区块链技术（www.ContentBC.com)

1. 哈希算法
2. 非对称加密算法
3. 数字签名
4. Base64字符编码解码

一、哈希算法

（一）、Hash的定义

1、hash的单词意思

- [hæʃ]
- n.剁碎的食物；蒸煮肉丁
- vt.把..弄乱；切碎；

2、hash （哈希或散列）算法是信息技术领域非常基础也非常重要的技术。它能任意长度的二进制值（明文）映射为较短的固定长度的二进制值（Hash 值），并且不同的明文很难映射为相同的 Hash 值。hash 值在应用中又被称为指纹（fingerprint）、摘要（digest）。

- 例如计算一段话 "hello blockchain" 的 MD5 hash 值为：78e6a8bcdef7c4a254c16054b082c783
- 这意味着我们只要对某文件进行 MD5 Hash 计算，得到结果为 78e6a8bcdef7c4a254c16054b082c783，这就说明文件内容极大概率上就是 "hello blockchain"。可见，Hash 的核心思想十分类似于基于内容的编址或命名。

3、一个优秀的 hash 算法，将能实现：

- 正向快速：给定明文和 hash 算法，在有限时间和有限资源内能计算出 hash 值。
- 逆向困难：给定（若干） hash 值，在有限时间内很难（基本不可能）逆推出明文。
- 输入敏感：原始输入信息修改一点信息，产生的 hash 值看起来应该都有很大不同。
- 抗冲突：
  - 对不同的关键字可能得到同一散列地址，或者说两段内容不同的明文，它们的 hash 值可能一致。这种现象称冲突或者碰撞，具有相同函数值的关键字对该散列函数来说称做同义词。
  - 抗冲突又称为“抗碰撞性”或冲突避免，哈希函数抗冲突就是不同的输入不能产生相同的输出。
  - 抗冲突并不是不会有冲突，只不过找到有冲突的两个输入的代价非常大。就好像暴力破解一个有效期为20年的密码，整个破解过程或许长达30年。这样虽然最后密码被破解了，但是也失去意义了。

（二）、流行的算法

- 目前流行的 Hash 算法包括 MD5、SHA-1 和 SHA-2系列（SHA-224、SHA-256、SHA-384，和 SHA-512 ）。

1、MD4

- MD4（RFC 1320）是 MIT (Massachusetts Institute of Technology，麻州理工大学) 的 Ronald L. Rivest（荣获2002年图灵奖）在 1990 年设计的，MD 是 Message Digest 的缩写。其输出为 128 位。

2、MD5

- MD5（RFC 1321）是 Rivest 于1991年对 MD4 的改进版本。其输出是 128 位。MD5 比 MD4 复杂，并且计算速度要慢一点，更安全一些。MD5 已被证明不具备“强抗碰撞性”。
- MD5 是一个经典的 hash 算法，其和 SHA-1 算法都已被证明安全性不足应用于商业场景。

2、MD5

- MD5（RFC 1321）是 Rivest 于1991年对 MD4 的改进版本。其输出是 128 位。MD5 比 MD4 复杂，并且计算速度要慢一点，更安全一些。MD5 已被证明不具备“强抗碰撞性”。
- MD5 是一个经典的 hash 算法，其和 SHA-1 算法都已被证明安全性不足应用于商业场景。

3、SHA

- SHA （Secure Hash Algorithm，[sɪˈkjʊə(r)] [ˈælgərɪðəm]，加密哈希算法）是一个 Hash 函数族，由 NIST（National Institute of Standards and Technology，美国国家标准技术研究所）于 1993 年发布第一个算法。目前知名的 SHA-1 在 1995 年面世，它的输出为长度 160 位的 hash 值，因此抗穷举性更好。SHA-1 设计时基于和 MD4 相同原理，SHA-1 已被证明不具备“强抗碰撞性”。
- 为了提高安全性，NIST 还设计出了 SHA-224、SHA-256、SHA-384，和 SHA-512 算法（统称为 SHA-2），跟 SHA-1 算法原理类似。
- 目前，一般认为 MD5 和 SHA1 已经不够安全，推荐至少使用 SHA2-256 算法。

4、RIPEMD-160(RACE Integrity Primitives Evaluation Message Digest 160， RACE完整的原始评估信息摘要）

- 是一个160位的加密哈希函数，旨在替代128位哈希函数MD4、MD5。

5、各种hash算法后的长度

- md5(1) ，128位长度
  - c4ca4238a0b923820dcc509a6f75849b //32位16进制数字
- RIPEMD-160[1] ，160位长度
  - c47907abd2a80492ca9388b05a0e382518ff3960 //40位16进制数字
- SHA1(1)，160位长度
  - 356a192b7913b04c54574d18c28d46e6395428ab //40位16进制数字
- SHA256[1] ，256位长度
  - 6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b //64位16进制数字
- SHA512[1]，512位长度
  - 4dff4ea340f0a823f15d3f4f01ab62eae0e5da579ccb851f8db9dfe84c58b2b37b89903a740e1ee172da793a6e79d560e5f7f9bd058a12a280433ed6fa46510a //128位16进制数字

