Factor 区块链技术报告书
Factor Blockchain Technical Whitepaper

{ Factor MX(Multi-X) Blockchain Technical Whitepaper}

收录处 「ＦＡＣＴＯＲ　区块链」, vol 1.0, 2019.05.07

写作人 Daniel Lee
FACTOR 区块链所属人

参考 本内容是根据 FACTOR 区块链技术叙述的技术报告书。因持续更新内容，内容会有变更。

目测

技术说明

Technical Description

Daniel Lee

概要

FACTOR 区块链的研究目的是，现在虽然在区块链生态系中上市许多具备多种多样技术的区块链，但是若未解决区块链技术的根本性问题的话，此技术性方面的活用度会逐渐降低，于是判断不能在实际生活中被广泛活用。

FACTOR 区块链是为了克服这些区块链技术问题，查看了许多现象，且以这些现象为基础研究了应用方法。研究方法是调节种类，形态，形式都各自不同的区块链的变因。而且重点集中在速度，保安性，扩展性方面来进行研究。此结果，这些具有先进性技术的区块链技术在现代和未来也具备着卓越性，也助于区块链生态系的活性化，而且具有能够克服现存问题的可能性，同时在启动过程及原理方面上也导出了结果。

Abstract

Paragraph typed in 9.5 pt styled in 'Main Paragraph 1'. Paragraph typed in 9.5 pt styled in 'Main Paragraph 1'. Paragraph typed in 9.5 pt styled in 'Main Paragraph 1'. Paragraph typed in 9.5 pt styled in 'Main Paragraph 1'. Paragraph typed in 9.5 pt styled in 'Main Paragraph 1'. Paragraph typed in 9.5 pt styled in 'Main Paragraph 1'. Paragraph typed in 9.5 pt styled in 'Main Paragraph 1'. Paragraph typed in 9.5 pt styled in 'Main Paragraph 1'. Paragraph typed in 9.5 pt styled in 'Main Paragraph 1'. Paragraph typed in 9.5 pt styled in 'Main Paragraph 1'.

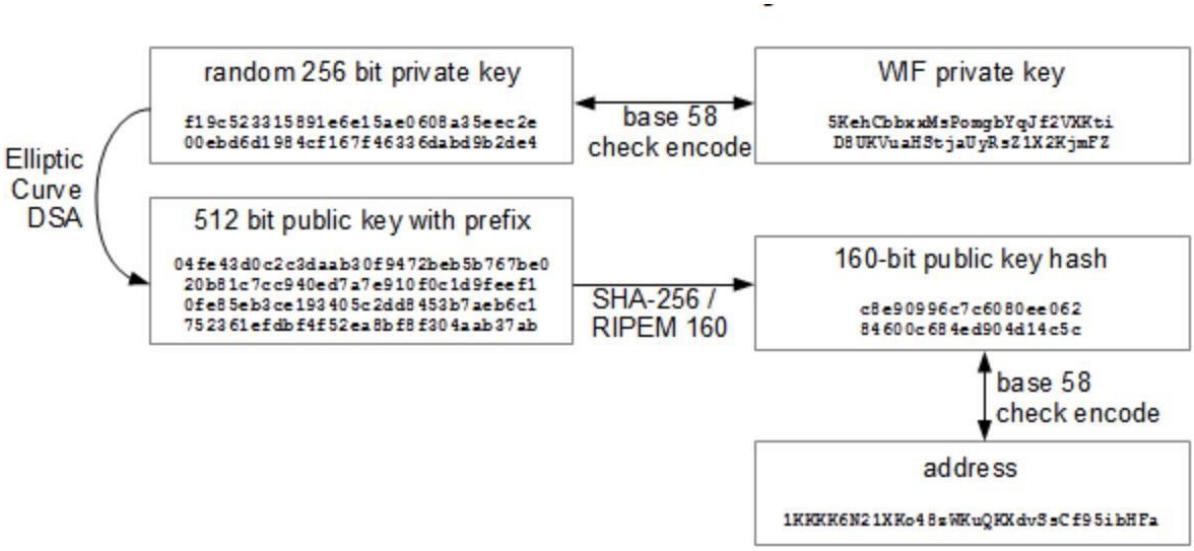
Ⅰ 导论

区块链也简称‘分散元帐’技术。
再说，把记录所有交易内容的元帐分散给多数人们来储存，管理的技术。
详细说明的话，区块链是结合多数网络交易记录，构成一个信息区块（block），利用散列（hash）值来连接之前区块和之后区块后，以 P2P 方式在全世界许多个电脑上复制这些全部或部分信息来分散，储存，管理的技术，而且不可随意伪造信息，于是虽然没有中介机关也可以信赖，具有安全交易及安全处理信息的优点。
区块链将会成为领导第四次产业革命的核心技术。这是依据现代信息社会的大信息，统合物理性世界和数码世界来影响到经济及产业等社会的所有领域。通过 O2O 进行物理性世界和数码世界的统合，且利用把人体的信息嫁接在数码世界的技术-智能手表等，生物学性世界将会实现为移动健身护理。
虚拟现实（VR）和增强现实（AR）及混合现实（MR）也将会融合物理性世界和数码世界。向后需要开发通过区块链和 IoT 来获取的‘政治性透明性’，“第四次产业革命将会带来的社会经济性典范变化”，“为了分散网络对区块链和 IoT 的技术性理解”，“区块链和 IoT 技术的活用事例”等适合多种多样的试图需要进行各方面的开发。
在第四次产业革命中必要的重要技术区块链是具有很多的信息，于是许多专门家提出从量子电脑上的安全保安性问题。
区块链大分为两种。是公共区块链和私人区块链。公共区块链是谁都能参加。需要购买货币的任何人都能参与，这就是公共区块链的代表举例。公共区块链是被开放且透明，于是所有使用者都可以查看所有交易内容。反面，私人区块链是在中央圈单独管理，于是为了加入需要获得承认。是单一企业或伙伴企业间主要使用的形态，于是只有获得承认的使用者加入区块链。不管公共区块链还是私人区块链，区块链是不可操作的。也不能随意变更各交易记录及‘区块’，这是因为这些区块与其它区块都相互连接在一起。因此保障保安。若需要在这些区块链添加新区块的话，需要获得其他使用者的同意。区块链技术确实依赖在应用软件及密码化技术，迄今在开发区块链技术提供的数百个创办企业中大多数都是使用还未检证的算法。 例如，比特币是把已检证的算法 SHA-256k1 使用在散列上。但是依据许多研究结果，因量子电脑最终这些算法都会被打破的。其它区块链是开发者‘保障’了安全，所以只好使用可信任的算法。股市分析家贝内指出‘实际上许多创办企业想要导入新的散列算法，但这些试图都被失败。’CIO 问卷调查结果，在区块链技术导入过程中发现了严重的软件病毒，甚至也具有重新开始所有工程等的风险。
FACTOR 为了解决，补完这些现存问题点，不断开发技术。现在，FACTOR 研究部主张‘区块链的代表-比特币和以太坊随着第四次产业革命，改善或更新速度，扩张性，安全性的变更路途就是与 FACTOR MX(Multi-X)区块链互换。也就是说，FACTOR 打破原有区块链，成功开发了新的区块链。
FACTOR 开发部在区块链的主区块中成功添加散列密码，这些技术是随着节点数的增加原有区块链的速度就逐渐下降，反面 FACTOR 区块链是随着节点数增加，速度会逐渐加快。于是速度很快，且没有出错等问题。
FACTOR MX(Multi-X)区块链是不可用区块链来操作。但是 MX(Multi-X)是可添加，更新应用软件技术的散列密码技能，于是若出现量子电脑也可对应这些问题，进而可以一起发展。
FACTOR 硬币是在保安技术装载着量子力学技能，于是安全上没有问题。

Ⅱ MX(Multi-X) Blockchain Algorithm

MX 区块链的总称是 Multi-X Blockchain 。Multi-X 区块链是含有多种多样的散列算法，各个散列通过相互作用启动许多种技能。
为了助于理解，先说明一下基本启动原理后再说明算法的特性及技能。

1）地址形成原理
[过程: Hash Algorithm -> ECDSA 密码化插入 Secp256r1 (公开键，密码键) -> Base58 Address -> 形成地址]
使用多种多样键和地址，通过按照以下图表可以说明键和地址。首先写作任意 256bit 个人键。个人键是在交易上签名后传送分散账簿时需要使用。因此非公开键需要维持秘密。
Elliptic Curve DSA 算法是从个人键形成 512 bit 公开键。这公开键是使用在确认交易签名。
Factor 协议是在公开键添加 Prefix Algorithm。与大部分公开公开键的系统不同，直到签名交易为止公开键不会被公开。



[1]图
下个阶段是形成与其它人分享的地址。512 bit 公开键不容易使用，于是使用 SHA-256 及 RIPEMD 散列算法，形成到 160bit。 下个键是使用使用者定义 Base58Check 编码，形成为 ASCII。结果地址 1KKKK6N21XKo48zWKuQKXdvSsCf95ibHFa 是为了收到分散账簿而人们揭示的地址。
在地址中不可判别公开键或个人键。若忘记个人键时，不可接近适当分散账簿。 最后 WIF(Wallet Interchange Format key)是 使用在客户, Wallet 软件上添加个人键。
这是非公开键的 ASCII 的 Base58Check 编码, 为了获取 256 bit 个人键可容易找回。 (像上图一样，有 WIF private key - 密码键, Address - 公开键, 160-bit public key hash - 160bit 公开散列键的 3 种类型键， 这些都是使用 Base58Check 编码，表现为 ASCII。非公开键是只能在完毕工作证明的适当分散账簿上具有接近权限。
使用以下 code 形成了 WIF 形式的个人键和地址。个人键单纯是任意 256bit 数字。 ECDSA 密码文件是在个人键形成公开键。 上图的地址是因将要在 1.Hash Algorithm 中说明的算法来形成，依据使用 Checksum 的 Base58 编码来形成。
最后个人键是在 Base58Check 形成编码后，形成在 Factor 客户软件中输入个人键时使用的 WIF 编码。 (下个 code 是举例说明)

[1] 图 [来源: Ken Shirriff's blog]

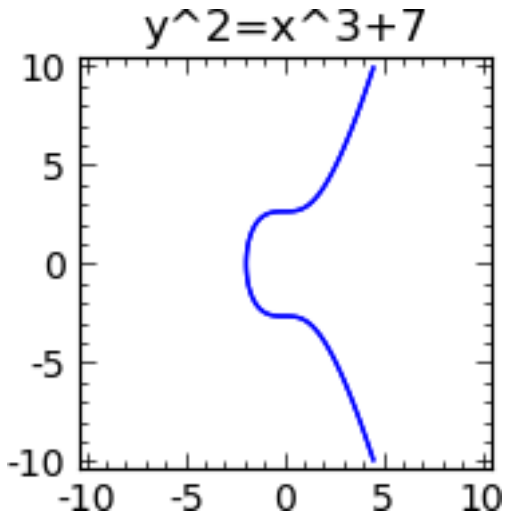
```
def
privateKeyToWif(key_hex):
                    return utils.base58CheckEncode(0x80, key_hex.decode('hex'))

                def privateKeyToPublicKey(s):
                    sk = ecdsa.SigningKey.from_string(s.decode('hex'),
                curve=ecdsa.SECP256k1)
                    vk = sk.verifying_key
                    return ('\04' + sk.verifying_key.to_string()).encode('hex')

                def pubKeyToAddr(s):
                    ripemd160 = hashlib.new('ripemd160')
                    ripemd160.update(hashlib.sha256(s.decode('hex')).digest())
                    return utils.base58CheckEncode(0, ripemd160.digest())


                def keyToAddr(s):
                    return pubKeyToAddr(privateKeyToPublicKey(s))


                # Warning: this random function is not cryptographically
                strong and is just for example
                private_key = ''.join(['%x' % random.randrange(16) for x in
                range(0, 64)])
                print keyUtils.privateKeyToWif(private_key)
                print keyUtils.keyToAddr(private_key)
```

2) 签名算法

现在区块链的 Secp256k1 键方式使用着属于 ECC(Elliptic Curve Cryptography, 椭圆曲线密码技术)方式的 ECDSA(Elliptic Curve Digital Signature Algorithm) 密码算法。在原有系统中以 ECDSA 的 parameter (指定定数集合)来使用 secp256k1 curve， secp256k1 curve 是使用为形成 elliptic curve。 也就是说, ECDSA 是指怎么形成密码化方式， secp256k1 键是在需要密码化的数中代入的定数集合。

因此类似方式的 RSA 密码技术是在实际生活中形成 WIFI 密码, ECC in JavaScript, window 运营体系 Cd-key, SSH-key 等等。但是在这里问题点就是密码化原有单词或用语来分别形成密码键（可译码的键）和公开键，但若出现量子电脑的话，因可预测密码键值，具有可进行译码的运算能力，于是在未来不能使用 ECC, RSA 密码方式。因此 MX Blockchain 是使用具有较高保安性的 secp256r1 方式及量子电脑的运算能力的保护散列。

例如 1) secp256k1 key



[2]

上图是满足非常简单的方程式 y ^ 2 = x ^ 3 + ax + b 的曲线。适当曲线是使用简单的方程式 y ^ 2 = x ^ 3 + 7，来使用 Secp256k1 的特征椭圆曲线。椭圆曲线的重要特性之一就是以简单的规则可以在曲线上添加点。 按照曲线画直线后三个点各个达到 A, B 及 C 时，因 A + B + C = 0. 椭圆曲线的特性，形成集团。而且可以定义整数乘法。（例如 : 4A = A + A + A + A). 在 ECDSA 中可活用椭圆曲线的理由是具有着可以利用多种多样整数乘法的性质。例如，12345678 * A = Q 一样的数式虽然可以快速运算，（算 2 的指数），但只知道 A 和 Q 时 n * A = Q 是较难运算。

在圆曲线密码化中密码 12345678 是个人键，曲线点 Q 是形成为公开键。在密码化中，代替使用曲线实数值，坐标值是以小数为基准的整数。椭圆曲线属性中，不管使用实数还是使用数字的运算，结果几乎都一样，于是很容易使用。

因此椭圆曲线是使用 256bit 点来形成。椭圆曲线电子签名算法 (Elliptic Curve Digital Signature Algorithm, ECDSA)是取短信散列后，应用短信，曲线点，使用个人键及乱数表，修行直线型椭圆曲线算术，且在提供签名的曲线上形成新点。具有公开键，短信及签名的人是为了确认签名是否有效，可修行简单的椭圆曲线运算。因此只有具有个人键的人才能在短信上签名，具有公开键的人不可签名，只能确认短信。

2. 参考 cp256r1 算法，就能知道原有 Secp256k1 的不同点。

[2] 图 [来源: Ken Shirriff's blog]

3) 形成及检证交易

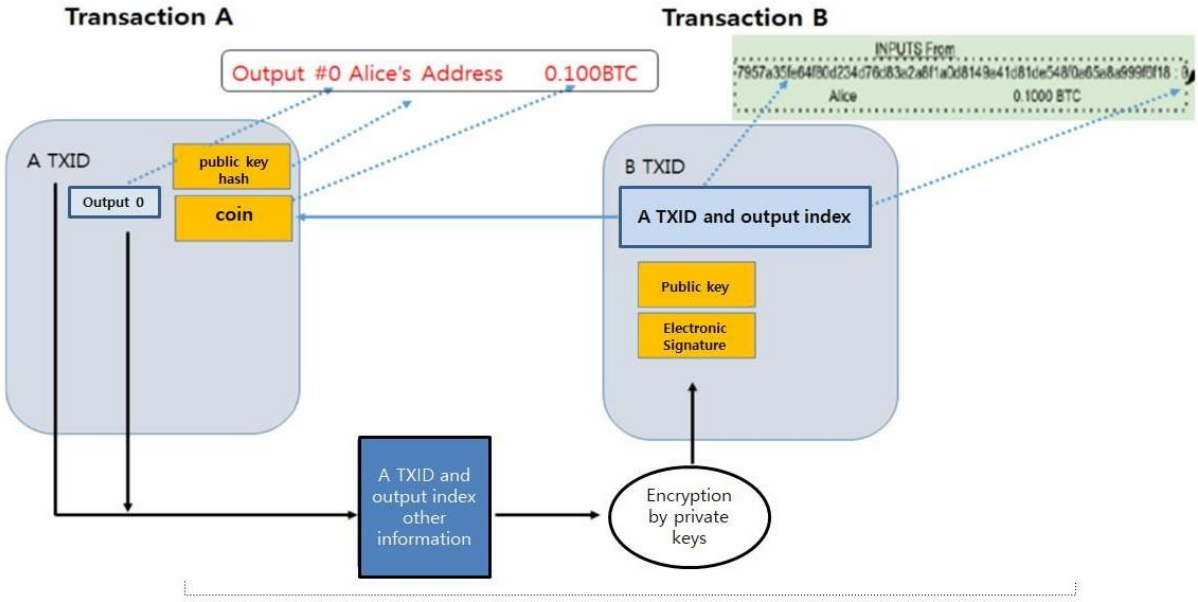
①形成交易的过程

在电子钱包可以把收信人的公开键利用为账户来传送分散账簿。在这里公开键起着发送，接收时活用的账户作用。电子签名是为了证明交易的妥当性来使用，

发送交易信息的发信人形成签名，通过收信人用发信人的公开键检证此签名，可以确认因他人的伪造与否。在输入栏输入电子签名时一起输入公开键，确认此公开键是否是公开键散列原本，用此公开键来确认电子签名。在交易 A 中公开键散列是否正当是通过把交易 B 的公开键变换为公开键散列时，此结果是否与公开键散列一致。检证公开键后，通过电子签名确认签名的内容，此公开键和个人键都被密码化，于是当然能解开密码。

确认此结果是否与交易 A 的信息一致后完毕检证。在网络上的所有节点上能够确认交易 B 的输入部是否正当。

此工程是在利用 script 语言的交易检证签名，这 script 语言非常简单，且限制为 push 和 pop 两个动作，不实行无限 loop。这叫做 turing incompleteness。



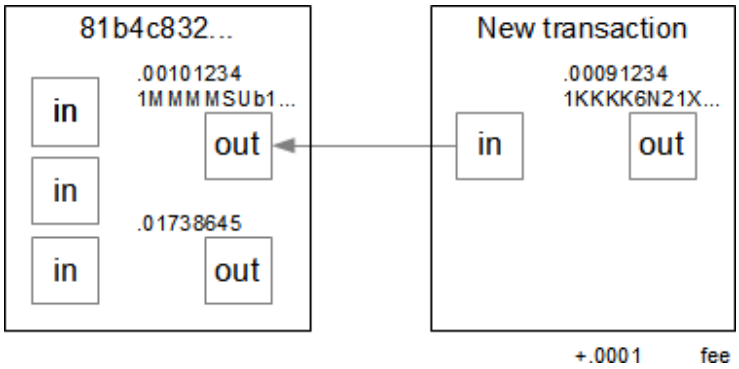
②传播阶段交易的检证

一开始设定节点时，依据下面秩序通过与其它节点的链接传达交易。传达到全世界的全体节点需要几秒左右。

[过程: DNSSEED(例如; seed.Factor.sipa.be, bitseed.xf2.org 等 6 个) 使用选项查询节点 -> 使用 SEEDNODE 选项，为了最初连接，与一个节点连接 -> 因把在网络上认知的节点目录保存到各客户的内部 DB，于是以此信息为基础试图连接其他节点] 形成交易后通过网络传达到邻居节点(实行电子钱包，采掘等电脑)， 结果传达到全体网络。且按照“交易检证目录”首先检证所发送的交易，判断此结果适当的话，持续传达到网络，若不适当时，在适当节点抛弃此交易。

网络的各节点是按照检证目录(checklist)独立检证所有交易后，发送到邻居节点，且切除不适合的交易。 确认交易的佣金(syntax)和信息构造(data structure)，确认交易输入值是否是 UTXO(Unspent Transaction Output)， 交易的输入值 是否比输出值小等通过以上交易检证目录， 成功检证的话， 此节点是给原节点发送“success message”，若检证失败时发送“rejection message”。

③形成交易



假设把 0.00101234 FTC 发送到 1MMMMSUb1piy2ufrSguNUdFmAcvqrQF8M5 地址时，形成 Transection txid(81b4c832...)，向适当钱包（1KKKK6N21XKo48zWKuQKXdvSsCf95ibHFa）附加手续费。

按照下面表形成启动构造。.

Version		01 00 00 00
input count		01
input	previous output hash (reversed)	48 4d 40 d4 5b 9e a0 d6 52 fc a8 25 8a b7 ca a4 25 41 eb 52 97 58 57 f9 6f b5 0c d7 32 c8 b4 81
	previous output index	00 00 00 00
	script length	
	scriptSig	script containing signature
	Sequence	ff ff ff ff
output count		01
output	Value	62 64 01 00 00 00 00 00
	script length	
	scriptPubKey	script containing destination address
block lock time		00 00 00 00

形成交易时使用的示例代码以下而同。

```
# Makes a
transaction
from the
inputs
            # outputs is a list of [redemptionSatoshis, outputScript]
            def makeRawTransaction(outputTransactionHash, sourceIndex, scriptSig, outputs):
                def makeOutput(data):
                    redemptionSatoshis, outputScript = data
                    return (struct.pack("<Q", redemptionSatoshis).encode('hex') +
                    '%02x' % len(outputScript.decode('hex')) + outputScript)
                formattedOutputs = ''.join(map(makeOutput, outputs))
                return (
                    "01000000" + # 4 bytes version
                    "01" + # varint for number of inputs
                    outputTransactionHash.decode('hex')[::-1].encode('hex') + # reverse
            outputTransactionHash
```

```
struct.pack('<L', sourceIndex).encode('hex') +
'%02x' % len(scriptSig.decode('hex')) + scriptSig +
"ffffffff" + # sequence
"%02x" % len(outputs) + # number of outputs
formattedOutputs +
"00000000" # lockTime
)
```

④交易签名方法

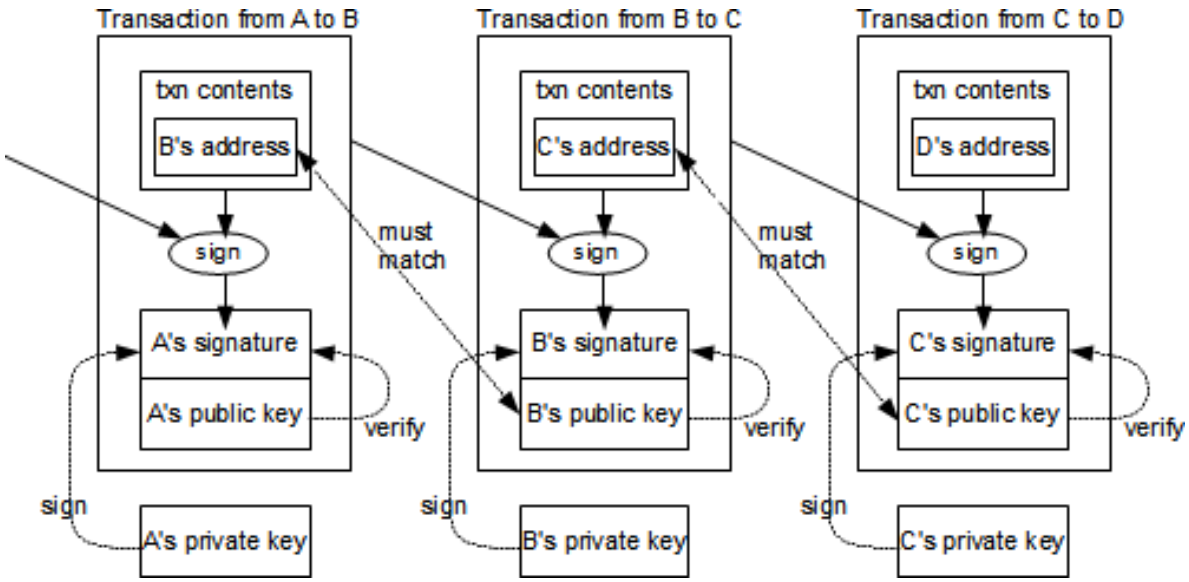
下图是简略提示交易签名后，一起链接的方法。

需要考虑从地址B 到地址 C 传送工作证明的分散账簿的中间交易。

交易内容(包含之前交易的散列)是签名为 B 的个人(private)键。

而且 B 的公开键(public)包含到交易中。修行许多个阶段，任何人都能确认交易是否被 B 受到承认。

首先证明公开键有效的B 的公开键要与之前交易的 B 地址一致。(地址是形成为 Base58 Algorithm)下面，交易 B 的签名是通过使用交易B 的公开键可以确认。 这阶段是保证交易 因 B 有效且具有权限。直到 B 的公开键使用在交易为止不会被公开，在系统上通过交易传达到地址。Chain 的各阶段能够确认分散账簿是否被有效使用。 交易是一般会具有许多输入输出，于是会形成多种多样的 Chain。

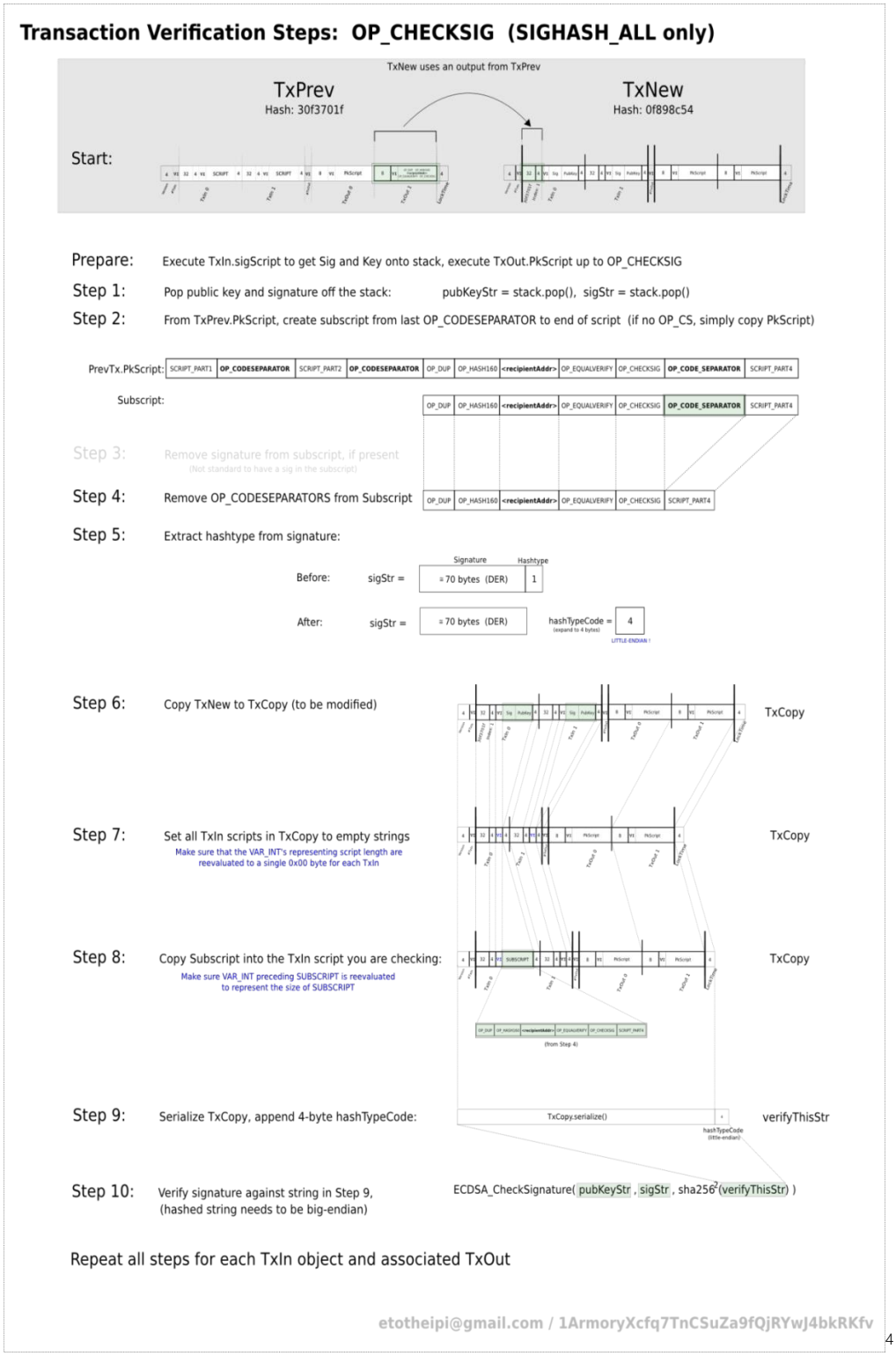


3

[3] 图 [来源: Ken Shirriff's blog]

⑤交易签名

基本主意是使用 ECDSA 椭圆曲线算法和个人键来形成交易的数码签名，但详细事项很复杂。签名工程是分为 19 个阶段，具体按照下面图一样。



[4]

[4] 图 [来源: etotheipi@gmail.com]

最大的问题是进行交易时出现签名， 签名之前提起在交易签名的方法。为了避免这个问题，scriptPubKey 在交易签名之前复制到支出交易（也就是说，正在签名中的交易）。之后签名变换为 script 语言的代码，写成包含在交易上的 scriptSig。签名时使用之前交易的 scriptPubKey 也是使用为记录的目的。 若是具有许多个输入的交易，按照各个输入需要分别签名。而且每个 Hash type 签名之前临时添加定数。Transaction 是 SIGHASH_ALL (0x00000001)。签名后这散列类型是在交易结尾删除，且添加到 scriptSig。签名是以 DER 编码形成编码，公开键是表示为一般 bite。 SIGHASH_ALL 是放在签名后面，04 类型是放在公开键前面。 因 ECDSA 算法使用乱数，签名 调试的难度更加提高。 因此签名是每次结算时都不同，不可与公开的签名进行比较。下面是交易签名示例代码。

```
def
makeSignedTransaction(privat
eKey, outputTransactionHash,
sourceIndex, scriptPubKey,
outputs):
                    myTxn_forSig = (makeRawTransaction(outputTransactionHash,
sourceIndex, scriptPubKey, outputs)
                            + "01000000") # hash code


                    s256 =
hashlib.sha256(hashlib.sha256(myTxn_forSig.decode('hex')).digest(
)).digest()
                    sk = ecdsa.SigningKey.from_string(privateKey.decode('hex'),
curve=ecdsa.SECP256k1)
                    sig = sk.sign_digest(s256,
sigencode=ecdsa.util.sigencode_der) + '\01' # 01 is hashtype
                    pubKey = keyUtils.privateKeyToPublicKey(privateKey)
                    scriptSig = utils.varstr(sig).encode('hex') +
utils.varstr(pubKey.decode('hex')).encode('hex')
                    signed_txn = makeRawTransaction(outputTransactionHash,
sourceIndex, scriptSig, outputs)
                    verifyTxnSignature(signed_txn)
                    return signed_txn
```

scriptSig 是包含对 source 地址的公开键和签名。
下面表是证明交易被许可，交易有效。

PUSHDATA 47		47
signature (DER)	sequence	30
	length	44
	integer	02
	length	20
	X	2c b2 65 bf 10 70 7b f4 93 46 c3 51 5d d3 d1 6f c4 54 61 8c 58 ec 0a 0f f4 48 a6 76 c5 4f f7 13
	integer	02
	length	20
	Y	6c 66 24 d7 62 a1 fc ef 46 18 28 4e ad 8f 08 67 8a c0 5b 13 c8 42 35 f1 65 4e 6a d1 68 23 3e 82
SIGHASH_ALL		01
PUSHDATA 41		41
public key	type	04
	X	14 e3 01 b2 32 8f 17 44 2c 0b 83 10 d7 87 bf 3d 8a 40 4c fb d0 70 4f 13 5b 6a d4 b2 d3 ee 75 13
	Y	10 f9 81 92 6e 53 a6 e8 c3 9b d7 d3 fe fd 57 6c 54 3c ce 49 3c ba c0 63 88 f2 65 1d 1a ac bf cd

最后在 scriptPubKey，下面结果证明交易成功实行。

OP_DUP	76
OP_HASH160	a9
PUSHDATA 14	14
public key hash	c8 e9 09 96 c7 c6 08 0e e0 62 84 60 0c 68 4e d9 04 d1 4c 5c
OP_EQUALVERIFY	88
OP_CHECKSIG	ac

⑥最终交易
经过以上过程后，最终完毕交易。

- 源码

```
privateKey =
keyUtils.wifToPrivateKey("5HusYj2b2x4nroApgfvaSf
KYZhRbKFH41bVyPooymbC6KfgSXdD") #1MMMM
```

```
signed_txn =
txnUtils.makeSignedTransaction(privateKey,

"81b4c832d70cb56ff957589752eb4125a4cab78a25
a8fc52d6a09e5bd4404d48", # output (prev)
transaction hash
```

```
        0, # sourceIndex

keyUtils.addrHashToScriptPubKey("1MMMMSUb1p
iy2ufrSguNUdFmAcvqrQF8M5"),
        [[91234, #satoshis

keyUtils.addrHashToScriptPubKey("1KKKK6N21X
Ko48zWKuQKXdvSsCf95ibHFa")]]
        )

txnUtils.verifyTxnSignature(signed_txn)
print 'SIGNED TXN', signed_txn
```

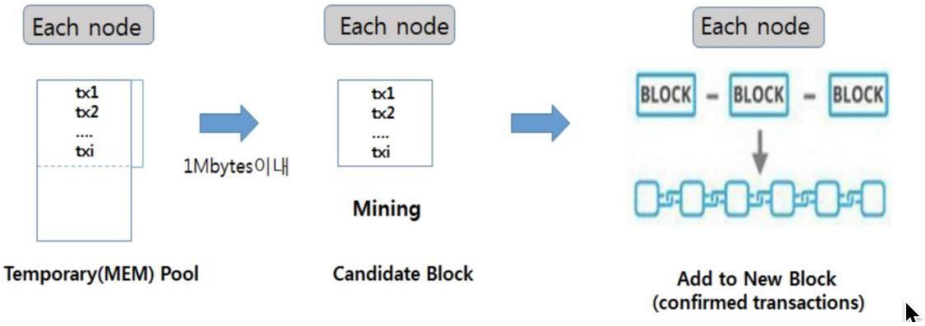
- 最终交易的启动构造是以下而同。
  把上面的 scriptSig 和 scriptPubKey 与未签名的交易结合。

version		01 00 00 00
input count		01
input	previous output hash (reversed)	48 4d 40 d4 5b 9e a0 d6 52 fc a8 25 8a b7 ca a4 25 41 eb 52 97 58 57 f9 6f b5 0c d7 32 c8 b4 81
	previous output index	00 00 00 00
	script length	8a
	scriptSig	47 30 44 02 20 2c b2 65 bf 10 70 7b f4 93 46 c3 51 5d d3 d1 6f c4 54 61 8c 58 ec 0a 0f f4 48 a6 76 c5 4f f7 13 02 20 6c 66 24 d7 62 a1 fc ef 46 18 28 4e ad 8f 08 67 8a c0 5b 13 c8 42 35 f1 65 4e 6a d1 68 23 3e 82 01 41 04 14 e3 01 b2 32 8f 17 44 2c 0b 83 10 d7 87 bf 3d 8a 40 4c fb d0 70 4f 13 5b 6a d4 b2 d3 ee 75 13 10 f9 81 92 6e 53 a6 e8 c3 9b d7 d3 fe fd 57 6c 54 3c ce 49 3c ba c0 63 88 f2 65 1d 1a ac bf cd
	sequence	ff ff ff ff
output count		01
output	value	62 64 01 00 00 00 00 00
	script length	19
	scriptPubKey	76 a9 14 c8 e9 09 96 c7 c6 08 0e e0 62 84 60 0c 68 4e d9 04 d1 4c 5c 88 ac
block lock time		00 00 00 00

4) 区块的构造

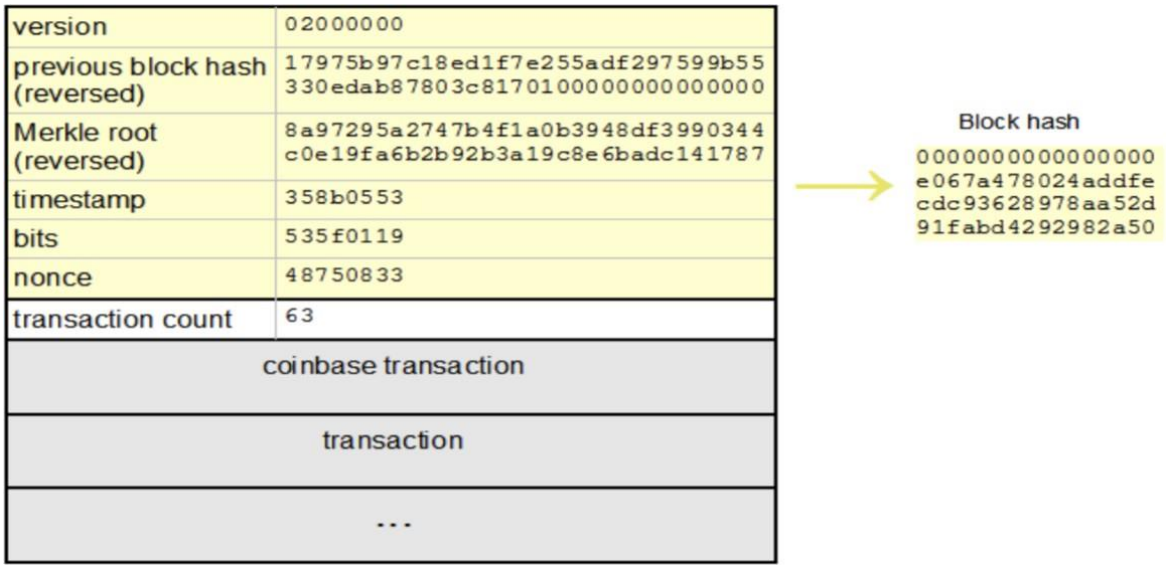
①区块构成

在各节点存在的Temporary Pool 上积蓄被检证的交易，
此交易中为了采掘(Mining)构成预备区块 (Candidate Block)。
在 Temporary Pool 形成预备区块，新诞生的区块连接于区块链。



