# HASH和数字签名在比特币中的应用模式

比特币本身并没有创造新的密码学成果，但比特币利用现有密码学成果构建了一个令人惊奇的全新的数字货币世界：去中心化、区块链、可编程货币等成果即使抛开比特币本身而言也是值得赞赏的洞见。

首先，现代密码学理论的共识遵循“柯克霍夫原则”：

柯克霍夫原则由奥古斯特·柯克霍夫在19世纪提出：密码系统应该就算被所有人知道系统的运作步骤，仍然是安全的。

这个是什么意思呢，拿钥匙和锁的例子来说，研制和生产锁具（包括钥匙）的工艺是完全公开的，锁具被攻破只有两种可能：一是证明工艺有漏洞，不需要拿到原装钥匙也能打开。二是穷尽各种钥匙可能，在可接受的时间里能够从概率意义上试出来（暴力破解）。

算法是公开的，唯一需要保护的是密钥，这是我们下文讨论的基础。

## 1.非对称加密

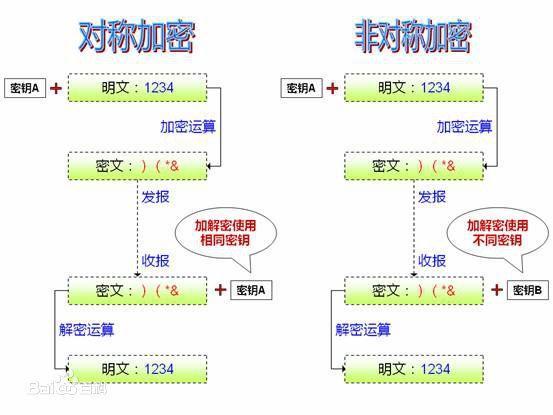
先看对称加密很好理解也符合直觉：

对称加密：对同一份敏感数据，加密解密密钥是相同的。

非对称加密呢：

非对称加密算法需要两个密钥：公开密钥（publickey）和私有密钥（privatekey）。公开密钥与私有密钥是一对，如果用公开密钥对数据进行加密，只有用对应的私有密钥才能解密；如果用私有密钥对数据进行加密，那么只有用对应的公开密钥才能解密。因为加密和解密使用的是两个不同的密钥，所以这种算法叫作非对称加密算法。 非对称加密算法实现机密信息交换的基本过程是：甲方生成一对密钥并将其中的一把作为公用密钥向其它方公开；得到该公用密钥的乙方使用该密钥对机密信息进行加密后再发送给甲方；甲方再用自己保存的另一把专用密钥对加密后的信息进行解密。

我们用一幅图来说明：

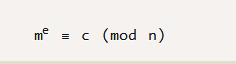


在上文的非对称加密部分，密钥A就是公开密钥（简称公钥），密钥B就是私有密钥（简称私钥）。道理都懂，问题有意思在什么地方呢？非对称加密方式下密钥A和密钥B是怎么找到的以及怎么实现的？我们举例说明：

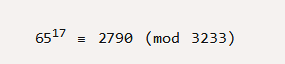
假设我们现在已经找到了（具体怎么找的后面会讲）公钥（3233,17）和私钥(3233, 2753)。

注意，公钥和私钥不一定只有一个数字，是可以有多个的，具体几个数字依赖于非对称加密算法。

假设现在明文待加密信息是数字65，我们首先给出加密公式：



解释一下，c代表加密后数字，（n，e）对应我们的公钥对，m代表明文，≡的意思是同模运算，比如60≡4（mod 7）,60对4取模后等于4。好了我们开始计算密文：



密文是2790，如何解密呢，解密公式：

https://upload-images.jianshu.io/upload_images/1367447-3e9ed8e72e05be9c.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/203

d对应私钥的2753，其余字母和加密过程相同，于是解密运算为：

https://upload-images.jianshu.io/upload_images/1367447-38c2af86757c72e3.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/293