〔三〕、Hash与加密解密的区别

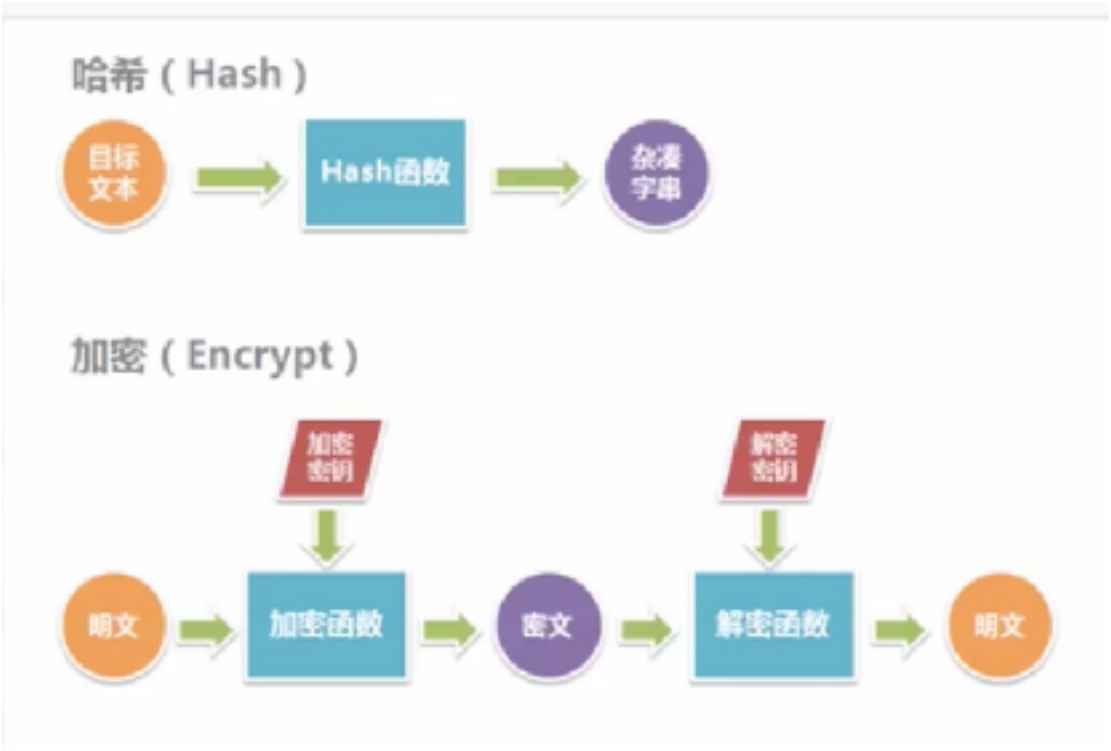
1、哈希（Hash）与加密（Encrypt）的区别

- 这两个概念不是很清晰，容易混淆两者。而正确区别两者是正确选择和使用哈希与加密的基础。
- 哈希（Hash）是将目标文本转换成具有相同长度的、不可逆的杂凑字符串（或叫做消息摘要），而加密（Encrypt）是将目标文本转换成具有不同长度的、可逆的密文。
  - 哈希算法往往被设计成生成具有相同长度的文本，而加密算法生成的文本长度与明文本身的长度有关。
    - 哈希的结果具有相同的长度，而加密的结果则长度不同。实际上，如果使用相同的哈希算法，不论你的输入有多么长，得到的结果长度是一个常数，而加密算法往往与明文的长度成正比。
  - 哈希算法是不可逆的，而加密算法是可逆的，
    - 哈希不是一一映射，是不可逆的。而加密算法是一一映射，是可逆的。
    - 哈希是不可能可逆的，因为如果可逆，那么哈希就是世界上最强悍的压缩方式了——能将任意大小的文件压缩成固定大小。