[5] 在预备区块完成采掘后，形成在区块链连接的新区块。

[5] 图 [来源: Mastering Cryptocurrency 2nd Edition, O'Reillly]



6

division	Description
version	Number of block version
Previous block hash	A hash value from hashing the previous block header twice using the sha256 hash function
Merkle hash root	A hash value that is located at the root of the tree when the transaction hash of the transaction information in the current block is constructed in a binary tree form.
Timestamp	Time of block creation, seconds since January 1 1970
Bits	Difficulty goal for the POW algorithm of block
Nonce	Counters for finding values lower than a specific target value

7

区块链的区块是构成为头部(header)和体部(body)，Header是像上图一样构成为80bytes，节点是检证收到的交易后，在Temporary pool上持续添加此交易。在交易pool中的交易中，形成预备区块，之后为了确定交易等待被采掘。

---

6 图 [来源: http://www.righto.com/2014/02/Factor-mining-hard-way-algorithms.html]

7 图 [来源: http://www.righto.com/2014/02/Factor-mining-hard-way-algorithms.html]

②区块检证

■ Block verification process

① Verification of Previous block existence
(hash value of the previous block header)

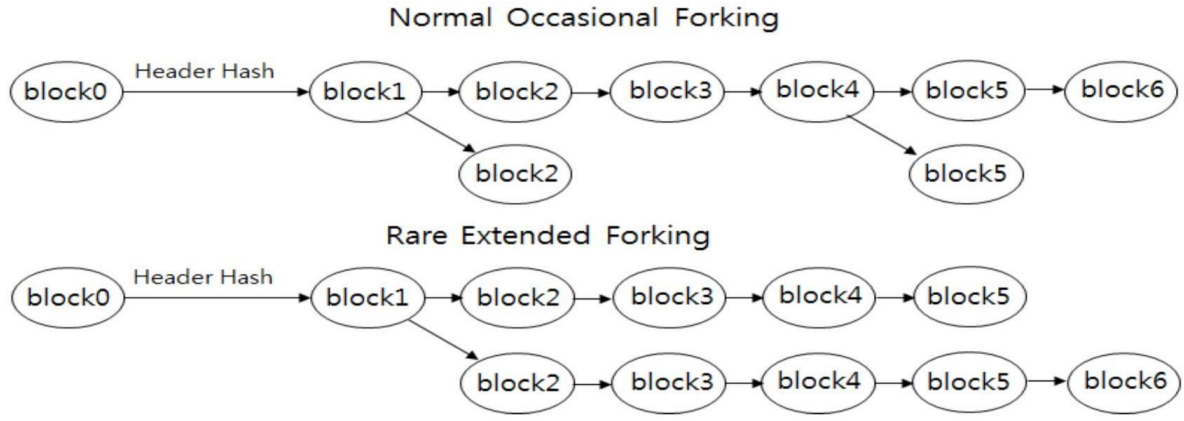
② Verification of Proof of work (POW)

SHA256(SHA256(| Ver | Prev Block hash | Merkle hash | | Time | bits | nonce |)) <-Desired Value

[8]

连接在网络上的各节点是对各个新形成的区块独立进行检证。检证新形成的区块信息构造是否适当，区块 header 散列值是否比目标值更低，区块大小是否比在系统上定义的大小更小等等。区块适当(valid)的话，适当节点是把 “success message”发送到形成区块的节点 (originator)，若区块不适当的话，把“reject message”传送到形成的节点。需要收到节点 一半以上 (51%)的合同才能承认区块，也就是说才能承认交易，以后不占全体节点的一半以上， 就不能伪造交易。以上是举例，是表现了 SHA256 算法，且使用在 1. MX blockchain Hash Algorithm 所说明的算法。

③形成区块



[9]

收到新区块的各节点是完毕检证以后，把这区块连接到原有区块链。是区块链分散的信息构造，于是被采掘的区块能够同时在相互不同的节点上诞生，在数千，数万个节点中会发生把首先收到的新区块连接到原有区块链的“区块链分歧” (Blockchain Forking)”现象。
若发生 Normal Occasional Forking(分歧)时，优先适用难度(difficulty)高的区块。 Rare Extended Forking 是随着时间，最长的区块链生存，短的区块链就会自动删除。

[8] 图［来源: 区块链构造与理论 p.120 再次构成］
[9] 图[来源: https://blog-archive.bitgo.com/the-challenges-of-block-chain-indexing]

1.MX Blockchain Hash Algorithm

A = 2, 3, 5, 6, 7, (8,8-1,8-2,8-3,8-4,8-5), 9, (13,13-1), (14,14-1), 15, 16, 17 (散列算法)
B = 2, 3, 5, 6, 7, 9, (13,13-1), 15, 16, 17 (散列连接算法)
C = 18, A, B 包含算法 (散列连接排列算法)
D = 4, (算法)
E = 拿 C, D, F 算法修行 AES 排列作用(散列算法)
F = SHA-1(F10), SHA-256(F11), SHA-512(F12). (SHA 算法)
(参考事项: 在 A 存在 A2, A3, A5,, 等等的算法，在 B 中存在 B2, B3, B5,,等等的算法。按照各个 A, B, C, D, E, F 存在各自不同的算法， A2, B2 的算法是相互不同的算法。)

MultiX Blockchain Hash Algorithm 是具有下面散列算法种类。在这里 A 和 B 聚集在 C 首先连接。因此把这样连接的算法 C, D, F 聚集在 E 形成 build。这时 D 通过自动插入 Factor Blockchain 的设定内容的技能，在 E “C, D, F” 以 AES 排列方式形成后启动。

E1 散列: 是散列算法，拿 C, D, F 算法修行 AES 排列作用

```
#include
"C,D,F"


                #define AESx(x)   ( ((SPH_C32(x) >> 24) & SPH_C32(0x000000FF)) \
                                  | ((SPH_C32(x) >>  8) & SPH_C32(0x0000FF00)) \
                                  | ((SPH_C32(x) <<  8) & SPH_C32(0x00FF0000)) \
                                  | ((SPH_C32(x) << 24) & SPH_C32(0xFF000000)))


                #define AES0      AES0_BE
                #define AES1      AES1_BE
                #define AES2      AES2_BE
                #define AES3      AES3_BE


                #define AES_ROUND_BE(X0, X1, X2, X3, K0, K1, K2, K3, Y0, Y1, Y2, Y3)   do
                { \
                                (Y0) = AES0[((X0) >> 24) & 0xFF] \
                                        ^ AES1[((X1) >> 16) & 0xFF] \
                                        ^ AES2[((X2) >> 8) & 0xFF] \
                                        ^ AES3[(X3) & 0xFF] ^ (K0); \
                                (Y1) = AES0[((X1) >> 24) & 0xFF] \
                                        ^ AES1[((X2) >> 16) & 0xFF] \
                                        ^ AES2[((X3) >> 8) & 0xFF] \
                                        ^ AES3[(X0) & 0xFF] ^ (K1); \
                                (Y2) = AES0[((X2) >> 24) & 0xFF] \
                                        ^ AES1[((X3) >> 16) & 0xFF] \
                                        ^ AES2[((X0) >> 8) & 0xFF] \
```

```
                ^ AES3[(X1) & 0xFF] ^ (K2); \
        (Y3) = AES0[((X3) >> 24) & 0xFF] \
                ^ AES1[((X0) >> 16) & 0xFF] \
                ^ AES2[((X1) >> 8) & 0xFF] \
                ^ AES3[(X2) & 0xFF] ^ (K3); \
    } while (0)


#define AES_ROUND_NOKEY_BE(X0, X1, X2, X3, Y0, Y1, Y2, Y3) \
    AES_ROUND_BE(X0, X1, X2, X3, 0, 0, 0, 0, Y0, Y1, Y2, Y3)


#else


#define AESx(x)   SPH_C32(x)
#define AES0      AES0_LE
#define AES1      AES1_LE
#define AES2      AES2_LE
#define AES3      AES3_LE


#define AES_ROUND_LE(X0, X1, X2, X3, K0, K1, K2, K3, Y0, Y1, Y2, Y3)   do
{ \
        (Y0) = AES0[(X0) & 0xFF] \
                ^ AES1[((X1) >> 8) & 0xFF] \
                ^ AES2[((X2) >> 16) & 0xFF] \
                ^ AES3[((X3) >> 24) & 0xFF] ^ (K0); \
        (Y1) = AES0[(X1) & 0xFF] \
                ^ AES1[((X2) >> 8) & 0xFF] \
                ^ AES2[((X3) >> 16) & 0xFF] \
                ^ AES3[((X0) >> 24) & 0xFF] ^ (K1); \
        (Y2) = AES0[(X2) & 0xFF] \
                ^ AES1[((X3) >> 8) & 0xFF] \
                ^ AES2[((X0) >> 16) & 0xFF] \
                ^ AES3[((X1) >> 24) & 0xFF] ^ (K2); \
        (Y3) = AES0[(X3) & 0xFF] \
                ^ AES1[((X0) >> 8) & 0xFF] \
                ^ AES2[((X1) >> 16) & 0xFF] \
                ^ AES3[((X2) >> 24) & 0xFF] ^ (K3); \
    } while (0)


#define AES_ROUND_NOKEY_LE(X0, X1, X2, X3, Y0, Y1, Y2, Y3) \
    AES_ROUND_LE(X0, X1, X2, X3, 0, 0, 0, 0, Y0, Y1, Y2, Y3)
```

```
#endif
```

A2 散列
使用 32 bit 单词，直到 256 bit 使用在电脑散列，使用 64 bit 单词，直到 512 bit 使用为计算散列。区块变换是输入 16word 及结合 16 个工作变数， 在区块间只保存 8 word (256 或 512 bit)。
使用 16 个 定数单词 (π 的分数部分 512 或 1024 bit)和 10 个 16 要素排列。

```
σ [0] = 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
σ [1] = 14 10 4 8 9 15 13 6 1 12 0 2 11 7 5 3
σ [2] = 11 8 12 0 5 2 15 13 10 14 3 6 7 1 9 4
σ [3] = 7 9 3 1 13 12 11 14 2 6 5 10 4 0 15 8
σ [4] = 9 0 5 7 2 4 10 15 14 1 11 12 6 8 3 13
σ [5] = 2 12 6 10 0 11 8 3 4 13 7 5 15 14 1 9
σ [6] = 12 5 1 15 14 13 4 10 0 7 6 3 9 2 8 11
σ [7] = 13 11 7 14 12 1 3 9 5 0 15 4 8 6 2 10
σ [8] = 6 15 14 9 11 3 0 8 12 2 13 7 1 4 10 5
σ [9] = 10 2 8 4 7 6 1 5 15 11 9 14 3 12 13 0
```

与 ChaCha 的 1/4 round 同等的核心工作是 4 单词或在对角线 a b c d 中启动，与 2 单词的短信 m[]和 2 个定数单词结合。n[]. 每 1round 修行 8 次。

```
j ← σ[r%10][2×i]            // Index computations
k ← σ[r%10][2×i+1]
a ← a + b + (m[j] ⊕ n[k])    // Step 1 (with input)
d ← (d ⊕ a) >>> 16
c ← c + d                    // Step 2 (no input)
b ← (b ⊕ c) >>> 12
a ← a + b + (m[k] ⊕ n[j])    // Step 3 (with input)
d ← (d ⊕ a) >>> 8
c ← c + d                    // Step 4 (no input)
b ← (b ⊕ c) >>> 7
```

A3 散列
依据 Digest sizes 224, 256, 384 or 512 bits, 无差别攻击有可能性发现冲突，于是不提供 160 bit 区块。 A3 散列是与短信一起启动分离为区块。

区块是按照顺序分为单词。区块及单词的大小是按照算法的特定实现而不同展现。下表是罗列 A3 散列算法的 4 种变形的主要属性。

一般属性

算法	短信大小	区块大小	单词大小	数码签名
A3 散列 224	$<2^{64}$	512	32	224
A3 散列 384	$<2^{64}$	512	32	384
A3 散列 224	$<2^{64}$	1024	64	224
A3 散列 512	$<2^{64}$	1024	64	512

参数，变数及定数

变数	说明
H	.n bit 的最终数码签名的最少两倍变更值.
Q	4 倍 TUBE
H (i)	第 i 个值为 H。 H (0)是初值。
H (N)	H 的最终值。使用为决定数码签名. n bit。
Q (i)	是 4 重 pipe 的 第 i 个值
H (i, j)	Je 是 H (i)的单词。在这里., H (i)是分割为提前决定的数的区块，也就是说分割为单词。
H (i, 0)	H (i)的最初(左边)
Q (i, j)	第 4 Q (i)的第 j 单词
Q (i, a)	从 Q (i)的 最前 16 个单词 , Q (i, j) 在这里 j = 0..15
Q (i, b)	在 Q (i)的最后 16 单词, Q (i, j)在这里 j = 16..31
I	短信长度 M (bit)
m	短信区块 M (i)的 bit 数为
M	Bit 长度 l 的输入短信
M (i)	I 是扯断输入短信。Bit 长度 m
M (i, j)	M (i). M (i, 0), M (i, 1), ..., M (i, 15)]

XL, XH	在 H (i)的计算上使用的两个临时单词(按照算法的修正，长度 32 或 64 bit)

散列第 3 构造的一般性特征

全处理

1. 输入短信
2. 把输入的短信以 m bit 区块分析句子
3. 初始化在计算 hf 时使用的初值

散列连算

1. 在收信的短信中计算短信寄存器
2. 使用这寄存器计算 H (i)值的次序
3. n 最下位 bit(*LSB* - *最下位bit*)被选为数码签名

提前计算阶段

按照算法修正，输入的短信按照以下进行处理

第 3 个散列 224 或第 3 个散列 256

假设把短信长度为 I。这短信是 连接 1，跟随第 k 个 0 的序列。在这里 k 是方程式 I + 1 + k = 448 mod512 。

第 3 个散列 384 或第 3 个散列 512

假设把短信长度为 I。在这短信分割短信，在后面添加 k 个 0.在这里 k 不是最小的负数。是方程式 1 + 1 + k = 960 mod1024 的解答。之后 分割数字 1 的 2 进表现 64bit 区块。例如 "abc"(按照 ASCII)

计算散列函数的阶段

在计算工程上使用三种技能

。

- 第 1 函数 F_0 : $\{0.1\}^{2m}$ - $\{0.1\}^{m}$ 。这是采取两个因素 M (i)和 H (i - 1)，形成 bijective mapping M (i) XOR H (i - 1)。在这里 M (i)是第 i 个短信区块， H (i - 1)是二进 pipe 的现在值。 此结果，记录在上述第 1 部分 quad tube : Q 是 (I, a) = F_0 (M (I), H (I- 1)) = **A2** (**A1** (M (I) H (I-1) XOR)).
- 第二函数 F_1: $\{0,1\}^{2m} \rightarrow \{0,1\}^{m}$. 短信区块 M (i) (长度 m bit)和 4 重 pipe Q (i, a)的第一个部分采取为因素。结果四重 pipe 记录在 Q (i, b) = F_1 (M (i), Q (i, a)) 的第二个部分。
- 第三函数 F_3 : $\{0,1\}^{3m} \rightarrow \{0,1\}^{n}$. 为了这个 collapsing (eng. *{{{1}}* - *folding*) 的用语是使用为强调第 3 次元空间的 m 次元的属性。短信区块 M (i)和四重 pipeQ (i) = Q (i, a) || Q (i, b)的现在值采取为因素。结果是按照下面一样使用为 H (i)的新值。: H (i) = F_2 (M (i), Q (i))

```
计算散列函数
for (i=1;i<N;i++)
{
  Q(i,a)=F0(M(i),H(i-1));
  Q(i,b)=F1(M(i),Q(i,a));
```

```
  Q(i)=Q(i,a)||Q(i,b);
  H(i)=F2(M(i),Q(i));
}
Hash=N_LeastSignificantBits(H(i));
```

为了决定函数 f，$_{0}$ F，$_{1}$ ，f $_{(2)}$ ，第 1 附加技能

BMW224/BMW256	BMW384/BMW512
$s_0(x) = SHR^1(x) \oplus SHL^3(x) \oplus ROTL^4(x) \oplus ROTL^{19}(x)$ $s_1(x) = SHR^1(x) \oplus SHL^2(x) \oplus ROTL^8(x) \oplus ROTL^{23}(x)$ $s_2(x) = SHR^2(x) \oplus SHL^1(x) \oplus ROTL^{12}(x) \oplus ROTL^{25}(x)$ $s_3(x) = SHR^2(x) \oplus SHL^2(x) \oplus ROTL^{15}(x) \oplus ROTL^{29}(x)$ $s_4(x) = SHR^1(x) \oplus x$ $s_5(x) = SHR^2(x) \oplus x$ $r_1(x) = ROTL^3(x)$ $r_2(x) = ROTL^7(x)$ $r_3(x) = ROTL^{13}(x)$ $r_4(x) = ROTL^{16}(x)$ $r_5(x) = ROTL^{19}(x)$ $r_6(x) = ROTL^{23}(x)$ $r_7(x) = ROTL^{27}(x)$	$s_0(x) = SHR^1(x) \oplus SHL^3(x) \oplus ROTL^4(x) \oplus ROTL^{37}(x)$ $s_1(x) = SHR^1(x) \oplus SHL^2(x) \oplus ROTL^{13}(x) \oplus ROTL^{43}(x)$ $s_2(x) = SHR^2(x) \oplus SHL^1(x) \oplus ROTL^{19}(x) \oplus ROTL^{53}(x)$ $s_3(x) = SHR^2(x) \oplus SHL^2(x) \oplus ROTL^{28}(x) \oplus ROTL^{59}(x)$ $s_4(x) = SHR^1(x) \oplus x$ $s_5(x) = SHR^2(x) \oplus x$ $r_1(x) = ROTL^5(x)$ $r_2(x) = ROTL^{11}(x)$ $r_3(x) = ROTL^{27}(x)$ $r_4(x) = ROTL^{32}(x)$ $r_5(x) = ROTL^{37}(x)$ $r_6(x) = ROTL^{43}(x)$ $r_7(x) = ROTL^{53}(x)$
$expand_1(j) = s_1(Q_{j-16}^{(i)}) + s_2(Q_{j-15}^{(i)}) + s_3(Q_{j-14}^{(i)}) + s_0(Q_{j-13}^{(i)}) + s_1(Q_{j-12}^{(i)}) + s_2(Q_{j-11}^{(i)}) + s_3(Q_{j-10}^{(i)}) + s_0(Q_{j-9}^{(i)}) + s_1(Q_{j-8}^{(i)}) + s_2(Q_{j-7}^{(i)}) + s_3(Q_{j-6}^{(i)}) + s_0(Q_{j-5}^{(i)}) + s_1(Q_{j-4}^{(i)}) + s_2(Q_{j-3}^{(i)}) + s_3(Q_{j-2}^{(i)}) + s_0(Q_{j-1}^{(i)}) + M_{(j-16) \bmod 16}^{(i)} + M_{(j-13) \bmod 16}^{(i)} - M_{(j-6) \bmod 16}^{(i)} + K_j$	$expand_1(j) = s_1(Q_{j-16}^{(i)}) + s_2(Q_{j-15}^{(i)}) + s_3(Q_{j-14}^{(i)}) + s_0(Q_{j-13}^{(i)}) + s_1(Q_{j-12}^{(i)}) + s_2(Q_{j-11}^{(i)}) + s_3(Q_{j-10}^{(i)}) + s_0(Q_{j-9}^{(i)}) + s_1(Q_{j-8}^{(i)}) + s_2(Q_{j-7}^{(i)}) + s_3(Q_{j-6}^{(i)}) + s_0(Q_{j-5}^{(i)}) + s_1(Q_{j-4}^{(i)}) + s_2(Q_{j-3}^{(i)}) + s_3(Q_{j-2}^{(i)}) + s_0(Q_{j-1}^{(i)}) + M_{(j-16) \bmod 16}^{(i)} + M_{(j-13) \bmod 16}^{(i)} - M_{(j-6) \bmod 16}^{(i)} + K_j$
$expand_2(j) = Q_{j-16}^{(i)} + r_1(Q_{j-15}^{(i)}) + Q_{j-14}^{(i)} + r_2(Q_{j-13}^{(i)}) + Q_{j-12}^{(i)} + r_3(Q_{j-11}^{(i)}) + Q_{j-10}^{(i)} + r_4(Q_{j-9}^{(i)}) + Q_{j-8}^{(i)} + r_5(Q_{j-7}^{(i)}) + Q_{j-6}^{(i)} + r_6(Q_{j-5}^{(i)}) + Q_{j-4}^{(i)} + r_7(Q_{j-3}^{(i)}) + s_5(Q_{j-2}^{(i)}) + s_4(Q_{j-1}^{(i)}) + M_{(j-16) \bmod 16}^{(i)} + M_{(j-13) \bmod 16}^{(i)} - M_{(j-6) \bmod 16}^{(i)} + K_j$	$expand_2(j) = Q_{j-16}^{(i)} + r_1(Q_{j-15}^{(i)}) + Q_{j-14}^{(i)} + r_2(Q_{j-13}^{(i)}) + Q_{j-12}^{(i)} + r_3(Q_{j-11}^{(i)}) + Q_{j-10}^{(i)} + r_4(Q_{j-9}^{(i)}) + Q_{j-8}^{(i)} + r_5(Q_{j-7}^{(i)}) + Q_{j-6}^{(i)} + r_6(Q_{j-5}^{(i)}) + Q_{j-4}^{(i)} + r_7(Q_{j-3}^{(i)}) + s_5(Q_{j-2}^{(i)}) + s_4(Q_{j-1}^{(i)}) + M_{(j-16) \bmod 16}^{(i)} + M_{(j-13) \bmod 16}^{(i)} - M_{(j-6) \bmod 16}^{(i)} + K_j$

A3 散列 224 / A3 散列 256	A3 散列 384 / A3 散列 512

$s_0(x) = SHR^1(x) \otimes SHL^3(x) \otimes ROTL^4(x) \otimes ROTL^{1}$	$s_0(x) = SHR^1(x) \otimes SHL^3(x) \otimes ROTL^4(x) \otimes ROTL^{37}$
$s_1(x) = SHR^1(x) \otimes SHL^2(x) \otimes ROTL^8(x) \otimes ROTL^{2}$	$s_1(x) = SHR^1(x) \otimes SHL^2(x) \otimes ROTL^{13}(x) \otimes ROTL^{4}$
$s_2(x) = SHR^2(x) \otimes SHL^1(x) \otimes ROTL^{12}(x) \otimes ROTL^{}$	$s_2(x) = SHR^2(x) \otimes SHL^1(x) \otimes ROTL^{19}(x) \otimes ROTL^{5}$
$s_3(x) = SHR^2(x) \otimes SHL^2(x) \otimes ROTL^{15}(x) \otimes ROTL^{}$	$s_3(x) = SHR^2(x) \otimes SHL^2(x) \otimes ROTL^{28}(x) \otimes ROTL^{5}$
$s_4(x) = SHR^1(x) \otimes x$	$s_4(x) = SHR^1(x) \otimes x$
$s_5(x) = SHR^2(x) \otimes x$	$s_5(x) = SHR^2(x) \otimes x$
$r_1(x) = ROTL^3(x)$	$r_1(x) = ROTL^5(x)$
$r_2(x) = ROTL^7(x)$	$r_2(x) = ROTL^{11}(x)$
$r_3(x) = ROTL^{13}(x)$	$r_3(x) = ROTL^{27}(x)$
$r_4(x) = ROTL^{16}(x)$	$r_4(x) = ROTL^{32}(x)$
$r_5(x) = ROTL^{19}(x)$	$r_5(x) = ROTL^{37}(x)$
$r_6(x) = ROTL^{23}(x)$	$r_6(x) = ROTL^{43}(x)$
$r_7(x) = ROTL^{27}(x)$	$r_7(x) = ROTL^{53}(x)$

$$
\begin{aligned}
\text{expand}_1(j) = {} & s_1(Q^{(i)}_{j-16}) + s_2(Q^{(i)}_{j-15}) + s_3(Q^{(i)}_{j-14}) + s_0(Q^{(i)}_{j-13}) \\
& + s_1(Q^{(i)}_{j-12}) + s_2(Q^{(i)}_{j-11}) + s_3(Q^{(i)}_{j-10}) + s_0(Q^{(i)}_{j-9}) \\
& + s_1(Q^{(i)}_{j-8}) + s_2(Q^{(i)}_{j-7}) + s_3(Q^{(i)}_{j-6}) + s_0(Q^{(i)}_{j-5}) \\
& + s_1(Q^{(i)}_{j-4}) + s_2(Q^{(i)}_{j-3}) + s_3(Q^{(i)}_{j-2}) + s_0(Q^{(i)}_{j-1}) \\
& + M^{(i)}_{(j-16)\bmod 16} + M^{(i)}_{(j-13)\bmod 16} - M^{(i)}_{(j-6)\bmod 16} + K_j \\
\text{expand}_2(j) = {} & Q^{(i)}_{j-16} + r_1(Q^{(i)}_{j-15}) + (Q^{(i)}_{j-14}) + r_2(Q^{(i)}_{j-13}) \\
& + Q^{(i)}_{j-12} + r_3(Q^{(i)}_{j-11}) + Q^{(i)}_{j-10} + r_4(Q^{(i)}_{j-9}) \\
& + Q^{(i)}_{j-8} + r_5(Q^{(i)}_{j-7}) + Q^{(i)}_{j-6} + r_6(Q^{(i)}_{j-5}) \\
& + Q^{(i)}_{j-4} + r_7(Q^{(i)}_{j-3}) + s_5(Q^{(i)}_{j-2} + r_4(Q^{(i)}_{j-1}) \\
& + M^{(i)}_{(j-16)\bmod 16} + M^{(i)}_{(j-13)\bmod 16} - M^{(i)}_{(j-6)\bmod 16} + K_j
\end{aligned}
$$

在这里定数 K 的 $_j$ = J × (0x05555555)和 K 的 (32 bit 版) $_j$= J × (0x0555555555555555 () 的 64 bit 版). j = 16, 17, ..., 31

技能扩张 $_1$ 及扩张 $_2$ 在计算函数 F 的阶段上使用的 $_1$ quad tube 的膨胀阶段，也就是说第一个函数 expand $_1$ 是 比第二个函数更复杂，但提供更好的结果。

函数 f $_0$:

Биективное отображение ***A1(M(i) xor H(i))***

$$
\begin{aligned}
W_0^{(i)} &= (M_5^{(i)} \oplus H_5^{(i-1)}) - (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)}) + (M_{14}^{(i)} \oplus H_{14}^{(i-1)}) \\
W_1^{(i)} &= (M_6^{(i)} \oplus H_6^{(i-1)}) - (M_8^{(i)} \oplus H_8^{(i-1)}) + (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) + (M_{14}^{(i)} \oplus H_{14}^{(i-1)}) - (M_{15}^{(i)} \oplus H_{15}^{(i-1)}) \\
W_2^{(i)} &= (M_0^{(i)} \oplus H_0^{(i-1)}) + (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_9^{(i)} \oplus H_9^{(i-1)}) - (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) + (M_{15}^{(i)} \oplus H_{15}^{(i-1)}) \\
W_3^{(i)} &= (M_0^{(i)} \oplus H_0^{(i-1)}) - (M_1^{(i)} \oplus H_1^{(i-1)}) + (M_8^{(i)} \oplus H_8^{(i-1)}) - (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)}) \\
W_4^{(i)} &= (M_1^{(i)} \oplus H_1^{(i-1)}) + (M_2^{(i)} \oplus H_2^{(i-1)}) + (M_9^{(i)} \oplus H_9^{(i-1)}) - (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) - (M_{14}^{(i)} \oplus H_{14}^{(i-1)}) \\
W_5^{(i)} &= (M_3^{(i)} \oplus H_3^{(i-1)}) - (M_2^{(i)} \oplus H_2^{(i-1)}) + (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) - (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) + (M_{15}^{(i)} \oplus H_{15}^{(i-1)}) \\
W_6^{(i)} &= (M_4^{(i)} \oplus H_4^{(i-1)}) - (M_0^{(i)} \oplus H_0^{(i-1)}) - (M_3^{(i)} \oplus H_3^{(i-1)}) - (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)}) \\
W_7^{(i)} &= (M_1^{(i)} \oplus H_1^{(i-1)}) - (M_4^{(i)} \oplus H_4^{(i-1)}) - (M_5^{(i)} \oplus H_5^{(i-1)}) - (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) - (M_{14}^{(i)} \oplus H_{14}^{(i-1)}) \\
W_8^{(i)} &= (M_2^{(i)} \oplus H_2^{(i-1)}) - (M_5^{(i)} \oplus H_5^{(i-1)}) - (M_6^{(i)} \oplus H_6^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)}) - (M_{15}^{(i)} \oplus H_{15}^{(i-1)}) \\
W_9^{(i)} &= (M_0^{(i)} \oplus H_0^{(i-1)}) - (M_3^{(i)} \oplus H_3^{(i-1)}) + (M_6^{(i)} \oplus H_6^{(i-1)}) - (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_{14}^{(i)} \oplus H_{14}^{(i-1)}) \\
W_{10}^{(i)} &= (M_8^{(i)} \oplus H_8^{(i-1)}) - (M_1^{(i)} \oplus H_1^{(i-1)}) - (M_4^{(i)} \oplus H_4^{(i-1)}) - (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_{15}^{(i)} \oplus H_{15}^{(i-1)}) \\
W_{11}^{(i)} &= (M_8^{(i)} \oplus H_8^{(i-1)}) - (M_0^{(i)} \oplus H_0^{(i-1)}) - (M_2^{(i)} \oplus H_2^{(i-1)}) - (M_5^{(i)} \oplus H_5^{(i-1)}) + (M_9^{(i)} \oplus H_9^{(i-1)}) \\
W_{12}^{(i)} &= (M_1^{(i)} \oplus H_1^{(i-1)}) + (M_3^{(i)} \oplus H_3^{(i-1)}) - (M_6^{(i)} \oplus H_6^{(i-1)}) - (M_9^{(i)} \oplus H_9^{(i-1)}) + (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) \\
W_{13}^{(i)} &= (M_2^{(i)} \oplus H_2^{(i-1)}) + (M_4^{(i)} \oplus H_4^{(i-1)}) + (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) + (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) \\
W_{14}^{(i)} &= (M_3^{(i)} \oplus H_3^{(i-1)}) - (M_5^{(i)} \oplus H_5^{(i-1)}) + (M_8^{(i)} \oplus H_8^{(i-1)}) - (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) - (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) \\
W_{15}^{(i)} &= (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) - (M_4^{(i)} \oplus H_4^{(i-1)}) - (M_6^{(i)} \oplus H_6^{(i-1)}) - (M_9^{(i)} \oplus H_9^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)})
\end{aligned}
$$

Дальнейшее отображение ***A2(A1(M(i) xor Hli))***

$$
\begin{aligned}
Q_0^{(i)} &= s_0(W_0^{(i)}); & Q_1^{(i)} &= s_1(W_1^{(i)}); & Q_2^{(i)} &= s_2(W_2^{(i)}); & Q_3^{(i)} &= s_3(W_3^{(i)}); \\
Q_4^{(i)} &= s_4(W_4^{(i)}); & Q_5^{(i)} &= s_0(W_5^{(i)}); & Q_6^{(i)} &= s_1(W_6^{(i)}); & Q_7^{(i)} &= s_2(W_7^{(i)}); \\
Q_8^{(i)} &= s_3(W_8^{(i)}); & Q_9^{(i)} &= s_4(W_9^{(i)}); & Q_{10}^{(i)} &= s_0(W_{10}^{(i)}); & Q_{11}^{(i)} &= s_1(W_{11}^{(i)}); \\
Q_{12}^{(i)} &= s_2(W_{12}^{(i)}); & Q_{13}^{(i)} &= s_3(W_{13}^{(i)}); & Q_{14}^{(i)} &= s_4(W_{14}^{(i)}); & Q_{15}^{(i)} &= s_0(W_{15}^{(i)});
\end{aligned}
$$

函数 f_1 ：

ExpandRound1 及 ExpandRound2 参数是依据在上面说明的扩张 $_1$ 及扩张 $_2$ 算法，决定反复次数。

```
For (j=0;j<ExpandRound1;j++)

  Q(i,j)=expand1(j+16);

For (j=ExpandRound1;j<ExpandRound2+ExpandRound1;j++)

  Q(i,j)=expand2(j+16);
```

F 2 技能：

1. 添加变数 XL 及 XH 计算

$$XL = Q_{16}^{(i)} \oplus Q_{17}^{(i)} \oplus \cdots \oplus Q_{23}^{(i)}$$
$$XH = XL \oplus Q_{24}^{(i)} \oplus Q_{25}^{(i)} \oplus \cdots \oplus Q_{31}^{(i)}$$

2. 计算 H (i)的新值。

$$\begin{aligned}
H_0^{(i)} &= (SHL^5(XH) \otimes SHR^5(Q_{16}^{(i)}) \otimes M_0^{(i)}) + (XL \otimes Q_{24}^{(i)} \otimes Q_0^{(i)}\\
H_1^{(i)} &= (SHR^7(XH) \otimes SHL^8(Q_{17}^{(i)}) \otimes M_1^{(i)}) + (XL \otimes Q_{25}^{(i)} \otimes Q_1^{(i)}\\
H_2^{(i)} &= (SHR^5(XH) \otimes SHL^5(Q_{18}^{(i)}) \otimes M_2^{(i)}) + (XL \otimes Q_{26}^{(i)} \otimes Q_2^{(i)}\\
H_3^{(i)} &= (SHR^1(XH) \otimes SHL^5(Q_{19}^{(i)}) \otimes M_3^{(i)}) + (XL \otimes Q_{27}^{(i)} \otimes Q_3^{(i)}\\
H_4^{(i)} &= (SHR^3(XH) \otimes Q_{20}^{(i)} \otimes M_4^{(i)}) + (XL \otimes Q_{28}^{(i)} \otimes Q_4^{(i)}\\
H_5^{(i)} &= (SHL^6(XH) \otimes SHR^6(Q_{21}^{(i)}) \otimes M_5^{(i)}) + (XL \otimes Q_{29}^{(i)} \otimes Q_5^{(i)}\\
H_6^{(i)} &= (SHR^4(XH) \otimes SHL^6(Q_{22}^{(i)}) \otimes M_6^{(i)}) + (XL \otimes Q_{30}^{(i)} \otimes Q_6^{(i)}\\
H_7^{(i)} &= (SHR^{11}(XH) \otimes SHL^2(Q_{23}^{(i)}) \otimes M_7^{(i)}) + (XL \otimes Q_{31}^{(i)} \otimes Q_7^{(i)}\\
H_8^{(i)} &= ROTL^9(H_4^{(i)}) + (XH \otimes Q_{24}^{(i)} \otimes M_8^{(i)}) + (SHR^8(XL) \otimes Q_{23}^{(i)} \otimes Q_8^{(i)}\\
H_9^{(i)} &= ROTL^{10}(H_5^{(i)}) + (XH \otimes Q_{25}^{(i)} \otimes M_9^{(i)}) + (SHR^6(XL) \otimes Q_{16}^{(i)} \otimes Q_9^{(i)}\\
H_{10}^{(i)} &= ROTL^{11}(H_6^{(i)}) + (XH \otimes Q_{26}^{(i)} \otimes M_{10}^{(i)}) + (SHR^6(XL) \otimes Q_{17}^{(i)} \otimes Q_{10}^{(i)}\\
H_{11}^{(i)} &= ROTL^{12}(H_7^{(i)}) + (XH \otimes Q_{27}^{(i)} \otimes M_{11}^{(i)}) + (SHR^4(XL) \otimes Q_{18}^{(i)} \otimes Q_{11}^{(i)}\\
H_{12}^{(i)} &= ROTL^{13}(H_0^{(i)}) + (XH \otimes Q_{28}^{(i)} \otimes M_{12}^{(i)}) + (SHR^3(XL) \otimes Q_{19}^{(i)} \otimes Q_{12}^{(i)}\\
H_{13}^{(i)} &= ROTL^{14}(H_1^{(i)}) + (XH \otimes Q_{29}^{(i)} \otimes M_{13}^{(i)}) + (SHR^4(XL) \otimes Q_{20}^{(i)} \otimes Q_{13}^{(i)}\\
H_{14}^{(i)} &= ROTL^{15}(H_2^{(i)}) + (XH \otimes Q_{30}^{(i)} \otimes M_{14}^{(i)}) + (SHR^7(XL) \otimes Q_{21}^{(i)} \otimes Q_{14}^{(i)}\\
H_{15}^{(i)} &= ROTL^{16}(H_3^{(i)}) + (XH \otimes Q_{31}^{(i)} \otimes M_{15}^{(i)}) + (SHR^2(XL) \otimes Q_{22}^{(i)} \otimes Q_{15}^{(i)}
\end{aligned}$$