我们得到了原来的明文数据65，这个例子可以看到加密和解密数据是不同的。

以上我们举例使用的算法是大名鼎鼎的地球主流非对称加密算法RSA，有关RSA的介绍参考[阮一峰老师的文章](https://link.jianshu.com?t=http:/www.ruanyifeng.com/blog/2013/06/rsa_algorithm_part_one.html)。实际上我上面用到的数字也出自该文，实在是时间有限无法深入讲解RSA的细节大家可以自行学习。比特币体系实际使用的非对称算法是椭圆曲线加密（ECC），可以到[这里](https://link.jianshu.com?t=http:/8btc.com/article-138-1.html)详细了解。

从上面的例子我们知道满足非对称加密的密钥对是存在的，同时我们也做了加解密尝试。实际上，非对称加密算法的核心依赖于特定的数学难题，比如上文的RSA依赖于大数分解难题，如果该难题被破解了，算法本身也就被攻破了。另外，我读研的时候还做了RSA矩阵扩展，将其算法基础从素数扩展到矩阵，有兴趣的可以继续交流。

非对称算法通过公私鈅分别加解密方式给信息交换带来了巨大的变化，具体表现在：

1）在不安全的环境中传递敏感信息成为可能。从上文可以知道，公钥是完全公开任意传播的，通过公钥是无法（或极其困难）推算出私钥的，私钥是不公开不发送不传播的，仅仅消息接收方知道就可以，其他任何人都不需要知道。

2）多方通信所需密钥数量急剧减少，密钥维护工作变得异常简化。比如N个互不信任且互有通信需求的人，如果使用对称加密算法，则需要N！/2个密钥，如果使用非对称加密仅需要N个即可（每个人只需要维护自己的公私鈅对，无论和多少人通信）。

3）基于非对称加密发展起来的数字签名技术（见下文详述）从数学意义上解决了自证身份问题，使得信息接收者可以确认消息发送方身份信息且不可更改。

4）密码学问题对数学问题的依赖空前提高。之前看似无用的甚至古老的数学难题比如数论、离散对数等在此大放异彩，颇有些笑来老师在《把时间当做朋友》里面表达的"技不压身"的现实映射。

## 2.散列（哈希）算法

我们使用各种云盘、虚拟存储空间应用的时候一定都有类似的体会，上传一个明明很大的文件，可是速度却非常快，而有时上传一个小得多的文件却似乎进度条要走很久。这种现象的可能的一个秘密就是接下来要讲的散列算法（也叫数据摘要或者哈希算法，哈希是HASH的音译）。

实际上，云盘类产品对相同文件只会保留一份真实的存储，多个使用相同文件的用户只需“索引”到该存储位置即可。判断两个文件是否相同用到的就是散列算法：

散列算法将任意长度的二进制值映射为较短的固定长度的二进制值，这个小的二进制值称为哈希值。哈希值是一段数据唯一且极其紧凑的数值表示形式。如果散列一段明文而且哪怕只更改该段落的一个字母，随后的散列值都将产生不同的值。要找到散列为同一个值的两个不同的输入，在计算上是不可能的，所以数据的散列值可以检验数据的完整性。一般用于快速查找和加密算法。

本质上，散列算法的目的不是为了“加密”而是为了抽取“数据特征”，你也可以把给定数据的散列值理解为该数据的“指纹信息”，一个可靠的散列算法F需要满足：

1. 对于给定的数据M，很容易算出哈希值X=F（M）；

2. 根据X无法算出M；

3. 很难找到M和N令F（M）=F（N）。

我来解释一下：

对于第一点，通常哈希值的长度是固定的（为什么固定，一个很重要的原因是便于构造通信包结构，散列的应用场景通常是不可靠的网络传输而非本地存储），比如比特币使用的SHA256摘要算法对任意长度的输入给出的是256bit的输出。

对于第二点，散列算法是不可逆的，甚至是以丢失部分原始信息作为代价，因此我个人认为散列加密算法的提法是有问题的，加密意味着可以解密，单纯的扰乱称为加密并不合适。也许所谓的散列加密是和散列校验对应的，也就说两种应用的场景不同，散列加密的核心是扰乱，散列校验的核心是校验。