（三）、Hash与加密解密的区别

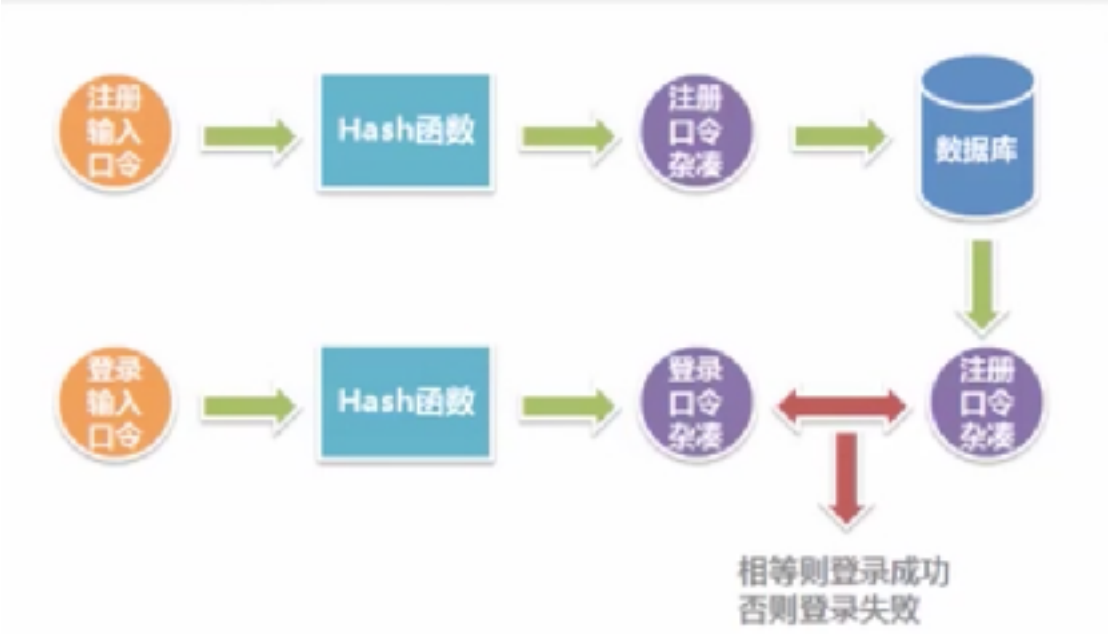
1、哈希（Hash）与加密（Encrypt）的区别

- 这两个概念不是很清晰，容易混淆两者。而正确区别两者是正确选择和使用哈希与加密的基础。
- 哈希（Hash）是将目标文本转换成具有相同长度的、不可逆的杂凑字符串（或叫做消息摘要），而加密（Encrypt）是将目标文本转换成具有不同长度的、可逆的密文。
  - 哈希算法往往被设计成生成具有相同长度的文本，而加密算法生成的文本长度与明文本身的长度有关。
    - 哈希的结果具有相同的长度，而加密的结果则长度不同。实际上，如果使用相同的哈希算法，不论你的输入有多么长，得到的结果长度是一个常数，而加密算法往往与明文的长度成正比。
  - 哈希算法是不可逆的，而加密算法是可逆的，
    - 哈希不是一一映射，是不可逆的。而加密算法是一一映射，是可逆的。
    - 哈希是不可能可逆的，因为如果可逆，那么哈希就是世界上最强悍的压缩方式了——能将任意大小的文件压缩成固定大小。



2、哈希（Hash）与加密（Encrypt）的选择

- 要实现数据保护，可以选择使用哈希或加密两种方式。那么在什么时候该选择哈希、什么时候该选择加密呢？
- 基本原则是：
  - 如果被保护数据仅仅用作比较验证，在以后不需要还原成明文形式，则使用哈希；
  - 如果被保护数据在以后需要被还原成明文，则需要使用加密。
  - 例如，你正在做一个系统，你打算当用户忘记自己的登录口令时，重置此用户口令为一个随机口令，而后将此随机口令发给用户，让用户下次使用此口令登录，则适合使用哈希。实际上很多网站都是这么做的，想想你以前登录过的很多网站，是不是当你忘记口令的时候，网站并不是将你忘记的口令发送给你，而是发送给你一个新的、随机的口令，然后让你用这个新口令登录。这是因为你在注册时输入的口令被哈希后存储在数据库里，而哈希算法不可逆，所以即使是网站管理员也不可能通过哈希结果复原你的口令，而只能重置口令。
  - 相反，如果你做的系统要求在用户忘记口令的时候必须将原口令发送给用户，而不是重置其口令，则必须选择加密而不是哈希。



3、对简单哈希（I-hash）的攻击

- 对于哈希的攻击，主要有寻找碰撞法和穷举法。
- 寻找碰撞法
  - 目前对于MD5和SHA1并不存在有效地寻找碰撞方法。
  - 我国杰出的数学家王小云教授曾经在国际密码学会议上发布了对于MD5和SHA1的碰撞寻找改进算法，但这种方法和“破解”相去甚远。该理论目前仅具有数学上的意义，她将破解MD5的预期步骤降低了好几个数量级，但对于实际应用来说仍然是一个天文数字。

MD4	碰撞攻击	2^{22}	1996
		2^{8}	2005
		2	2007
	第二原像攻击(弱消息)	2^{56}	2005
	原像攻击	2^{102}	2007
MD5	碰撞攻击	2^{37}	2005
		2^{16}	2009
	原像攻击	$2^{123.4}$	2009
RIPEMD	碰撞攻击	2^{18}	2005
SHA-0	碰撞攻击	2^{61}	1997
		2^{61}	1998
		2^{39}	2005
	52轮原像攻击	$2^{156.6}$	2009
SHA-1	碰撞攻击	2^{69}	2005
		2^{61}	2006
	48轮原像攻击	$2^{159.3}$	2009
HAVAL-128	碰撞攻击	2^{7}	2005
3-pass HAVAL	碰撞攻击	2^{29}	2003
	第二原像攻击	2^{114}	2006
	原像攻击	2^{224}	2008
4-pass HAVAL	碰撞攻击	2^{43}	2006
5-pass HAVAL	碰撞攻击	2^{123}	2006

- 穷举法（或暴力破解法）
  - 通俗来说，就是在一个范围内，如从000000到999999，将其中所有值一个一个用哈希算法哈希，然后将结果和杂凑串比较，如果相同，则这个值就一定是源字串或源字串的一个碰撞，于是就可以用这个值非法登录了。
  - 穷举法看似笨拙，但目前几乎所有的MD5破解机或MD5在线破解都是用这种穷举法。
  - 纠其缘由，就是相当一部分口令是非常简单的，如“123456”或“000000”。穷举法是否能成功很大程度上取决于口令的复杂性。因为穷举法扫描的区间往往是单字符集、规则的区间，或者由字典数据进行组合，因此，如果使用复杂的口令，例如“!@#$%^&*()”这种口令，穷举法就很难奏效了。

二、非对称加密算法

二、非对称加密算法

（一）、发展史

1、该思想最早由瑞夫·墨克（Ralph C. Merkle）在1974年提出，之后在1975年，惠特菲尔德·迪菲（Whitfield Diffe）与马丁·赫尔曼（Martin Heiman）两位斯坦福的学者以单向函数为基础，创建了DH密码交换算法。

2、RSA公钥加密算法是1977年美国麻省理工学院的Rivest、Shamir和Adleman开发的。RSA取名来自开发他们三者的名字。RSA是目前最有影响力的公钥加密算法，它能够抵抗到目前为止已知的所有密码攻击，已被ISO推荐为公钥数据加密标准。

3、其他常见的公钥加密算法有：ElGamal、背包算法、Rabin（RSA的特例）、椭圆曲线加密算法（Elliptic Curve Cryptography, ECC）。

