是不能算作单向散列函数的。比如说，按照 Java 的 hashCode() 方法的实现，32 位整数的哈希值是这个整数本身，所以逆向运算一点难度都没有，当然不能算作单向散列函数。

单向散列函数是一定要逆向运算困难的。

至于构造碰撞困难，我用现成的单向散列函数给你举一个例子，比如 SHA-1 算法，它是一个常见的适用于密码学的单向散列函数。

现在，你面前有两句话，分别是“Hello, world!”和“Hello, vorld!”，这两句话只有一位的差异 (w: 119/01110111, v: 118/01110110)，我把它们的 SHA-1 算法计算出来的散列值，列在了下面。

你可以对比两个散列值，感受一下一个位的输入数据差异，计算出的散列值能有多大的差异。

复制代码

```
SHA-1("Hello, world!):
10010100 00111010 01110000 00101101 00000110 11110011 01000101 10011001 101011

SHA-1("Hello, vorld!):
11001011 11111111 11111011 10010011 01010111 11000010 10001101 01011000 001000
```

是不是差异还挺大的？这种现象，我们把它叫做雪崩效应。

雪崩效应（Avalanche Effect）是密码学算法一个常见的特点，指的是输入数据的微小变换，就会导致输出数据的巨大变化。严格雪崩效应是雪崩效应的一个形式化指标，我们也常用来衡量均匀分布。**严格雪崩效应指的是，如果输入数据的一位反转，输出数据的每一位都有 50% 的概率会发生变化。**

一个适用于密码学的单向散列函数，就要具有雪崩效应的特点，也就是说，如果一个单向散列函数具有雪崩效应，那么对于给定的数据，构造出一个新的、具有相同散列值的数据是困难的。

在这一讲的一开始，我们说过，密码学的单向散列函数是用来解决数据完整性问题的。那么，单向散列函数是怎么解决数据完整性问题的呢？

怎么解决完整性问题？

想要解决完整性问题，我们就要知道完整性问题的背后逻辑是什么。

完整性意味着什么？完整性的核心是**数据未经授权，不得更改**。对于“不得更改”这四个字，你最直观的感受是什么？是不是无论如何，数据都没有办法改动？这是一个很强的解读。一般情况下，也很难有满足的场景。

还有一种站在反面看的、曲线的解读，就是如果数据有变动，能够被检测出来，我们就不采纳被篡改的数据。使用单向散列函数，就可以通过检查数据是否有变动，来解决数据完整性问题。

我们刚才说了，在单向散列函数里，一段数据，无论它是少了一个字，多了一个字，或者修改了一个字，原始数据和修改后的数据的散列值都可能相差巨大。

而且，由于逆向运算困难，虽然存在具有相同散列值的两个或者多个数据，但是对于一个好的单向散列函数来说，刻意寻找这样的数据是困难的。如果困难程度足够大，我们就有足够信心认为，如果散列值没有变化，它对应的输入数据也没有变化。

所以，单向函数和散列函数的组合，单向散列函数，就可以帮助我们解决完整性问题。

假如我们收到了一段数据，我们就可以重新计算这段数据的散列值。如果我们还可以获得数据发送者计算的散列值，我们就可以对比新计算的散列值和接收到的散列值。如果两个散列值是相同的，我们就可以认为这段数据是完整的；否则，这段数据就是被篡改过的。

复制代码

```
输入：
    1、数据D
    2、原始数据的散列值H
    3、计算散列值使用的散列函数
输出：
    数据D是不是完整的？


运算：
    1、使用散列函数计算数据D的散列值H'；
    2、对比数据的散列值H和计算获得的散列值，如果两个散列值相同，则数据D是完整的；否则，数据D
```

