247

比特币客户端：

<https://bitcoin.org/en/download>

比特币客户端执行程序说明

bin文件夹下：

bitcoin-qt.exe 比特币的核心节点以及一个钱包的前端功能

比特币测试网络

可以通过配置文件来进行配置。比特币的

配置文件名为bitcoin.conf，可以在数据目录也就是钱包数据文件

wallet.dat所在目录下创建一个文本文件，命名为bitcoin.conf即可，这就

是bitcoin-qt默认读取的配置文件了。接下来我们就来配置一下以进入测

试网络，只需在bitcoin.conf中写入如下配置项：

testnet=1

通过命令行启动测试网络：

bitcoin-qt -testnet

更换数据目录：

bitcoin-qt -datadir="D:\mybitcoin\_data"

重新指定配置文件：

bitcoin-qt -conf="c:\mybitcoin.conf"

bitcoind.exe 这个其实就可以看作不带界面的bitcoin-qt.exe

bitcoin-qt与bitcoind是互相兼容的，有同样的命令行参数，读取相同

格式的配置文件，也读写相同的数据文件，使用的时候，这两个程序根

据需要启动一个即可，同时启动也不会出错

bitcoind默认读取的配置文件，在不同操作系统下路径也不尽相

同，如下所示：

·Windows：%APPDATA%\Bitcoin\

·OS X：$HOME/Library/Application Support/Bitcoin/

·Linux：$HOME/.bitcoin/

bitcoind -datadir="c:\bitcoin\_data" -conf="C:\mybitcoin.conf"