（二）、非对称加密（Asymmetric Cryptography）

1、非对称加密又叫做公开密钥加密（Public key cryptography）或公钥加密。

2、公钥加密，需要两个密钥，一个是公开密钥，另一个是私有密钥；一个用作加密的时候，另一个则用作解密。

- 使用其中一个密钥把明文加密后所得的密文，只能用相对应的另一个密钥才能解密得到原本的明文；甚至连最初用来加密的密钥也不能用作解密。由于加密和解密需要两个不同的密钥，故被称为非对称加密。
- 不同于加密和解密都使用同一个密钥的对称加密。虽然两个密钥在数学上相关，但如果知道了其中一个，并不能凭此计算出另外一个。因此其中一个可以公开，称为公钥，任意向外发布；不公开的密钥为私钥，必须由用户自行严格秘密保管，绝不通过任何途径向任何人提供。
- 公钥用来加密消息、验证签名；私钥用来解密信息和进行签名。加密消息的密钥是不能解密消息的。

（三）、对称加密与非对称加密的区别

- 对称加密算法（data encryption algorithm, DEA），也叫做私钥加密算法。是加密和解密使用相同密钥的加密算法，也叫做单密钥算法，传统密钥算法。

算法类型	特点	优势	缺陷	代表算法
对称加密	加解密密钥相同或可推算	计算效率高，加密强度高	需提前共享密钥；易泄露	DES、3DES、AES、IDEA
非对称加密	加解密密钥不相关	无需提前共享密钥	计算效率低，中间人攻击可能性低	RSA、ElGamal、椭圆曲线系列算法

（四）、RSA原理

1、RSA算法基于一个十分简单的数论事实：

- 将两个大素数相乘十分容易
- 想要对其乘积进行因式分解极其困难，因此可以将乘积公开作为加密密钥。

2、密钥对的生成步骤

1. 随机选择两个不相等的质数p和q
2. 计算p和q的乘积N
3. 计算p-1和q-1的乘积φ(N)
4. 随机选个整数e，e与m要互质，且0<e<φ(N)
5. 计算e的模反元素d
6. 公钥是（N,e），私钥是(N,d)

3、加解密步骤

1. 假设一个明文数m（0<=m<N)
2. 对明文m加密成密文c
    a. c=m^e mod N
3. 对密文c解密成明文m
    a. m=c^d mod N

4、举例说明

1. p=11， q=3
2. N=pq = 33
3. φ(N) =(p-1)(1-1)=20
4. 选择20的互质数e=3
5. 计算满足ed=1 mod 20的d，也就是模反元素d=7
6. 公钥为(33,3)，私钥为(33,7)
7. 假设明文m=8， (0<8<33)
8. 密文c=m^e mod N = 8 ^ 3 mod 33 = 512 mod 33 = 17 mod 33，得出c=17
9. 明文m=c^d mod N = 17 ^ 7 mod 33 = 8 mod 33， 得出m=8

三、数字签名

（一）、数字签名的概念

1、所谓数字签名(Digital Signature)（又称公开密钥数字签名、电子签章）

- 是一种类似写在纸上的普通的物理签名，但是使用了公钥加密领域的技术实现，用于鉴别数字信息的方法。一套数字签名通常定义两种互补的运算，一个用于签名，另一个用于验证。

2、数字签名如何工作

- 数字签名由两部分组成：第一部分是使用私钥（签名密钥）从消息（交易）创建签名的算法；第二部分是

（二）、数字签名应该满足的要求

1、签名不可伪造性；

2、签名不可抵赖的（简直通俗易懂~）；

3、签名可信性，签名的识别和应用相对容易，任何人都可以验证签名的有效性；

4、签名是不可复制的，签名与原文是不可分割的整体；

5、签名消息不可篡改，因为任意比特数据被篡改，其签名便被随之改变，那么任何人可以验证而拒绝接受此签名。

（三）、比特币系统中的数字签名

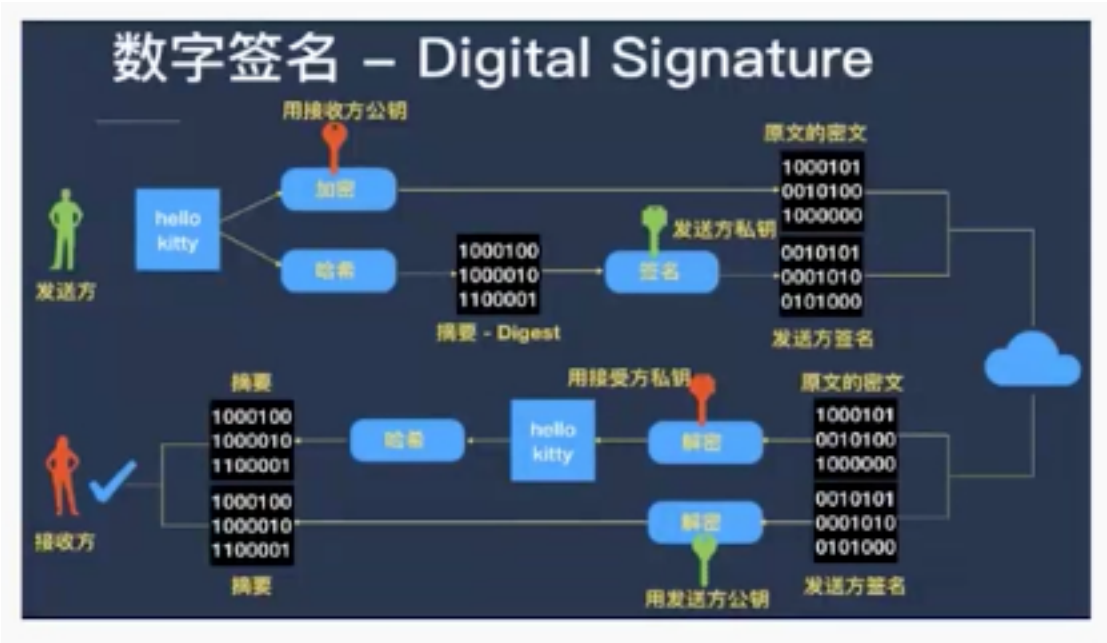
1、只有转账人才能生成的一段防伪造的字符串，通过验证该字符串，一方面证明该交易是转出方本人发起的，另一方面证明交易信息在传输过程中没有被更改。