可是，这里面依然有两个遗留问题，也是我们使用单向散列函数需要特别关注的两个问题。

第一个问题是，我们该选择什么样的散列函数，它的破解难度才能足够大？这样，我们才有足够的信心根据散列值判断数据的完整性。

第二个问题是，我们怎么能够安全地获得数据发送者计算的散列值？如果我们接收到的是被修改过的数据和修改过的散列值，我们是没有办法判断数据是不是完整的。

第二个问题，我们放在稍后一点讨论。下一次，我们讨论第一个问题。

Take Away（今日收获）

今天，我们讨论了单向函数、散列函数以及单向散列函数，还有怎么使用单向散列函数来解决数据和信息的完整性问题。

为什么我要先讲单向散列函数？因为，单向散列函数是密码学的基础。在一个应用系统里，如果单向散列函数选择失误，整个系统的安全性就无从谈起。之后，我们还会讨论单向散列函数是怎样和加密算法以及签名算法结合起来，构建宏大的信息安全基础架构的。

我们常说，铁打的营盘流水的兵。**在密码学里，最基础的概念就像是铁打的营盘，具有长久的生命力；而密码学算法就像是流水的兵，隔一阵儿就会换一茬。**

所以，每一次讨论，我总是会先交代清楚基本概念和基础诉求，然后再带你去看具体的算法。基本概念和基础诉求可以跟随你几十年，随着你对它们理解的加深，会逐渐加厚你的功力。理解了基本概念和基础诉求，你就可以得心应手地调度、安排生命只有十数年的密码学算法了。

这一讲，通过对单向散列函数的讨论，我们要：

理解单向散列函数的以下三个特点：

单向散列函数正向计算容易，逆向运算困难；

单向散列函数运算结果均匀分布，构造碰撞困难；

对于相同的单向散列函数， 给定数据的散列值是确定的，长度是固定的。

知道单向散列函数解决数据完整性问题的基本思路。

思考题

我们回头看看上一次讨论过的牛郎织女的约会问题。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，来防范约会信息被恶意修改？然后，你再想想，你建议的办法还有没有缺陷？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。

提建议

上一篇 01 | 学习密码学有什么用?

下一篇 03 | 如何设置合适的安全强度?

精选留言 (11)

写留言

天天有吃的

2020-11-25

希望老师可以把课程中用到的代码整理一份到git，比如今天的SHA1，虽然之前没接触过但是可以看看代码跑一跑理解一下，不然自己再去找博客去调试还是挺费时间的

> 作者回复: 课程里尽量不使用代码，这样大家看起来就不用顾及语言问题了。但是，后面我会把涉及到的算法的用法，用Java写出来放到加餐或者Github上。

4

25ma
2020-11-26

突然有个大胆的想法，给老师这个课想了一个有趣的名字[趣谈密码学]，哈哈😁，让枯燥的密码学算法的学习变得更加有趣，生动，形象，感谢老师

展开

作者回复: 哈哈，明明不是一个讲故事的专栏。真是有点这个意思。不过，等着吧，烧脑的马上就来。

1 2

tony
2020-11-26

牛郎利用非对称算法，生成一个公钥与私钥。公钥对外公开。
织女利用非对称算法，生成一个公钥宜私钥。
公钥对外公开。

牛郎生成对称密钥（对称算法织女知道）。牛郎用自己的单向散列函数计算信件内容输...

展开

作者回复: 几乎是完美的。有两个小问题可以再细化一下。第一个问题是，牛郎怎么确认织女的公钥是织女的，织女怎么知道牛郎宣称的公钥真的是牛郎的？第二个问题是，使用使用非对称密钥加解密，有没有什么安全隐患？哪些算法可以这么干，那些算法不行？很遗憾，这个专栏没有计划讨论非对称密钥，这些都是很有意思的话题。

不过，仅仅使用对称密钥也能解决问题。

2 1

Anjou
2020-11-25

1）织女生成一对密钥，一个是公钥，另一个是私钥；
2）织女把公钥寄给牛郎；
3）牛郎写信给织女，对信对内容进行hash；用公约对hash值加密作为数字签名；用公钥对信的内容进行加密；把加密后的信和信的数字签名一起寄给织女；
4）织女收信后，用私钥解密信的内容；用私钥解密数字签名得到信的hash值，比对信...