散列函数的最终值

计算 H (N)的最终值后，需要选择对应散列函数 *Hash = Hash (H (N))* 值的 n bit。

- 散列 = H (N, 8) || H (N, 9) || ... || H (n, 15) - 版 A3 散列 256 及 A3 散列 512 用
- 散列 = H (N, 10) || H (N, 11) || ... || H (n, 15) - 版 A3 散列 384
- 散列 = H (N, 9) || H (N, 10) || ... || H (n, 15) - 版 A3 散列 224

A5 散列
具有 128 bit，使用宽阔的 pipe 构造，是以 ARX 基础的算法。短信区块是 128 bit 的初期 bit，形成 XOR 经过区块之间的 r round 全体变换阶段。初期在 NIST 提案书（"A5 散列 8 / 1"）是每 bit 需要约 200cycle。创作人是通过 NIST 的说明，把提案书在 "A5 散列比 8 / 1 快 16 倍，在参照平台中变更为 "A5 散列 16 / 32". 主要推荐事项是 A5 散列 512，定义为 A5 散列 16 + 16 / 32 + 32-512。

* 启动原理

有 5 个参数，特定 变量是表示为 A5 散列 `i + r / b + f - h`。

- `i` 是初期 round 数。

- **r** 是每区块 round 数。
- **b** 是对 {1, 2, 3, ... 128}所定义的区块大小（bytes）
- **f** 是最终 round 数。
- **h** 是对 {8, 16, 24, 32, ... 512}所定义的散列输出的 bit 大小。

在日语 NIST, **I** 及 **F是在** (10)固定的**R** . 标记法 A5 散列**r** / **b** - **h** 是 **i** 和 **f** 暗示为 10 **r。**

内部状态在两个层次上都被定义为 0-1 单词的 5 个层次排列(4bytes 定数)。 单词是参照坐标 [00000] ~ [11111]。 单词被视为 little endian。

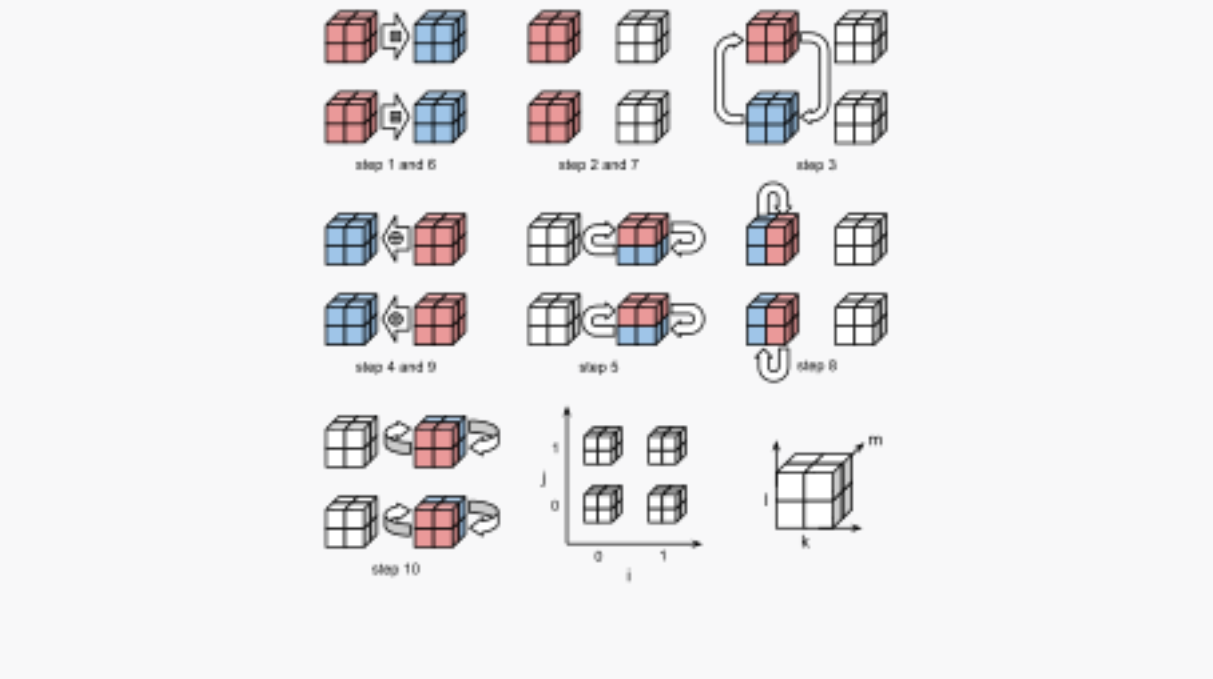
内部状态是把最初三个词([000000],[00001],[00010]分别设定为 h / 8,b 及 r ,把其他所有词设定为 0,并进行初始化。 然后通过 i round 实施,完成初始化阶段。 状态是初始化矢量(IV)。 IV 是可存储 h,b ,r 的组合并进行再使用。

信息被分成 b byte 区块。 增加 1bit 后,根据需要增加 0 位数,形成完整的区块。

各区块是把首个 b bytes 用 XOR 输入后修行下个 r round。

输出散列包含在最终状态的第一个 h/8bytes。

ROUND 技能



Mixing 技能的 10 阶段。在 5 个次元中展开 2 种。
A5 散列 round 技能是构成为下面 10 个阶段。

1. 对各个 (j, k, l, m)添加 x [**1** jklm] modulo 2 ^{32}에 x [**0** jklm]。
2. 对各个 (j, k, l, m)把 x [**0** jklm]往上旋转 7 bit
3. 对各个 (k, l, m)把 x [**00** klm]转换为 x [**01** klm]
4. 以 X [**1** jklm] X, [**0** 对各个 (J, K, L, M) jklm].
5. X [**1** JK **0** X [m] **1** JK **1** 对各个 (j, K, m)的 m].
6. 对各个 (j, k, l, m)添加 x [**1** jklm] modulo 2 ^{32}에 x [**0** jklm]
7. 对各个 (j, k, l, m)把 x [**0** jklm]往上旋转 11 bit

8. 对各个 (j, l, m)把 x[0 j 0 lm]转换为 x[0 j 1 lm]
9. 以 X[1 jklm] X[0 对各个(J, K, L, M)的 jklm].
10. 对各个 (j, k, l)把 x[1 jkl 0]转换为 x[1 jkl 1]

散列例子

在这例子中使用 A5 散列 80 + 8 / 1 + 80-512。 初始化矢量在所有 80 + 8 / 1 + f - 512 散列中都相同,内容如下。

```
5df39869c73009fb108994600f1626e6f37c07360c0d8bb53d19cf57b8e74133 \
5b8034a3eff9892014c4ff315038ef2a391812fe52a440e9a293527d12ca4570  \
6e0958933470bf814aa4909adb3ec39384e9c314d0db874af21d45bcacb31252  \
1ce5ab6a3bf6f05de88abbdd0fcfd3fafb8225d546242eada52540095c3da221
```

散列 ASCII 短信 "Hello"(16 真数 : 0x48, 0x65, 0x6c, 0x6c, 0x6f)是使用 6 个短信区块。在短信中有 5 区块，是 bit 排列的输入，有 1 个为了 padding 区块。 512 bit 散列值以下而同。

```
7ce309a25e2e1603ca0fc369267b4d43f0b1b744ac45d6213ca08e7567566444 \
8e2f62fdbf7bbd637ce40fc293286d75b9d09e8dda31bd029113e02ecccfd39b
```

若短信有一点改变的话因 Avalanche effect 效果，大大变更散列输出。 "hello"(在 1 bit 位置与"Hello"的不同点) 散列短信提供下列散列值

```
01ee7f4eb0e0ebfdb8bf77460f64993faf13afce01b55b0d3d2a63690d25010f \
7127109455a7c143ef12254183e762b15575e0fcc49c79a0471a970ba8a66638
```

变更参数

A5 散列是许可在决定散列输出时使用的多种多样参数。使用者决定需要使用的参数。下面是使用相互不同的参数短信的许多散列事例。短信都为 ASCII。

```
message: ""  (the zero-length string)
A5 散列 160+16/32+160-512:
4a1d00bbcfcb5a9562fb981e7f7db3350fe2658639d948b9d57452c22328bb32\

f468b072208450bad5ee178271408be0b16e5633ac8a1e3cf9864cfbfc8e043a

A5 散列 80+8/1+80-512:
90bc3f2948f7374065a811f1e47a208a53b1a2f3be1c0072759ed49c9c6c7f28\

f26eb30d5b0658c563077d599da23f97df0c2c0ac6cce734ffe87b2e76ff7294

A5 散列 10+1/1+10-512:
3f917707df9acd9b94244681b3812880e267d204f1fdf795d398799b584fa8f1\

f4a0b2dbd52fd1c4b6c5e020dc7a96192397dd1bce9b6d16484049f85bb71f2f
```

A5 散列 160+16/32+160-256:
44c6de3ac6c73c391bf0906cb7482600ec06b216c7c54a2a8688a6a42676577d

A5 散列 80+8/1+80-256:
38d1e8a22d7baac6fd5262d83de89cacf784a02caa866335299987722aeabc59

A5 散列 10+1/1+10-256:
80f72e07d04ddadb44a78823e0af2ea9f72ef3bf366fd773aa1fa33fc030e5cb
message: "Hello"
A5 散列 160+16/32+160-512:
dcc0503aae279a3c8c95fa1181d37c418783204e2e3048a081392fd61bace883\

a1f7c4c96b16b4060c42104f1ce45a622f1a9abaeb994beb107fed53a78f588c

A5 散列 80+8/1+80-512:

7ce309a25e2e1603ca0fc369267b4d43f0b1b744ac45d6213ca08e7567566444\

8e2f62fdbf7bbd637ce40fc293286d75b9d09e8dda31bd029113e02ecccfd39b

A5 散列 10+1/1+10-512:

13cf99c1a71e40b135f5535bee02e151eb4897e4de410b9cb6d7179c677074eb\

6ef1ae9a9e685ef2d2807509541f484d39559525179d53838eda95eb3f6a401d

A5 散列 160+16/32+160-256:

e712139e3b892f2f5fe52d0f30d78a0cb16b51b217da0e4acb103dd0856f2db0

A5 散列 80+8/1+80-256: 692638db57760867326f851bd2376533f37b640bd
47a0ddc607a9456b692f70f

A5 散列 10+1/1+10-256:
f63041a946aa98bd47f3175e6009dcb2ccf597b2718617ba46d56f27ffe35d49
message: "The quick brown fox jumps over the lazy dog"
A5 散列 160+16/32+160-512:
bdba44a28cd16b774bdf3c9511def1a2baf39d4ef98b92c27cf5e37beb8990b7\

cdb6575dae1a548330780810618b8a5c351c1368904db7ebdf8857d596083a86

A5 散列 80+8/1+80-512:

ca942b088ed9103726af1fa87b4deb59e50cf3b5c6dcfbcebf5bba22fb39a6be\

9936c87bfdd7c52fc5e71700993958fa4e7b5e6e2a3672122475c40f9ec816ba

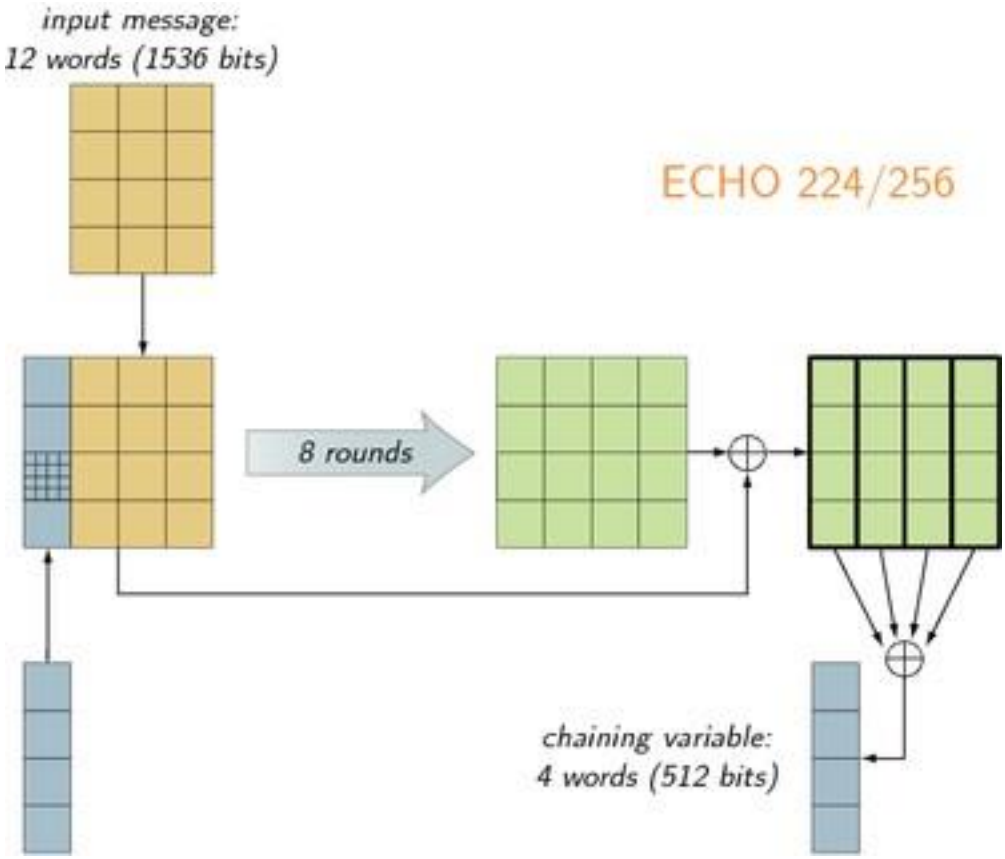
```
A5 散列 10+1/1+10-512:
eb7f5f80706e8668c61186c3c710ce57f9094fbfa1dbdc7554842cdbb4d10ce4\

2fce72736d10b152f6216f23fc648bce810a7af4d58e571ec1b852fa514a0a8e

A5 散列 160+16/32+160-256:
5151e251e348cbbfee46538651c06b138b10eeb71cf6ea6054d7ca5fec82eb79

A5 散列 80+8/1+80-256:
94e0c958d85cdfaf554919980f0f50b945b88ad08413e0762d6ff0219aff3e55

A5 散列 10+1/1+10-256:
217a4876f2b24cec489c9171f85d53395cc979156ea0254938c4c2c59dfdf8a4
```

关于表示的四种变形初始化矢量都不同。

例如 A5 散列 80 + 8 / 1 + 80-512 的初始化矢量是如上图一样， A5 散列 80 + 8 / 1 + 80-256 的 IV 是以下而同。

```
830b2bd5273d616fd785876a4a500218a5388963eeb702fb47547842459f8d89 \
8727a1c8ba40bd48cef47fe82543c2735c033052ae9fcd632d4541bde6b6cb0d \
cb8a9cdf579f5b67b2ae00968180af6e51ebdf0ca597cd2bf91f7ab29a62 \  01a
d72d946e6c075c6d1337e0a293d6f90c438ac38be153f32aa288ffc5eca8a
```

A6 散列

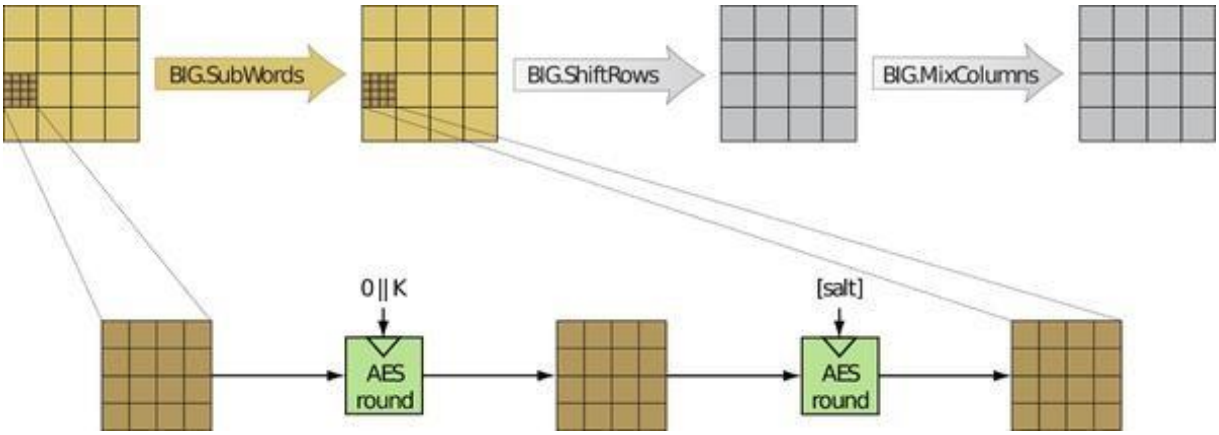


是散列函数的上位水平例子，ROUND 技能是在下图有说明。

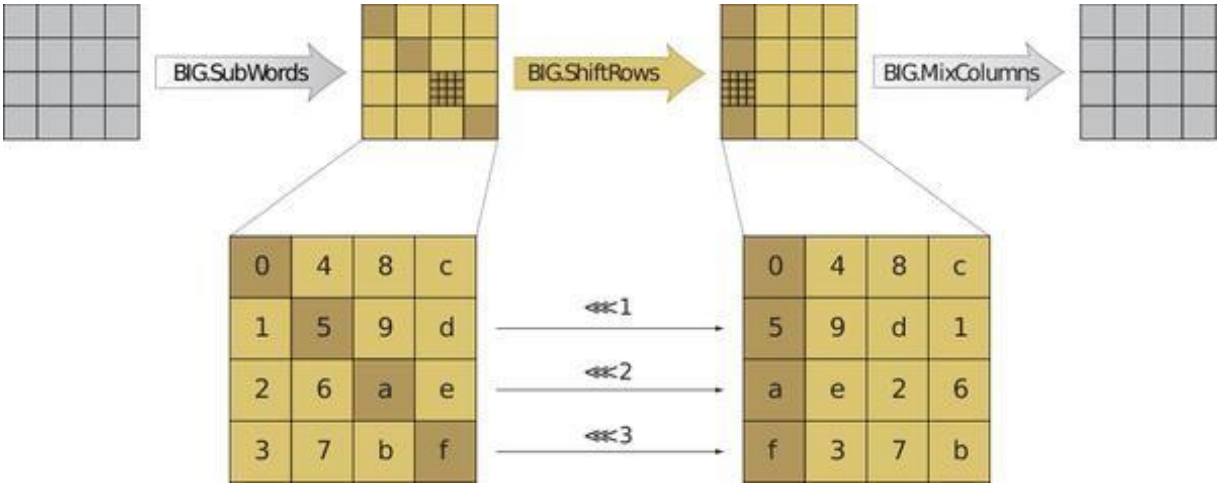
ROUND 技能

A6 散列的 ROUND 技能是带来 AES 构造，按照下图一样。

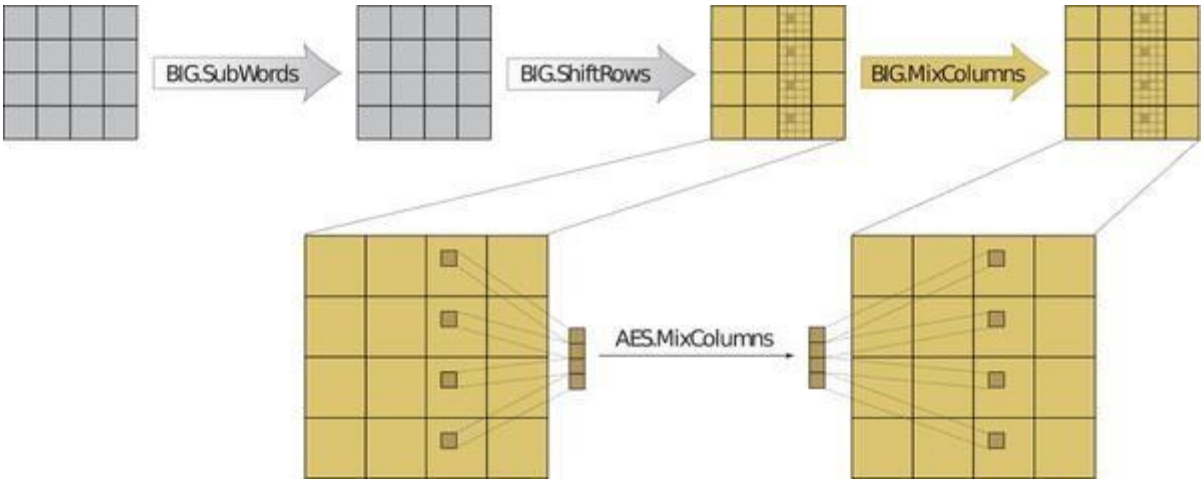
在 16 单词各个适用 2 个 AES ROUND 的 BIG.SubWords 变换 ：



- 模仿 AES 的 ShiftRows，模仿 128bit 单词的 BIG.ShiftRows 变换：



- BIG.MixColumns 变换是通过 AES MixColumns 状态，适用在 bytes 的 4 TYUPLE。



A7
与 D5 / SHA 系列的其它散列函数一样，把输入分成区块，反复计算 h i = f (h i -1 , m i)。但 Grøstl 是把散列状态维持为最终输出(512 或 1024 bit)大小的两倍以上。 散列状态是在散列结算的末端被删除。 压缩函数 f 是以一双 256 或 512bit 函数 P 和 Q 为基础，如下定义。 f (h , m) = P (h + m) + Q (m) + h 顺列函数 P 和 Q 是以 Rijndael (AES) 区块密码为基础，但比 4x4，启动在 bite 的 8x8 或 8x16 排列。与 AES 一样，各 ROUND 构成为四种工作。 AddRoundKey (Grøstl ROUND 键是被固定，但 P 和 Q 相互不同) SubBytes (使用 Rijndael S-box 可与 AES 分享) ShiftBytes (与 AES 比较后扩张，在 P 和 Q，512 bit 及 1024 bit 版上也有差异) MixColumns (比 Rijndael 的 4x4 更常使用 8x8) 与 Rijndael 不同，所有 ROUND 都统一，没有最终 AddRoundKey 工作。512bit 顺列是推荐 10ROUND，1024bit 版推荐 14ROUND。最后两倍宽度散列是收到如下最终输出变换。Ω (h) = h + P (h) 被裁剪为所需要的宽度。这都是使用了 0 短信区块 m，适用压缩函数的最终反复后，跟随固定的定数 Q (0) 和排他性 (不是很重要) 值。

散列例子

空文字列的散列值

```
Grøstl-224("")
```

```
0x f2e180fb5947be964cd584e22e496242c6a329c577fc4ce8c36d34c3
Grøstl-256("")
0x 1a52d11d550039be16107f9c58db9ebcc417f16f736adb2502567119f0083467
Grøstl-384("")
0x  ac353c1095ace21439251007862d6c62f829ddbe6de4f78e68d310a9205a736d8b11
d99 bffe448f57a1cfa2934f044a5
Grøstl-512("")
0x
6d3ad29d279110eef3adbd66de2a0345a77baede1557f5d099fce0c03d6dc2ba8e6d4a6
633dfbd66053c20faa87d1a11f39a7fbe4a6c2f009801370308fc4ad8
```

尽管短信的小小变化，因 Avalanche effect 效果，几乎都会获得其它散列。
例如在文章末端添加句号时：

```
Grøstl-256("The quick brown fox jumps over the lazy dog.")
0x 8c7ad62eb26a21297bc39c2d7293b4bd4d3399fa8afab29e970471739e28b301
Grøstl-256("The quick brown fox jumps over the lazy dog.")
0x f48290b1bcacee406a0429b993adb8fb3d065f4b09cbcdb464a631d4a0080aaf
```

Hmac

定义

$$\mathrm{HMAC}(K,m) = \mathrm{H}\left(\left(K' \oplus opad\right) \parallel \mathrm{H}\left(\left(K' \oplus ipad\right) \parallel m\right)\right)$$

$$K' = \begin{cases} \mathrm{H}(K) & K \text{ is larger than block size} \\ K & \text{otherwise} \end{cases}$$

H 是密码化散列函数。
m 是将要被认证的短信。
K 是密码键。
K' 是从密码键 *K* 导出的区块大小的键。从区块大小往 0 向右侧 padding。
// 表示连接
⊕是 bit 单位的排他性逻辑综合 (XOR)
opad 是区块大小的外部 padding，构成为 0x5c 的值反复形成的 bytes。
ipad 是区块大小的内部 padding，构成为 0x36 的值反复形成的 bytes。

实现

下面展现 HMAC 的实现反复。区块大小是在使用 SHA-1, MD5, A14 散列-128 / 160 散列函数之一时的 64 (bit)。

```
Function hmac
   Inputs:
      key:         Bytes      //array of bytes
```

```
      message:    Bytes     //array of bytes to be hashed
      hash:       Function  //the hash function to use (e.g. SHA-1)
      blockSize:  Integer   //the block size of the underlying hash
function (e.g. 64 bytes for SHA-1)
      outputSize: Integer   //the output size of the underlying hash
function (e.g. 20 bytes for SHA-1)

   //Keys longer than blockSize are shortened by hashing them
   if (length(key) > blockSize) then
      key ← hash(key) //Key becomes outputSize bytes long

   //Keys shorter than blockSize are padded to blockSize by padding
with zeros on the right
   if (length(key) < blockSize) then
      key ← Pad(key, blockSize) //pad key with zeros to make it
blockSize bytes long

   o_key_pad = key xor [0x5c * blockSize]    //Outer padded key
   i_key_pad = key xor [0x36 * blockSize]    //Inner padded key

   return hash(o_key_pad ∥ hash(i_key_pad ∥ message)) //Where ∥ is c
oncatenation
```

例如
下面是部分空置的 HMAC 值。

```
HMAC_MD5 ( "", "") = 74e6f7298a9c2d168935f58c001bad88  HMAC_SH
A1 ( "", "") = fbdb1d1b18aa6c08324b7d64b71fb76370690e1d 8 号散列
( "", "") =
b613679a0814d9ec772f95d778c35fc5ff1697c493715653c6c712144292c5ad
```

下面是假设 8 bit ASCII 或 UTF-8 编码，是未空置的 HMAC 值。

```
HMAC_MD5("key", "The quick brown fox jumps over the lazy dog")     =
80070713463e7749b90c2dc24911e275
HMAC_SHA1("key", "The quick brown fox jumps over the lazy dog")    =
de7c9b85b8b78aa6bc8a7a36f70a90701c9db4d9
8 号散列 ("key", "The quick brown fox jumps over the lazy dog") =
f7bc83f430538424b13298e6aa6fb143ef4d59a14946175997479dbc2d1a3cd8
```

A8 散列 A
8-1 散列
A8-2 散列
A8-3 散列

A8-4 散列
A8-5 散列

（在这里 A8-1 散列 ~ A8-5 散列是 A8 散列的连动散列）

A9 散列
具有 1024 bit 状态，在 512 bit 输入区块中启动。
输入区块处理是构成为下面 3 个阶段

1. 把输入区块状态的左侧一半为 XOR

2. 在适当状态上适用 42 ROUND 没有键的顺列（密码化技能） 。这构成为下面 42 反复。

1.把输入分割为 256 个 4 bit 区块，在两个 4bitS-box 中通过一个各个连接。选择是以 256bit round 键来实现。把各个输入区块与键 bit 结合，通过 5 → 4 bit S- box 结合此结果。
2. 关于 GF (2 4)使用最大距离分离代码，混合临近的 4bit 区块。
3. 在下个 ROUND 为了能够接近相互不同的区块，指定 4bit 区块。

3. 把输入区块状态的右侧一半为 XOR

结果文摘在 1024 bit 最终值，是第一个 224, 256, 384 或 512 bit。 SSE2
命令使用 set，适合与实现 bit 分割，每 bytes 提供 16.8 cycle 的速度。

A13 散列, A13-1 散列 (SHA-3)
Padding
为了使信息均匀分割成 rbit 区块,就需要 padding。 SHA-3 在 padding 功能中使用图案 10 * 1。 在 1 bit 后面出现 0bit 以上(最大 r - 1)及最后 1bit。 r - 1 zero bit 的最大值在最后一个信息区块为 r - 1 bit 长度时出现。 然后在最后 1bit 后添加包含 r -1 zerobit 的其它区块。 两个 1bit 在信息长度已经可分为 r 的情况下也可以添加。 [24] :5.1 在这种情况下,其他区块被添加到信息中,包括 1 bit,r -2 zero 区块及其他 1 bit。 这是为了不生成与 padding 相同的长度的长度"r"的信息与"Bit"相同的散列。 一开始需要 1bit,因此在末端的几个追加 0bit 的其他信息不会生成相同的散列。 最后 1bit 的位置是显示使用的比率 r(多重速度 padding),对其他散列变形需要安全证明。 若不存在的话，同样短信息的其他散列变种在切割之前都会是一样的。

区块顺列

对 SHA-3 的 1A3 散列-f [1600] 区块变换 f 是使用 **XOR**, **AND** 及 NOT 连算的顺列，设计为能够在软件或硬件中容易被实现。

这是定义 2 的 2 单词大小， $w = 2^{\ell}$ bit。主要 SHA-3 提出是使用 64 bit 单词， $\ell = 6$。

能够考虑状态的 $5 \times 5 \times w$ bit 的排列。使用主要 indexing，假设 $a[i][j][k]$ 是输入的 bit$(5i + j) \times w + k$ 。也就是说， 选择*前*。 把 J 列, k *为 bit*

INDEX 算术是第一第二次元修行为 5，第三次元修行为 w。

基本区块交替技能是构成为 $12 + 2\ell$ 的 5 个阶段。

θ

5 w (320, w = 64 时) 计算5 bit列的各平价，计算对正规图案的两个临近列的排他性逻辑综合。详细说明的话， $a[i][j][k] \oplus a[i][j][k] \oplus$ 平价 (a [0 ... 4] [j -1] [k]) $\oplus$ 平价 (a [0 ... 4] [j +1] [k -1])

ρ

把25 个单词各个以不同的三角形数 0, 1, 3, 6, 10, 15, ...旋转为bit 单位。详细说明的话 [a] [0]不旋转，$0 \leqslant t < 24$ ， [I]是 [j是 [K是] ← [I] [J] [$K - (t+1)(t+2)/2$] ，

π

把25 单词移动到固定的图案。$a[j][2i+3j] \leftarrow a[i][j]$

χ

使用$x \leftarrow x \oplus (\neg y \ \& \ z)$按照行结合bit。详细说明的话， [I] [J] [k是] ← [I] [J] [K] $\oplus$ (¬ [I] [$J+1$] [K] [I] [j是 $+2$] [k]) . 这是 SHA-3 的唯一非线性连算。

ι

以排他性或定数表现状态的一个单词。若 $0 \leqslant m \leqslant \ell$ 时，在round n 中 [0] [0] [2 〃 -1] 是 8 LFSR 的bit$m + 7n$ 与 XOR 进行连算。这样的话，依据其他阶段被保存的对称性会被打破。

变量

对短信 M 及输出长度 d，定义下面变量。

例如	输出大小 $= d$	比率 R = 区块大小	容量 c	定义	Bit 单位的保安强度		
					冲突	FREE 图片	第二个图片
SHA3-224 (M)	224	1152	448	A13 散列 [448] (M || 01, 224)	112	224	224
SHA3-256 (M)	256	1088	512	A13 散列 [512] (M || 01, 256)	128	256	256
SHA3-384 (M)	384	832	768	A13 散列 [768] (M || 01,	192	384	384

				384)			
SHA3-512 (*M*)	512	576	1024	A13 散列 [1024] (*M* || 01, 512)	256	512	512
SHAKE128 (*M* , *d*)	D	1344	256	A13 散列 [256] (*M* || 1111, *d*)	分 *d* /2,128)	≥min(*d* ,128)	分(*d* , 128)
SHAKE256 (*M* , *d*)	D	1088	512	A13 散列 [512] (*M* || 1111, *d*)	分 (*d* /2,256)	≥min(*d* ,256)	分(*d* , 256)

- A13 散列 [*c*] (*N* , *d*) = sponge [A13 散列-f [1600], pad10 * 1, *r*] (*N* , *d*)
- A13 散列-f [1600] = A13 散列-p [1600, 24]
- *c* 是容量
- *r* 是 比率 = 1600- *c*。
- *N* 是输入 bit 文字列。

增加的接尾语不是 16 码,而是用 bit 文字列来写作。

SHA-3 变 量 是与相同的安全索赔一起作为 SHA-2 的 Drop In 替代品。 Shake instance 是 XOF 的可扩展的输出函数。 例如,SHAKE128 (M, 256)可使用为 256 bit 长度和 128 bit 整体具有安全的散列函数。

所有变量在信息中添加几个 bit,右侧显示域名分离接尾词。 目的是,不能对 1A3 散列函数的各种应用程序构成生成同一散列输出的信息。 有如下域名分离接尾词:

结尾词	意义
... 0	为了向后使用已预约

01	SHA-3
... 11	RawSHAKE

RawSHAKE 是尚未标准化的为了进行散列的 Sakura 涂层的基础。 但 SHAKE 的接尾词是为了能够与 Sakura 互换是被慎重选择的。 Sakura 在适用 RawSHAKE 之前,在链条 Hop 添加 0,或在信息中添加 1,然后对非最终(内部)节点添加 10 * 0 或对最终节点添加 1。 依次散列是相当于拥有单一信息的 Hop Tree。即,在 RawSHAKE 适用之前,在信息添加 11。 因此,SHAKE XOF 在信息中添加 1111,在信息中添加 1,在最终节点中添加 1,在 RawSHAKE 域名分离接尾语中添加 11。

10 * 1 padding 常添加至少 2 bit,因此,在 bit 排列文件中经常有 6 个未使用的 0bit。 因此,这样添加的 bit 使短信变得更长。

添加变量

在 SHA-3 派生的添加技能能在下列表确认。

例如	技术
cSHAKE128 (*X* , *L* , *N* , *S*)	通过参数支持明确域名分离的 SHAKE 版本。.
cSHAKE256 (*X* , *L* , *N* , *S*)	
KMAC128 (*K* , *X* , *L* , *S*)	以 A13 散列为基础的键顺序散列函数。用一般散列没有键也可以使用。
KMAC256 (*K* , *X* , *L* , *S*)	
KMACXOF128 (*K* , *X* , *L* , *S*)	
KMACXOF256 (*K* , *X* , *L* , *S*)	
TupleHash128 (*X* , *L* , *S*)	这是散列文字列 TUPLE 的函数。这函数的输出是按照输入文字列的内容和顺序不同。
TupleHash256 (*X* , *L* , *S*)	
TupleHashXOF128 (*X* , *L* , *S*)	

TupleHashXOF256 (*X* , *L* , *S*)	
ParallelHash128 (*X* , *B* , *L* , *S*)	为了更快的散列，是为了能够在最新工程使用并列处理来设计的函数。 与 KangarooTwelve 不同，不使用被缩小的 A13 散列。
ParallelHash256 (*X* , *B* , *L* , *S*)	
ParallelHashXOF128 (*X* , *B* , *L* , *S*)	
ParallelHashXOF256 (*X* , *B* , *L* , *S*)	

对量子攻击的保安

一般的计算机(Grover 算法)可以进行量子计算机 √ 2 d = 2 d / 2 结构化的自由图像攻击,而古典的 brute-force 攻击则需要 d2。 结构化自由图像攻击意味着第二次自由图像攻击和冲突攻击。 量子计算机还根据特定指定日期进行攻击,破坏碰撞耐性,3 √ 2 D = 2 D / 3 . 若最大强度为 c / 2 时对 SHA-3 的量子保安提供下列上位界限:

例如	Bit 单位的保安强度			
	冲突 (Brassard et al.)	冲突 (burnstarin)	自由图像	第二个图像
SHA3-224 (*M*)	74/3/3	112	112	112
SHA3-256 (*M*)	85 1/3	128	128	128
SHA3-384 (*M*)	128	192	192	192
SHA3-512 (*M*)	170/3/3	256	256	256
SHAKE128 (*M* , *d*)	分 (*d* / 3,128)	分 (*d* / 2,128)	≥min (*d* / 2,128)	分 (*d* / 2,128)
SHAKE256 (*M* , *d*)	分 (*d* / 3,256)	分 (*d* / 2,256)	≥min (*d* / 2,256)	分 (*d* / 2,256)

SHA-3 的变形举例

```
SHA3-224("")
6b4e03423667dbb73b6e15454f0eb1abd4597f9a1b078e3f5b5a6bc7
SHA3-256("")
a7ffc6f8bf1ed76651c14756a061d662f580ff4de43b49fa82d80a4b80f8434a
SHA3-384("")
0c63a75b845e4f7d01107d852e4c2485c51a50aaaa94fc61995e71bbee983a2ac371383
1264adb47fb6bd1e058d5f004
SHA3-512("")
a69f73cca23a9ac5c8b567dc185a756e97c982164fe25859e0d1dcc1475c80a615b2123
af1f5f94c11e3e9402c3ac558f500199d95b6d3e301758586281dcd26
SHAKE128("", 256)
7f9c2ba4e88f827d616045507605853ed73b8093f6efbc88ebA16eacfa66ef26
SHAKE256("", 512)
46b9dd2b0ba88d13233b3feb743eeb243fcd52ea62b81b82b50c27646ed5762fd75dc4d
dd8c0f200cb05019d67b592f6fc821c49479ab48640292eacb3b7c4be
```

变更单一 bit，输出的各 bit 变更为显示 Avalanche effect 效果的 50%概率。

```
SHAKE128("The quick brown fox jumps over the lazy dog", 256)  f42
02e3c5852f9182a0430fd8144f0a74b95e7417ecae17db0f8cfeed0e3e66e SHA
KE128("The quick brown fox jumps over the lazy dof", 256)  853f45
38be0db962A16cea659a06c1107b1f83f02b13d18297bd39d7411cf10c
```

A14 散列-160 散列技能
不仅是 A14 散列-128,原来的 A14 散列是 128bit 结果太小,且因设计上的弱点(原来 A14 散列)导致安全问题,因此被视为不安全。 256 及 320 Bit 版本的 A14 散列是分别提供与 A14 散列-128 及 A14 散列-160 同等水平的保安。 安全水平足够,但设计为需要更长的散列结果的应用程序用。
160 Bit A14 散列-160 散列(RIPE 信息,亦称 Digest)是一般以 40 位数的 16 实数来表示。 下面显示 43Bytes ASCII 输入及相应 A14 散列-160 散列。

```
A14 散列-160("The quick brown fox jumps over the lazy dog") =
 37f332f68db77bd9d7edd4969571ad671cf9dd3b
```

A14 散列-160 是与密码化散列函数（小变更，例如把 d 变更为 c 的话成为完全不同的散列）的 everlanch 效果一起启动。

```
A14 散列-160("The quick brown fox jumps over the lazy cog") =
 132072df690933835eb8b6ad0b77e7b6f14acad7
```

长度为 0 的文字列的散列值如下。

```
A14 散列-160("") =
 9c1185a5c5e9fc54612808977ee8f548b2258d31
```

A14 散列
A14-1 散列
(A14-1 散列是 A14 散列连动散列)

A15 散列
区分对完全的散列函数的结果和 building 区块结果的两种情况。
推荐保安参数 : 12 round (n = 224,256); 14 round (n = 384,512)
散列技能
在这里我们罗列对散列函数的结果。唯一变更是变更保安参数。

分析类型	散列大小(n)	参数	压缩函数	储存要求事项
第二个图像	512	10 round	2^{497}	2^{16}
第二个图像	512	9 round	2^{496}	2^{16}

Building 区块
在这里我们是罗列对基本构成区块的结果和除了保安参数以外的方法来修正的散列函数。
这样的结果是通过部分内部变数（也就是说，FREE
START，意图，压缩机能，区块密码或顺列攻击）假设直接性控制或接近。

分析类型	散列函数部分	散列大小 (n)	参数/ 变形	压缩函数	储存要求事项
观测	散列	ALL			
自由图像	压缩	512	14 round	2^{384+s}	2^{128-s}
类似冲突	压缩	512	14 round	2^{192}	2^{128}
类似冲突	压缩	ALL	Full (1 round)		
类似冲突	压缩	256	Full (1 round)		
不可能的级别	区块密码	224,256	5 round	-	-
不可能的级别	区块密码	384,512	9 round	-	-

A16 散列

A16 散列是新的散列函数系列,已提交为 SHA-3 候选。 A16 散列是以熟悉的 Merkle-Damgård 设计为基础。压缩功能采用与 Davies-Meyer 模式的 Feistel 相同的密码。 但是这个设计中有创新。 内部状态是输出大小的两倍,使用强力的信息扩张,在压缩功能上使用经过修改的 Pid Poward。 这个运作模式从一般的攻击中安全。 A16 散列的最重要构成要素是信息扩展，设计为能够提供最小的距离。 通过这个方法,可以防止 A16 散列加差等密码的解读。 A16 散列在压缩功能内部以小小的并列处理为特征。 这可用于使用矢量指令(A/16 散列)来编制有效的体现。 这些指示可以在广泛使用的许多架构中使用。 :x86 的 SSE, PowerPC 的 Altivec,ARM 的 IwMMXt。因此 A16 散列在 NIST 参考平台上具有非常快的速度。

.