2、数字签名由：数字摘要和非对称加密技术组成。

- 数字摘要把交易信息hash成固定长度的字符串；
- 再用私钥对hash后的交易信息进行加密形成数字签名。

3、交易中，需要将完整的交易信息和数字签名一起广播给矿工。

- 矿工节点用转账人公钥对签名验证，验证成功说明该交易确实是转账人发起的；
- 矿工节点将交易信息进行hash后与签名中的交易信息摘要进行比对，如果一致，则表明交易信息在传输过程中没有被篡改。


数字签名 – Digital Signature
用接收方公钥
加密
hello kitty
哈希
发送方
1000100
1000010
1100001
摘要 - Digest
签名
发送方私钥
原文的密文
1000101
0010100
1000000
0010101
0001010
0101000
发送方签名
摘要
1000100
1000010
1100001
哈希
hello kitty
用接受方私钥
解密
原文的密文
1000101
0010100
1000000
0010101
0001010
0101000
发送方签名
接收方
1000100
1000010
1100001
摘要
解密
用发送方公钥

四、字符编码/解码


、Base64
1、Base64就是一种基于64个可打印字符来表示二进制数据的方法
• Base64使用了26个小写字母、26个大写字母、10个数字以及两个符号（例如“+”和“/”），用于在电子邮件这样的基于文本的媒介中传输二进制数据。
• Base64通常用于编码邮件中的附件。
2、Base64字符集：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/

3、Base64的步骤

1. 将每个字符转成ASCII编码（10进制）
2. 将10进制编码转成2进制编码
3. 将2进制编码按照6位一组进行平分
4. 将6位一组的2进制数高位补零，然后转成10进制数
5. 将10进制数作为索引，从Base64编码表中查找字符
6. 每3个字符的文本将编码为4个字符长度（3*8=4*6）
    a. 若文本为3个字符，则正好编码为4个字符长度；
    b. 若文本为2个字符，则编码为3个字符，由于不足4个字符，则在尾部用一个“=”补齐；
    c. 若文本为1个字符，则编码为2个字符，由于不足4个字符，则在尾部用两个“=”补齐。

文本	M								a								n							
ASCII编码	77								97								110							
二进制位	0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0
索引	19								22								5							
Base64编码	T								W								F							
索引	46																							
Base64编码	u																							

文本（1 Byte）	A																							
二进制位	0	1	0	0	0	0	0	1																
二进制位（补0）	0	1	0	0	0	0	0	1	0	0	0	0												
Base64编码	Q				Q				=				=											
文本（2 Byte）	B								C															
二进制位	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	1			x	x	x	x	x	x
二进制位（补0）	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	x	x	x	x	x	x
Base64编码	Q				k				M				=											

4、Java中的 Base64编码/解码的核心代码

(1)、Base64.getEncoder().encode(bytes);

(2)、Base64.getDecoder().decode(str);

（二）、Base58

1、Base58是一种基于文本的二进制编码格式，用在比特币和其它的加密货币中。这种编码格式不仅实现了数据压缩，保持了易读性，还具有错误诊断功能。

2、Base58是Base64编码格式的子集，同样使用大小写字母和10个数字，但舍弃了一些容易错读和在特定字体中容易混淆的字符。

- Base58不含Base64中的0（数字0）、O（大写字母o）、l（小写字母L）、I（大写字母I），以及“+”和“/”两个字符。目的就是去除容易混淆的字符。
- 简而言之，Base58就是由不包括（0，O，l，I）的大小写字母和数字组成。
- 比特币的Base58字母表：123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz

（三）、Base58Check

1、Base58Check是一种常用在比特币中的Base58编码格式，增加了错误检验码来检查数据在转录中出现的错误。

- 校验码长4个字节，添加到需要编码的数据之后。
- 校验码是从需要编码的数据的哈希值中得到的，所以可以用来检测并避免转录和输入中产生的错误。
- 使用Base58check编码格式时，编码软件会计算原始数据的校验码并和结果数据中自带的校验码进行对比。二者不匹配则表明有错误产生，那么这个Base58Check格式的数据就是无效的。例如，一个错误比特币地址就不会被钱包认为是有效的地址，否则这种错误会造成资金的丢失。

2、为了使用Base58Check编码格式对数据（数字）进行编码，首先我们要对数据添加一个称作"版本字节"的前缀，这个前缀用来明确需要编码的数据的类型。

- 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）。

一、比特币地址的生成步骤

(一)、概述

1、比特币是建立在数学加密学基础上的，中本聪用了椭圆曲线数字签名算法（ECDSA）来产生比特币的私钥和公钥。

- ECDSA算法（Elliptic Curve Digital Signature Algorithm，椭圆曲线数字签名算法），是使用椭圆曲线加密学（ECC，Elliptic Curve Cryptography）对数字签名算法（DSA，Digital Signature Algorithm）的模拟。ECDSA在1998年就被ISO所接受，于1999年成为ANSI标准，并于2000年成为IEEE和NIST标准。
- 具体来说，比特币系统中的椭圆曲线是Certicom推荐的SECP256K1算法。
- Certicom是国际上著名的椭圆曲线密码技术公司。而SECP256K1算法由于其构造的特殊性，优化后性能比其它曲线算法提高30%。