展开

作者回复: 整体思路大方向没问题，还有小细节你要在挑战一下：既然公钥是公开的，王母娘娘会不会也截获了公钥，然后同样的步骤3，篡改信的内容？另外，公钥一般不能用来加密像信这样的信息，我们讲非对成密钥的时候在讨论为什么。

1 1

Baldwin
2020-11-25

牛郎和织女约定好好加密方法。这几个字的实际存储比特取反然后用gb2312编码。最后用孙悟空的浑号做关键字；计算写一个校验和。当然加密方法和关键字 和签名算法只有牛郎和织女知道

作者回复: 也就是说，除了加密方法之外，没有别的秘密？这个加密方法怎么传递给对方呢？

1 1

明
2020-12-03

老师 小白问一个很简单的问题（评论里大佬说的我都不会😂😂）单向散了函数说的是一个函数 既满足单向性质，又满足散列性质吧 而不是两个只满足其中一个性质函数的组合使用

作者回复: 是的，只满足一个是不行的。

2020-11-26

使用RSA方式 私钥对数据进行加密，并携带数据和加密算法，公钥解密。
比如：A：加密算法，B：数据，C：hash值
整个信息为：A.B.hash(A+B+私钥）组成的，如果B 被篡改了，公钥验证hash里面的就会验证失败。
而RSA方式公钥不能对数据加密和篡改，只能验证。...
展开

作者回复: 你的意思是使用私钥加密hash值对不对？hash的计算不能有私钥吧？要不然，对方怎么重算、对比散列值呢？

1

成

2020-11-25

老师，我觉得可以把信息内容用单向散列函数计算出散列值，然后把这个值和函数算法包含在信件里发出，然后接受者可以用相同函数来计算信件里的内容，得到的散列值同附带的散列值比较，如果相同说明内容没有被篡改。但是有个问题，如果王母娘娘得到这封信把内容改了，用算法再计算一个新的散列值，发出去这就不好办了。所以还需要一个新的协定来约束。

展开

作者回复: 是的，还需要一个办法来防止篡改。单向散列函数单独只能解决一部分问题。

彩色的沙漠

2020-11-25

老师，接口数据采用的AES加密之后通信，对于明文还有必要在做哈希让后端在做完整性校验吗？如果做了哈希把哈希值放到某一个字段里面，然后在整体加密传输，感觉必要性不大，因为数据已经加密，如果被别人破译了加密，那完整性校验就不攻自破了。
需要一种混合算法，把明文的哈希值和明文的加密数据通过某种算法混合到一块，这种安全性高一点。

展开

作者回复: 这要看AES具体的加密模式，如果是CBC或者ECB这些传统的加密模式，数据完整性是没有办法保证的。我们后面还会讨论这些安全问题，和带有自我验证的加密模式。也可以想一想一个例子，就是如果中间环节故意丢弃几个数据包，会有什么问题？

Charlie Guo

2020-11-25

可以把牛郎要发给织女的信的每个字的笔画数记录下来，然后加上声调数算出来的整数总值用二进制表示，然后通过Sha-256算出值加在信的最后。如果再可靠一点的话可以在第一步算出来的值上再加一个只有牛郎和织女才会知道的幸运数字再做Sha运算。

展开

作者回复: 这个思路清奇啊！这是假设只有织女牛郎知道用笔画加声调计算散列值吗？如果知道了幸运数字，还需要数笔画和算声调吗？

3

Litt1eQ
2020-11-25

牛郎喝织女可以先约定在消息后面拼接一个固定的字符串 这个字符串是提前约定好的 只有牛郎和织女知道 然后对如下信息作散列算法 原始信息 + 固定字符串+时间戳 发送的信息是 原始信息 散列算法的值 时间戳 这样织女在收到消息之后 按照同样的方法对消息作散列 如果值一样 表示消息没有被篡改过

展开

作者回复: 固定字符串和时间戳有什么用呢？是不是要假设王母娘娘不知道使用固定字符串和时间戳来伪造散列值？

1