bitcoind启动后可以通过bitcoin-cli进行访问

·8333，用于与其他节点进行通信的监听端口，节点之间的通信是

通过bitcoin protocol进行的，通过这个端口才能进入比特币的P2P网络。

·8332，这是提供JSON-RPC通信的端口，通过这个端口可以访问节

点的数据。

·如果是测试网络，分别是18333和18332。

以上端口是可以另外指定的，通过参数-port与-rpcport参数可以分别

重新指定。

bitcoin-cli.exe

命令行客户端 用来通过RPC方式访问bitcoind的RPC服务

bitcoin-cli -version

bitcoin-cli getinfo

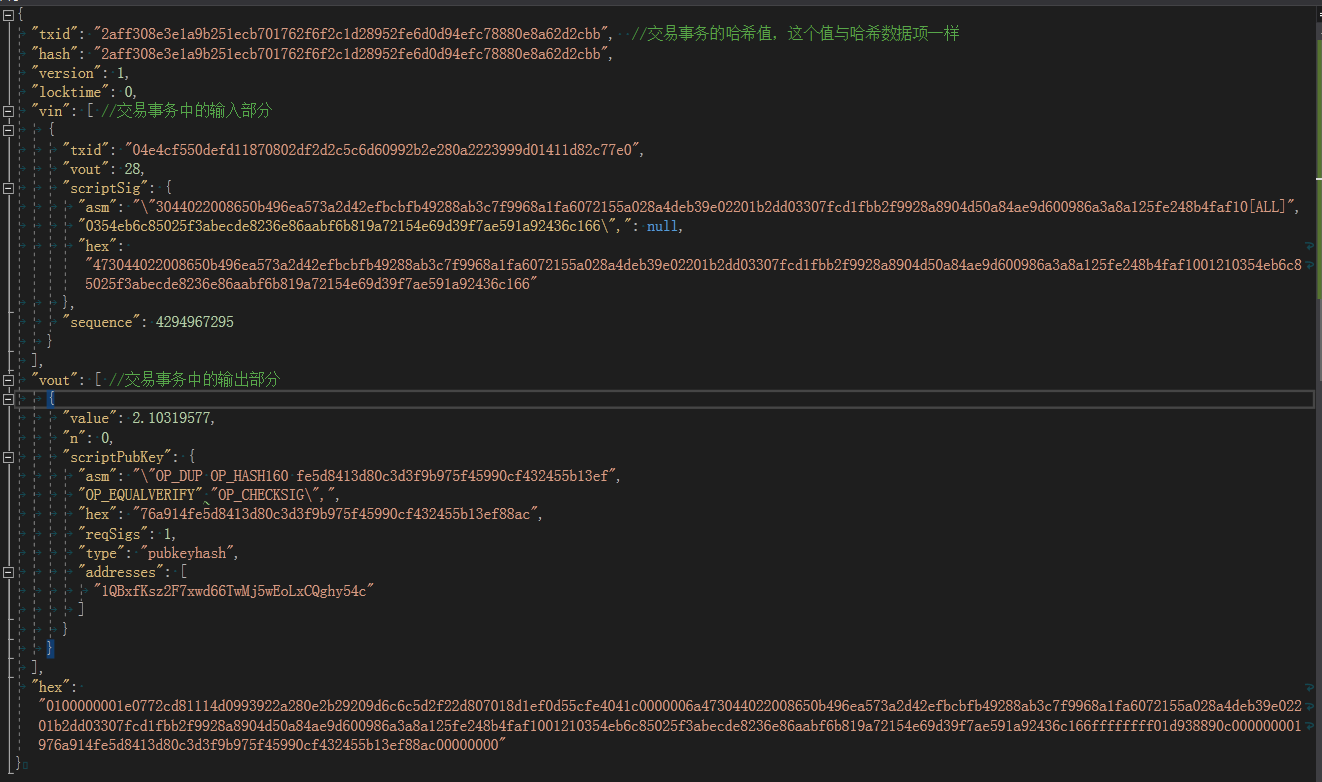
bitcoin-cli stop

bitcoind -datadir="c:\bitcoin\_data" -conf="C:\bitcoin.conf"

bitcoin-cli -datadir="c:\bitcoin\_data" -conf="c:\bitcoin.conf" getinfo

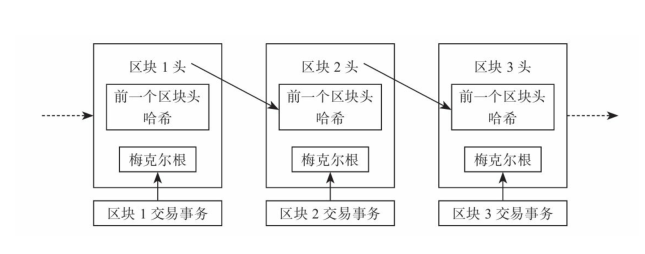
bticoin-tx.exe

bitcoin-tx -json 二进制数据



test\_bitcoin.exe

比特币中区块链数据的组成示意图：



共识算法：

PoW（Proof of Work，工作量证明）、

难度值压缩处理规则，最高位字节用来存储难度值的有效字节数。后三个字节标示目标值有效位最高三字节值

目标值=系数\*2^（8\*（指数-3））

挖矿程序从内存池中获取用来打包区块的交易数据，进行sha256运算，对参数进行两

次SHA256计算，如果计算出来的值小于那个TARGET（也就是难度目标值），那就算是挖矿成功了。挖矿就是重复计算区块头的哈希值，不断修改该参数，直到与难度目标值匹配的一个过程。

那么这里奖励在哪呢？这个奖励其实是作为一条交易事务包含在区块的交易事务中的，相当于系统给矿工转账了一笔比特币，这种交易事务由于特殊性，通常称为coinbase交易



PoS（Proof of Stake，权益证

明）、DPoS（Delegate Proof of Stake，委托权益证明）、

PBFT（Practical Byzantine Fault Tolerance，实用拜占庭容错算法）等，

各种不同的算法

新难度值的计算公式是这样的：新难度值=当前难度值×（最近的

2016个区块的实际出块时间/20160分钟）。

网络路由:

节点发现协议

连接时，会发送一条包含认证内容的消息进行“握手”确认，比特币网络中是靠彼此共享节点信息来寻找其他节点的，当一个节点建立与其他节点的连接后，会发送一条包含自身IP地址的消息给相邻的节点，而邻居收到后会再次发送给自己的其他邻居，当然节点也不是只能被动地等别人来告诉自己，也可以自己发送请求给其他节点索取这些地址信息，如果与发现的节点之间能够成功连接，那么就会被记录下来，下次启动时就会自动去寻找上次成功连接过的节点。