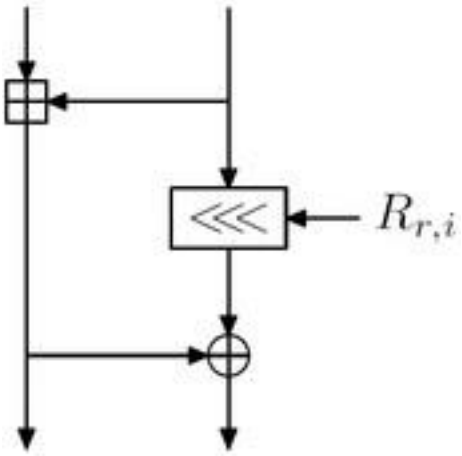
OS 模式 :	32 bit	64 bit
在 Core2 上 A16 散列速度		
A16 散列-256	12 cpb	11 cpb
A16 散列-512	13 cpb	12 cpb

eBASH 的独立性测定结果如下图一样。

Processor:	Core i5	Core 2 (45 nm)	Core 2 (65 nm)
Speed measured by eBASH (2010.05.15)			
A16 散列-256	7.51 cpb	9.18 cpb	11.34 cpb
A16 散列-512	8.63 cpb	10.02 cpb	12.05 cpb

而且为了提高 A16 散列的压缩技能，能够在 2 个 core 进行并列处理。

A17 散列
技能
支持 256,512 及 1024bit 的内部状态大小和任意输出大小。 笔者在 Intel Core 2 Duo 上对 64bit 模式的输出大小要求每 bytes6.1cycle。 Threefish 的核心是以 MIX 函数为基础。 MIX 函数再添加一次定数和 XOR 来变换两个 64bit 的词。 UBI Chening 模式是结合输入链值和任意长度输入文字列，生成固定大小输出。 Threefish 的非线性来自加法运算和排他性逻辑合的结合。 它不使用 S 箱。
该功能最适合 64 Bit 处理器,A17 散列定义为随机散列,并列解码,流线密码,个人设定及键诱导功能等选择性功能。



Threefish Mix 技能

A17 散列 散列举例
空文字列的散列值

```
A17 散列-256-256("")  c8877087da56e072870daa843f176e9453115929094c
3a40c463a196c29bf7ba A17 散列-512-256("")
39ccc4554a8b31853b9de7a1fe638a24cce6b35a55f2431009e18780335d2621
A17 散列-512-512("")
bc5b4c50925519c290cc634277ae3d6257212395cba733bbad37a4af0fa06af41fca790
3d06564fea7a2d3730dbdb80c1f85562dfcc070334ea4d1d9e72cba7a
```

虽然短信的小变化，因 Avalanche effect，几乎都会获得其它散列。例如在文章末端添加句号时

```
A17 散列 -512-256("The quick brown fox jumps over the lazy dog.")
b3250457e05d3060b1a4bbc1428bc75a3f525ca389aeab96cfa34638d96e492a
A17 散列 -512-256("The quick brown fox jumps over the lazy dog."))
41e829d7fca71c7d7154ed8fc8a069f274dd664ae0ed29d365d919f4e575eebb
A17 散列 -512-512("The quick brown fox jumps over the lazy dog."))
94c2ae036dba8783d0b3f7d6cc111ff810702f5c77707999be7e1c9486ff238a7044de7
34293147359b4ac7e1d09cd247c351d69826b78dcddd951f0ef912713
A17 散列 -512-512("The quick brown fox jumps over the lazy dog."))
658223cb3d69b5e76e3588ca63feffba0dc2ead38a95d0650564f2a39da8e83fbb42c9d
6ad9e03fbfde8a25a880357d457dbd6f74cbcb5e728979577dbce5436
```

包含 D4 散列 - Factorconfig.h 的命令内容，修行下列命令句子。

```
#if
defined(D4 散列)
                    #include "factor-config.h"
                    #endif


                    #include <stdint.h>


                    #if defined(HAVE_ENDIAN_H)
                    #include <endian.h>
                    #endif


                    uint32_t static inline ReadLE32(const unsigned char* ptr)
                    {
                    #if HAVE_DECL_LE32TOH == 1
                        return le32toh(*((uint32_t*)ptr));
                    #elif !defined(WORDS_BIGENDIAN)
                        return *((uint32_t*)ptr);
```

```
#else
    return ((uint32_t)ptr[3] << 24 | (uint32_t)ptr[2] << 16 |
(uint32_t)ptr[1] << 8 | (uint32_t)ptr[0]);
#endif
}


uint64_t static inline ReadLE64(const unsigned char* ptr)
{
#if HAVE_DECL_LE64TOH == 1
    return le64toh(*((uint64_t*)ptr));
#elif !defined(WORDS_BIGENDIAN)
    return *((uint64_t*)ptr);
#else
    return ((uint64_t)ptr[7] << 56 | (uint64_t)ptr[6] << 48 |
(uint64_t)ptr[5] << 40 | (uint64_t)ptr[4] << 32 |
            (uint64_t)ptr[3] << 24 | (uint64_t)ptr[2] << 16 |
(uint64_t)ptr[1] << 8 | (uint64_t)ptr[0]);
#endif
}


void static inline WriteLE32(unsigned char* ptr, uint32_t x)
{
#if HAVE_DECL_HTOLE32 == 1
    *((uint32_t*)ptr) = htole32(x);
#elif !defined(WORDS_BIGENDIAN)
    *((uint32_t*)ptr) = x;
#else
    ptr[3] = x >> 24;
    ptr[2] = x >> 16;
    ptr[1] = x >> 8;
    ptr[0] = x;
#endif
}


void static inline WriteLE64(unsigned char* ptr, uint64_t x)
{
#if HAVE_DECL_HTOLE64 == 1
    *((uint64_t*)ptr) = htole64(x);
#elif !defined(WORDS_BIGENDIAN)
    *((uint64_t*)ptr) = x;
#else
    ptr[7] = x >> 56;
    ptr[6] = x >> 48;
```

```
    ptr[5] = x >> 40;
    ptr[4] = x >> 32;
    ptr[3] = x >> 24;
    ptr[2] = x >> 16;
    ptr[1] = x >> 8;
    ptr[0] = x;
#endif
}


uint32_t static inline ReadBE32(const unsigned char* ptr)
{
#if HAVE_DECL_BE32TOH == 1
    return be32toh(*((uint32_t*)ptr));
#else
    return ((uint32_t)ptr[0] << 24 | (uint32_t)ptr[1] << 16 |
(uint32_t)ptr[2] << 8 | (uint32_t)ptr[3]);
#endif
}


uint64_t static inline ReadBE64(const unsigned char* ptr)
{
#if HAVE_DECL_BE64TOH == 1
    return be64toh(*((uint64_t*)ptr));
#else
    return ((uint64_t)ptr[0] << 56 | (uint64_t)ptr[1] << 48 |
(uint64_t)ptr[2] << 40 | (uint64_t)ptr[3] << 32 |
            (uint64_t)ptr[4] << 24 | (uint64_t)ptr[5] << 16 |
(uint64_t)ptr[6] << 8 | (uint64_t)ptr[7]);
#endif
}


void static inline WriteBE32(unsigned char* ptr, uint32_t x)
{
#if HAVE_DECL_HTOBE32 == 1
    *((uint32_t*)ptr) = htobe32(x);
#else
    ptr[0] = x >> 24;
    ptr[1] = x >> 16;
    ptr[2] = x >> 8;
    ptr[3] = x;
#endif
}
```

```
void static inline WriteBE64(unsigned char* ptr, uint64_t x)
{
#if HAVE_DECL_HTOBE64 == 1
    *((uint64_t*)ptr) = htobe64(x);
#else
    ptr[0] = x >> 56;
    ptr[1] = x >> 48;
    ptr[2] = x >> 40;
    ptr[3] = x >> 32;
    ptr[4] = x >> 24;
    ptr[5] = x >> 16;
    ptr[6] = x >> 8;
    ptr[7] = x;
#endif
}
```

SHA 散列
Sha-1, F10 散列
Sha-256, F11 散列
Sha-512, F12 散列
B = 2,3,5,6,7,9,(13,13-1),15,16,17

散列连接算法 B
2 散列连接算法

```
#define A2 散列


#include "C"


#define SIZE_B2 散列连接算法 224   224
#define SIZE_B2 散列连接算法 256   256
#define SIZE_B2 散列连接算法 384   384
#define SIZE_B2 散列连接算法 512   512


#endif
```

B3 散列连接算法

```
#define A3 散列


#include "C"
```

```
#define SIZE_B3 散列连接算法 224   224
#define SIZE_B3 散列连接算法 256   256
#define SIZE_B3 散列连接算法 384   384
#define SIZE_B3 散列连接算法 512   512


#endif
```

B5 散列连接算法

```
#define A5 散列


#include "C"


#define SIZE_B5 散列连接算法 224   224

#define SIZE_B5 散列连接算法 256   256
#define SIZE_B5 散列连接算法 384   384
#define SIZE_B5 散列连接算法 512   512


#endif
```

B6 散列连接算法

```
#define A6 散列


#include "C"


#define SIZE_B6 散列连接算法 224   224
#define SIZE_B6 散列连接算法 256   256
#define SIZE_B6 散列连接算法 384   384
#define SIZE_B6 散列连接算法 512   512


#endif
```

B7 散列连接算法

```
#define A7 散列


#include "C"
```

```
#define SIZE_B7 散列连接算法 224   224
#define SIZE_B7 散列连接算法 256   256
#define SIZE_B7 散列连接算法 384   384
#define SIZE_B7 散列连接算法 512   512


#endif
```

B9 散列连接算法

```
#define A9 散列


#include "C"


#define SIZE_B9 散列连接算法 224   224
#define SIZE_B9 散列连接算法 256   256
#define SIZE_B9 散列连接算法 384   384
#define SIZE_B9 散列连接算法 512   512


#endif
```

B13 散列连接算法

```
#define A13 散列


#include "C"


#define SIZE_B13 散列连接算法 224   224
#define SIZE_B13 散列连接算法 256   256
#define SIZE_B13 散列连接算法  384   384
#define SIZE_B13 散列连接算法 512   512


#endif
```

B13-1 散列连接算法

```
#define A13-1 散列


#include "C"


#define SIZE_B13-1 散列连接算法 224   224
```

```
#define SIZE_B13-1 散列连接算法 256   256
#define SIZE_B13-1 散列连接算法 384   384
#define SIZE_B13-1 散列连接算法 512   512


#endif
```

B15 散列连接算法

```
#define A15 散列


#include "C"


#define SIZE_B15 散列连接算法 224   224
#define SIZE_B15 散列连接算法 256   256
#define SIZE_B15 散列连接算法 384   384
#define SIZE_B15 散列连接算法 512   512


#endif
```

B16 散列连接算法

```
#define A16 散列


#include "C"


#define SIZE_B16 散列连接算法 224   224
#define SIZE_B16 散列连接算法 256   256
#define SIZE_B16 散列连接算法 384   384
#define SIZE_B16 散列连接算法 512   512


#endif
```

B17 散列连接算法

```
#define A17 散列


#include "C"


#define SIZE_B17 散列连接算法 224   224
#define SIZE_B17 散列连接算法 256   256
```

```
#define SIZE_B17 散列连接算法 384   384
#define SIZE_B17 散列连接算法 512   512


#endif
```

C18 散列排列链接算法（包含 A,B 算法）

```
#include
<E>


        #ifdef UINT32_MAX
        typedef uint32_t A 散列,B 散列连接算法_32;
        typedef int32_t A 散列,B 散列连接算法 _32;
        #else
        typedef uint_fast32_t A 散列,B 散列连接算法 _32;
        typedef int_fast32_t A 散列,B 散列连接算法 _32;
        #endif
        #if !_NO_64
        #ifdef UINT64_MAX
        typedef uint64_t A 散列,B 散列连接算法 _64;
        typedef int64_t A 散列,B 散列连接算法 _64;
        #else
        typedef uint_fast64_t A 散列,B 散列连接算法 _64;
        typedef int_fast64_t A 散列,B 散列连接算法 _64;
        #endif
        #endif


        #define C32(x)     ((A 散列,B 散列连接算法 _32)(x))
        #if !NO_64
        #define C64(x)     ((A 散列,B 散列连接算法 _64)(x))
        #define 64  1
        #endif
```

2. Secp256r1

ECkey 形成公开键和密键两个键对。 有将该密码以 RSA 方式或 DSA 方式进行密码化的方法。 在这里 Factor 的 secp256r1 的键对形成方式是(A,B)=DSA + RSA(C,D)方式结合的组合形成方式。 这是 E 回避 Secp256r1 的重复形成方式的设计算法,由 H,F 在 DSA,RSA 双重密码中回避重复的加密算法组合组成。 E,G,I 在 1.MX Blockchain Hash 算法添加键对后，添加 J 的技能来形成键对。

因此,与使用现有 secp256k1 的拐点 3 的方式相比,如下表所示,secp256r1 具有更优秀的安全性。

-

- 特征

Parameters	Strength	Size	RSA/DSA	Koblitz or random
secp256r1	128	256	3072	r

Parameters	n ANSI X9.62	ANSI X9.63	echeck	IEEE P1363	IPSec	NIST	WAP
secp256r1	r	r	c	c	c	r	c

-启动原理

可验证的任意椭圆曲线域名参量 secp256r1 是定义为与下列变数。

T = (p,a,b,G,n,h)

p = FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF

$= 2^{224}(2^{32}-1)+2^{192}+2^{96}-1$

The curve E: $y^2 = x^3 + ax + b$ over $\mathbb{F}_p$ is defined by:

a = FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFC

b = 5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E 27D2604B

E was chosen verifiably at random as specified in ANSI X9.62 [1] from the seed:

S = C49D3608 86E70493 6A6678E1 139D26B7 819F7E90

The base point G in compressed form is:

G = 03 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296

and in uncompressed form is:

G = 04 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296 4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357 6B315ECE CBB64068 37BF51F5

Finally the order n of G and the cofactor are:

n = FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2 FC632551

h = 01 10

- 算法

形成 A - ECDSA parse,serialize,verify,sign,recover Code

```
#ifndef
_SECP256r1_ECDSA_
                    #define _SECP256r1_ECDSA_


                    #include "I"
                    #include "H"
```

10 图片 [来源: https://www.secg.org/SEC2-Ver-1.0.pdf]

```
static void secp256r1_ecsda_start(void);
static void secp256r1_ecdsa_stop(void);


typedef struct {
    secp256r1_scalar_t r, s;
} secp256r1_ecdsa_sig_t;


static int secp256r1_ecdsa_sig_parse(secp256r1_ecdsa_sig_t *r,
const unsigned char *sig, int size);
static int secp256r1_ecdsa_sig_serialize(unsigned char *sig, int *size,
const secp256r1_ecdsa_sig_t *a);
static int secp256r1_ecdsa_sig_verify(const secp256r1_ecdsa_sig_t *sig,
const secp256r1_ge_t *pubkey, const secp256r1_scalar_t *message);
static int secp256r1_ecdsa_sig_sign(secp256r1_ecdsa_sig_t *sig, const
secp256r1_scalar_t *seckey, const secp256r1_scalar_t *message,
const secp256r1_scalar_t *nonce, int *recid);
static int secp256r1_ecdsa_sig_recover(const secp256r1_ecdsa_sig_t
*sig, secp256r1_ge_t *pubkey, const secp256r1_scalar_t *message,
int recid);
static void secp256r1_ecdsa_sig_set_rs(secp256r1_ecdsa_sig_t *sig,
const secp256r1_scalar_t *r, const secp256r1_scalar_t *s);


#endif
```

B - ECDSA 修行密码化功能

```
#ifndef
_SECP256r1_ECDSA_IMPL_H
_
      #define _SECP256r1_ECDSA_IMPL_H_


      #include "I"
      #include "G"
      #include "H"
      #include "E"
      #include "ecmult_gen.h"
      #include "ecdsa.h"
```

```
typedef struct {
    secp256r1_fe_t order_as_fe;
    secp256r1_fe_t p_minus_order;
} secp256r1_ecdsa_consts_t;


static const secp256r1_ecdsa_consts_t *secp256r1_ecdsa_consts
= NULL;


static void secp256r1_ecdsa_start(void) {
    if (secp256r1_ecdsa_consts != NULL)
        return;


    /* Allocate. */
    secp256r1_ecdsa_consts_t *ret =
(secp256r1_ecdsa_consts_t*)malloc(sizeof(secp256r1_ecdsa_consts_t
));


    static const unsigned char order[] = {
        0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
        0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFE,
        0xBA,0xAE,0xDC,0xE6,0xAF,0x48,0xA0,0x3B,
        0xBF,0xD2,0x5E,0x8C,0xD0,0x36,0x41,0x41
    };


    secp256r1_fe_set_b32(&ret->order_as_fe, order);
    secp256r1_fe_negate(&ret->p_minus_order, &ret->order_as_fe,
1);
    secp256r1_fe_normalize(&ret->p_minus_order);


    /* Set the global pointer. */
    secp256r1_ecdsa_consts = ret;
}


static void secp256r1_ecdsa_stop(void) {
    if (secp256r1_ecdsa_consts == NULL)
        return;


    secp256r1_ecdsa_consts_t *c =
```

```
(secp256r1_ecdsa_consts_t*)secp256r1_ecdsa_consts;
    secp256r1_ecdsa_consts = NULL;
    free(c);
}


static int secp256r1_ecdsa_sig_parse(secp256r1_ecdsa_sig_t *r,
const unsigned char *sig, int size) {
    if (sig[0] != 0x30) return 0;
    int lenr = sig[3];
    if (5+lenr >= size) return 0;
    int lens = sig[lenr+5];
    if (sig[1] != lenr+lens+4) return 0;
    if (lenr+lens+6 > size) return 0;
    if (sig[2] != 0x02) return 0;
    if (lenr == 0) return 0;
    if (sig[lenr+4] != 0x02) return 0;
    if (lens == 0) return 0;
    const unsigned char *sp = sig + 6 + lenr;
    while (lens > 0 && sp[0] == 0) {
        lens--;
        sp++;
    }
    if (lens > 32) return 0;
    const unsigned char *rp = sig + 4;
    while (lenr > 0 && rp[0] == 0) {
        lenr--;
        rp++;
    }
    if (lenr > 32) return 0;
    unsigned char ra[32] = {0}, sa[32] = {0};
    memcpy(ra + 32 - lenr, rp, lenr);
    memcpy(sa + 32 - lens, sp, lens);
    int overflow = 0;
    secp256r1_scalar_set_b32(&r->r, ra, &overflow);
    if (overflow) return 0;
    secp256r1_scalar_set_b32(&r->s, sa, &overflow);
    if (overflow) return 0;
    return 1;
}


static int secp256r1_ecdsa_sig_serialize(unsigned char *sig, int
*size, const secp256r1_ecdsa_sig_t *a) {
    unsigned char r[33] = {0}, s[33] = {0};
    secp256r1_scalar_get_b32(&r[1], &a->r);
```

```
    secp256r1_scalar_get_b32(&s[1], &a->s);
    unsigned char *rp = r, *sp = s;
    int lenR = 33, lenS = 33;
    while (lenR > 1 && rp[0] == 0 && rp[1] < 0x80) { lenR--;
rp++; }
    while (lenS > 1 && sp[0] == 0 && sp[1] < 0x80) { lenS--;
sp++; }
    if (*size < 6+lenS+lenR)
        return 0;
    *size = 6 + lenS + lenR;
    sig[0] = 0x30;
    sig[1] = 4 + lenS + lenR;
    sig[2] = 0x02;
    sig[3] = lenR;
    memcpy(sig+4, rp, lenR);
    sig[4+lenR] = 0x02;
    sig[5+lenR] = lenS;
    memcpy(sig+lenR+6, sp, lenS);
    return 1;
}


static int secp256r1_ecdsa_sig_recompute(secp256r1_scalar_t
*r2, const secp256r1_ecdsa_sig_t *sig, const secp256r1_ge_t
*pubkey,  const secp256r1_scalar_t *message) {
    if (secp256r1_scalar_is_zero(&sig->r) ||
secp256r1_scalar_is_zero(&sig->s))
        return 0;


    int ret = 0;
    secp256r1_scalar_t sn, u1, u2;
    secp256r1_scalar_inverse_var(&sn, &sig->s);
    secp256r1_scalar_mul(&u1, &sn, message);
    secp256r1_scalar_mul(&u2, &sn, &sig->r);
    secp256r1_gej_t pubkeyj; secp256r1_gej_set_ge(&pubkeyj,
pubkey);
    secp256r1_gej_t pr; secp256r1_ecmult(&pr, &pubkeyj, &u2,
&u1);
    if (!secp256r1_gej_is_infinity(&pr)) {
        secp256r1_fe_t xr; secp256r1_gej_get_x_var(&xr, &pr);
        secp256r1_fe_normalize(&xr);
        unsigned char xrb[32]; secp256r1_fe_get_b32(xrb, &xr);
        secp256r1_scalar_set_b32(r2, xrb, NULL);
        ret = 1;
    }
```

```
    return ret;
}


static int secp256r1_ecdsa_sig_recover(const
secp256r1_ecdsa_sig_t *sig, secp256r1_ge_t *pubkey, const
secp256r1_scalar_t *message, int recid) {
    if (secp256r1_scalar_is_zero(&sig->r) ||
secp256r1_scalar_is_zero(&sig->s))
        return 0;


    unsigned char brx[32];
    secp256r1_scalar_get_b32(brx, &sig->r);
    secp256r1_fe_t fx;
    VERIFY_CHECK(secp256r1_fe_set_b32(&fx, brx)); /* brx comes
from a scalar, so is less than the order; certainly less than p
*/
    if (recid & 2) {
        if (secp256r1_fe_cmp_var(&fx, &secp256r1_ecdsa_consts-
>p_minus_order) >= 0)
            return 0;
        secp256r1_fe_add(&fx, &secp256r1_ecdsa_consts-
>order_as_fe);
    }
    secp256r1_ge_t x;
    if (!secp256r1_ge_set_xo(&x, &fx, recid & 1))
        return 0;
    secp256r1_gej_t xj;
    secp256r1_gej_set_ge(&xj, &x);
    secp256r1_scalar_t rn, u1, u2;
    secp256r1_scalar_inverse_var(&rn, &sig->r);
    secp256r1_scalar_mul(&u1, &rn, message);
    secp256r1_scalar_negate(&u1, &u1);
    secp256r1_scalar_mul(&u2, &rn, &sig->s);
    secp256r1_gej_t qj;
    secp256r1_ecmult(&qj, &xj, &u2, &u1);
    secp256r1_ge_set_gej_var(pubkey, &qj);
    return !secp256r1_gej_is_infinity(&qj);
}


static int secp256r1_ecdsa_sig_verify(const secp256r1_ecdsa_sig_t
*sig, const secp256r1_ge_t *pubkey, const secp256r1_scalar_t
*message) {
    secp256r1_scalar_t r2;
```

```
    int ret = 0;
    ret = secp256r1_ecdsa_sig_recompute(&r2, sig, pubkey,
message) && secp256r1_scalar_eq(&sig->r, &r2);
    return ret;
}


static int secp256r1_ecdsa_sig_sign(secp256r1_ecdsa_sig_t
*sig, const secp256r1_scalar_t *seckey, const
secp256r1_scalar_t
*message, const secp256r1_scalar_t *nonce, int *recid) {
    secp256r1_gej_t rp;
    secp256r1_ecmult_gen(&rp, nonce);
    secp256r1_ge_t r;
    secp256r1_ge_set_gej(&r, &rp);
    unsigned char b[32];
    secp256r1_fe_normalize(&r.x);
    secp256r1_fe_normalize(&r.y);
    secp256r1_fe_get_b32(b, &r.x);
    int overflow = 0;
    secp256r1_scalar_set_b32(&sig->r, b, &overflow);
    if (recid)
        *recid = (overflow ? 2 : 0) |
(secp256r1_fe_is_odd(&r.y) ? 1 : 0);
    secp256r1_scalar_t n;
    secp256r1_scalar_mul(&n, &sig->r, seckey);
    secp256r1_scalar_add(&n, &n, message);
    secp256r1_scalar_inverse(&sig->s, nonce);
    secp256r1_scalar_mul(&sig->s, &sig->s, &n);
    secp256r1_scalar_clear(&n);
    secp256r1_gej_clear(&rp);
    secp256r1_ge_clear(&r);
    if (secp256r1_scalar_is_zero(&sig->s))
        return 0;
    if (secp256r1_scalar_is_high(&sig->s)) {
        secp256r1_scalar_negate(&sig->s, &sig->s);
        if (recid)
            *recid ^= 1;
    }
    return 1;
}


static void secp256r1_ecdsa_sig_set_rs(secp256r1_ecdsa_sig_t
*sig, const secp256r1_scalar_t *r, const secp256r1_scalar_t *s) {
    sig->r = *r;
    sig->s = *s;
```

```
                        }


                        #endif
```

形成 C - ECKEY parse,serialize,parse,serialize Code

```
#ifndef
_SECP256r1_ECKEY_
                    #define _SECP256r1_ECKEY_


                    #include "H"
                    #include "I"


                    static int secp256r1_eckey_pubkey_parse(secp256r1_ge_t *elem,
                    const unsigned char *pub, int size);
                    static int secp256r1_eckey_pubkey_serialize(secp256r1_ge_t *elem,
                    unsigned char *pub, int *size, int compressed);


                    static int secp256r1_eckey_privkey_parse(secp256r1_scalar_t *key,
                    const unsigned char *privkey, int privkeylen);
                    static int secp256r1_eckey_privkey_serialize(unsigned char *privkey,
                    int *privkeylen, const secp256r1_scalar_t *key, int compressed);


                    static int secp256r1_eckey_privkey_tweak_add(secp256r1_scalar_t
                    *key, const secp256r1_scalar_t *tweak);
                    static int secp256r1_eckey_pubkey_tweak_add(secp256r1_ge_t *key, const
                    secp256r1_scalar_t *tweak);
                    static int secp256r1_eckey_privkey_tweak_mul(secp256r1_scalar_t *key,
                    const secp256r1_scalar_t *tweak);
                    static int secp256r1_eckey_pubkey_tweak_mul(secp256r1_ge_t *key, const
                    secp256r1_scalar_t *tweak);


                    #endif
```

修行 D - ECKEY 功能

```
#ifndef
_SECP256r1_ECKEY_IMP
L_H_
                        #define _SECP256r1_ECKEY_IMPL_H_
```

```
#include "C"


#include "I"
#include "G"
#include "H"
#include "ecmult_gen.h"


static int secp256r1_eckey_pubkey_parse(secp256r1_ge_t *elem,
const unsigned char *pub, int size) {
    if (size == 33 && (pub[0] == 0x02 || pub[0] == 0x03)) {
        secp256r1_fe_t x;
        return secp256r1_fe_set_b32(&x, pub+1) &&
secp256r1_ge_set_xo(elem, &x, pub[0] == 0x03);
    } else if (size == 65 && (pub[0] == 0x04 || pub[0] == 0x06 ||
pub[0] == 0x07)) {
        secp256r1_fe_t x, y;
        if (!secp256r1_fe_set_b32(&x, pub+1)
|| !secp256r1_fe_set_b32(&y, pub+33)) {
            return 0;
        }
        secp256r1_ge_set_xy(elem, &x, &y);
        if ((pub[0] == 0x06 || pub[0] == 0x07) &&
secp256r1_fe_is_odd(&y) != (pub[0] == 0x07))
            return 0;
        return secp256r1_ge_is_valid(elem);
    } else {
        return 0;
    }
}


static int secp256r1_eckey_pubkey_serialize(secp256r1_ge_t
*elem, unsigned char *pub, int *size, int compressed) {
    if (secp256r1_ge_is_infinity(elem)) {
        return 0;
    }
    secp256r1_fe_normalize(&elem->x);
    secp256r1_fe_normalize(&elem->y);
    secp256r1_fe_get_b32(&pub[1], &elem->x);
    if (compressed) {
        *size = 33;
        pub[0] = 0x02 | (secp256r1_fe_is_odd(&elem->y) ? 0x01 :
0x00);
    } else {
```

```
        *size = 65;
        pub[0] = 0x04;
        secp256r1_fe_get_b32(&pub[33], &elem->y);
    }
    return 1;
}


static int secp256r1_eckey_privkey_parse(secp256r1_scalar_t
*key, const unsigned char *privkey, int privkeylen) {
    const unsigned char *end = privkey + privkeylen;
    /* sequence header */
    if (end < privkey+1 || *privkey != 0x30)
        return 0;
    privkey++;
    /* sequence length constructor */
    int lenb = 0;
    if (end < privkey+1 || !(*privkey & 0x80))
        return 0;
    lenb = *privkey & ~0x80; privkey++;
    if (lenb < 1 || lenb > 2)
        return 0;
    if (end < privkey+lenb)
        return 0;
    /* sequence length */
    int len = 0;
    len = privkey[lenb-1] | (lenb > 1 ? privkey[lenb-2] << 8 : 0);
    privkey += lenb;
    if (end < privkey+len)
        return 0;
    /* sequence element 0: version number (=1) */
    if (end < privkey+3 || privkey[0] != 0x02 || privkey[1] != 0x01
|| privkey[2] != 0x01)
        return 0;
    privkey += 3;
    /* sequence element 1: octet string, up to 32 bytes */
    if (end < privkey+2 || privkey[0] != 0x04 || privkey[1] > 0x20
|| end < privkey+2+privkey[1])
        return 0;
    int overflow = 0;
    unsigned char c[32] = {0};
    memcpy(c + 32 - privkey[1], privkey + 2, privkey[1]);
    secp256r1_scalar_set_b32(key, c, &overflow);
    memset(c, 0, 32);
    return !overflow;
}
```

```
static int secp256r1_eckey_privkey_serialize(unsigned char
*privkey, int *privkeylen, const secp256r1_scalar_t *key,
int compressed) {
    secp256r1_gej_t rp;
    secp256r1_ecmult_gen(&rp, key);
    secp256r1_ge_t r;
    secp256r1_ge_set_gej(&r, &rp);
    if (compressed) {
        static const unsigned char begin[] = {
            0x30,0x81,0xD3,0x02,0x01,0x01,0x04,0x20
        };
        static const unsigned char middle[] = {

0xA0,0x81,0x85,0x30,0x81,0x82,0x02,0x01,0x01,0x30,0x2C,0x06,0x07,0x
2A,0x86,0x48,

0xCE,0x3D,0x01,0x01,0x02,0x21,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x
FF,0xFF,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x
FF,0xFF,0xFF,

0xFF,0xFF,0xFE,0xFF,0xFF,0xFC,0x2F,0x30,0x06,0x04,0x01,0x00,0x04,0x
01,0x07,0x04,

0x21,0x02,0x79,0xBE,0x66,0x7E,0xF9,0xDC,0xBB,0xAC,0x55,0xA0,0x62,0x
95,0xCE,0x87,

0x0B,0x07,0x02,0x9B,0xFC,0xDB,0x2D,0xCE,0x28,0xD9,0x59,0xF2,0x81,0x
5B,0x16,0xF8,

0x17,0x98,0x02,0x21,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x
FF,0xFF,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFE,0xBA,0xAE,0xDC,0xE6,0xAF,0x48,0xA0,0x3B,0x
BF,0xD2,0x5E,

0x8C,0xD0,0x36,0x41,0x41,0x02,0x01,0x01,0xA1,0x24,0x03,0x22,0x00
        };
        unsigned char *ptr = privkey;
        memcpy(ptr, begin, sizeof(begin)); ptr += sizeof(begin);
        secp256r1_scalar_get_b32(ptr, key); ptr += 32;
        memcpy(ptr, middle, sizeof(middle)); ptr += sizeof(middle);
        int pubkeylen = 0;
```

```
        if (!secp256r1_eckey_pubkey_serialize(&r, ptr, &pubkeylen,
1)) {
            return 0;
        }
        ptr += pubkeylen;
        *privkeylen = ptr - privkey;
    } else {
        static const unsigned char begin[] = {
            0x30,0x82,0x01,0x13,0x02,0x01,0x01,0x04,0x20
        };
        static const unsigned char middle[] = {

0xA0,0x81,0xA5,0x30,0x81,0xA2,0x02,0x01,0x01,0x30,0x2C,0x06,0x07,0x
2A,0x86,0x48,

0xCE,0x3D,0x01,0x01,0x02,0x21,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x
FF,0xFF,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x
FF,0xFF,0xFF,

0xFF,0xFF,0xFE,0xFF,0xFF,0xFC,0x2F,0x30,0x06,0x04,0x01,0x00,0x04,0x
01,0x07,0x04,

0x41,0x04,0x79,0xBE,0x66,0x7E,0xF9,0xDC,0xBB,0xAC,0x55,0xA0,0x62,0x
95,0xCE,0x87,

0x0B,0x07,0x02,0x9B,0xFC,0xDB,0x2D,0xCE,0x28,0xD9,0x59,0xF2,0x81,0x
5B,0x16,0xF8,

0x17,0x98,0x48,0x3A,0xDA,0x77,0x26,0xA3,0xC4,0x65,0x5D,0xA4,0xFB,0x
FC,0x0E,0x11,

0x08,0xA8,0xFD,0x17,0xB4,0x48,0xA6,0x85,0x54,0x19,0x9C,0x47,0xD0,0x
8F,0xFB,0x10,

0xD4,0xB8,0x02,0x21,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x
FF,0xFF,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFE,0xBA,0xAE,0xDC,0xE6,0xAF,0x48,0xA0,0x3B,0x
BF,0xD2,0x5E,

0x8C,0xD0,0x36,0x41,0x41,0x02,0x01,0x01,0xA1,0x44,0x03,0x42,0x00
        };
        unsigned char *ptr = privkey;
        memcpy(ptr, begin, sizeof(begin)); ptr += sizeof(begin);
```

```
        secp256r1_scalar_get_b32(ptr, key); ptr += 32;
        memcpy(ptr, middle, sizeof(middle)); ptr += sizeof(middle);
        int pubkeylen = 0;
        if (!secp256r1_eckey_pubkey_serialize(&r, ptr, &pubkeylen,
0)) {
            return 0;
        }
        ptr += pubkeylen;
        *privkeylen = ptr - privkey;
    }
    return 1;
}


static int secp256r1_eckey_privkey_tweak_add(secp256r1_scalar_t
*key, const secp256r1_scalar_t *tweak) {
    secp256r1_scalar_add(key, key, tweak);
    if (secp256r1_scalar_is_zero(key))
        return 0;
    return 1;
}


static int secp256r1_eckey_pubkey_tweak_add(secp256r1_ge_t
*key, const secp256r1_scalar_t *tweak) {
    secp256r1_gej_t pt;
    secp256r1_gej_set_ge(&pt, key);
    secp256r1_scalar_t one;
    secp256r1_scalar_set_int(&one, 1);
    secp256r1_ecmult(&pt, &pt, &one, tweak);


    if (secp256r1_gej_is_infinity(&pt))
        return 0;
    secp256r1_ge_set_gej(key, &pt);
    return 1;
}


static int secp256r1_eckey_privkey_tweak_mul(secp256r1_scalar_t
*key, const secp256r1_scalar_t *tweak) {
    if (secp256r1_scalar_is_zero(tweak))
        return 0;


    secp256r1_scalar_mul(key, key, tweak);
```

```
    return 1;
}

static int secp256r1_eckey_pubkey_tweak_mul(secp256r1_ge_t
*key, const secp256r1_scalar_t *tweak) {
    if (secp256r1_scalar_is_zero(tweak))
        return 0;

    secp256r1_scalar_t zero;
    secp256r1_scalar_set_int(&zero, 0);
    secp256r1_gej_t pt;
    secp256r1_gej_set_ge(&pt, key);
    secp256r1_ecmult(&pt, &pt, tweak, &zero);
    secp256r1_ge_set_gej(key, &pt);
    return 1;
}

#endif
```