2、由私钥可以计算出公钥，公钥经过一系列数字签名运算会得到比特币地址。

3、因为由公钥可以算出比特币地址，所以有人经常把公钥和比特币地址相混淆。不过可以把比特币地址理解成是另一种格式的公钥。

4、从比特币私钥得到比特币地址需要10个步骤。中间依次会用到SECP256K1、SHA256、RIPEMD160以及BASE58编码。我们平时使用的比特币地址都是经过BASE58编码之后的结果。一般以"1"、"3"开头的34位字符串。（在比特币testnet网络中比特币地址以2开头）

（二）、Base58Check

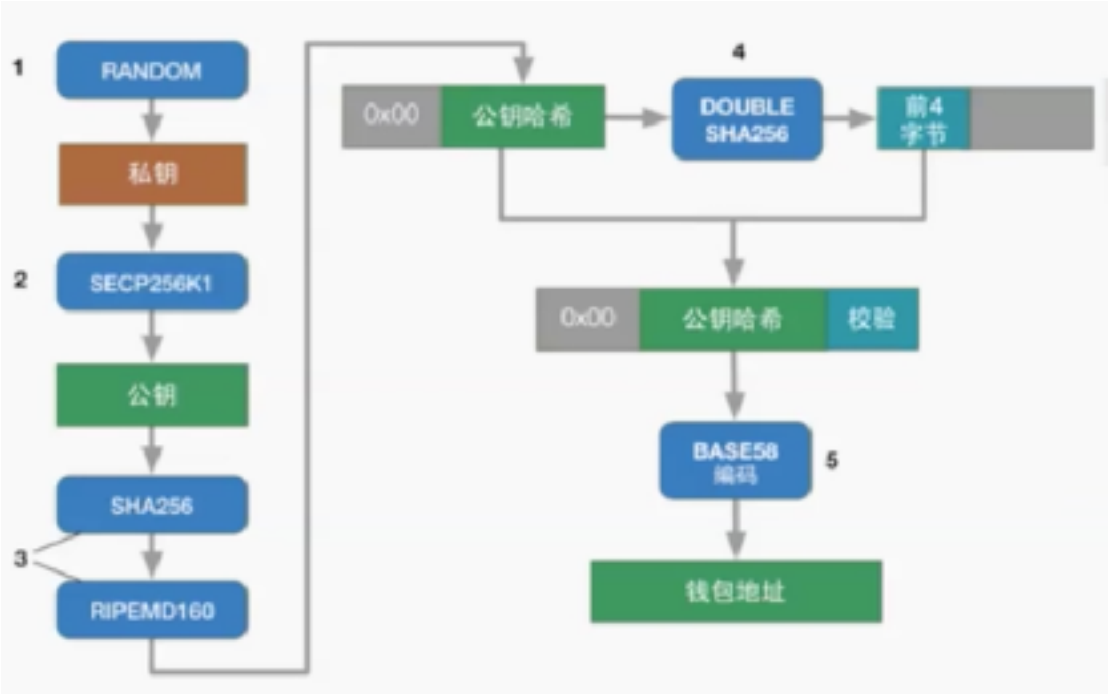
1、Base58Check是一种常用在比特币中的Base58编码格式，增加了错误校验码来检查数据在传输中出现的错误。

2、在生成比特币地址以及生成WIF-compressed格式私钥的过程中都用到了Base58Check。

3、在Base58Check中，对数据添加了一个称作"版本字节"的前缀，这个前缀用来明确需要编

4、Base58Check的作用

- 既然有了Base58编码，已经不会搞错0和O，l和I，也把大整数转换成了可读字符串
  Base58Check这个环节呢？
- 假设一种情况，你在程序中输入一个Base58编码的地址，尽管你已经不会搞错0和O，
  小心输错一个字符，或者少写多写一个字符，会咋样？你可能会说，没啥大不了的，错个字符而已，这不是很常见嘛，重新输入不就可以了吗？但是当用户给一个比特币地址转账，如果输入错误，那么对方就不会收到资金，更关键的是该笔资金发给了一个根本不存在的比特币地址，那么这笔资金也就永远不可能被交易，也就是说比特币丢失了。
- 使用Base58check编码格式时，程序会计算原始数据的校验码并和自带的校验码进行对比，二者若不匹配则表明有错误产生。
- 实际上，在比特币交易中，都会校验比特币地址是否合法，如果经过Base58Check的
  包程序判定是无效的，当然会阻止交易继续进行，就避免了资金损失。



（二）、比特币地址生成步骤

1、第一步，随机选取一个32字节的数，大小介于1～0xFFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFE BAAE DCE6 AF48 A03B BFD2 5E8C D036 4141之间，作为私钥。私钥为64位16进制数字。

A376D3867EA8F2DDB6D80B0A73D1C67FE51B669D780EE2549490D02983624344C

2、第二步，使用椭圆曲线加密算法（ECDSA-secp256k1）计算私钥所对应的非压缩公钥。（共65字节，1字节0x04，32字节为x坐标，32字节为y坐标），公钥为130位16进制数字。