对于第三点，这是散列函数真正的挑战。假设找到了一对（M,N）使得等式成立就成为该算法找到了一次碰撞，对散列算法健壮性的分析就是分析其抗强碰撞能力。可以理解的是，碰撞的出现将使得散列算法本身存在的意义消失了，因为发现了不同的人拥有相同的指纹。

比特币使用的散列算法是SHA256，他是安全散列算法SHA（Secure Hash Algorithm）系列算法的一种（另外还有SHA-1、SHA-224、SHA-384 和 SHA-512 等变体），SHA是美国国家安全局 （NSA） 设计，美国国家标准与技术研究院（NIST） 发布的，主要适用于数字签名标准（DigitalSignature Standard DSS）里面定义的数字签名算法（Digital Signature Algorithm DSA）。可以到[这里](https://link.jianshu.com?t=http:/8btc.com/article-136-1.html)详细了解。

最后，我要解释一下二次散列问题，看过比特币协议的读者可能注意到工作量证明和密钥编码过程中多次使用了二次哈希，如SHA256（SHA256（k））或者RIPEMD160(SHA256(K))，这种方式带来的好处是增加了工作量或者在不清楚协议的情况下增加破解难度，从安全性角度并没有显著增加（如果是暴力穷举破解的话，需要增加一倍破解时间）。真正的二次哈希是基于加盐的哈希，什么意思呢？对于特定的待散列数据和特定的散列算法，可以知道散列值是一定的，这种情况下如果用散列保护敏感数据（理论上是不合适的，但是国内外存在大量的使用散列来保护用户密码的情况），那就很容易使用字典攻击反向推算，比如我计算123456的SHA256值就可以反推了，解决办法就是先做一次散列，再加一次随机数据以后再做一次散列，随机数据就是所谓的盐。

## 3.数字签名

有了非对称加密和散列算法的准备，我们现在可以进一步认识数字签名了。顾名思义，数字签名就是在数字世界里用于身份辨识的一种解决方案。不需要骑缝章，骑缝签名，也不需要笔迹专家，数字签名即具有不可抵赖性。

简单地说,所谓数字签名就是附加在数据单元上的一些数据,或是对数据单元所作的密码变换。这种数据或变换允许数据单元的接收者用以确认数据单元的来源和数据单元的完整性并保护数据,防止被人(例如接收者)进行伪造。它是对电子形式的消息进行签名的一种方法,一个签名消息能在一个通信网络中传输。基于公钥密码体制和私钥密码体制都可以获得数字签名,主要是基于公钥密码体制的数字签名。

上述解释稍显学术，我们换种说法表述：所谓数字签名就是签名人用自己的私钥对待签名数据的摘要进行加密得到的值就是签名值。签名者发送数据签名时需要把待签名数据和签名值一起发送给对方。

注意，签名对象并非待签名数据而是待签名数据的摘要，为什么呢？因为非对称加密的速度通常都比较慢，直接对原始数据私钥加密是很慢的而且也没有必要。

如何验证签名呢？接收方首先使用签名者的公钥对签名值解密即可得到摘要值，然后使用约定的算法对待签名数据进行散列运算后和解密得到的摘要值进行比较即可验证。

[这里](https://link.jianshu.com?t=http:/www.youdzone.com/signature.html)有一个图形化的数字签名过程有助于理解数字签名的整个过程。

从以上数字签名的整个过程描述来看，数字签名的核心在于签名，在于证明这份数据是签名者发出的、不可抵赖的。待签名数据本身的保密不是数字签名方案要考虑的问题。

## 4.可读性编码

严格来讲，编码部分不是密码学理论的核心内容，不过为了便于下文能说清楚，我就把可读性编码放到这里简单说说。

可读性编码很好理解，就是不改变信息内容仅改变内容的表现形式，部分编码方式还加入了容错校验功能，通常是为了保证更好的通信传输。

比如二进制的1111对应十进制的15这就是一次编码。是用十进制对二进制进行编码，现在有个问题，我拿到一个编码后的数据，如何知道该数据是使用了哪种形式的编码呢？这个是通过前缀来实现的，例如，比特币地址的前缀是0（十六进制是0x00），而对私钥编码时前缀是128（十六进制是0x80）。