E - 当实行 J,H 的算法时，E 的代入值可在 H 进行修正。

```
#ifndef
_SECP256r1_ECMULT_
                    #define _SECP256r1_ECMULT_

                    #include "J"
                    #include "H"

                    static void secp256r1_ecmult_start(void);
                    static void secp256r1_ecmult_stop(void);

                    /** Double multiply: R = na*A + ng*G */
                    static void secp256r1_ecmult(secp256r1_gej_t *r, const secp256r1_gej_t
                    *a, const secp256r1_scalar_t *na, const secp256r1_scalar_t *ng);

                    #endif
```

H - 按集团别再次形成算法

```
#ifndef
_SECP256r1_GROUP_
                    #define _SECP256r1_GROUP_


                    #include "J"
                    #include "G"
                    #include "E"


                    /** A group element of the secp256r1 curve, in affine coordinates. */
                    typedef struct {
                        secp256r1_fe_t x;
                        secp256r1_fe_t y;
                        int infinity; /* whether this represents the point at infinity */
                    } secp256r1_ge_t;


                    /** A group element of the secp256r1 curve, in jacobian coordinates. */
                    typedef struct {
                        secp256r1_fe_t x; /* actual X: x/z^2 */
                        secp256r1_fe_t y; /* actual Y: y/z^3 */
                        secp256r1_fe_t z;
                        int infinity; /* whether this represents the point at infinity */
                    } secp256r1_gej_t;


                    /** Global constants related to the group */
                    typedef struct {
                        secp256r1_ge_t g; /* the generator point */


                    #ifdef USE_ENDOMORPHISM
                        /* constants related to secp256r1's efficiently computable
                    endomorphism */
                        secp256r1_fe_t beta;
                    #endif
                    } secp256r1_ge_consts_t;


                    static const secp256r1_ge_consts_t *secp256r1_ge_consts = NULL;


                    /** Initialize the group module. */
                    static void secp256r1_ge_start(void);
```

```
/** De-initialize the group module. */
static void secp256r1_ge_stop(void);

/** Set a group element equal to the point at infinity */
static void secp256r1_ge_set_infinity(secp256r1_ge_t *r);

/** Set a group element equal to the point with given X and
Y coordinates */
static void secp256r1_ge_set_xy(secp256r1_ge_t *r, const secp256r1_fe_t
*x, const secp256r1_fe_t *y);

/** Set a group element (affine) equal to the point with the given
X coordinate, and given oddness
 *  for Y. Return value indicates whether the result is valid. */
static int secp256r1_ge_set_xo(secp256r1_ge_t *r, const secp256r1_fe_t
*x, int odd);

/** Check whether a group element is the point at infinity. */
static int secp256r1_ge_is_infinity(const secp256r1_ge_t *a);

/** Check whether a group element is valid (i.e., on the curve). */
static int secp256r1_ge_is_valid(const secp256r1_ge_t *a);

static void secp256r1_ge_neg(secp256r1_ge_t *r, const secp256r1_ge_t
*a);

/** Get a hex representation of a point. *rlen will be overwritten
with the real length. */
static void secp256r1_ge_get_hex(char *r, int *rlen, const
secp256r1_ge_t *a);

/** Set a group element equal to another which is given in
jacobian coordinates */
static void secp256r1_ge_set_gej(secp256r1_ge_t *r, secp256r1_gej_t
*a);
```

```
/** Set a batch of group elements equal to the inputs given in jacobian
coordinates */
static void secp256r1_ge_set_all_gej_var(size_t len, secp256r1_ge_t
r[len], const secp256r1_gej_t a[len]);



/** Set a group element (jacobian) equal to the point at infinity. */
static void secp256r1_gej_set_infinity(secp256r1_gej_t *r);


/** Set a group element (jacobian) equal to the point with given X
and Y coordinates. */
static void secp256r1_gej_set_xy(secp256r1_gej_t *r, const
secp256r1_fe_t *x, const secp256r1_fe_t *y);


/** Set a group element (jacobian) equal to another which is given
in affine coordinates. */
static void secp256r1_gej_set_ge(secp256r1_gej_t *r, const
secp256r1_ge_t *a);


/** Get the X coordinate of a group element (jacobian). */
static void secp256r1_gej_get_x_var(secp256r1_fe_t *r, const
secp256r1_gej_t *a);


/** Set r equal to the inverse of a (i.e., mirrored around the X axis)
*/
static void secp256r1_gej_neg(secp256r1_gej_t *r, const secp256r1_gej_t
*a);


/** Check whether a group element is the point at infinity. */
static int secp256r1_gej_is_infinity(const secp256r1_gej_t *a);


/** Set r equal to the double of a. */
static void secp256r1_gej_double_var(secp256r1_gej_t *r, const
secp256r1_gej_t *a);


/** Set r equal to the sum of a and b. */
static void secp256r1_gej_add_var(secp256r1_gej_t *r, const
```

```
secp256r1_gej_t *a, const secp256r1_gej_t *b);

/** Set r equal to the sum of a and b (with b given in
affine coordinates, and not infinity). */
static void secp256r1_gej_add_ge(secp256r1_gej_t *r, const
secp256r1_gej_t *a, const secp256r1_ge_t *b);

/** Set r equal to the sum of a and b (with b given in
affine coordinates). This is more efficient
    than secp256r1_gej_add_var. It is identical to secp256r1_gej_add_ge
but without constant-time
    guarantee, and b is allowed to be infinity. */
static void secp256r1_gej_add_ge_var(secp256r1_gej_t *r, const
secp256r1_gej_t *a, const secp256r1_ge_t *b);

/** Get a hex representation of a point. *rlen will be overwritten
with the real length. */
static void secp256r1_gej_get_hex(char *r, int *rlen, const
secp256r1_gej_t *a);

#ifdef USE_ENDOMORPHISM
/** Set r to be equal to lambda times a, where lambda is chosen in a
way such that this is very fast. */
static void secp256r1_gej_mul_lambda(secp256r1_gej_t *r, const
secp256r1_gej_t *a);
#endif

/** Clear a secp256r1_gej_t to prevent leaking sensitive information.
*/
static void secp256r1_gej_clear(secp256r1_gej_t *r);

/** Clear a secp256r1_ge_t to prevent leaking sensitive information. */
static void secp256r1_ge_clear(secp256r1_ge_t *r);
```

F – 检证查看功能

```
#ifndef
_SECP256r1_GROUP_IMPL_H_
```

```
#define _SECP256r1_GROUP_IMPL_H_


#include <string.h>


#include "J"
#include "G"
#include "H"


static void secp256r1_ge_set_infinity(secp256r1_ge_t *r) {
    r->infinity = 1;
}


static void secp256r1_ge_set_xy(secp256r1_ge_t *r, const
secp256r1_fe_t *x, const secp256r1_fe_t *y) {
    r->infinity = 0;
    r->x = *x;
    r->y = *y;
}


static int secp256r1_ge_is_infinity(const secp256r1_ge_t *a) {
    return a->infinity;
}


static void secp256r1_ge_neg(secp256r1_ge_t *r, const
secp256r1_ge_t *a) {
    r->infinity = a->infinity;
    r->x = a->x;
    r->y = a->y;
    secp256r1_fe_normalize(&r->y);
    secp256r1_fe_negate(&r->y, &r->y, 1);
}


static void secp256r1_ge_get_hex(char *r, int *rlen,
const secp256r1_ge_t *a) {
    char cx[65]; int lx=65;
    char cy[65]; int ly=65;
    secp256r1_fe_get_hex(cx, &lx, &a->x);
    secp256r1_fe_get_hex(cy, &ly, &a->y);
    lx = strlen(cx);
```

```
    ly = strlen(cy);
    int len = lx + ly + 3 + 1;
    if (*rlen < len) {
        *rlen = len;
        return;
    }
    *rlen = len;
    r[0] = '(';
    memcpy(r+1, cx, lx);
    r[1+lx] = ',';
    memcpy(r+2+lx, cy, ly);
    r[2+lx+ly] = ')';
    r[3+lx+ly] = 0;
}


static void secp256r1_ge_set_gej(secp256r1_ge_t *r,
secp256r1_gej_t *a) {
    r->infinity = a->infinity;
    secp256r1_fe_inv(&a->z, &a->z);
    secp256r1_fe_t z2; secp256r1_fe_sqr(&z2, &a->z);
    secp256r1_fe_t z3; secp256r1_fe_mul(&z3, &a->z, &z2);
    secp256r1_fe_mul(&a->x, &a->x, &z2);
    secp256r1_fe_mul(&a->y, &a->y, &z3);
    secp256r1_fe_set_int(&a->z, 1);
    r->x = a->x;
    r->y = a->y;
}


static void secp256r1_ge_set_gej_var(secp256r1_ge_t *r,
secp256r1_gej_t *a) {
    r->infinity = a->infinity;
    if (a->infinity) {
        return;
    }
    secp256r1_fe_inv_var(&a->z, &a->z);
    secp256r1_fe_t z2; secp256r1_fe_sqr(&z2, &a->z);
    secp256r1_fe_t z3; secp256r1_fe_mul(&z3, &a->z, &z2);
    secp256r1_fe_mul(&a->x, &a->x, &z2);
    secp256r1_fe_mul(&a->y, &a->y, &z3);
    secp256r1_fe_set_int(&a->z, 1);
    r->x = a->x;
    r->y = a->y;
}
```

```
static void secp256r1_ge_set_all_gej_var(size_t len,
secp256r1_ge_t r[len], const secp256r1_gej_t a[len]) {
    size_t count = 0;
    secp256r1_fe_t az[len];
    for (size_t i=0; i<len; i++) {
        if (!a[i].infinity) {
            az[count++] = a[i].z;
        }
    }

    secp256r1_fe_t azi[count];
    secp256r1_fe_inv_all_var(count, azi, az);

    count = 0;
    for (size_t i=0; i<len; i++) {
        r[i].infinity = a[i].infinity;
        if (!a[i].infinity) {
            secp256r1_fe_t *zi = &azi[count++];
            secp256r1_fe_t zi2; secp256r1_fe_sqr(&zi2, zi);
            secp256r1_fe_t zi3; secp256r1_fe_mul(&zi3, &zi2,
zi);
            secp256r1_fe_mul(&r[i].x, &a[i].x, &zi2);
            secp256r1_fe_mul(&r[i].y, &a[i].y, &zi3);
        }
    }
}

static void secp256r1_gej_set_infinity(secp256r1_gej_t *r) {
    r->infinity = 1;
    secp256r1_fe_set_int(&r->x, 0);
    secp256r1_fe_set_int(&r->y, 0);
    secp256r1_fe_set_int(&r->z, 0);
}

static void secp256r1_gej_set_xy(secp256r1_gej_t *r, const
secp256r1_fe_t *x, const secp256r1_fe_t *y) {
    r->infinity = 0;
    r->x = *x;
    r->y = *y;
    secp256r1_fe_set_int(&r->z, 1);
}
```

```
static void secp256r1_gej_clear(secp256r1_gej_t *r) {
    r->infinity = 0;
    secp256r1_fe_clear(&r->x);
    secp256r1_fe_clear(&r->y);
    secp256r1_fe_clear(&r->z);
}


static void secp256r1_ge_clear(secp256r1_ge_t *r) {
    r->infinity = 0;
    secp256r1_fe_clear(&r->x);
    secp256r1_fe_clear(&r->y);
}


static int secp256r1_ge_set_xo(secp256r1_ge_t *r, const
secp256r1_fe_t *x, int odd) {
    r->x = *x;
    secp256r1_fe_t x2; secp256r1_fe_sqr(&x2, x);
    secp256r1_fe_t x3; secp256r1_fe_mul(&x3, x, &x2);
    r->infinity = 0;
    secp256r1_fe_t c; secp256r1_fe_set_int(&c, 7);
    secp256r1_fe_add(&c, &x3);
    if (!secp256r1_fe_sqrt(&r->y, &c))
        return 0;
    secp256r1_fe_normalize(&r->y);
    if (secp256r1_fe_is_odd(&r->y) != odd)
        secp256r1_fe_negate(&r->y, &r->y, 1);
    return 1;
}


static void secp256r1_gej_set_ge(secp256r1_gej_t *r, const
secp256r1_ge_t *a) {
   r->infinity = a->infinity;
   r->x = a->x;
   r->y = a->y;
   secp256r1_fe_set_int(&r->z, 1);
}


static void secp256r1_gej_get_x_var(secp256r1_fe_t *r,
const secp256r1_gej_t *a) {
    secp256r1_fe_t zi2; secp256r1_fe_inv_var(&zi2, &a->z);
```

```
secp256r1_fe_sqr(&zi2, &zi2);
    secp256r1_fe_mul(r, &a->x, &zi2);
}


static void secp256r1_gej_neg(secp256r1_gej_t *r, const
secp256r1_gej_t *a) {
    r->infinity = a->infinity;
    r->x = a->x;
    r->y = a->y;
    r->z = a->z;
    secp256r1_fe_normalize(&r->y);
    secp256r1_fe_negate(&r->y, &r->y, 1);
}


static int secp256r1_gej_is_infinity(const secp256r1_gej_t *a) {
    return a->infinity;
}


static int secp256r1_gej_is_valid(const secp256r1_gej_t *a) {
    if (a->infinity)
        return 0;
    /** y^2 = x^3 + 7
     *  (Y/Z^3)^2 = (X/Z^2)^3 + 7
     *  Y^2 / Z^6 = X^3 / Z^6 + 7
     *  Y^2 = X^3 + 7*Z^6
     */
    secp256r1_fe_t y2; secp256r1_fe_sqr(&y2, &a->y);
    secp256r1_fe_t x3; secp256r1_fe_sqr(&x3, &a->x);
secp256r1_fe_mul(&x3, &x3, &a->x);
    secp256r1_fe_t z2; secp256r1_fe_sqr(&z2, &a->z);
    secp256r1_fe_t z6; secp256r1_fe_sqr(&z6, &z2);
secp256r1_fe_mul(&z6, &z6, &z2);
    secp256r1_fe_mul_int(&z6, 7);
    secp256r1_fe_add(&x3, &z6);
    secp256r1_fe_normalize(&y2);
    secp256r1_fe_normalize(&x3);
    return secp256r1_fe_equal(&y2, &x3);
}


static int secp256r1_ge_is_valid(const secp256r1_ge_t *a) {
    if (a->infinity)
        return 0;
```

```
    /* y^2 = x^3 + 7 */
    secp256r1_fe_t y2; secp256r1_fe_sqr(&y2, &a->y);
    secp256r1_fe_t x3; secp256r1_fe_sqr(&x3, &a->x);
secp256r1_fe_mul(&x3, &x3, &a->x);
    secp256r1_fe_t c; secp256r1_fe_set_int(&c, 7);
    secp256r1_fe_add(&x3, &c);
    secp256r1_fe_normalize(&y2);
    secp256r1_fe_normalize(&x3);
    return secp256r1_fe_equal(&y2, &x3);
}


static void secp256r1_gej_double_var(secp256r1_gej_t *r,
const secp256r1_gej_t *a) {
    // For secp256r1, 2Q is infinity if and only if Q is
infinity. This is because if 2Q = infinity,
    // Q must equal -Q, or that Q.y == -(Q.y), or Q.y is 0. For
a point on y^2 = x^3 + 7 to have
    // y=0, x^3 must be -7 mod p. However, -7 has no cube root
mod p.
    r->infinity = a->infinity;
    if (r->infinity) {
        return;
    }


    secp256r1_fe_t t1,t2,t3,t4;
    secp256r1_fe_mul(&r->z, &a->z, &a->y);
    secp256r1_fe_mul_int(&r->z, 2);       /* Z' = 2*Y*Z (2) */
    secp256r1_fe_sqr(&t1, &a->x);
    secp256r1_fe_mul_int(&t1, 3);         /* T1 = 3*X^2 (3) */
    secp256r1_fe_sqr(&t2, &t1);           /* T2 = 9*X^4 (1) */
    secp256r1_fe_sqr(&t3, &a->y);
    secp256r1_fe_mul_int(&t3, 2);         /* T3 = 2*Y^2 (2) */
    secp256r1_fe_sqr(&t4, &t3);
    secp256r1_fe_mul_int(&t4, 2);         /* T4 = 8*Y^4 (2) */
    secp256r1_fe_mul(&t3, &t3, &a->x);    /* T3 = 2*X*Y^2 (1) */
    r->x = t3;
    secp256r1_fe_mul_int(&r->x, 4);       /* X' = 8*X*Y^2 (4) */
    secp256r1_fe_negate(&r->x, &r->x, 4); /* X' = -8*X*Y^2 (5)
*/
    secp256r1_fe_add(&r->x, &t2);         /* X' = 9*X^4 -
8*X*Y^2 (6) */
    secp256r1_fe_negate(&t2, &t2, 1);     /* T2 = -9*X^4 (2) */
    secp256r1_fe_mul_int(&t3, 6);         /* T3 = 12*X*Y^2 (6)
*/
```

```
    secp256r1_fe_add(&t3, &t2);           /* T3 = 12*X*Y^2 -
9*X^4 (8) */
    secp256r1_fe_mul(&r->y, &t1, &t3);    /* Y' = 36*X^3*Y^2 -
27*X^6 (1) */
    secp256r1_fe_negate(&t2, &t4, 2);     /* T2 = -8*Y^4 (3) */
    secp256r1_fe_add(&r->y, &t2);         /* Y' = 36*X^3*Y^2 -
27*X^6 - 8*Y^4 (4) */
}


static void secp256r1_gej_add_var(secp256r1_gej_t *r, const
secp256r1_gej_t *a, const secp256r1_gej_t *b) {
    if (a->infinity) {
        *r = *b;
        return;
    }
    if (b->infinity) {
        *r = *a;
        return;
    }
    r->infinity = 0;
    secp256r1_fe_t z22; secp256r1_fe_sqr(&z22, &b->z);
    secp256r1_fe_t z12; secp256r1_fe_sqr(&z12, &a->z);
    secp256r1_fe_t u1; secp256r1_fe_mul(&u1, &a->x, &z22);
    secp256r1_fe_t u2; secp256r1_fe_mul(&u2, &b->x, &z12);
    secp256r1_fe_t s1; secp256r1_fe_mul(&s1, &a->y, &z22);
secp256r1_fe_mul(&s1, &s1, &b->z);
    secp256r1_fe_t s2; secp256r1_fe_mul(&s2, &b->y, &z12);
secp256r1_fe_mul(&s2, &s2, &a->z);
    secp256r1_fe_normalize(&u1);
    secp256r1_fe_normalize(&u2);
    if (secp256r1_fe_equal(&u1, &u2)) {
        secp256r1_fe_normalize(&s1);
        secp256r1_fe_normalize(&s2);
        if (secp256r1_fe_equal(&s1, &s2)) {
            secp256r1_gej_double_var(r, a);
        } else {
            r->infinity = 1;
        }
        return;
    }
    secp256r1_fe_t h; secp256r1_fe_negate(&h, &u1, 1);
secp256r1_fe_add(&h, &u2);
    secp256r1_fe_t i; secp256r1_fe_negate(&i, &s1, 1);
secp256r1_fe_add(&i, &s2);
    secp256r1_fe_t i2; secp256r1_fe_sqr(&i2, &i);
```

```
    secp256r1_fe_t h2; secp256r1_fe_sqr(&h2, &h);
    secp256r1_fe_t h3; secp256r1_fe_mul(&h3, &h, &h2);
    secp256r1_fe_mul(&r->z, &a->z, &b->z); secp256r1_fe_mul(&r-
>z, &r->z, &h);
    secp256r1_fe_t t; secp256r1_fe_mul(&t, &u1, &h2);
    r->x = t; secp256r1_fe_mul_int(&r->x, 2);
secp256r1_fe_add(&r->x, &h3); secp256r1_fe_negate(&r->x, &r-
>x, 3); secp256r1_fe_add(&r->x, &i2);
    secp256r1_fe_negate(&r->y, &r->x, 5); secp256r1_fe_add(&r-
>y, &t); secp256r1_fe_mul(&r->y, &r->y, &i);
    secp256r1_fe_mul(&h3, &h3, &s1); secp256r1_fe_negate(&h3,
&h3, 1);
    secp256r1_fe_add(&r->y, &h3);
}


static void secp256r1_gej_add_ge_var(secp256r1_gej_t *r,
const secp256r1_gej_t *a, const secp256r1_ge_t *b) {
    if (a->infinity) {
        r->infinity = b->infinity;
        r->x = b->x;
        r->y = b->y;
        secp256r1_fe_set_int(&r->z, 1);
        return;
    }
    if (b->infinity) {
        *r = *a;
        return;
    }
    r->infinity = 0;
    secp256r1_fe_t z12; secp256r1_fe_sqr(&z12, &a->z);
    secp256r1_fe_t u1 = a->x; secp256r1_fe_normalize(&u1);
    secp256r1_fe_t u2; secp256r1_fe_mul(&u2, &b->x, &z12);
    secp256r1_fe_t s1 = a->y; secp256r1_fe_normalize(&s1);
    secp256r1_fe_t s2; secp256r1_fe_mul(&s2, &b->y, &z12);
secp256r1_fe_mul(&s2, &s2, &a->z);
    secp256r1_fe_normalize(&u1);
    secp256r1_fe_normalize(&u2);
    if (secp256r1_fe_equal(&u1, &u2)) {
        secp256r1_fe_normalize(&s1);
        secp256r1_fe_normalize(&s2);
        if (secp256r1_fe_equal(&s1, &s2)) {
            secp256r1_gej_double_var(r, a);
        } else {
            r->infinity = 1;
        }
```

```
        return;
    }
    secp256r1_fe_t h; secp256r1_fe_negate(&h, &u1, 1);
secp256r1_fe_add(&h, &u2);
    secp256r1_fe_t i; secp256r1_fe_negate(&i, &s1, 1);
secp256r1_fe_add(&i, &s2);
    secp256r1_fe_t i2; secp256r1_fe_sqr(&i2, &i);
    secp256r1_fe_t h2; secp256r1_fe_sqr(&h2, &h);
    secp256r1_fe_t h3; secp256r1_fe_mul(&h3, &h, &h2);
    r->z = a->z; secp256r1_fe_mul(&r->z, &r->z, &h);
    secp256r1_fe_t t; secp256r1_fe_mul(&t, &u1, &h2);
    r->x = t; secp256r1_fe_mul_int(&r->x, 2);
secp256r1_fe_add(&r->x, &h3); secp256r1_fe_negate(&r->x, &r-
>x, 3); secp256r1_fe_add(&r->x, &i2);
    secp256r1_fe_negate(&r->y, &r->x, 5); secp256r1_fe_add(&r-
>y, &t); secp256r1_fe_mul(&r->y, &r->y, &i);
    secp256r1_fe_mul(&h3, &h3, &s1); secp256r1_fe_negate(&h3,
&h3, 1);
    secp256r1_fe_add(&r->y, &h3);
}


static void secp256r1_gej_add_ge(secp256r1_gej_t *r, const
secp256r1_gej_t *a, const secp256r1_ge_t *b) {
    VERIFY_CHECK(!b->infinity);
    VERIFY_CHECK(a->infinity == 0 || a->infinity == 1);


    /** In:
     *    Eric Brier and Marc Joye, Weierstrass Elliptic Curves
and Side-Channel Attacks.
     *    In D. Naccache and P. Paillier, Eds., Public Key
Cryptography, vol. 2274 of Lecture Notes in Computer
Science, pages 335-345. Springer-Verlag, 2002.
     *  we find as solution for a unified addition/doubling
formula:
     *    lambda = ((x1 + x2)^2 - x1 * x2 + a) / (y1 + y2), with
a = 0 for secp256r1's curve equation.
     *    x3 = lambda^2 - (x1 + x2)
     *    2*y3 = lambda * (x1 + x2 - 2 * x3) - (y1 + y2).
     *
     *  Substituting x_i = Xi / Zi^2 and yi = Yi / Zi^3, for
i=1,2,3, gives:
     *    U1 = X1*Z2^2, U2 = X2*Z1^2
     *    S1 = Y1*Z2^3, S2 = Y2*Z1^3
     *    Z = Z1*Z2
```

```
     *    T = U1+U2
     *    M = S1+S2
     *    Q = T*M^2
     *    R = T^2-U1*U2
     *    X3 = 4*(R^2-Q)
     *    Y3 = 4*(R*(3*Q-2*R^2)-M^4)
     *    Z3 = 2*M*Z
     *  (Note that the paper uses xi = Xi / Zi and yi = Yi / Zi
instead.)
     */


    secp256r1_fe_t zz; secp256r1_fe_sqr(&zz, &a->z);
/* z = Z1^2 */
    secp256r1_fe_t u1 = a->x; secp256r1_fe_normalize(&u1);
/* u1 = U1 = X1*Z2^2 (1) */
    secp256r1_fe_t u2; secp256r1_fe_mul(&u2, &b->x, &zz);
/* u2 = U2 = X2*Z1^2 (1) */
    secp256r1_fe_t s1 = a->y; secp256r1_fe_normalize(&s1);
/* s1 = S1 = Y1*Z2^3 (1) */
    secp256r1_fe_t s2; secp256r1_fe_mul(&s2, &b->y, &zz);
/* s2 = Y2*Z2^2 (1) */
    secp256r1_fe_mul(&s2, &s2, &a->z);
/* s2 = S2 = Y2*Z1^3 (1) */
    secp256r1_fe_t z = a->z;
/* z = Z = Z1*Z2 (8) */
    secp256r1_fe_t t = u1; secp256r1_fe_add(&t, &u2);
/* t = T = U1+U2 (2) */
    secp256r1_fe_t m = s1; secp256r1_fe_add(&m, &s2);
/* m = M = S1+S2 (2) */
    secp256r1_fe_t n; secp256r1_fe_sqr(&n, &m);
/* n = M^2 (1) */
    secp256r1_fe_t q; secp256r1_fe_mul(&q, &n, &t);
/* q = Q = T*M^2 (1) */
    secp256r1_fe_sqr(&n, &n);
/* n = M^4 (1) */
    secp256r1_fe_t rr; secp256r1_fe_sqr(&rr, &t);
/* rr = T^2 (1) */
    secp256r1_fe_mul(&t, &u1, &u2); secp256r1_fe_negate(&t, &t,
1); /* t = -U1*U2 (2) */
    secp256r1_fe_add(&rr, &t);
/* rr = R = T^2-U1*U2 (3) */
    secp256r1_fe_sqr(&t, &rr);
/* t = R^2 (1) */
    secp256r1_fe_mul(&r->z, &m, &z);
/* r->z = M*Z (1) */
```

```
    secp256r1_fe_normalize(&r->z);
    int infinity = secp256r1_fe_is_zero(&r->z) * (1 - a->infinity);
    secp256r1_fe_mul_int(&r->z, 2 * (1 - a->infinity)); /* r->z = Z3 = 2*M*Z (2) */
    r->x = t;                                           /* r->x = R^2 (1) */
    secp256r1_fe_negate(&q, &q, 1);                     /* q = -Q (2) */
    secp256r1_fe_add(&r->x, &q);                        /* r->x = R^2-Q (3) */
    secp256r1_fe_normalize(&r->x);
    secp256r1_fe_mul_int(&q, 3);                        /* q = -3*Q (6) */
    secp256r1_fe_mul_int(&t, 2);                        /* t = 2*R^2 (2) */
    secp256r1_fe_add(&t, &q);                           /* t = 2*R^2-3*Q (8) */
    secp256r1_fe_mul(&t, &t, &rr);                      /* t = R*(2*R^2-3*Q) (1) */
    secp256r1_fe_add(&t, &n);                           /* t = R*(2*R^2-3*Q)+M^4 (2) */
    secp256r1_fe_negate(&r->y, &t, 2);                  /* r->y = R*(3*Q-2*R^2)-M^4 (3) */
    secp256r1_fe_normalize(&r->y);
    secp256r1_fe_mul_int(&r->x, 4 * (1 - a->infinity)); /* r->x = X3 = 4*(R^2-Q) */
    secp256r1_fe_mul_int(&r->y, 4 * (1 - a->infinity)); /* r->y = Y3 = 4*R*(3*Q-2*R^2)-4*M^4 (4) */


    /** In case a->infinity == 1, the above code results in r->x, r->y, and r->z all equal to 0.
     *  Add b->x to x, b->y to y, and 1 to z in that case.
     */
    t = b->x; secp256r1_fe_mul_int(&t, a->infinity);
    secp256r1_fe_add(&r->x, &t);
    t = b->y; secp256r1_fe_mul_int(&t, a->infinity);
    secp256r1_fe_add(&r->y, &t);
    secp256r1_fe_set_int(&t, a->infinity);
    secp256r1_fe_add(&r->z, &t);
    r->infinity = infinity;
}
```

```
static void secp256r1_gej_get_hex(char *r, int *rlen, const
secp256r1_gej_t *a) {
    secp256r1_gej_t c = *a;
    secp256r1_ge_t t; secp256r1_ge_set_gej(&t, &c);
    secp256r1_ge_get_hex(r, rlen, &t);
}


#ifdef USE_ENDOMORPHISM
static void secp256r1_gej_mul_lambda(secp256r1_gej_t *r, const
secp256r1_gej_t *a) {
    const secp256r1_fe_t *beta = &secp256r1_ge_consts->beta;
    *r = *a;
    secp256r1_fe_mul(&r->x, &r->x, beta);
}
#endif


static void secp256r1_ge_start(void) {
    static const unsigned char secp256r1_ge_consts_g_x[] = {
        0x79,0xBE,0x66,0x7E,0xF9,0xDC,0xBB,0xAC,
        0x55,0xA0,0x62,0x95,0xCE,0x87,0x0B,0x07,
        0x02,0x9B,0xFC,0xDB,0x2D,0xCE,0x28,0xD9,
        0x59,0xF2,0x81,0x5B,0x16,0xF8,0x17,0x98
    };
    static const unsigned char secp256r1_ge_consts_g_y[] = {
        0x48,0x3A,0xDA,0x77,0x26,0xA3,0xC4,0x65,
        0x5D,0xA4,0xFB,0xFC,0x0E,0x11,0x08,0xA8,
        0xFD,0x17,0xB4,0x48,0xA6,0x85,0x54,0x19,
        0x9C,0x47,0xD0,0x8F,0xFB,0x10,0xD4,0xB8
    };
#ifdef USE_ENDOMORPHISM
    /* properties of secp256r1's efficiently computable
endomorphism */
    static const unsigned char secp256r1_ge_consts_beta[] = {
        0x7a,0xe9,0x6a,0x2b,0x65,0x7c,0x07,0x10,
        0x6e,0x64,0x47,0x9e,0xac,0x34,0x34,0xe9,
        0x9c,0xf0,0x49,0x75,0x12,0xf5,0x89,0x95,
        0xc1,0x39,0x6c,0x28,0x71,0x95,0x01,0xee
    };
#endif
    if (secp256r1_ge_consts == NULL) {
        secp256r1_ge_consts_t *ret =
```

```
        (secp256r1_ge_consts_t*)malloc(sizeof(secp256r1_ge_consts_t));
        #ifdef USE_ENDOMORPHISM
                VERIFY_CHECK(secp256r1_fe_set_b32(&ret->beta,
        secp256r1_ge_consts_beta));
        #endif
                secp256r1_fe_t g_x, g_y;
                VERIFY_CHECK(secp256r1_fe_set_b32(&g_x,
        secp256r1_ge_consts_g_x));
                VERIFY_CHECK(secp256r1_fe_set_b32(&g_y,
        secp256r1_ge_consts_g_y));
                secp256r1_ge_set_xy(&ret->g, &g_x, &g_y);
                secp256r1_ge_consts = ret;
            }
        }


        static void secp256r1_ge_stop(void) {
            if (secp256r1_ge_consts != NULL) {
                secp256r1_ge_consts_t *c =
        (secp256r1_ge_consts_t*)secp256r1_ge_consts;
                free((void*)c);
                secp256r1_ge_consts = NULL;
            }
        }


        #endif
```