04

1BF6C2F0F5DD3F669C7F2A942B97C38AB797EB7E73BA80C88371B77EA5C2ED40

8AE99486B8D2141FEED69590509BD5FE8B939FEEF6A0BAE0D11D2412D878B2B

3、第三步，计算公钥的SHA-256哈希值

F2487CF9A2378C2F9C18E3EB4B4C8A3CA16E9F6C076560CAD48BC8D783830D89

4、第四步，取上一步结果，计算RIPEMD-160哈希值

9353BD455CDBCAEC7BDBA6AE7224CBF3D9B6D50C

5、第五步，取上一步结果，前面加入地址版本号（比特币主网版本号"0x00"）

009353BD455CDBCAEC7BDBA6AE7224CBF3D9B6D50C

6、第六步，取上一步结果，计算SHA-256哈希值

907202A7758189FFA8BD65C2FE0C4DEA76BBBE6766E7758B893DD3E97AAAE613

7、第七步，取上一步结果，再计算一下SHA-256哈希值

D0295602554E9234DDCEB016F9D6D4E48015548243 4DEBD29797297B7A3E7BA22

8、第八步，取上一步结果的前4个字节（8位十六进制）作为校验码

D0295602

9、第九步，将校验码加在第五步的结果后面，这就形成了比特币地址的16进制格式。

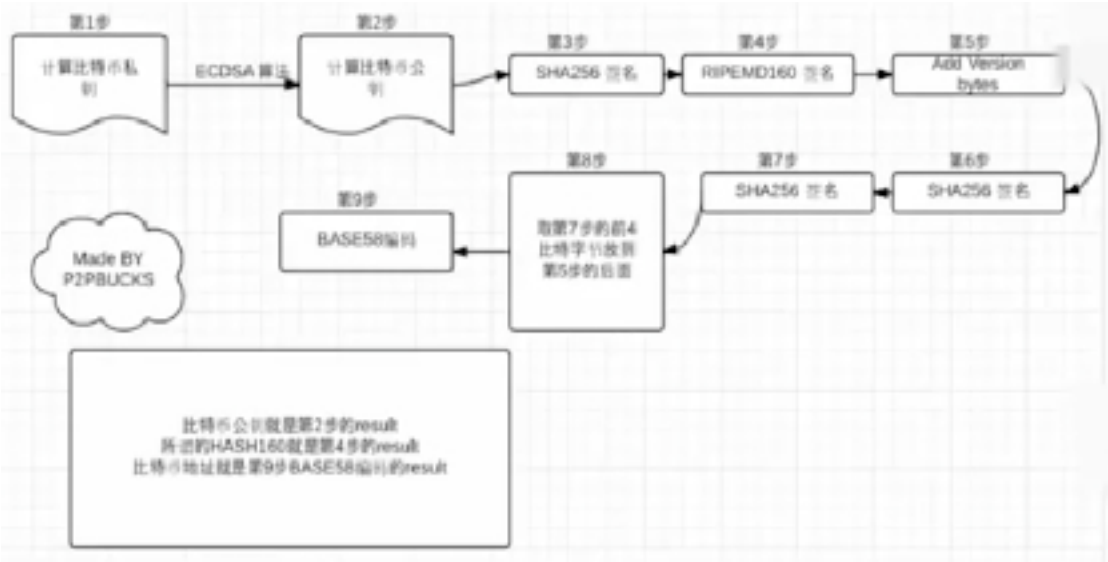
009353BD455CDBCAEC7BDBA6AE7224CBF3D9B6D50CD0295602

10、第十步，用Base58编码（这是最常使用的比特币地址格式）。

1ERzeWbxZsZYu7JzR4H2JYx4gNK5Y7Abb

我们经常说的比特币公钥就是指的图中第二步所产生的结果，而HASH160指的是第四步RIPEMD160签名所产生的结果，由于RIPEMD也是一种HASH算法所以就统称为HASH160。

【备注：生成比特币地址的步骤详见网站——http://gobittest.appspot.com/Address】



二、生成WIF或WIF-compressed格式私钥的步骤

1、第一步，随机选取一个32字节的数作为私钥。私钥为64位16进制数字。

例如：18E14A7B6A307F426A94F8114701E7C8E774E7F9A47E2C2035DB29A206321725

2、第二步，取上一步结果，前面加入版本号（版本号"0x80"），末尾添加压缩标识符（"0x01"）

- 根据是否添加压缩标识符，将私钥格式分为WIF和WIF-compressed格式，最终经过Base58编码后分别为51位长度和52位长度。
- WIF格式以“5”开头，WIF-compressed格式以“L”或“K”开头。