区块链1.0架构



区块链2.0架构



区块链3.0架构



比特币的发行：挖矿

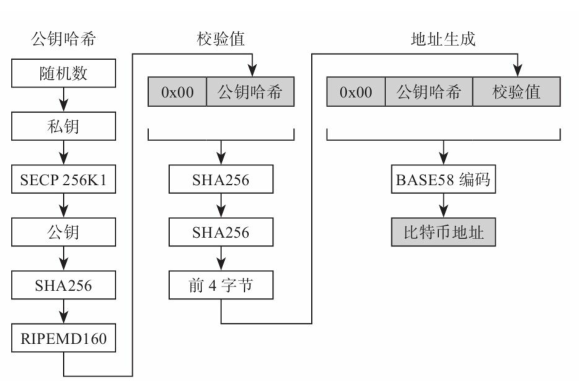
抢夺区块打包权

·验证交易事务

奖励发行新币

·广播新区块

比特币地址的生成过程



1）首先使用随机数发生器生成一个私钥，私钥在比特币中的作用

非常重要，可以用来证明用户的身份，也可以签发交易事务。

2）私钥经过SECP256K1算法处理生成了公钥，SECP256K1是一种

特定的椭圆曲线算法，需要注意的是，通过算法可以从私钥生成公钥，

但是却无法反向从公钥生成私钥，这也是公钥为什么可以公开的原因。

3）公钥接下来先使用SHA256哈希算法计算，再使用RIPEMD160

哈希算法计算，计算出公钥哈希。比特币的代码通过2次哈希来计算地

址值，这样能进一步确保哈希后的数值唯一性，进一步降低不同数据进

行哈希后相同的概率。与SHA256一样，RIPEMD160也是一种哈希算

法。

4）将一个地址版本号连接到公钥哈希（比特币主网版本号为

0x00），然后对其进行两次SHA256运算，将计算得到的结果取前面4字

节作为公钥哈希的校验值

5）将0x00版本号与公钥哈希以及校验值连接起来，然后进行

BASE58编码转换，最终得到了比特币地址。

SPV钱包的大致过程如下所示

1）首先下载完整的区块头数据，注意是区块头，而不是所有的区

块链数据，这样可以大大减少需要获取的账本数据量，区块头中包含有

区块的梅克尔根，SPV方式主要就是靠它来实现的。

2）如果想要验证某笔支付交易，则计算出这笔交易事务的哈希值

txHash。

3）找到txHash所在的区块，验证一下所在区块的区块头是否包含

在账本数据中。

1. 获得所在区块中计算梅克尔根所需要的哈希值。

5）计算出梅克尔根。

6）若计算结果与所在区块的梅克尔根相等，则支付交易是存在

的。

7）根据该区块所处的高度位置，还可以确定该交易得到了多少个

确认。

比特币的交易事务结构：



在比特币中，是使用输入脚本和输出脚本程序实现的，有时候也称为“锁定脚本”和“解锁脚本”。简单地说，就是通过“锁定脚本”，利用私钥签名解锁自己的某一条UTXO（也就是之前的“输出”），然后使用对方的公钥锁定新的“输出”，成功后，这笔新的“输出”就成为了对方的UTXO。同样，对方也可以使用“锁定脚本”和“解锁脚本”来实现转账。

rsa加密算法：

我们通过一个例子来理解RSA算法。假设Alice要与Bob进行加密通信，她该怎么生成公钥和私钥呢？

1）选择两个质数。通常是随机选择两个不同的质数，我们不妨称为p和q，本例中Alice选择了61和53，当然实际应用中，这两个质数越大越好，这样就越难破解。

2）计算p和q的乘积n。Alice把61和53相乘：n=61×53=3233。

n的长度就是密钥长度，3233写成二进制是110010100001，一共有12位，所以这个密钥就是12位，实际应用中，RSA密钥一般是1024位，重要场合则为2048位，还是那句话，越长越好。

1. 计算n的欧拉函数Φ（n）。

根据公式：φ（n）=（p-1）（q-1），Alice算出φ（3233）等于60×52，即3120，实际上就是两个质数分别减1后的乘积。

1. 选择一个整数e

这个整数是随机选择的，并且有个条件，条件是1<e<Φ（n），且e与Φ（n）互质。Alice就在1到3120之间，随机选择了17，实际应用中，常常选择65537。

1. 计算e对于Φ（n）的模反元素d

所谓“模反元素”就是指有一个整数d，可以使得e\*d被φ（n）除的余数为1，表达式

如下：

e\*d≡1（modφ（n））

这个式子等价于

e\*d-1=kφ（n）

于是找到模反元素d，实质上就是对下面这个二元一次方程求解：

e\*x+φ（n）y=1

已知e=17，φ（n）=3120，则17x+3120y=1。

这个方程可以用“扩展欧几里得算法”求解，此处省略具体过程。总之，Alice算出一组整数解为（x，y）=（2753，-15），即d=2753。

1. 产生公钥和私钥。

将n和e封装成公钥，n和d封装成私钥，在Alice的例子中，n=3233，e=17，d=2753，所以公钥就是（3233，17），私钥就是（3233，2753）。

至此所有计算就完成了，可以看到RSA的算法过程其实还是很简单的，最关键的就是找到两个足够大的质数。

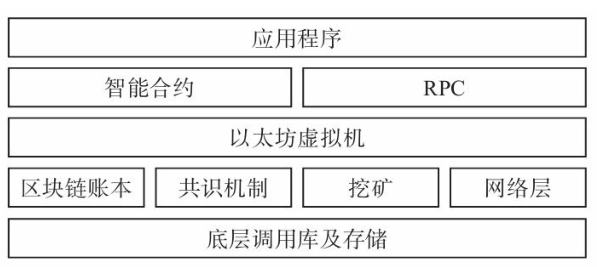
以太坊

<https://stats.ethfans.org> 星火节点

<https://github.com/ethereum> 以太坊源码

<https://github.com/ethereum/mist/releases> 以太坊钱包客户端

以太坊的组成结构



以太坊的每一个区块头，都包含了指向三棵树的指针，分别是：状态树、交易树、收据树。

交易树指针就类似于比特币区块头中的梅克尔树根，交易树是用来代表区块中发生的所有交易历史的；

状态树代表访问区块后的整个状态；

收据树代表每笔交易对应的收据，所谓的收据是指每一笔交易影响的数据条，或者说是每一笔交易影响的结果。