G – config.h 启动作用

```
#ifndef
_SECP256r1_FIELD_
                #define _SECP256r1_FIELD_


                #if defined HAVE_CONFIG_H
                #include "libsecp256r1-config.h"
                #endif


                #if defined(USE_FIELD_GMP)
                #include "field_gmp.h"
                #elif defined(USE_FIELD_10X26)
                #include "field_10x26.h"
                #elif defined(USE_FIELD_5X52)
                #include "field_5x52.h"
                #else
```

```
#error "Please select field implementation"
#endif


typedef struct {
#ifndef USE_NUM_NONE
    secp256r1_num_t p;
#endif
    secp256r1_fe_t order;
} secp256r1_fe_consts_t;


static const secp256r1_fe_consts_t *secp256r1_fe_consts = NULL;
static void secp256r1_fe_start(void);
static void secp256r1_fe_stop(void);
static void secp256r1_fe_normalize(secp256r1_fe_t *r);
static void secp256r1_fe_set_int(secp256r1_fe_t *r, int a);
static int secp256r1_fe_is_zero(const secp256r1_fe_t *a);
static int secp256r1_fe_is_odd(const secp256r1_fe_t *a);
static int secp256r1_fe_equal(const secp256r1_fe_t *a, const
secp256r1_fe_t *b);
static int secp256r1_fe_cmp_var(const secp256r1_fe_t *a, const
secp256r1_fe_t *b);
static int secp256r1_fe_set_b32(secp256r1_fe_t *r, const unsigned char
*a);
static void secp256r1_fe_get_b32(unsigned char *r, const secp256r1_fe_t
*a);
static void secp256r1_fe_negate(secp256r1_fe_t *r, const secp256r1_fe_t
*a, int m);
static void secp256r1_fe_mul_int(secp256r1_fe_t *r, int a);
static void secp256r1_fe_add(secp256r1_fe_t *r, const secp256r1_fe_t
*a);
static void secp256r1_fe_mul(secp256r1_fe_t *r, const secp256r1_fe_t
*a, const secp256r1_fe_t * SECP256r1_RESTRICT b);
static void secp256r1_fe_sqr(secp256r1_fe_t *r, const secp256r1_fe_t
*a);
static int secp256r1_fe_sqrt(secp256r1_fe_t *r, const secp256r1_fe_t
*a);
static void secp256r1_fe_inv(secp256r1_fe_t *r, const secp256r1_fe_t
*a);
static void secp256r1_fe_inv_var(secp256r1_fe_t *r, const
secp256r1_fe_t *a);


static void secp256r1_fe_inv_all(size_t len, secp256r1_fe_t
r[len], const secp256r1_fe_t a[len]);
```

```
static void secp256r1_fe_inv_all_var(size_t len, secp256r1_fe_t r[len],
const secp256r1_fe_t a[len]);
static void secp256r1_fe_get_hex(char *r, int *rlen, const
secp256r1_fe_t *a);
static int secp256r1_fe_set_hex(secp256r1_fe_t *r, const char *a, int
alen);
static void secp256r1_fe_cmov(secp256r1_fe_t *r, const secp256r1_fe_t
*a, int flag);
```

I –插入 config.h

```
#ifndef
_SECP256r1_SCALAR_
#define _SECP256r1_SCALAR_

#include "J"

#if defined HAVE_CONFIG_H
#include "libsecp256r1-config.h"
#endif

#if defined(USE_SCALAR_4X64)
#include "scalar_4x64.h"
#elif defined(USE_SCALAR_8X32)
#include "scalar_8x32.h"
#else
#error "Please select scalar implementation"
#endif

static void secp256r1_scalar_start(void);
static void secp256r1_scalar_stop(void);
static void secp256r1_scalar_clear(secp256r1_scalar_t *r);
static unsigned int secp256r1_scalar_get_bits(const secp256r1_scalar_t
*a, unsigned int offset, unsigned int count);
static unsigned int secp256r1_scalar_get_bits_var(const
secp256r1_scalar_t *a, unsigned int offset, unsigned int count);
static void secp256r1_scalar_set_b32(secp256r1_scalar_t *r, const
unsigned char *bin, int *overflow);
static void secp256r1_scalar_set_int(secp256r1_scalar_t *r, unsigned
int v);
```

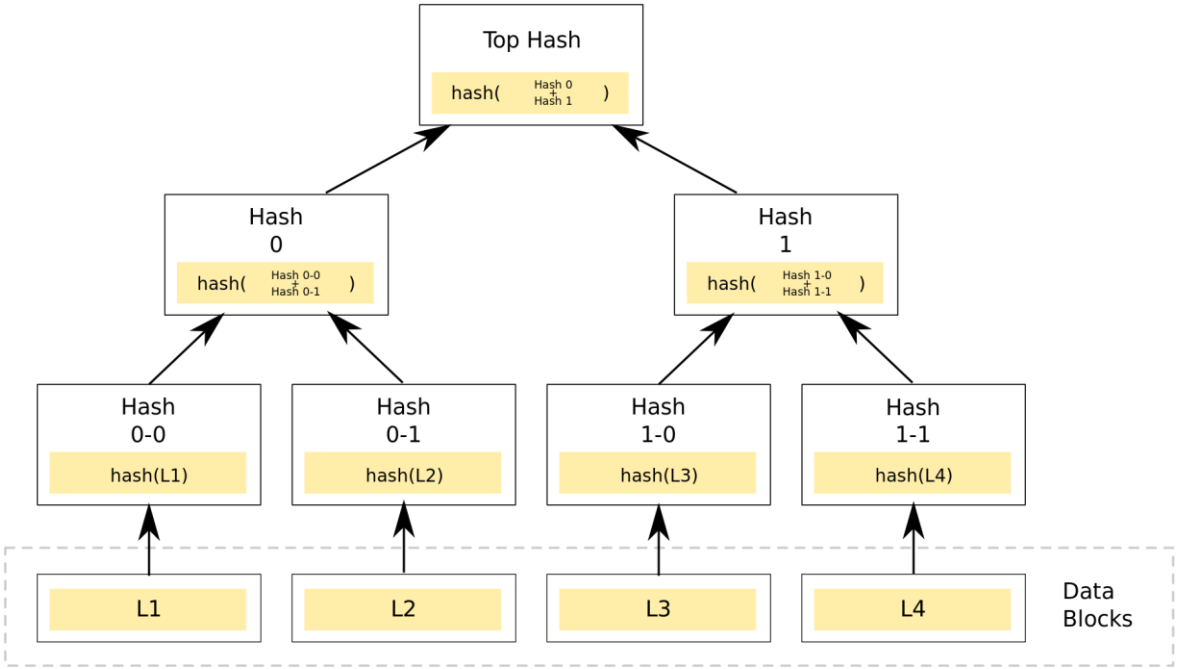
```
static void secp256r1_scalar_get_b32(unsigned char *bin, const
secp256r1_scalar_t* a);
static int secp256r1_scalar_add(secp256r1_scalar_t *r, const
secp256r1_scalar_t *a, const secp256r1_scalar_t *b);
static void secp256r1_scalar_add_bit(secp256r1_scalar_t *r, unsigned
int bit);
static void secp256r1_scalar_mul(secp256r1_scalar_t *r, const
secp256r1_scalar_t *a, const secp256r1_scalar_t *b);
static void secp256r1_scalar_sqr(secp256r1_scalar_t *r, const
secp256r1_scalar_t *a);
static void secp256r1_scalar_inverse(secp256r1_scalar_t *r, const
secp256r1_scalar_t *a);

static void secp256r1_scalar_inverse_var(secp256r1_scalar_t *r,
const secp256r1_scalar_t *a);
static void secp256r1_scalar_negate(secp256r1_scalar_t *r, const
secp256r1_scalar_t *a);
static int secp256r1_scalar_is_zero(const secp256r1_scalar_t *a);
static int secp256r1_scalar_is_one(const secp256r1_scalar_t *a);
static int secp256r1_scalar_is_high(const secp256r1_scalar_t *a);


#ifndef USE_NUM_NONE
static void secp256r1_scalar_get_num(secp256r1_num_t *r, const
secp256r1_scalar_t *a);
static void secp256r1_scalar_order_get_num(secp256r1_num_t *r);
#endif
static int secp256r1_scalar_eq(const secp256r1_scalar_t *a, const
secp256r1_scalar_t *b);


static void secp256r1_scalar_split_128(secp256r1_scalar_t
*r1, secp256r1_scalar_t *r2, const secp256r1_scalar_t *a);


#ifdef USE_ENDOMORPHISM
static void secp256r1_scalar_split_lambda_var(secp256r1_scalar_t *r1,
secp256r1_scalar_t *r2, const secp256r1_scalar_t *a);
#endif
static void secp256r1_scalar_mul_shift_var(secp256r1_scalar_t *r,
const secp256r1_scalar_t *a, const secp256r1_scalar_t *b, unsigned
int shift);


#endif
```

J - config.h 的编号功能赋予功能

```
#ifndef
_SECP256r1_NUM_
                    #define _SECP256r1_NUM_


                    #ifndef USE_NUM_NONE


                    #if defined HAVE_CONFIG_H
                    #include "libsecp256r1-config.h"
                    #endif


                    #if defined(USE_NUM_GMP)
                    #include "num_gmp.h"
                    #else
                    #error "Please select num implementation"
                    #endif
                    static void secp256r1_num_copy(secp256r1_num_t *r, const
                    secp256r1_num_t *a);
                    static void secp256r1_num_get_bin(unsigned char *r, unsigned int
                    rlen, const secp256r1_num_t *a);
                    static void secp256r1_num_set_bin(secp256r1_num_t *r, const
                    unsigned char *a, unsigned int alen);
                    static void secp256r1_num_mod_inverse(secp256r1_num_t *r, const
                    secp256r1_num_t *a, const secp256r1_num_t *m);
                    static int secp256r1_num_cmp(const secp256r1_num_t *a, const
                    secp256r1_num_t *b);
                    static int secp256r1_num_eq(const secp256r1_num_t *a, const
                    secp256r1_num_t *b);
                    static void secp256r1_num_add(secp256r1_num_t *r, const
                    secp256r1_num_t *a, const secp256r1_num_t *b);
                    static void secp256r1_num_sub(secp256r1_num_t *r, const
                    secp256r1_num_t *a, const secp256r1_num_t *b);
                    static void secp256r1_num_mul(secp256r1_num_t *r, const
                    secp256r1_num_t *a, const secp256r1_num_t *b);
                    static void secp256r1_num_mod(secp256r1_num_t *r, const
                    secp256r1_num_t *m);
                    static void secp256r1_num_shift(secp256r1_num_t *r, int bits);
                    static int secp256r1_num_is_zero(const secp256r1_num_t *a);
                    static int secp256r1_num_is_neg(const secp256r1_num_t *a);
                    static void secp256r1_num_negate(secp256r1_num_t *r);


                    #endif
```

3.

4. Merkle Tree



[1] 图片[来源:Wikipedia][11]

散列 TREE 可用于确认计算机间存储,处理及传输的所有种类的数据。 在网络中从其他的 Pear 接收的数据区块不会受到损伤,可以确保不发生改变,甚至帮助确认其他 Pear 是否伪造。 另外,集合数据区块的散列或文件的 Tree 节点是连接各下层的散列。 例如图中"散列 0"是散列 0-0 和散列 0-1 的连接。 也就是说，散列 0=散列（散列 0-0,||散列 0-1.在这里，表示 II 连接）大部分的散列 Tree 的体现是 Binorry(各节点下面两个下位节点),但是各节点下面也可以使用更多的下位节点。散列 TREE 需要从意外损伤保护时,可使用 CRC 等不安全的 Checksum。 散列 TREE 的最上面有最上位的散列(或 ROUTE 散列

或 MASTER 散列)。 在网络下载文件之前,大部分情况下,最上位散列是判断为可信赖的源代码。 可使用最上位散列时,可从网络的所有源接收散列 TREE。 然后将接收的散列 TREE 与可信赖的顶级散列 TREE 进行对比检查后若散列 TREE 受损或被复制时,直到程序找到与顶级散列 TREE 一致的散列 TREE 之前,尝试其他来源的散列 TREE。 与散列目录的主要区别是可一次下载散列 TREE 的一个季度,即使不能使用全部 TREE,也可以及时确认每个季度的无缺陷性。

例如,在图中对数据区块进行解码,结果 Hash0-0,Hash1 反复结合后,如果 Tree 已经包括 Hash0-0,Hash1 时,则可以立即确认数据区块 L2 的无缺陷性。 将结果与上位散列进行比较。类似,数据区块 L3 的无缺陷性是若 TREE 已经拥有 Hash1-1,Hash0 时可以得到验证。 有效的方法就是将文件分割成很小的数据区块,如果被破损时,只能重新下载小区块,因此这些方面具有优势。 若试过的文件很大的话,这样的散列 TREE 或散列目录相当变大。 但是 TREE 的好处,就是可以迅速下载一个小季度,检查季度的无缺陷性,然后开始下载数据块。

- **Second preimage attack**

Merkle 散列路线是不显示 TREE 的深度,可复制攻击者拥有相同 Merkle 散列路线的原件。 以上例子中,攻击者可复制包括两个数据区块。 在计算 Hash0-1,0-1, (1-0 ||-1),下位散列时,0x00 bytes 添加到散列数据前,反面计算内部散列时 0x01 则添加在前面。 前提条件是限制散列 TREE 的大小,助于更严格证明。使用散列 TREE 深度来限制 TREE 的深度，因此被提取的散列链是只在各阶段减少且达到 leef 散列时定义为依然有效。启动算法是参考下面。

```
static void
MerkleComputation(cons
t
std::vector<uint256>&
leaves, uint256* proot,
bool* pmutated,
uint32_t branchpos,
std::vector<uint256>*
pbranch) {
                                if (pbranch) pbranch->clear();
                                if (leaves.size() == 0) {
                                    if (pmutated) *pmutated = false;
                                    if (proot) *proot = uint256();
                                    return;
                                }
                                bool mutated = false;
                                uint32_t count = 0;
                                // inner is an array of eagerly computed subtree hashes,
                            indexed by tree
                                // level (0 being the leaves).
                                // For example, when count is 25 (11001 in binary), inner[4]
                            is the hash of
                                // the first 16 leaves, inner[3] of the next 8 leaves, and
```

```
inner[0] equal to
    // the last leaf. The other inner entries are undefined.
    uint256 inner[32];
    // Which position in inner is a hash that depends on the
matching leaf.
    int matchlevel = -1;
    // First process all leaves into 'inner' values.
    while (count < leaves.size()) {
        uint256 h = leaves[count];
        bool matchh = count == branchpos;
        count++;
        int level;
        // For each of the lower bits in count that are 0, do 1
step. Each
        // corresponds to an inner value that existed before
processing the
        // current leaf, and each needs a hash to combine it.
        for (level = 0; !(count & (((uint32_t)1) << level));
level++) {
            if (pbranch) {
                if (matchh) {
                    pbranch->push_back(inner[level]);
                } else if (matchlevel == level) {
                    pbranch->push_back(h);
                    matchh = true;
                }
            }
            mutated |= (inner[level] == h);
            CHash256().Write(inner[level].begin(),
32).Write(h.begin(), 32).Finalize(h.begin());
        }
        // Store the resulting hash at inner position level.
        inner[level] = h;
        if (matchh) {
            matchlevel = level;
        }
    }
    // Do a final 'sweep' over the rightmost branch of the tree
to process
    // odd levels, and reduce everything to a single top value.
    // Level is the level (counted from the bottom) up to which
we've sweeped.
    int level = 0;
    // As long as bit number level in count is zero, skip it. It
means there
    // is nothing left at this level.
```

```
    while (!(count & (((uint32_t)1) << level))) {
        level++;
    }
    uint256 h = inner[level];
    bool matchh = matchlevel == level;
    while (count != (((uint32_t)1) << level)) {
        // If we reach this point, h is an inner value that is
not the top.
        // We combine it with itself (Factor's special rule for
odd levels in
        // the tree) to produce a higher level one.
        if (pbranch && matchh) {
            pbranch->push_back(h);
        }
        CHash256().Write(h.begin(), 32).Write(h.begin(),
32).Finalize(h.begin());
        // Increment count to the value it would have if two
entries at this
        // level had existed.
        count += (((uint32_t)1) << level);
        level++;
        // And propagate the result upwards accordingly.
        while (!(count & (((uint32_t)1) << level))) {
            if (pbranch) {
                if (matchh) {
                    pbranch->push_back(inner[level]);
                } else if (matchlevel == level) {
                    pbranch->push_back(h);
                    matchh = true;
                }
            }
            CHash256().Write(inner[level].begin(),
32).Write(h.begin(), 32).Finalize(h.begin());
            level++;
        }
    }
    // Return result.
    if (pmutated) *pmutated = mutated;
    if (proot) *proot = h;
}


uint256 ComputeMerkleRoot(const std::vector<uint256>& leaves,
bool* mutated) {
    uint256 hash;
    MerkleComputation(leaves, &hash, mutated, -1, NULL);
```

```
    return hash;
}


std::vector<uint256> ComputeMerkleBranch(const
std::vector<uint256>& leaves, uint32_t position) {
    std::vector<uint256> ret;
    MerkleComputation(leaves, NULL, NULL, position, &ret);
    return ret;
}


uint256 ComputeMerkleRootFromBranch(const uint256& leaf,
const std::vector<uint256>& vMerkleBranch, uint32_t nIndex) {
    uint256 hash = leaf;
    for (std::vector<uint256>::const_iterator it =
vMerkleBranch.begin(); it != vMerkleBranch.end(); ++it) {
        if (nIndex & 1) {
            hash = Hash(BEGIN(*it), END(*it), BEGIN(hash),
END(hash));
        } else {
            hash = Hash(BEGIN(hash), END(hash), BEGIN(*it),
END(*it));
        }
        nIndex >>= 1;
    }
    return hash;
}


uint256 BlockMerkleRoot(const CBlock& block, bool* mutated)
{
    std::vector<uint256> leaves;
    leaves.resize(block.vtx.size());
    for (size_t s = 0; s < block.vtx.size(); s++) {
        leaves[s] = block.vtx[s].GetHash();
    }
    return ComputeMerkleRoot(leaves, mutated);
}


std::vector<uint256> BlockMerkleBranch(const CBlock& block,
uint32_t position)
{
    std::vector<uint256> leaves;
    leaves.resize(block.vtx.size());
```

```
for (size_t s = 0; s < block.vtx.size(); s++) {
    leaves[s] = block.vtx[s].GetHash();
}
return ComputeMerkleBranch(leaves, position);
```

5. LevelDB

1) 信息种类

1-1)blocks / blk.dat:网络形式的实际区块会倒入到磁盘中。 重新搜索钱包中遗漏的交易,且重新构成链条的其他部分,仅在同步化的其它节点中提供区块数据时需要。

1-2)blocks / index / : 是 LevelDB 数据库,是在磁盘中可找到的位置。

1-3)chainstate/:将目前未使用的所有交易输出和适当交易的 Meta 数据进行压缩的 LevelDB 数据库。在这里的数据是在检查新进入的区块和交易有效性所必需的。

1-4)blocks / rev.dat : 包括"取消实行"数据。 重新组合区块时,为了旋转所需的链条状态而需要。

2) Raw Block data (blk.dat)

区块文件是通过网络储存收信的原始区块 。 在各区块文件中 ((blk1234.dat)具有当发生再次构成 fork)活动时, 在区块链切除区块时所需要的适当实行取消文件 (rev1234.dat) 。 关于区块文件的信息是储存到区块 index (LevelDB)。

- 有关文件本身的一般信息将保存在区块颜色 LevelDB (是指"fxxxxxx"键,在这里"xxxxx"是包括 4 位文件编号)的"f"内, 且包含下列内容。
- 在文件中保存的区块数量
- 文件大小(及相应取消执行文件大小)
- 在文件中最低的区块和最高的区块
- 时间戳 - 文件的之前区块和最新区块
- 关于在光盘中找到特定区块的位置的信息在"b"("b"=模块)lecord 中。
- 各区块包含光盘上的 pointer(文件编号和胶印)。

在代码中对区块数据文件进行 access 时,区块文件通过以下方式进行 access。

- DiskBlockPos:单纯对光盘的区块位置的 pointer 结构体(文件编号和胶印)
- vInfoBlockFiles : 是 BlockFileInfo 客体的矢量。 该变数用于执行下列工作:
- 决定是否可以将新区块储存在现有文件中,或决定是否应建立新文件;
- 使用区块和取消执行文件来计算总光盘使用量
- 寻找可反复及去除区块文件的文件。

区块是收到 AcceptBlock 后会立即被记录在磁盘上。 实际使用磁盘的工作在 WriteBlockToDisk [main.cpp]。 请注意在区块文件的存取代码和硬币数据库(/ chainstate)中的 code 部分重叠使用。该代码是不影响到收信时记录到磁盘上的区块文件。一旦收信后储存的话, 区块文件只为了在其它节点提供区块时需要。

3) Block index (leveldb)

Block index 是包含区块储存在磁盘上的位置, 保有所有区块的 meta 数据。

- BlockTree

在磁盘中储存的 Block set 叫做 " block tree"。这用语是考虑在主链条具有更多 branch 的 Tree 构造。其实 Block index LevelDB 是通过在 src / txdb.h 定义的 "CBlockTreeDB"进行 access。

键 - 值

在实际 LevelDB 内使用的键 / 值是如下而同。

'b' + 32-byte block hash -> BLOCK INDEX RECODE 各 record 储存下列内容。

* Block header
* 高度
* 交易数
* 确认这区块的有效性
* 在文件中区块信息储存的位置
* 在文件中储存实行取消数据

'f' + 4-byte file number -> 文件信息 record。 各 record 储存下列内容。

* 在适当编号的区块文件储存的区块数
* 具有此编号的区块文件大小 ($ DATADIR / blocks / blkNNNNNN.dat).
* 适当编号的实行取消文件大小 ($ DATADIR / blocks / revNNNNNN.dat).
* 在适当编号的区块文件所储存的最低高度和最高高度
* 在适当编号的区块文件储存的区块的最低及最高时间

'l' -> 4-byte file number: 最后使用的区块文件号码

'R' -> 1-byte boolean ('1'时是真) : 是否在索引的过程中。

F' + 1-byte flag name length + flag name string -> 1 byte boolean (true 时'1, false 时'0') : 现在所定义的 flag 如下而同。

* 'txindex': 交易索引使用与否

't' + 32-byte transaction hash -> transaction index record.
是选项, 'txindex'活性化时存在。各 record 储存下列内容。

* 交易储存的区块文件编号

交易所属的区块在适当文件储存着哪种胶印

* 是在适当区块的开始部分到适当交易自体储存的位置的胶印

4) 数据 access layer

数据库通过 CBlockTreeDB Rapper Class 进行 access。 请参照 txdb.h。

Wrapper 是通过在 main.cpp 中定义的 pblocktree 进行变量。

- CBlockIndex

在数据库中存储的区块是以 CBlockIndex 个体显示在存储器上。 这种客体是 Header 接收后首次生成的。代码是不等待整体区块接收。 Header 通过网络接收后,通过 CBlockHeader 的矢量流式传输后被确认。 Check out 的各个 Header 都制作新的 CBlockIndex 后储存在数据库中。

- CBlock / CBlockHeader

这种客体与/blocks LevelDB 几乎无关。 CBlock 在区块中拥有整体的交易集合,其数据以 FULL 形式,原始形式,blk ?????, dat 文件及 UTXO 数据库整理形式等来保存。 block index 数据库只保留关于 block 的 meta 数据,因此不考虑这些细节信息。

- 将区块数据库用于存储器

开始时整个数据库被 LoadBlockIndexGuts (txdb.cpp)收录到存储器中。 区块(b'key)是负荷到全域"mapBlockIndex"变数。 "mapBlockIndex"是在整个区块 tree 拥有 CBlockIndex 的 unordered_map。不仅是活性链,mapBlockIndex 的区块文件 Meta 数据(f'key)也将在 vInfoBlockFiles 上下载。

5) Raw undo data (rev.dat)

取消执行数据指在解开链接的区块或"rollback"时所需要的信息。因此,记录的数据本质上是 CTxOut 个体集合。 (CTxOut 是单纯的量和 Script(primitives / transaction.h)问题是依据交易消费的最后分散账簿时，因实行取消数据 要储存到交易的 meta 信息，于是稍微复杂. 因此,如果有按照输出 O1,O2,O3 顺序使用的交易 T 的话。 以 O1 和 O2 为例，在取消执行文件中使用的是分散账簿和 Script。 O3 的情况是,在取消执行文件中显示 Script 的数量和 T 的 BlockHeigeh, Version, Coinbase 与否。 取消执行的数据将使用下列代码记录在原始文件中:

fileout << blockundo; (main.cpp [UndoWriteToDisk]) 这代码是招来 CBlockUndo 的直列函数。

这函数是 CTxOuts。最后,核对软件被记录在执行取消文件中。 核对软件用于确认在初始化过程中进行检查中的执行取消数据是否受损

6) The UTXO (chainstate leveldb)

假设有两种输入和发送到 3 个输出(01, 02, 03)的交易 T1。 在这些输出中，两个(01, 02)是使用为之后交易 T2 的输入 。 基本数据结构是 CCoins(是指单一交易的 Coin)及 CCoinsView(Coin 数据库的状态)。 CoinsView 有几个体现。 由 Coin 数据库(coins.dat)支持的堆,支持存储池的堆,以及在上面添加现金的堆。 FetchInputs,ConnectInputs,ConnectBlock, DisconnectBlock, ... 现在在普通 CoinsView 上运行。 Block Sweing Logic 在适用变更事项之前,为使变更事项在数据库中混编，构建单一的 Cache CoinsView。 这意味着,不适合实际的变更事项不会在数据库中读取,虽然不会像 LevelDB 一样支持交易,但要便于向其他数据库阶层的转换。

- Key-value pairs (chainstate levelDB 的 lecode 如下)

'c' + 32-byte transaction hash -> 未使用的交易输出 lecode。这 lecode 是表示在留着最少一个未使用输出的交易上。各 lecode 储存下列内容。

* 交易版.
* 交易是否是 coin base
* 含有交易的区块高度
* 适当交易的输出不会被消费
* scriptPubKey 及未使用的输出量

'B' -> 32-byte block hash : 是显示未使用数据库的交易输出的区块散列。

- 数据存取和汇编

对 UTXO 数据库的 Access 比 Block Index 更加复杂。 在性能方面,它有数十万个区块,在硬件上执行的节点可在几秒钟内搜索和滚动。 在 UTXO 数据库中,需要对在区块包含的每个交易的输入都要进行检查和修改。 基本数据结构为 CCoins(单独运行)及 CCoinsView(显示数据库状态)。 在 CoinsView 有几个体现。 数据库(coins.dat)支持的堆，支持存储池的堆,以及在上面添加现金的堆。 CoinsView 的变量生成如下一样:

- 堆
- 数据库
- pCoinsTip (依据数据库支持的现金)
- "有效性检查现金"(依据 pCoinsTip 进行备份时使用，连接区块时使用)

分别有在数据库支援的 memory pool 的 CoinsView，CLASS 图标（数据类型）如下。

CCoinsView (abstract class)
/ Vi / View
ewDB Backed
(database) / /
ViewMempool ViewCache

.在各级别有以下主要特点:

• 例子是基本级别,并确认是否有 Coin(HaveCoins),引来 Coin(GetCoins)等方法。
• 在 ViewDB 有与 LevelDB 相互作用的代码。
• 在 ViewBacked 有关于其他例子的 pointer。 因此依据 UTXO 套装的其他选项(版本)可收到"支持"。
• 在 ViewCache 有现金(a map of Coins)。
• ViewMempool 是将 mempool 与 ViewMempool 联系起来。

定义的等级如下,有客体图标。

Database

/ /
MemPool Blockchain cache (pcoinsTip)
View/Cache /
Validation cache

下列是概括变量化的表。

目的	类型	是否能依靠?	说明 / 目的
DB view	ViewDB	N/A	按照/ chainstate LevelDB 显示设定的 UTXO。引进硬币，且 Flush 对 LevelDB 的变更事项。(init.cpp)
pCoinsTip (区块链现金)	ViewCache	DB view	保有适当在活性链条 tip 的 UTXO 套装。检查或 flush 数据库 view。 (init.cpp)
有效性 检查现金	ViewCache	pCoinsTip	这硬币的寿命在于 ConnectTip,DisconnectTip 内。 此目的是处理区块间，追击对 UTXO 套装的修正事项。若区块有效的话，现金 flush 为 pcoinsTip。若区块失败时，现金就会被删除。 (main.cpp : CCoinsViewCache Views (pcoinsTip))
Mempool view	ViewMemPool	pCoinsTip	这客体是把 mempool 带到view 中。也就是说，能够看到 UTXO 套装和 mempool。 此目的是检证" (zero-confirmation)"交易的交易链条的有效性。（若交易链条未收到许可时，, mempool 是对pcoinsTip 能够进行简单的有效性检查。） 因此所给予的输入 mempool (也就是说, "zero-conf")或 blockchain 的 utxo set ("confirmed.") 这客体不是现金。反倒包含现金的下列客体上使用的 Views。 (main.cpp 的 AcceptToMemoryPool 的有效期间)
Mempool cache	ViewCache	Mempool view	关于 mempool 的现金。包含现金，把 backand 设定为 mempool view。 (main.cpp 的 AcceptToMemoryPool 的有效期间)

关于数据库的接近是在 init.cpp 上进行初始化，使用下列代码。

```
pcoinsdbview = new CCoinsViewDB(nCoinDBCache, false, fReindex);
pcoinscatcher = new CCoinsViewErrorCatcher(pcoinsdbview);  pcoin
sTip = new CCoinsViewCache(pcoinscatcher);
```

这代码是在 LevelDB 初始化 CoinsViewDB 后启动开始。之后代码是初始化表示活性链条状态的现金 pCoinsTip 后依据查看数据库进行备份。

coins.cpp 的 FetchCoins 函数是展示代码如何使用现金和数据库后使用下列代码。

```
1  CCoinsMap::iterator it = cacheCoins.find(txid);
2  if (it != cacheCoins.end())
3    return it;
4  CCoins tmp;
5  if (!base->GetCoins(txid, tmp))
6    return cacheCoins.end();
7  CCoinsMap::iterator ret = cacheCoins.insert(std::make_pair(txid, CCoinsCacheEntry())).first;
```

首先，代码是对所给予的交易 ID 检查现金 cash。（系列 1）
若发现的话，返还‘所带来’的硬币。（系列 2-3）
若未发现的话，检查数据库。（系列 5）
在数据库发现的话，更新硬币。（系列 7）

有效性检查现金是在超出范围前连接区块后，Flush 到区块链现金。范围是从 ConnectTip 中截取,尤其在代码区块 main.cpp:2231-2243 中截取。 在相应代码区块对 ConnectBlock 有呼叫,在该代码中的有效性检查现金中储存新的 Coin。 (尤其是，请注意参考 main.cpp 的 UpdateCoins ()。
在代码区块末端中 Flush 有效性检查现金。 "parent view"也是 cash(pcoinsTip, block chain cash),因此代码把招来 parent 的 ViewCache: BatchWrite 来进行更新的 Coin 项目转换为自体现金。 在多形性执行中,以后区块链现金 flush 为查看数据库时的代码为 CoinsViewDB : 运行 BatchWrite,最后一行使用在 LevelDB。

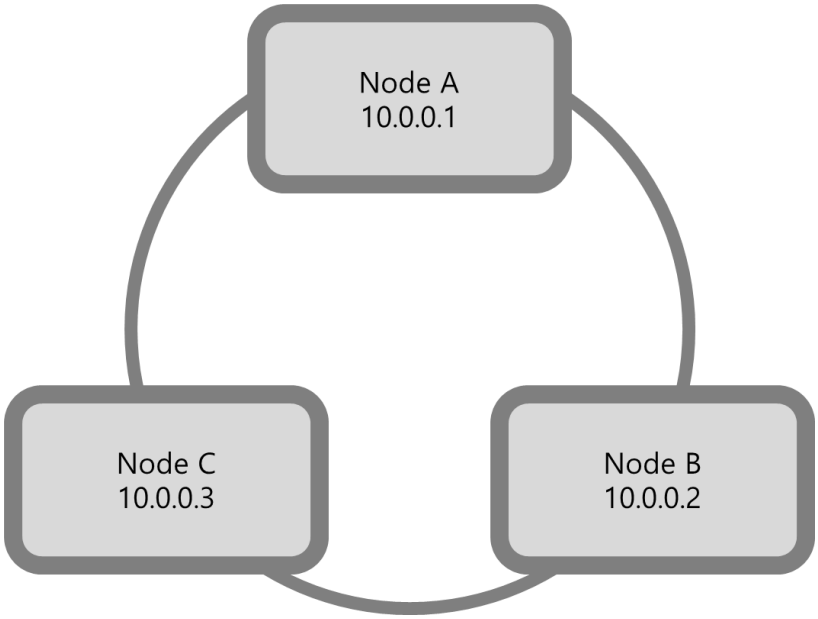
把区块链现金 flush 到数据库时，使用如下代码。

Flush 有效性检查现金的方法很简单。 因为代码只混合了存储器的两个现金(无人识别 cashing 代码外部)之间的项目。把区块链现金 flush 到数据库的方法是稍微复杂。 在最低水平上，Flush 区块链现金的机制(pcoinsTip)是与有效性检查现金相同。 Flush () Messard 在后台("base" point)呼叫 BatchWrite。 这种情况下, BatchWrite 从数据库,FlushStateToDisk (FSTD) - main.cpp 中呼叫 Flush ()。 FlushStateToDisk 可在所给予的几个不同地点呼叫。

III Node Algorithm

1. Seednode

Seed node,即种子节点,是同步化(Syncronizing),经常拥有已验证的分散账簿。 用户在使用 Cliet 模块时,可最先找出并连接节点。 但如果计算出其他节点的速度比先连接的 seednode 更快的话,则连接到先连接 seednode 但连接到维持其它节点快速的网络,首先带来分散账簿数据。 MX Seednode 技术将这些原理应用于所有种类的节点来计算整体节点的速度,并按顺序以速度优先为主维持连接节点。
这是在网络上速度快的节点持续添加的假设下，具有加速化速度的效果，同时添加速度一般或慢的节点时，在速度慢的节点上保留连接，连接到速度快的节点，能够快速招来分散账簿数据，这样进行同步化这是最大的特征。因这样的理由，速度慢的节点在网络上形成，也能在速度方面具备优秀性。关于构成各个节点的多数服务器间集群的方法请参考下图。



作为测试，假设利用 3 大节点构成一个集群时，在上面构成假设 A=Seednode, C= Masternode, B= 新生成的 Node。在这集群上有 3 个节点时，A 节点时起着告诉网络构成的作用，同时快速连接其它节点，MasterNode 是起着提高各节点间连接性，速度性，保安性的作用。 (参考 2. Masternode 内容).
各个节点的速度假设为 C=1000Mbps, A=100 Mbps, B=10Mbps 时，若在这里新生成 B 节点的话，先连接 NodeA。如果在这里 NodeA 不是 SeedNode 的话，会经过依据协议查看所有 IP 的过程。 因此能够减少在节点间的连接上所需要的时间，虽然先连接到 A 节点，但 A 节点已经与 C 节点连接，于是自动与 C 节点连接，而且速度是 C 节点的速度更高，于是比 A 和 B 连接的速度，减少消耗的时间，且带来 C 节点的分散账簿数据。通过以上过程，随着比网络的平均流量更快的节点添加， 速度就越快,而且存在很多节点时,】以 Spread 方式反复连接全体网络的节点。算法和连接概括过程请参考下面内容。

B 节点 **-(其它节点 IP Scan 及确认协议时间缩短)>** A 节点 **-(连接与速度快的节点)>** C 节点

- SeedNode Generate

```
from_future  im
port  print_fun
ction,
```

division

```
                from base64 import b32decode
                from binascii import a2b_hex
                import sys, os
                import re


                # ipv4 in ipv6 prefix
                pchIPv4 = bytearray([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0xff, 0xff])
                # tor-specific ipv6 prefix
                pchOnionCat = bytearray([0xFD,0x87,0xD8,0x7E,0xEB,0x43])


                def name_to_ipv6(addr):
                    if len(addr)>6 and addr.endswith('.onion'):
                        vchAddr = b32decode(addr[0:-6], True)
                        if len(vchAddr) != 16-len(pchOnionCat):
                            raise ValueError('Invalid onion %s' % s)
                        return pchOnionCat + vchAddr
                    elif '.' in addr: # IPv4
                        return pchIPv4 + bytearray((int(x) for x in addr.split('.')))
                    elif ':' in addr: # IPv6
                        sub = [[], []] # prefix, suffix
                        x = 0
                        addr = addr.split(':')
                        for i,comp in enumerate(addr):
                            if comp == '':
                                if i == 0 or i == (len(addr)-1): # skip empty component
                at beginning or end
                                    continue
                                x += 1 # :: skips to suffix
                                assert(x < 2)
                            else: # two bytes per component
                                val = int(comp, 16)
                                sub[x].append(val >> 8)
                                sub[x].append(val & 0xff)
                        nullbytes = 16 - len(sub[0]) - len(sub[1])
                        assert((x == 0 and nullbytes == 0) or (x == 1 and nullbytes > 0))
                        return bytearray(sub[0] + ([0] * nullbytes) + sub[1])
                    elif addr.startswith('0x'): # IPv4-in-little-endian
                        return pchIPv4 + bytearray(reversed(a2b_hex(addr[2:])))
                    else:
                        raise ValueError('Could not parse address %s' % addr)


                def parse_spec(s, defaultport):
```

```
    match = re.match('\[([0-9a-fA-F:]+)\](?::([0-9]+))?$', s)
    if match: # ipv6
        host = match.group(1)
        port = match.group(2)
    else:
        (host,_,port) = s.partition(':')


    if not port:
        port = defaultport
    else:
        port = int(port)


    host = name_to_ipv6(host)


    return (host,port)


def process_nodes(g, f, structname, defaultport):
    g.write('static SeedSpec6 %s[] = {\n' % structname)
    first = True
    for line in f:
        comment = line.find('#')
        if comment != -1:
            line = line[0:comment]
        line = line.strip()
        if not line:
            continue
        if not first:
            g.write(',\n')
        first = False


        (host,port) = parse_spec(line, defaultport)
        hoststr = ','.join(('0x%02x' % b) for b in host)
        g.write('    {{%s}, %i}' % (hoststr, port))
    g.write('\n};\n')


def main():
    if len(sys.argv)<2:
        print(('Usage: %s <path_to_nodes_txt>' % sys.argv[0]),
file=sys.stderr)
        exit(1)
```

```
    g = sys.stdout
    indir = sys.argv[1]
    g.write('#ifndef FACTOR_CHAINPARAMSSEEDS_H\n')
    g.write('#define FACTOR_CHAINPARAMSSEEDS_H\n')
    g.write('/**\n')
    g.write(' * List of fixed seed nodes for the factor network\n')
    g.write(' * AUTOGENERATED by share/seeds/generate-seeds.py\n')
    g.write(' *\n')
    g.write(' * Each line contains a 16-byte IPv6 address and a port.\n')
    g.write(' * IPv4 as well as onion addresses are wrapped inside a IPv6
address accordingly.\n')
    g.write(' */\n')
    with open(os.path.join(indir,'nodes_main.txt'),'r') as f:
        process_nodes(g, f, 'pnSeed6_main', 1993)
    g.write('\n')
    with open(os.path.join(indir,'nodes_test.txt'),'r') as f:
        process_nodes(g, f, 'pnSeed6_test', 11993)
    g.write('#endif // FACTOR_CHAINPARAMSSEEDS_H\n')

if__name__== '__main__':
    main()
```