801 8E14A7B6A307F426A94F8114701E7C8E774E7F9A47E2C2035DB29A20632172501

3、第三步，取上一步结果，计算两次SHA-256后的哈希值

281938ff642c734eb0ffec8a394c5490dcc5d67a79257832eb80118570188eaef

4、第四步，取上一步结果的前4个字节（8位十六进制）作为校验码

281938ff

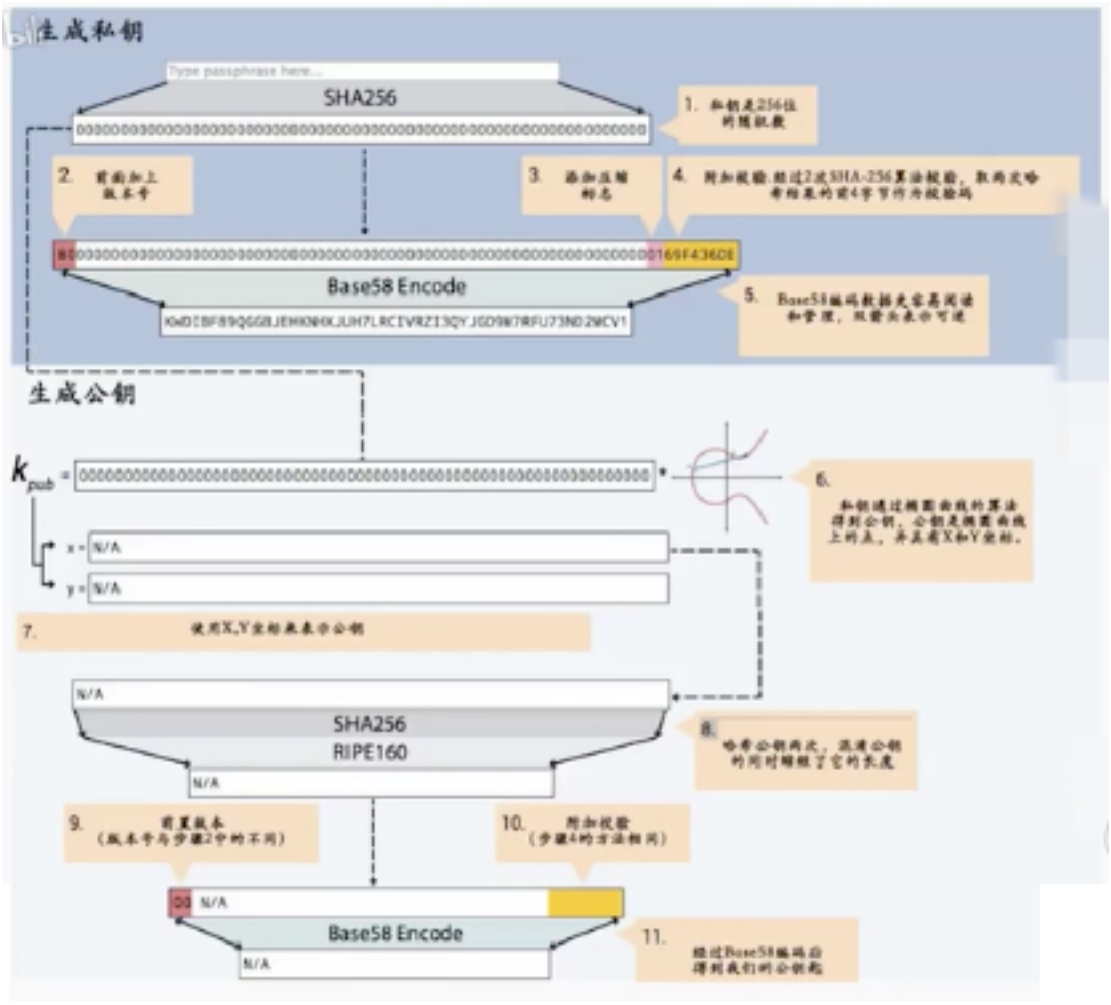
5、第五步，把校验码加在第二步的结果后面，形成增加校验码之后的16进制格式

801 8E14A7B6A307F426A94F8114701E7C8E774E7F9A47E2C2035DB29A20632172501281938ff

6、第六步，用base58表示法变换一下地址（这就是最常见的比特币地址形态）。

Kx45GeUBSMPReYQwgXiKhG9FzNXrnCeutJp4yjTd5kKxCitadm3C

【备注：生成私钥的步骤详见网站——http://gobittest.appspot.com/PrivateKey】


生成私钥
Type passphrase here...
SHA256
1.
2.
3.
4.
Base58 Encode
5.
生成公钥
6.
x = N/A
y = N/A
7.
N/A
SHA256
RIPE160
8.
N/A
9.
10.
N/A
Base58 Encode
11.
N/A


TP's Go Bitcoin Tests – Addresses
<< Go back
Random
2 – Private ECDSA Key
Send
1 – Public ECDSA Key
Send
P – SHA-256 hash of 1
3 – RIPEMD-160 Hash of 2
Send
4 – Adding network bytes to 3
Send
5 – SHA-256 hash of 4
6 – SHA-256 hash of 5

三、Base58及Secp256k1椭圆曲线算法的核心代码

（一）、封装Base58编码工具类（Base58Util）

1、Base58编码

- public static String encode(byte[] input)

2、Base58解码

- public static String decode(String input)

（二）、封装Secp256k1椭圆曲线算法工具类（Secp256k1Util）

1、利用私钥生成对应的公钥

- public static byte[] generatePublicKey(byte[] privateKey)

2、随机生成私钥

- public static byte[] generatePrivateKey()

四、封装比特币地址工具类（BitcoinAddressUtil）

1、通过64位长度的私钥生成Base58编码的比特币地址

- 参数：64位16进制私钥字符串
- public static String generateAddressByHexPrivKey(String hexPrivKey)

2、通过64位长度的私钥生成Base58编码的比特币地址

- 参数：64位16进制私钥字符串
- public static String generateAddressByB58PrivKey(String base58PrivKey)

3、通过公钥生成Base58编码的比特币地址

- 参数：130位16进制公钥字符串
- public static String generateAddressByPubKey(String hexPubKey)

4、将64位16进制私钥数值转成Base58编码

- 参数：64位16进制私钥字符串
- public static String generatePrivKeyB58(String hexStr)

5、将Base58编码的私钥返回到16进制字符串

- 参数：Base58编码格式的私钥字符串
- public static String getHexPrivKey(String base58privKey)

6、验证Base58编码的私钥（WIF-compressed格式）是否有效

- 参数：WIF-compressed格式的私钥字符串（以K或L开头的51位长度字符串）
- public static boolean validateWifPrivKey(String b58privKey)

在线工具：

http://tool.oschina.net/encrypt?type=2

http://tools.jb51.net/password/hash_md5_sha

https://tool.lu/hexconvert