- Seednode

```
NSEEDS=5
12


MAX_SEEDS_PER_ASN=2


MIN_BLOCKS = 200000


# These are hosts that have been observed to be behaving strangely (e.g.
# aggressively connecting to every node).
SUSPICIOUS_HOSTS = set([
    "10.0.0.1", "127.0.0.1”
])


import re
import sys
import dns.resolver
```

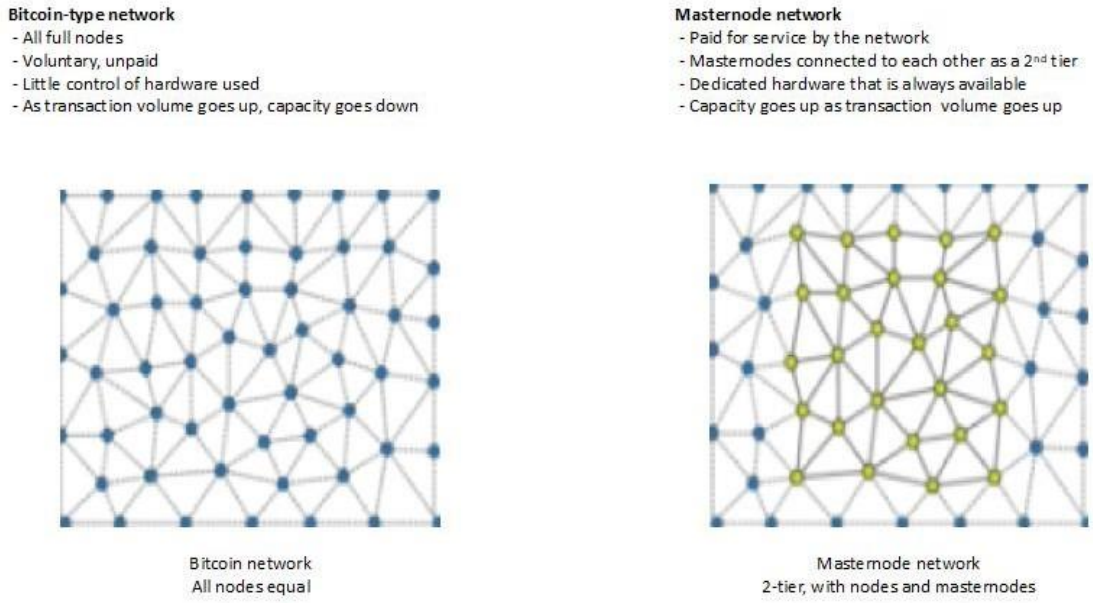
```
PATTERN_IPV4 =
re.compile(r"^((\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})):1993$")
PATTERN_AGENT =
re.compile(r"^(\/Satoshi:0.8.6\/|\/Satoshi:0.9.(2|3)\/|\/Core:0.1(0|1|2).\d{1,2
}.\d{1,2}\/)$")


def parseline(line):
    sline = line.split()
    if len(sline) < 11:
       return None
    # Match only IPv4
    m = PATTERN_IPV4.match(sline[0])
    if m is None:
        return None
    # Do IPv4 sanity check
    ip = 0
    for i in range(0,4):
        if int(m.group(i+2)) < 0 or int(m.group(i+2)) > 255:
            return None
        ip = ip + (int(m.group(i+2)) << (8*(3-i)))
    if ip == 0:
        return None
    # Skip bad results.
    if sline[1] == 0:
        return None
    # Extract uptime %.
    uptime30 = float(sline[7][:-1])
    # Extract Unix timestamp of last success.
    lastsuccess = int(sline[2])
    # Extract protocol version.
    version = int(sline[10])
    # Extract user agent.
    agent = sline[11][1:-1]
    # Extract service flags.
    service = int(sline[9], 16)
    # Extract blocks.
    blocks = int(sline[8])
    # Construct result.
    return {
        'ip': m.group(1),
        'ipnum': ip,
        'uptime': uptime30,
        'lastsuccess': lastsuccess,
        'version': version,
```

```
        'agent': agent,
        'service': service,
        'blocks': blocks,
    }


# Based on Greg Maxwell's seed_filter.py
def filterbyasn(ips, max_per_asn, max_total):
    result = []
    asn_count = {}
    for ip in ips:
        if len(result) == max_total:
            break
        try:
            asn = int([x.to_text() for x in
dns.resolver.query('.'.join(reversed(ip['ip'].split('.'))) +
'.origin.asn.cymru.com',
'TXT').response.answer][0].split('\"')[1].split(' ')[0])
            if asn not in asn_count:
                asn_count[asn] = 0
            if asn_count[asn] == max_per_asn:
                continue
            asn_count[asn] += 1
            result.append(ip)
        except:
            sys.stderr.write('ERR: Could not resolve ASN for "' + ip['ip'] +
'"\n')
    return result


def main():
    lines = sys.stdin.readlines()
    ips = [parseline(line) for line in lines]
    # Skip entries with valid IPv4 address.
    ips = [ip for ip in ips if ip is not None]
    # Skip entries from suspicious hosts.
    ips = [ip for ip in ips if ip['ip'] not in SUSPICIOUS_HOSTS]
    # Enforce minimal number of blocks.
    ips = [ip for ip in ips if ip['blocks'] >= MIN_BLOCKS]
    # Require service bit 1.
    ips = [ip for ip in ips if (ip['service'] & 1) == 1]
    # Require at least 50% 30-day uptime.
    ips = [ip for ip in ips if ip['uptime'] > 50]
    # Require a known and recent user agent.
    ips = [ip for ip in ips if PATTERN_AGENT.match(ip['agent'])]
    # Sort by availability (and use last success as tie breaker)
```

```
    ips.sort(key=lambda x: (x['uptime'], x['lastsuccess'], x['ip']),
reverse=True)
    # Look up ASNs and limit results, both per ASN and globally.
    ips = filterbyasn(ips, MAX_SEEDS_PER_ASN, NSEEDS)
    # Sort the results by IP address (for deterministic output).
    ips.sort(key=lambda x: (x['ipnum']))


    for ip in ips:
        print ip['ip']


if__name__== '__main__': main()
```

2. Masternode

Masternode 是具有着 POS(poorf of stake), POW(poorf of work)的特征，是能够给予全体网络的系统。 MX Masternode System 是与指定特定节点来运营，支付高费用来提高速度的原有方式不同，只在所设定的 Coin 个数中具有充足股份的话谁都能参与。要持续驱动 masternode (这一说法的用语为 Staking)使用补偿系统 Protocole 支付一定的补偿，能够在全体网络上提高速度性，保安性，连接性。



[12] 上图表示，能够查看在一般网络上的节点形式和 masternode 形式的不同点。
MX-Masternode 具有下列功能。

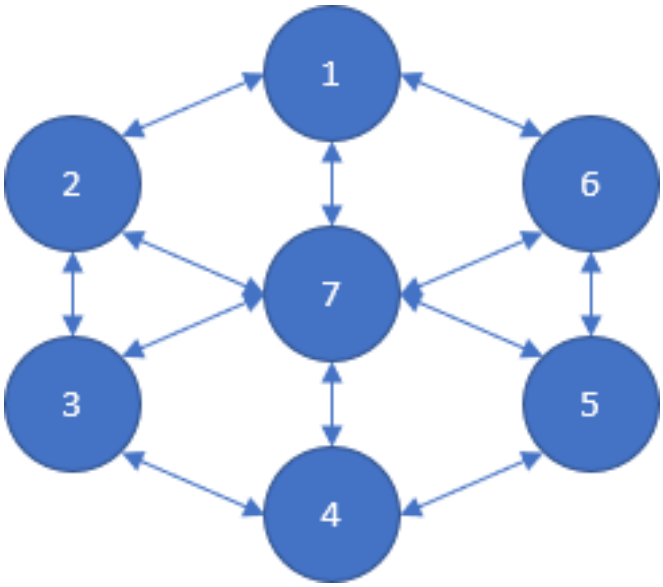
- 功能

①Transaction Volume 提高交易 volume
②Distributing network traffic (分散网络通信量)
②提高节点 Spread(喷射)连接方式连接性
③可使用 Private Send 功能
④ 贡献低电力高效率网络的功能
⑤有效性检查
⑥Transection Broadcasting
⑦PrivateSend (Darksend)
⑧InstantX

驱动 MX-Masternode 时不要求过多的能量和时间，高价装备等，反倒会提高使用者交易的效率性和匿名性。而且因持续运营的特征，不但记录在区块链内所发生的所有交易内容，监督黑客的活动，而且能够防御 DDOS 攻击等网络攻击。

[12] 图片 [来源: TTM Agency - Medium, Differences between Bitcoin-type and masternode networks]

- 构造



1=SeedNode（速度 100Mbps） 2=
Masternode（速度 10000Mbps）3
=Mining node（速度 1000Mbps）
4=Nomal Node（速度 10Mbps） 5
=Nomal Node（速度 10Mbps） 6=
Nomal Node（速度 10Mbps） 7
=新规节点(速度 10Mbps)

在上述假设下，7 号新规节点最先连接到 SeedNode，且连接到速度最快的 Masternode 上。在这时的连接过程，通过 1 号 Seednode 试图连接到全体节点，但最先连接的是 2 号的 Masternode。这时 Seednode 试图连接时的 FACTOR 的 Spread 连接方式，也就是说意味着使用 Spread 连接方式。Seednode 技术和 spread 喷射连接方式是网络的所有节点都在使用，于是 2 号 masternode 也是使用喷射方式，同时连接到速度最快的其它节点，于是能够与 3 号 mining node 连接。（连接是按照以上方法无限反复） 因此，实际上最先连接的是 1 号 seednode，但分散账簿数据是在 2 号 masternode 中带来信息，且对连接的所有节点修行上述说明的功能。 各功能的详细算法请参考下列代码。

```
map<uint256, int> mapSeenMasternodeScanningErrors;
// cache block hashes as we calculate them
std::map<int64_t, uint256> mapCacheBlockHashes;


//Get the last hash that matches the modulus given. Processed in reverse order
bool GetBlockHash(uint256& hash, int nBlockHeight)
{
    if (chainActive.Tip() == NULL) return false;


    if(nBlockHeight == 0)
```

```
        nBlockHeight = chainActive.Tip()->nHeight;


    if(mapCacheBlockHashes.count(nBlockHeight)){
        hash = mapCacheBlockHashes[nBlockHeight];
        return true;
    }


    const CBlockIndex *BlockLastSolved = chainActive.Tip();
    const CBlockIndex *BlockReading = chainActive.Tip();


    if (BlockLastSolved == NULL || BlockLastSolved->nHeight == 0
|| chainActive.Tip()->nHeight+1 < nBlockHeight) return false;


    int nBlocksAgo = 0;
    if(nBlockHeight > 0) nBlocksAgo = (chainActive.Tip()->nHeight+1)-nBlockHeight;
    assert(nBlocksAgo >= 0);


    int n = 0;
    for (unsigned int i = 1; BlockReading && BlockReading->nHeight > 0; i++) {
        if(n >= nBlocksAgo){
            hash = BlockReading->GetBlockHash();
            mapCacheBlockHashes[nBlockHeight] = hash;
            return true;
        }
        n++;


        if (BlockReading->pprev == NULL) { assert(BlockReading); break; }
        BlockReading = BlockReading->pprev;
    }


    return false;
}


CMasternode::CMasternode()
{
    LOCK(cs);
    vin = CTxIn();
    addr = CService();
```

```
    pubkey = CPubKey();
    pubkey2 = CPubKey();
    sig = std::vector<unsigned char>();
    activeState = MASTERNODE_ENABLED;
    sigTime = GetAdjustedTime();
    lastPing = CMasternodePing();
    cacheInputAge = 0;
    cacheInputAgeBlock = 0;
    unitTest = false;
    allowFreeTx = true;
    protocolVersion = PROTOCOL_VERSION;
    nLastDsq = 0;
    nScanningErrorCount = 0;
    nLastScanningErrorBlockHeight = 0;
    lastTimeChecked = 0;
}


CMasternode::CMasternode(const CMasternode& other)
{
    LOCK(cs);
    vin = other.vin;
    addr = other.addr;
    pubkey = other.pubkey;
    pubkey2 = other.pubkey2;
    sig = other.sig;
    activeState = other.activeState;
    sigTime = other.sigTime;
    lastPing = other.lastPing;
    cacheInputAge = other.cacheInputAge;
    cacheInputAgeBlock = other.cacheInputAgeBlock;
    unitTest = other.unitTest;
    allowFreeTx = other.allowFreeTx;
    protocolVersion = other.protocolVersion;
    nLastDsq = other.nLastDsq;
    nScanningErrorCount = other.nScanningErrorCount;
    nLastScanningErrorBlockHeight = other.nLastScanningErrorBlockHeight;
    lastTimeChecked = 0;
}


CMasternode::CMasternode(const CMasternodeBroadcast& mnb)
{
    LOCK(cs);
    vin = mnb.vin;
    addr = mnb.addr;
```

```
    pubkey = mnb.pubkey;
    pubkey2 = mnb.pubkey2;
    sig = mnb.sig;
    activeState = MASTERNODE_ENABLED;
    sigTime = mnb.sigTime;
    lastPing = mnb.lastPing;
    cacheInputAge = 0;
    cacheInputAgeBlock = 0;
    unitTest = false;
    allowFreeTx = true;
    protocolVersion = mnb.protocolVersion;
    nLastDsq = mnb.nLastDsq;
    nScanningErrorCount = 0;
    nLastScanningErrorBlockHeight = 0;
    lastTimeChecked = 0;
}


//
// When a new masternode broadcast is sent, update our information
//
bool CMasternode::UpdateFromNewBroadcast(CMasternodeBroadcast& mnb)
{
    if(mnb.sigTime > sigTime) {
        pubkey2 = mnb.pubkey2;
        sigTime = mnb.sigTime;
        sig = mnb.sig;
        protocolVersion = mnb.protocolVersion;
        addr = mnb.addr;
        lastTimeChecked = 0;
        int nDoS = 0;
        if(mnb.lastPing == CMasternodePing() || (mnb.lastPing != CMasternodePing()
&& mnb.lastPing.CheckAndUpdate(nDoS, false))) {
            lastPing = mnb.lastPing;
            mnodeman.mapSeenMasternodePing.insert(make_pair(lastPing.GetHash(),
lastPing));
        }
        return true;
    }
    return false;
}


//
// Deterministically calculate a given "score" for a Masternode depending on how
close it's hash is to
```

```
// the proof of work for that block. The further away they are the better, the
furthest will win the election
// and get paid this block
//
uint256 CMasternode::CalculateScore(int mod, int64_t nBlockHeight)
{
    if(chainActive.Tip() == NULL) return 0;


    uint256 hash = 0;
    uint256 aux = vin.prevout.hash + vin.prevout.n;


    if(!GetBlockHash(hash, nBlockHeight)) {
        LogPrintf("CalculateScore ERROR - nHeight %d - Returned 0\n",
nBlockHeight);
        return 0;
    }


    CHashWriter ss(SER_GETHASH, PROTOCOL_VERSION);
    ss << hash;
    uint256 hash2 = ss.GetHash();


    CHashWriter ss2(SER_GETHASH, PROTOCOL_VERSION);
    ss2 << hash;
    ss2 << aux;
    uint256 hash3 = ss2.GetHash();


    uint256 r = (hash3 > hash2 ? hash3 - hash2 : hash2 - hash3);


    return r;
}


void CMasternode::Check(bool forceCheck)
{
    if(ShutdownRequested()) return;


    if(!forceCheck && (GetTime() - lastTimeChecked <
MASTERNODE_CHECK_SECONDS)) return;
    lastTimeChecked = GetTime();
```

```
//once spent, stop doing the checks
if(activeState == MASTERNODE_VIN_SPENT) return;



if(!IsPingedWithin(MASTERNODE_REMOVAL_SECONDS)){
    activeState = MASTERNODE_REMOVE;
    return;
}


if(!IsPingedWithin(MASTERNODE_EXPIRATION_SECONDS)){
    activeState = MASTERNODE_EXPIRED;
    return;
}


if(!unitTest){
    CValidationState state;
    CMutableTransaction tx = CMutableTransaction();
    CTxOut vout = CTxOut(999.99*COIN, darkSendPool.collateralPubKey);
    tx.vin.push_back(vin);
    tx.vout.push_back(vout);


    {
        TRY_LOCK(cs_main, lockMain);
        if(!lockMain) return;


        if(!AcceptableInputs(mempool, state, CTransaction(tx), false, NULL)){
            activeState = MASTERNODE_VIN_SPENT;
            return;


        }
    }
}


activeState = MASTERNODE_ENABLED; // OK
```

```
}

int64_t CMasternode::SecondsSincePayment() {
    CScript pubkeyScript;
    pubkeyScript = GetScriptForDestination(pubkey.GetID());

    int64_t sec = (GetAdjustedTime() - GetLastPaid());
    int64_t month = 60*60*24*30;
    if(sec < month) return sec; //if it's less than 30 days, give seconds

    CHashWriter ss(SER_GETHASH, PROTOCOL_VERSION);
    ss << vin;
    ss << sigTime;
    uint256 hash =  ss.GetHash();

    // return some deterministic value for unknown/unpaid but force it to be
more than 30 days old
    return month + hash.GetCompact(false);
}

int64_t CMasternode::GetLastPaid() {
    CBlockIndex* pindexPrev = chainActive.Tip();
    if(pindexPrev == NULL) return false;

    CScript mnpayee;
    mnpayee = GetScriptForDestination(pubkey.GetID());

    CHashWriter ss(SER_GETHASH, PROTOCOL_VERSION);
    ss << vin;
    ss << sigTime;
    uint256 hash =  ss.GetHash();

    // use a deterministic offset to break a tie -- 2.5 minutes
    int64_t nOffset = hash.GetCompact(false) % 150;

    if (chainActive.Tip() == NULL) return false;
```

```
    const CBlockIndex *BlockReading = chainActive.Tip();


    int nMnCount = mnodeman.CountEnabled()*1.25;
    int n = 0;
    for (unsigned int i = 1; BlockReading && BlockReading->nHeight > 0; i++) {
        if(n >= nMnCount){
            return 0;
        }
        n++;


        if(masternodePayments.mapMasternodeBlocks.count(BlockReading->nHeight)){
            /*
                Search for this payee, with at least 2 votes. This will aid in
consensus allowing the network
                to converge on the same payees quickly, then keep the same
schedule.
            */
            if(masternodePayments.mapMasternodeBlocks[BlockReading-
>nHeight].HasPayeeWithVotes(mnpayee, 2)){
                return BlockReading->nTime + nOffset;
            }
        }


        if (BlockReading->pprev == NULL) { assert(BlockReading); break; }
        BlockReading = BlockReading->pprev;
    }


    return 0;
}


CMasternodeBroadcast::CMasternodeBroadcast()
{
    vin = CTxIn();
    addr = CService();
    pubkey = CPubKey();
    pubkey2 = CPubKey();
    sig = std::vector<unsigned char>();
    activeState = MASTERNODE_ENABLED;
    sigTime = GetAdjustedTime();
    lastPing = CMasternodePing();
```

```
    cacheInputAge = 0;
    cacheInputAgeBlock = 0;
    unitTest = false;
    allowFreeTx = true;
    protocolVersion = PROTOCOL_VERSION;
    nLastDsq = 0;
    nScanningErrorCount = 0;
    nLastScanningErrorBlockHeight = 0;
}


CMasternodeBroadcast::CMasternodeBroadcast(CService newAddr, CTxIn newVin,
CPubKey newPubkey, CPubKey newPubkey2, int protocolVersionIn)
{
    vin = newVin;
    addr = newAddr;
    pubkey = newPubkey;
    pubkey2 = newPubkey2;
    sig = std::vector<unsigned char>();
    activeState = MASTERNODE_ENABLED;
    sigTime = GetAdjustedTime();
    lastPing = CMasternodePing();
    cacheInputAge = 0;
    cacheInputAgeBlock = 0;
    unitTest = false;
    allowFreeTx = true;
    protocolVersion = protocolVersionIn;
    nLastDsq = 0;
    nScanningErrorCount = 0;
    nLastScanningErrorBlockHeight = 0;
}


CMasternodeBroadcast::CMasternodeBroadcast(const CMasternode& mn)
{
    vin = mn.vin;
    addr = mn.addr;
    pubkey = mn.pubkey;
    pubkey2 = mn.pubkey2;
    sig = mn.sig;
    activeState = mn.activeState;
    sigTime = mn.sigTime;
    lastPing = mn.lastPing;
    cacheInputAge = mn.cacheInputAge;
    cacheInputAgeBlock = mn.cacheInputAgeBlock;
    unitTest = mn.unitTest;
```

```
    allowFreeTx = mn.allowFreeTx;
    protocolVersion = mn.protocolVersion;
    nLastDsq = mn.nLastDsq;
    nScanningErrorCount = mn.nScanningErrorCount;
    nLastScanningErrorBlockHeight = mn.nLastScanningErrorBlockHeight;
}


bool CMasternodeBroadcast::CheckAndUpdate(int& nDos)
{
    nDos = 0;


    // make sure signature isn't in the future (past is OK)
    if (sigTime > GetAdjustedTime() + 60 * 60) {
        LogPrintf("mnb - Signature rejected, too far into the future %s\n",
vin.ToString());
        nDos = 1;
        return false;
    }


    if(protocolVersion < masternodePayments.GetMinMasternodePaymentsProto()) {
        LogPrintf("mnb - ignoring outdated Masternode %s protocol version %d\n",
vin.ToString(), protocolVersion);
        return false;
    }


    CScript pubkeyScript;
    pubkeyScript = GetScriptForDestination(pubkey.GetID());


    if(pubkeyScript.size() != 25) {
        LogPrintf("mnb - pubkey the wrong size\n");
        nDos = 100;
        return false;
    }


    CScript pubkeyScript2;
    pubkeyScript2 = GetScriptForDestination(pubkey2.GetID());


    if(pubkeyScript2.size() != 25) {
        LogPrintf("mnb - pubkey2 the wrong size\n");
```

```
        nDos = 100;
        return false;
    }


    if(!vin.scriptSig.empty()) {
        LogPrintf("mnb - Ignore Not Empty ScriptSig %s\n",vin.ToString());
        return false;
    }


    // incorrect ping or its sigTime
    if(lastPing == CMasternodePing() || !lastPing.CheckAndUpdate(nDos, false,
true))
        return false;


    std::string strMessage;
    std::string errorMessage = "";


    if(protocolVersion < 70201) {
        std::string vchPubKey(pubkey.begin(), pubkey.end());
        std::string vchPubKey2(pubkey2.begin(), pubkey2.end());
        strMessage = addr.ToString(false) +
boost::lexical_cast<std::string>(sigTime) +
                        vchPubKey + vchPubKey2 +
boost::lexical_cast<std::string>(protocolVersion);


        LogPrint("masternode", "mnb - sanitized strMessage: %s, pubkey address:
%s, sig: %s\n",
            SanitizeString(strMessage), CBitcoinAddress(pubkey.GetID()).ToString(),
            EncodeBase64(&sig[0], sig.size()));


        if(!darkSendSigner.VerifyMessage(pubkey, sig, strMessage, errorMessage)){
            if (addr.ToString() != addr.ToString(false))
            {
                // maybe it's wrong format, try again with the old one
                strMessage = addr.ToString() +
boost::lexical_cast<std::string>(sigTime) +
                                vchPubKey + vchPubKey2 +
boost::lexical_cast<std::string>(protocolVersion);
```

```
                LogPrint("masternode", "mnb - sanitized strMessage: %s, pubkey
address: %s, sig: %s\n",
                    SanitizeString(strMessage),
CBitcoinAddress(pubkey.GetID()).ToString(),
                    EncodeBase64(&sig[0], sig.size()));


                if(!darkSendSigner.VerifyMessage(pubkey, sig, strMessage,
errorMessage)){
                    // didn't work either
                    LogPrintf("mnb - Got bad Masternode address signature,
sanitized error: %s\n", SanitizeString(errorMessage));
                    // there is a bug in old MN signatures, ignore such MN but do
not ban the peer we got this from
                    return false;
                }
            } else {
                // nope, sig is actually wrong
                LogPrintf("mnb - Got bad Masternode address signature, sanitized
error: %s\n", SanitizeString(errorMessage));
                // there is a bug in old MN signatures, ignore such MN but do not
ban the peer we got this from
                return false;
            }
        }
    } else {
        strMessage = addr.ToString(false) +
boost::lexical_cast<std::string>(sigTime) +
                        pubkey.GetID().ToString() + pubkey2.GetID().ToString() +
                        boost::lexical_cast<std::string>(protocolVersion);


        LogPrint("masternode", "mnb - strMessage: %s, pubkey address:
%s, sig: %s\n",
            strMessage, CBitcoinAddress(pubkey.GetID()).ToString(),
EncodeBase64(&sig[0], sig.size()));


        if(!darkSendSigner.VerifyMessage(pubkey, sig, strMessage, errorMessage)){
            LogPrintf("mnb - Got bad Masternode address signature, error: %s\n",
errorMessage);
            nDos = 100;
            return false;
        }
    }
```

```
    if(Params().NetworkID() == CBaseChainParams::MAIN) {
        if(addr.GetPort() != 1993) return false;
    } else if(addr.GetPort() == 1993) return false;


    //search existing Masternode list, this is where we update existing
Masternodes with new mnb broadcasts
    CMasternode* pmn = mnodeman.Find(vin);


    // no such masternode, nothing to update
    if(pmn == NULL) return true;


    // this broadcast is older or equal than the one that we already have -
it's bad and should never happen
    // unless someone is doing something fishy
    // (mapSeenMasternodeBroadcast in CMasternodeMan::ProcessMessage should filter
legit duplicates)
    if(pmn->sigTime >= sigTime) {
        LogPrintf("CMasternodeBroadcast::CheckAndUpdate - Bad sigTime %d for
Masternode %20s %105s (existing broadcast is at %d)\n",
                      sigTime, addr.ToString(), vin.ToString(), pmn->sigTime);
        return false;
    }


    // masternode is not enabled yet/already, nothing to update
    if(!pmn->IsEnabled()) return true;


    // mn.pubkey = pubkey, IsVinAssociatedWithPubkey is validated once below,
    //   after that they just need to match
    if(pmn->pubkey == pubkey && !pmn-
>IsBroadcastedWithin(MASTERNODE_MIN_MNB_SECONDS)) {
        //take the newest entry
        LogPrintf("mnb - Got updated entry for %s\n", addr.ToString());
        if(pmn->UpdateFromNewBroadcast((*this))){
            pmn->Check();
            if(pmn->IsEnabled()) Relay();
        }
        masternodeSync.AddedMasternodeList(GetHash());
    }
```

```
    return true;
}

bool CMasternodeBroadcast::CheckInputsAndAdd(int& nDoS)
{
    // we are a masternode with the same vin (i.e. already activated) and this mnb
is ours (matches our Masternode privkey)
    // so nothing to do here for us
    if(fMasterNode && vin.prevout == activeMasternode.vin.prevout && pubkey2 ==
activeMasternode.pubKeyMasternode)
        return true;


    // incorrect ping or its sigTime
    if(lastPing == CMasternodePing() || !lastPing.CheckAndUpdate(nDoS, false,
true))
        return false;


    // search existing Masternode list
    CMasternode* pmn = mnodeman.Find(vin);


    if(pmn != NULL) {
        // nothing to do here if we already know about this masternode and it's
enabled
        if(pmn->IsEnabled()) return true;
        // if it's not enabled, remove old MN first and continue
        else mnodeman.Remove(pmn->vin);
    }


    CValidationState state;
    CMutableTransaction tx = CMutableTransaction();
    CTxOut vout = CTxOut(999.99*COIN, darkSendPool.collateralPubKey);
    tx.vin.push_back(vin);
    tx.vout.push_back(vout);


    {
        TRY_LOCK(cs_main, lockMain);
        if(!lockMain) {
            // not mnb fault, let it to be checked again later
            mnodeman.mapSeenMasternodeBroadcast.erase(GetHash());
            masternodeSync.mapSeenSyncMNB.erase(GetHash());
```

```
            return false;
        }

        if(!AcceptableInputs(mempool, state, CTransaction(tx), false, NULL)) {
            //set nDos
            state.IsInvalid(nDoS);
            return false;
        }
    }

    LogPrint("masternode", "mnb - Accepted Masternode entry\n");

    if(GetInputAge(vin) < MASTERNODE_MIN_CONFIRMATIONS){
        LogPrintf("mnb - Input must have at least %d confirmations\n",
MASTERNODE_MIN_CONFIRMATIONS);
        // maybe we miss few blocks, let this mnb to be checked again later
        mnodeman.mapSeenMasternodeBroadcast.erase(GetHash());
        masternodeSync.mapSeenSyncMNB.erase(GetHash());
        return false;
    }

    // verify that sig time is legit in past
    // should be at least not earlier than block when 1000 FACTOR tx got
MASTERNODE_MIN_CONFIRMATIONS
    uint256 hashBlock = 0;
    CTransaction tx2;
    GetTransaction(vin.prevout.hash, tx2, hashBlock, true);
    BlockMap::iterator mi = mapBlockIndex.find(hashBlock);
    if (mi != mapBlockIndex.end() && (*mi).second)
    {
        CBlockIndex* pMNIndex = (*mi).second; // block for 1000 FACTOR tx -> 1
confirmation
        CBlockIndex* pConfIndex = chainActive[pMNIndex->nHeight +
MASTERNODE_MIN_CONFIRMATIONS - 1]; // block where tx
got MASTERNODE_MIN_CONFIRMATIONS
        if(pConfIndex->GetBlockTime() > sigTime)
        {
            LogPrintf("mnb - Bad sigTime %d for Masternode %20s %105s (%i conf
block is at %d)\n",
                      sigTime, addr.ToString(), vin.ToString(),
MASTERNODE_MIN_CONFIRMATIONS, pConfIndex->GetBlockTime());
            return false;
```

```
        }
    }

    LogPrintf("mnb - Got NEW Masternode entry - %s - %s - %s - %lli
\n", GetHash().ToString(), addr.ToString(), vin.ToString(), sigTime);
    CMasternode mn(*this);
    mnodeman.Add(mn);

    // if it matches our Masternode privkey, then we've been remotely activated
    if(pubkey2 == activeMasternode.pubKeyMasternode && protocolVersion ==
PROTOCOL_VERSION){
        activeMasternode.EnableHotColdMasterNode(vin, addr);
    }

    bool isLocal = addr.IsRFC1918() || addr.IsLocal();
    if(Params().NetworkID() == CBaseChainParams::REGTEST) isLocal = false;

    if(!isLocal) Relay();

    return true;
}

void CMasternodeBroadcast::Relay()
{
    CInv inv(MSG_MASTERNODE_ANNOUNCE, GetHash());
    RelayInv(inv);
}

bool CMasternodeBroadcast::Sign(CKey& keyCollateralAddress)
{
    std::string errorMessage;

    std::string vchPubKey(pubkey.begin(), pubkey.end());
    std::string vchPubKey2(pubkey2.begin(), pubkey2.end());

    sigTime = GetAdjustedTime();
```

```
    std::string strMessage = addr.ToString(false) +
boost::lexical_cast<std::string>(sigTime) + vchPubKey + vchPubKey2
+ boost::lexical_cast<std::string>(protocolVersion);


    if(!darkSendSigner.SignMessage(strMessage, errorMessage, sig,
keyCollateralAddress)) {
        LogPrintf("CMasternodeBroadcast::Sign() - Error: %s\n", errorMessage);
        return false;
    }


    return true;
}


bool CMasternodeBroadcast::VerifySignature()
{
    std::string errorMessage;


    std::string vchPubKey(pubkey.begin(), pubkey.end());
    std::string vchPubKey2(pubkey2.begin(), pubkey2.end());


    std::string strMessage = addr.ToString() +
boost::lexical_cast<std::string>(sigTime) + vchPubKey + vchPubKey2
+ boost::lexical_cast<std::string>(protocolVersion);


    if(!darkSendSigner.VerifyMessage(pubkey, sig, strMessage, errorMessage)) {
        LogPrintf("CMasternodeBroadcast::VerifySignature() - Error: %s\n",
errorMessage);
        return false;
    }


    return true;
}


CMasternodePing::CMasternodePing()
{
    vin = CTxIn();
    blockHash = uint256(0);
```

```
    sigTime = 0;
    vchSig = std::vector<unsigned char>();
}


CMasternodePing::CMasternodePing(CTxIn& newVin)
{
    vin = newVin;
    blockHash = chainActive[chainActive.Height() - 12]->GetBlockHash();
    sigTime = GetAdjustedTime();
    vchSig = std::vector<unsigned char>();
}




bool CMasternodePing::Sign(CKey& keyMasternode, CPubKey& pubKeyMasternode)
{
    std::string errorMessage;
    std::string strMasterNodeSignMessage;


    sigTime = GetAdjustedTime();
    std::string strMessage = vin.ToString() + blockHash.ToString() +
boost::lexical_cast<std::string>(sigTime);


    if(!darkSendSigner.SignMessage(strMessage, errorMessage,
vchSig, keyMasternode)) {
        LogPrintf("CMasternodePing::Sign() - Error: %s\n", errorMessage);
        return false;
    }


    if(!darkSendSigner.VerifyMessage(pubKeyMasternode, vchSig,
strMessage, errorMessage)) {
        LogPrintf("CMasternodePing::Sign() - Error: %s\n", errorMessage);
        return false;
    }


    return true;
}
bool CMasternodePing::VerifySignature(CPubKey& pubKeyMasternode, int &nDos) {
    std::string strMessage = vin.ToString() + blockHash.ToString() +
boost::lexical_cast<std::string>(sigTime);
```

```
    std::string errorMessage = "";


    if(!darkSendSigner.VerifyMessage(pubKeyMasternode, vchSig,
strMessage, errorMessage))
    {
        LogPrintf("CMasternodePing::VerifySignature - Got bad Masternode ping
signature %s Error: %s\n", vin.ToString(), errorMessage);
        nDos = 33;
        return false;
    }
    return true;
}


bool CMasternodePing::CheckAndUpdate(int& nDos, bool fRequireEnabled,
bool fCheckSigTimeOnly)
{
    if (sigTime > GetAdjustedTime() + 60 * 60) {
        LogPrintf("CMasternodePing::CheckAndUpdate - Signature rejected, too far
into the future %s\n", vin.ToString());
        nDos = 1;
        return false;
    }
    if (sigTime <= GetAdjustedTime() - 60 * 60) {
        LogPrintf("CMasternodePing::CheckAndUpdate - Signature rejected, too far
into the past %s - %d %d \n", vin.ToString(), sigTime, GetAdjustedTime());
        nDos = 1;
        return false;
    }
    if(fCheckSigTimeOnly) {
        CMasternode* pmn = mnodeman.Find(vin);
        if(pmn) return VerifySignature(pmn->pubkey2, nDos);
        return true;
    }
    LogPrint("masternode", "CMasternodePing::CheckAndUpdate - New Ping - %s - %s -
 %lli\n", GetHash().ToString(), blockHash.ToString(), sigTime);
    // see if we have this Masternode
    CMasternode* pmn = mnodeman.Find(vin);
    if(pmn != NULL && pmn->protocolVersion >=
masternodePayments.GetMinMasternodePaymentsProto())
    {
        if (fRequireEnabled && !pmn->IsEnabled()) return false;
        // LogPrintf("mnping - Found corresponding mn for vin: %s\n",
vin.ToString());
        // update only if there is no known ping for this masternode or
```

```
        // last ping was more then MASTERNODE_MIN_MNP_SECONDS-60 ago comparing to
this one
        if(!pmn->IsPingedWithin(MASTERNODE_MIN_MNP_SECONDS - 60, sigTime))
        {
            if(!VerifySignature(pmn->pubkey2, nDos))
                return false;
            BlockMap::iterator mi = mapBlockIndex.find(blockHash);
            if (mi != mapBlockIndex.end() && (*mi).second)
            {
                if((*mi).second->nHeight < chainActive.Height() - 24)
                {
                    LogPrintf("CMasternodePing::CheckAndUpdate - Masternode %s
block hash %s is too old\n", vin.ToString(), blockHash.ToString());
                    // Do nothing here (no Masternode update, no mnping relay)
                    // Let this node to be visible but fail to accept mnping
                    return false;
                }
            } else {
                if (fDebug) LogPrintf("CMasternodePing::CheckAndUpdate -
Masternode %s block hash %s is unknown\n", vin.ToString(), blockHash.ToString());
                // maybe we stuck so we shouldn't ban this node, just fail to
accept it
                // TODO: or should we also request this block?
                return false;
            }
            pmn->lastPing = *this;
            //mnodeman.mapSeenMasternodeBroadcast.lastPing is probably outdated, so
we'll update it
            CMasternodeBroadcast mnb(*pmn);
            uint256 hash = mnb.GetHash();
            if(mnodeman.mapSeenMasternodeBroadcast.count(hash)) {
                mnodeman.mapSeenMasternodeBroadcast[hash].lastPing = *this;
            }
            pmn->Check(true);
            if(!pmn->IsEnabled()) return false;
            LogPrint("masternode", "CMasternodePing::CheckAndUpdate - Masternode
ping accepted, vin: %s\n", vin.ToString());
            Relay();
            return true;
        }
        LogPrint("masternode", "CMasternodePing::CheckAndUpdate - Masternode ping
arrived too early, vin: %s\n", vin.ToString());
        //nDos = 1; //disable, this is happening frequently and causing banned
peers
        return false;
    }
```

```
    LogPrint("masternode", "CMasternodePing::CheckAndUpdate - Couldn't find
compatible Masternode entry, vin: %s\n", vin.ToString());
    return false;
}
void CMasternodePing::Relay()
{
    CInv inv(MSG_MASTERNODE_PING, GetHash());
    RelayInv(inv);
}
```

3. Mining node

Mining(采掘)是指代入任意Nonce值，形成区块散列结果值，寻找生成的结果值比提示的Target更小的区块散列的过程。因散列函数的特征,不可知道要代入哪种Nonce值才能寻找比提示的Target更小的区块散列信息。也就是说，
为了寻找正确的结果值，把Nonce的值从0增加1个,直到出现比提示的Target更小的结果值为止要无限反复修行工作。这样在1秒能够修行几次解开数学问题的过程等的数值信息叫做散列力量，越是散列力量大的minor，能够解开更多的问题，能够解开更多问题的采掘者具有更高的可能性寻找新的区块。散列力量越大的采掘者，意味着修行更多的工作，若修行更多的工作时，有更大的可能性收到更多的补偿。因此在定义PoW(Proof of work)时,它表现为对做更多工作的采掘者给予更多补偿的方式。 为了连接区块数据散列，在区块中加入以区块Head数据为解码函数计算的区块解码,通过此方法来验证散列连接性,确认区块链数据中间没有发生伪造。 这种工作证明算法的必要性是使网络的所有节点无法同时形成区块。 只有通过工作证明才能生成区块，为此会消耗巨大的能量。 然后利用调节难度(Difiiculty)算法,保证每10分钟生成1~2个区块。 求区块散列的方法和在整个过程中贡献交易的方法请参考下列内容。

- Source代码

求区块散列时需要的值: ver, prev_block, mrkl_root, time, bits, nonce

```
import hashlib
import struct



little_endian = lambda value: struct.pack('<L', value).hex()
reverse_order_pair = lambda value: ''.join([value[i - 2:i] for i in range(len(value), 0, -2)])

# 第 125552 个区块信息
# 以下列信息出来 00000000000000001e8d6829a8a21adc5d38d0a473b144b6765798e61f98bd1d 结果时成功
block_info = {
    'hash': '00000000000000001e8d6829a8a21adc5d38d0a473b144b6765798e61f98bd1d',
    'ver': 1,
    'prev_block': '00000000000008a3a41b85b8b29ad444def299fee21793cd8b9e567eab02cd81',
    'mrkl_root':  '2b12fcf1b09288fcaff797d71e950e71ae42b91e8bdb2304758dfcffc2b620e3',  'tim
    e': 1305998791,
    'bits': 440711666,
    'fee': 1000000,
    'nonce': 2504433986,
    }

convert_block_info = {}

# 1. 把版本，时间，bits, nonce 信息变形为 incoding 形态  convert_
block_info['ver'] = little_endian(block_info['ver'])  convert_bl
ock_info['time'] = little_endian(block_info['time'])  convert_bl
ock_info['bits'] = little_endian(block_info['bits'])  convert_bl
ock_info['nonce'] = little_endian(block_info['nonce'])
```

```
# 2. 把之前区块散列，merckleroute 结果值变形为相反顺序  convert_block_info['prev
_block'] = reverse_order_pair(block_info['prev_block']) convert_block_info['mrk
l_root'] = reverse_order_pair(block_info['mrkl_root'])

# 3 合算区块 header 信息（需要按照顺序）
header_hex = convert_block_info['ver'] + convert_block_info['prev_block'] +  convert_bloc
k_info['mrkl_root'] + convert_block_info['time'] + convert_block_info['bits']
+ convert_block_info['nonce']

# 4. 变形为 binary 形态
header_bin = bytes.fromhex(header_hex)

# 5. 把变换为 SHA 256 的结果再次变换为 SHA 256 (进行 2 次)
hash = hashlib.sha256(hashlib.sha256(header_bin).digest()).digest()

# 6. 颠倒 hash 结果后变更为 16 真数
result_header_hex = hash[::-1].hex()


# result : 00000000000000001e8d6829a8a21adc5d38d0a473b144b6765798e61f98bd1d
# 成功
```

实行上面代码就能确认形成了现在区块的散列值。采掘是把 Nonce 从 0 开始增加 1 个与 Target 值比较后直到出现更小或同一值之前反复实行。像上面一样成功采掘后，若 Nonce 为 2504433986 时，散列函数的结果值为 00000000000000001e8d6829a8a21adc5d38d0a473b144b6765798e61f98bd1d。而且按照下列可以整理计算过程。

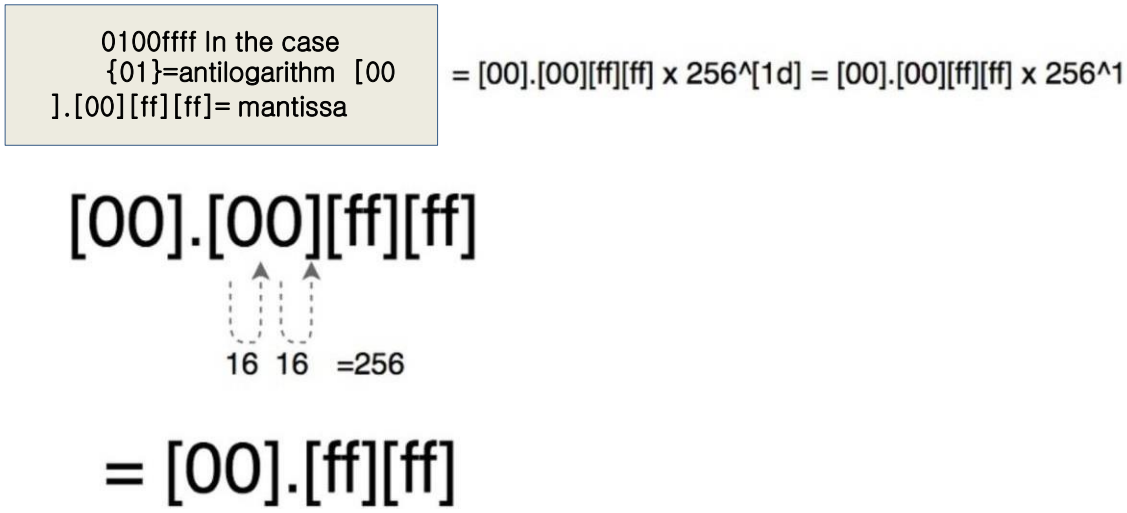
1. ver, time, bits, nonce 是变形为 Little Endian 形态
2. prev_hash, mrkl_root 是把结果值变更为相反顺序（是 16 真数，2 个一起变更为相反顺序）
3. 合算 header 信息（连接文字列值，这时不可变更顺序）
4. 把合算的 header 信息变更为 binary 形态
5. 把变更为 SHA256 形态的结果再次变形为 SHA256
6. 在 5 号出来的结果颠倒后变更为 16 真数

- 工作证明值导出方法

为了用 Nonce 成功证明工作，与 bits 提示的数字相同或更小的话就成功。但是在 bits 利用指数和可数所有有点复杂。 例如，在数学表现大数或小数时，表现为 0.1234*10^5 或 0.1234*10^(-5) 。一开始举例的数字是 12,340 最后数字是 0.000001234 。这时把 0.1234 称为可数，5 或 -5 称为 指数。在上面用. Little Endian 计算的 bits 是 ffff001d，但需要变换为 Big Endian 计算为 1d00ffff。用 Endian 计算的数是 16 真数，所以要把握成 1d, 00, ff, ff。在这里 bits 是需要把 1d 看成指数，00ffff 看成可数。

也就是说，1d 是真数， 00ffff 是可数。 (可数是在前面省略小数点 0.，是指 0.00ffff 。在这里 bits 提示的数字计算法是 00.00ffff*256^1d 。万一指数为 01 时，往 256^1 小数点向右侧移动一个，就成为 00.ffff。

(* 在这里需要注意的是 16 真数，所以每 16 是 1 栏。

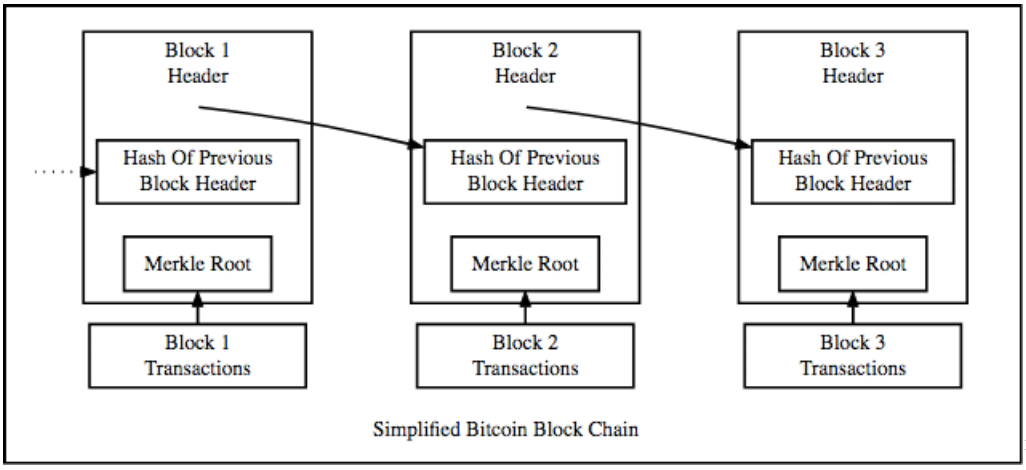


[13]

回到上面 bits，是[00].[00][ff][ff]*256^1d [00].[00][ff][ff]*256^29
(把 1d 放入到 https://ko.calcuworld.com/%EC%88%98%ED%95%99/16%EC%A7%84%EB%B2%95-%EA%B3%84%EC%82%B0%EA%B8%B0/ 变换为 10 真数就出来 29 的值) 把小数点向右侧移动 29 个栏。再说在[ff][ff] 后 26 次写作 [00]就可以。在这里结束的话，形成为 28 个，所以在导出的 28 个值前面出来 添加 4 个[00]的[00][00][00][00][ff][ff][00][00]„[00] (32bytes) - (64 字)。结果 bits 提示的数字为 000000ff0000...00 (32bytes) - (64 字)。变更 Nonce，在上面出来的散列结果值要比这个值更小或一样才可以成功证明工作。

工作难度
上面说明过工作证明是找出区块散列低于特定数字时的nonce值。 前面说过Block Hash是32bytes的数字，拿它来解释因太复杂，所以假设是没有符号的1bytes。 那么散列函数的结果值在0~255之间。 如果说区块散列必须小于128的话,那么小于128的区块散列结果值出现的概率为128/256,所以是50%的概率。如果区块散列结果值要小于64的话,则64/256,是25%的概率，因此可获得Nonce。 在这里举的例子,128,64这个值就是bits。

- 概括说明和全体过程



[14]

[13] 图片 [来源: 火熊, Tistory]
[14] 图片 [来源: Genier, brunch]

在区块链上区块的构造形成为 Version, Pre Block Hash, Merkle Hash, Timestamp, Bits, Nonce，在这里寻找Nonce 值证明交易的有效性。找出Nonce值的人是以证明交易有效性的代价收到硬币。全体过程如下。

第一阶段：用户在 Wallet 应用程序的交易上签名,试图将特定密码或硬币传送给他人。

第二阶段：交易由钱包应用软件中 broad casting,目前等待相关区块链的采掘者选择。 不被选择的话就留在‘未确认交易 pool’。这 pool 是待处理的网络未经确认的交易集合。 这种未经确认的交易通常不是从一个巨大的 pool 中收集的,而大部分是从一个小细分的 local pool 中收集的。

第三阶段：网络上的采掘者在这种 pool 中选择交易，把这构成为"区块"。 区块基本上除了部分 meta 数据外， 也是交易(此时尚未确认的交易)的集合。 所有采掘都构成自体交易区块。 多名采掘者可以选择在自己的区块中包含的同一交易。

第三阶段 举例):两名采掘者 A,B 可把交易包含在区块中。 每个区块链都有最大区块大小。 在适当区块添加交易之前， 采掘需确认交易是否可以按照区块链记录来执行。 若在发信人的钱包余额上具有充分的资金时，视为交易有效，于是可添加到区块中。

第四阶段：:选择交易,添加到区块,开采生成交易区块。 如果想添加区块链 A 交易区块时(要想让其他所有采掘和节点登录交易的话),首先需要在区块上签名(也叫作业证明)。 该签名是通过解决各交易区块固有的非常复杂的数学问题而形成的。 每个区块都有不同的数学题,所有的采掘都会针对他们所制造的区块中固有的其他问题进行工作。 所有这些问题都很难完全解决。 为了解决这一数学问题,需要使用许多计算能力(因此使用很多电力)。 这叫做采掘,过程请参照前面所说明的源代码。

第五阶段：首先,找到适合该区块签字的采矿者,将该区块和签字给其他所有采矿者 broad casting。

第六阶段：其他采掘者现在带来了 broad casting 区块的数据文字列,为了确认包含输出散列的签名与实际是否一致，于是 hashing 后确认签名的合法性。如果它有效的话,其他的开采会确认它的可行性,并同意此区块能够添加到区块链。 (经 MX 协议算法决定)， 因此使用"一致算法"一词。 签字是执行工作的"证明"(消费的计算能力)。 现在可以把区块添加到区块链中,分散到网络的其他所有节点。 只要区块内的交易与相应时刻的现在钱包余额(交易历史)完全一致的话，其他节点就承认区块，储存到交易数据中。

第 7 阶段：区块被添加到链条上后,上面增加的其他所有区块都会经过确认计算过程。 对这个区块,比如说，我的交易包含在 Block 502 中,如果区块链条长度为 507 区块的话，那么意味着在交易中有 5 次确认(507-502)。 每次增加其他区块时,区块链都会对包括交易及区块块在内的全部交易明细重新达成协议,因此叫做确认。 在那个时刻,可以说贵方的交易被区块链子确认了 5 次。 交易验证越来越多，从黑客伪造分散账簿的可能性就越低。如果新的区块被添加到区块链中,则所有采掘者必须要形成新的交易区块,在第三阶段重新开始。 不能持续采掘（但在这个报道不那么重要,)采掘因这些理由，可解决自己工作的区块问题。

即以上结论表明,采矿者(Miner)可对交易的签名,验证,传输和区块生成作出贡献。
MXblockchain的Ming node在CPU,GPU方式的开采上,两者都可以在开采中使用,支持GPU的多重堆叠加速方式,在依靠现有的CPU进行开采的方式上可提高电力消费量的效率性来降低电力消费量。 因此能够有效增加对冲力,这是能够有效加速交易的签名,验证,传输,区块生成,比现有区块链保持相同电量对比更高的对冲力。
详细算法代码请参考下列内容。

```
class
COrphan
        {
        public:
            const CTransaction* ptx;
            set<uint256> setDependsOn;
            CFeeRate feeRate;
            double dPriority;


            COrphan(const CTransaction* ptxIn) : ptx(ptxIn), feeRate(0), dPriority(0)
            {
            }
        };


        uint64_t nLastBlockTx = 0;
        uint64_t nLastBlockSize = 0;


        // We want to sort transactions by priority and fee rate, so:
        typedef boost::tuple<double, CFeeRate, const CTransaction*> TxPriority;
        class TxPriorityCompare
        {
            bool byFee;


        public:
            TxPriorityCompare(bool _byFee) : byFee(_byFee) { }


            bool operator()(const TxPriority& a, const TxPriority& b)
            {
                if (byFee)
                {
                    if (a.get<1>() == b.get<1>())
                        return a.get<0>() < b.get<0>();
                    return a.get<1>() < b.get<1>();
                }
```

```
        else
        {
            if (a.get<0>() == b.get<0>())
                return a.get<1>() < b.get<1>();
            return a.get<0>() < b.get<0>();
        }
    }
};


void UpdateTime(CBlockHeader* pblock, const CBlockIndex* pindexPrev)
{
    pblock->nTime = std::max(pindexPrev->GetMedianTimePast()+1,
GetAdjustedTime());


    // Updating time can change work required on testnet:
    if (Params().AllowMinDifficultyBlocks())
        pblock->nBits = GetNextWorkRequired(pindexPrev, pblock);
}


CBlockTemplate* CreateNewBlock(const CScript& scriptPubKeyIn)
{
    // Create new block
    auto_ptr<CBlockTemplate> pblocktemplate(new CBlockTemplate());
    if(!pblocktemplate.get())
        return NULL;
    CBlock *pblock = &pblocktemplate->block; // pointer for convenience


    // -regtest only: allow overriding block.nVersion with
    // -blockversion=N to test forking scenarios
    if (Params().MineBlocksOnDemand())
        pblock->nVersion = GetArg("-blockversion", pblock->nVersion);


    // Create coinbase tx
    CMutableTransaction txNew;
    txNew.vin.resize(1);
    txNew.vin[0].prevout.SetNull();
    txNew.vout.resize(1);
    txNew.vout[0].scriptPubKey = scriptPubKeyIn;


    // Largest block you're willing to create:
```

```
    unsigned int nBlockMaxSize = GetArg("-blockmaxsize", DEFAULT_BLOCK_MAX_SIZE);
    // Limit to betweeen 1K and MAX_BLOCK_SIZE-1K for sanity:
    nBlockMaxSize = std::max((unsigned int)1000, std::min((unsigned
int)(MAX_BLOCK_SIZE-1000), nBlockMaxSize));


    // How much of the block should be dedicated to high-priority transactions,
    // included regardless of the fees they pay
    unsigned int nBlockPrioritySize = GetArg("-blockprioritysize",
DEFAULT_BLOCK_PRIORITY_SIZE);
    nBlockPrioritySize = std::min(nBlockMaxSize, nBlockPrioritySize);


    // Minimum block size you want to create; block will be filled with
free transactions
    // until there are no more or the block reaches this size:
    unsigned int nBlockMinSize = GetArg("-blockminsize", DEFAULT_BLOCK_MIN_SIZE);
    nBlockMinSize = std::min(nBlockMaxSize, nBlockMinSize);


    // Collect memory pool transactions into the block
    CAmount nFees = 0;


    {
        LOCK2(cs_main, mempool.cs);


        CBlockIndex* pindexPrev = chainActive.Tip();
        const int nHeight = pindexPrev->nHeight + 1;
        CCoinsViewCache view(pcoinsTip);


        // Add our coinbase tx as first transaction
        pblock->vtx.push_back(txNew);
        pblocktemplate->vTxFees.push_back(-1); // updated at end
        pblocktemplate->vTxSigOps.push_back(-1); // updated at end


        // Priority order to process transactions
        list<COrphan> vOrphan; // list memory doesn't move
        map<uint256, vector<COrphan*> > mapDependers;
        bool fPrintPriority = GetBoolArg("-printpriority", false);


        // This vector will be sorted into a priority queue:
```

```
        vector<TxPriority> vecPriority;
        vecPriority.reserve(mempool.mapTx.size());
        for (map<uint256, CTxMemPoolEntry>::iterator mi = mempool.mapTx.begin();
             mi != mempool.mapTx.end(); ++mi)
        {
            const CTransaction& tx = mi->second.GetTx();
            if (tx.IsCoinBase() || !IsFinalTx(tx, nHeight))
                continue;


            COrphan* porphan = NULL;
            double dPriority = 0;
            CAmount nTotalIn = 0;
            bool fMissingInputs = false;
            BOOST_FOREACH(const CTxIn& txin, tx.vin)
            {
                // Read prev transaction
                if (!view.HaveCoins(txin.prevout.hash))
                {
                    // This should never happen; all transactions in the memory
                    // pool should connect to either transactions in the chain
                    // or other transactions in the memory pool.
                    if (!mempool.mapTx.count(txin.prevout.hash))
                    {
                        LogPrintf("ERROR: mempool transaction missing input\n");
                        if (fDebug) assert("mempool transaction missing input" ==
0);

                        fMissingInputs = true;
                        if (porphan)
                            vOrphan.pop_back();
                        break;
                    }


                    // Has to wait for dependencies
                    if (!porphan)
                    {
                        // Use list for automatic deletion
                        vOrphan.push_back(COrphan(&tx));
                        porphan = &vOrphan.back();
                    }
                    mapDependers[txin.prevout.hash].push_back(porphan);
                    porphan->setDependsOn.insert(txin.prevout.hash);
                    nTotalIn +=
mempool.mapTx[txin.prevout.hash].GetTx().vout[txin.prevout.n].nValue;
                    continue;
```

```
                }
                const CCoins* coins = view.AccessCoins(txin.prevout.hash);
                assert(coins);

                CAmount nValueIn = coins->vout[txin.prevout.n].nValue;
                nTotalIn += nValueIn;

                int nConf = nHeight - coins->nHeight;

                dPriority += (double)nValueIn * nConf;
            }
            if (fMissingInputs) continue;

            // Priority is sum(valuein * age) / modified_txsize
            unsigned int nTxSize = ::GetSerializeSize(tx, SER_NETWORK,
PROTOCOL_VERSION);
            dPriority = tx.ComputePriority(dPriority, nTxSize);

            uint256 hash = tx.GetHash();
            mempool.ApplyDeltas(hash, dPriority, nTotalIn);

            CFeeRate feeRate(nTotalIn-tx.GetValueOut(), nTxSize);

            if (porphan)
            {
                porphan->dPriority = dPriority;
                porphan->feeRate = feeRate;
            }
            else
                vecPriority.push_back(TxPriority(dPriority, feeRate, &mi-
>second.GetTx()));
        }

        // Collect transactions into block
        uint64_t nBlockSize = 1000;
        uint64_t nBlockTx = 0;
        int nBlockSigOps = 100;
        bool fSortedByFee = (nBlockPrioritySize <= 0);
```

```
        TxPriorityCompare comparer(fSortedByFee);
        std::make_heap(vecPriority.begin(), vecPriority.end(), comparer);


        while (!vecPriority.empty())
        {
            // Take highest priority transaction off the priority queue:
            double dPriority = vecPriority.front().get<0>();
            CFeeRate feeRate = vecPriority.front().get<1>();
            const CTransaction& tx = *(vecPriority.front().get<2>());


            std::pop_heap(vecPriority.begin(), vecPriority.end(), comparer);
            vecPriority.pop_back();


            // Size limits
            unsigned int nTxSize = ::GetSerializeSize(tx, SER_NETWORK,
PROTOCOL_VERSION);
            if (nBlockSize + nTxSize >= nBlockMaxSize)
                continue;


            // Legacy limits on sigOps:
            unsigned int nTxSigOps = GetLegacySigOpCount(tx);
            if (nBlockSigOps + nTxSigOps >= MAX_BLOCK_SIGOPS)
                continue;


            // Skip free transactions if we're past the minimum block size:
            const uint256& hash = tx.GetHash();
            double dPriorityDelta = 0;
            CAmount nFeeDelta = 0;
            mempool.ApplyDeltas(hash, dPriorityDelta, nFeeDelta);
            if (fSortedByFee && (dPriorityDelta <= 0) && (nFeeDelta <= 0) &&
(feeRate < ::minRelayTxFee) && (nBlockSize + nTxSize >= nBlockMinSize))
                continue;


            // Prioritise by fee once past the priority size or we run out
of high-priority
            // transactions:
            if (!fSortedByFee &&
                ((nBlockSize + nTxSize >= nBlockPrioritySize)
```

```
|| !AllowFree(dPriority)))
            {
                fSortedByFee = true;
                comparer = TxPriorityCompare(fSortedByFee);
                std::make_heap(vecPriority.begin(), vecPriority.end(), comparer);
            }


            if (!view.HaveInputs(tx))
                continue;


            CAmount nTxFees = view.GetValueIn(tx)-tx.GetValueOut();


            nTxSigOps += GetP2SHSigOpCount(tx, view);
            if (nBlockSigOps + nTxSigOps >= MAX_BLOCK_SIGOPS)
                continue;


            // Note that flags: we don't want to set mempool/IsStandard()
            // policy here, but we still have to ensure that the block we
            // create only contains transactions that are valid in new blocks.
            CValidationState state;
            if (!CheckInputs(tx, state, view, true,
MANDATORY_SCRIPT_VERIFY_FLAGS, true))
                continue;


            CTxUndo txundo;
            UpdateCoins(tx, state, view, txundo, nHeight);


            // Added
            pblock->vtx.push_back(tx);
            pblocktemplate->vTxFees.push_back(nTxFees);
            pblocktemplate->vTxSigOps.push_back(nTxSigOps);
            nBlockSize += nTxSize;
            ++nBlockTx;
            nBlockSigOps += nTxSigOps;
            nFees += nTxFees;


            if (fPrintPriority)
            {
                LogPrintf("priority %.1f fee %s txid %s\n",
```

```
                    dPriority, feeRate.ToString(), tx.GetHash().ToString());
            }


            // Add transactions that depend on this one to the priority queue
            if (mapDependers.count(hash))
            {
                BOOST_FOREACH(COrphan* porphan, mapDependers[hash])
                {
                    if (!porphan->setDependsOn.empty())
                    {
                        porphan->setDependsOn.erase(hash);
                        if (porphan->setDependsOn.empty())
                        {
                            vecPriority.push_back(TxPriority(porphan->dPriority,
porphan->feeRate, porphan->ptx));
                            std::push_heap(vecPriority.begin(),
vecPriority.end(), comparer);
                        }
                    }
                }
            }
        }


        // Masternode and general budget payments
        FillBlockPayee(txNew, nFees);


        // Make payee
            if(txNew.vout.size() > 1){
            pblock->payee = txNew.vout[1].scriptPubKey;
        }


        nLastBlockTx = nBlockTx;
        nLastBlockSize = nBlockSize;
        LogPrintf("CreateNewBlock(): total size %u\n", nBlockSize);


        // Compute final coinbase transaction.
        txNew.vin[0].scriptSig = CScript() << nHeight << OP_0;
        pblock->vtx[0] = txNew;
        pblocktemplate->vTxFees[0] = -nFees;
```

```
        // Fill in header
        pblock->hashPrevBlock  = pindexPrev->GetBlockHash();
        UpdateTime(pblock, pindexPrev);
        pblock->nBits          = GetNextWorkRequired(pindexPrev, pblock);
        pblock->nNonce         = 0;
        pblocktemplate->vTxSigOps[0] = GetLegacySigOpCount(pblock->vtx[0]);


        CValidationState state;
        if (!TestBlockValidity(state, *pblock, pindexPrev, false, false))
            throw std::runtime_error("CreateNewBlock() : TestBlockValidity
failed");
    }


    return pblocktemplate.release();
}


void IncrementExtraNonce(CBlock* pblock, CBlockIndex* pindexPrev, unsigned
int& nExtraNonce)
{
    // Update nExtraNonce
    static uint256 hashPrevBlock;
    if (hashPrevBlock != pblock->hashPrevBlock)
    {
        nExtraNonce = 0;
        hashPrevBlock = pblock->hashPrevBlock;
    }
    ++nExtraNonce;
    unsigned int nHeight = pindexPrev->nHeight+1; // Height first in coinbase
required for block.version=2
    CMutableTransaction txCoinbase(pblock->vtx[0]);
    txCoinbase.vin[0].scriptSig = (CScript() << nHeight <<
CScriptNum(nExtraNonce)) + COINBASE_FLAGS;
    assert(txCoinbase.vin[0].scriptSig.size() <= 100);


    pblock->vtx[0] = txCoinbase;
    pblock->hashMerkleRoot = pblock->BuildMerkleTree();
}


#ifdef ENABLE_WALLET
//////////////////////////////////////////////////////////////////////////////
//
```

```
// Internal miner
//
double dHashesPerSec = 0.0;
int64_t nHPSTimerStart = 0;


// ***TODO*** ScanHash is not yet used in Factor
//
// ScanHash scans nonces looking for a hash with at least some zero bits.
// The nonce is usually preserved between calls, but periodically or if the
// nonce is 0xffff0000 or above, the block is rebuilt and nNonce starts over at
// zero.
//
//bool static ScanHash(const CBlockHeader *pblock, uint32_t& nNonce, uint256
*phash)
//{
//    // Write the first 76 bytes of the block header to a double-SHA256 state.
//    CHash256 hasher;
//    CDataStream ss(SER_NETWORK, PROTOCOL_VERSION);
//    ss << *pblock;
//    assert(ss.size() == 80);
//    hasher.Write((unsigned char*)&ss[0], 76);


//    while (true) {
//        nNonce++;


//        // Write the last 4 bytes of the block header (the nonce) to a copy of
//        // the double-SHA256 state, and compute the result.
//        CHash256(hasher).Write((unsigned char*)&nNonce, 4).Finalize((unsigned
char*)phash);


//        // Return the nonce if the hash has at least some zero bits,
//        // caller will check if it has enough to reach the target
//        if (((uint16_t*)phash)[15] == 0)
//            return true;


//        // If nothing found after trying for a while, return -1
//        if ((nNonce & 0xffff) == 0)
//            return false;
//        if ((nNonce & 0xfff) == 0)
//            boost::this_thread::interruption_point();
//    }
```

```
//}


CBlockTemplate* CreateNewBlockWithKey(CReserveKey& reservekey)
{
    CPubKey pubkey;
    if (!reservekey.GetReservedKey(pubkey))
        return NULL;


    CScript scriptPubKey = CScript() << ToByteVector(pubkey) << OP_CHECKSIG;
    return CreateNewBlock(scriptPubKey);
}


bool ProcessBlockFound(CBlock* pblock, CWallet& wallet, CReserveKey& reservekey)
{
    LogPrintf("%s\n", pblock->ToString());
    LogPrintf("generated %s\n", FormatMoney(pblock->vtx[0].vout[0].nValue));


    // Found a solution
    {
        LOCK(cs_main);
        if (pblock->hashPrevBlock != chainActive.Tip()->GetBlockHash())
            return error("FactorMiner : generated block is stale");
    }


    // Remove key from key pool
    reservekey.KeepKey();


    // Track how many getdata requests this block gets
    {
        LOCK(wallet.cs_wallet);
        wallet.mapRequestCount[pblock->GetHash()] = 0;
    }


    // Process this block the same as if we had received it from another node
    CValidationState state;
    if (!ProcessNewBlock(state, NULL, pblock))
        return error("FactorMiner : ProcessNewBlock, block not accepted");
```

```
    return true;
}


// ***TODO*** that part changed in bitcoin, we are using a mix with old one
here for now
void static BitcoinMiner(CWallet *pwallet)
{
    LogPrintf("FactorMiner started\n");
    SetThreadPriority(THREAD_PRIORITY_LOWEST);
    RenameThread("factor-miner");


    // Each thread has its own key and counter
    CReserveKey reservekey(pwallet);
    unsigned int nExtraNonce = 0;


    try {
        while (true) {
            if (Params().MiningRequiresPeers()) {
                // Busy-wait for the network to come online so we don't waste
time mining
                // on an obsolete chain. In regtest mode we expect to fly solo.
                do {
                    bool fvNodesEmpty;
                    {
                        LOCK(cs_vNodes);
                        fvNodesEmpty = vNodes.empty();
                    }
                    if (!fvNodesEmpty && !IsInitialBlockDownload())
                        break;
                    MilliSleep(1000);
                } while (true);
            }


            //
            // Create new block
            //
            unsigned int nTransactionsUpdatedLast =
mempool.GetTransactionsUpdated();
            CBlockIndex* pindexPrev = chainActive.Tip();
            if(!pindexPrev) break;
```

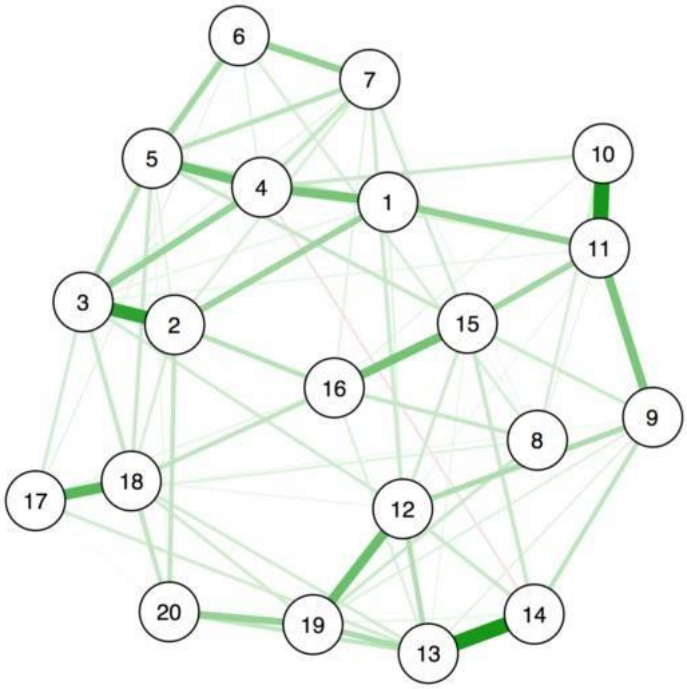
```
            auto_ptr<CBlockTemplate>
pblocktemplate(CreateNewBlockWithKey(reservekey));
            if (!pblocktemplate.get())
            {
                LogPrintf("Error in FactorMiner: Keypool ran out, please call
keypoolrefill before restarting the mining thread\n");
                return;
            }
            CBlock *pblock = &pblocktemplate->block;
            IncrementExtraNonce(pblock, pindexPrev, nExtraNonce);
            LogPrintf("Running FactorMiner with %u transactions in block (%u
bytes)\n", pblock->vtx.size(),
                ::GetSerializeSize(*pblock, SER_NETWORK, PROTOCOL_VERSION));
            //
            // Search
            //
            int64_t nStart = GetTime();
            uint256 hashTarget = uint256().SetCompact(pblock->nBits);
            while (true)
            {
                unsigned int nHashesDone = 0;


                uint256 hash;
                while (true)
                {
                    hash = pblock->GetHash();
                    if (hash <= hashTarget)
                    {
                        // Found a solution
                        SetThreadPriority(THREAD_PRIORITY_NORMAL);
                        LogPrintf("BitcoinMiner:\n");
                        LogPrintf("proof-of-work found  \n  hash: %s
\ntarget: %s\n", hash.GetHex(), hashTarget.GetHex());
                        ProcessBlockFound(pblock, *pwallet, reservekey);
                        SetThreadPriority(THREAD_PRIORITY_LOWEST);


                        // In regression test mode, stop mining after a block is
found. This
                        // allows developers to controllably generate a block on
demand.
                        if (Params().MineBlocksOnDemand())
                            throw boost::thread_interrupted();
                        break;
                    }
```

```
                    pblock->nNonce += 1;
                    nHashesDone += 1;
                    if ((pblock->nNonce & 0xFF) == 0)
                        break;
                }
                // Meter hashes/sec
                static int64_t nHashCounter;
                if (nHPSTimerStart == 0)
                {
                    nHPSTimerStart = GetTimeMillis();
                    nHashCounter = 0;
                }
                else
                    nHashCounter += nHashesDone;
                if (GetTimeMillis() - nHPSTimerStart > 4000)
                {
                    static CCriticalSection cs;
                    {
                        LOCK(cs);
                        if (GetTimeMillis() - nHPSTimerStart > 4000)
                        {
                            dHashesPerSec = 1000.0 * nHashCounter /
(GetTimeMillis() - nHPSTimerStart);
                            nHPSTimerStart = GetTimeMillis();
                            nHashCounter = 0;
                            static int64_t nLogTime;
                            if (GetTime() - nLogTime > 30 * 60)
                            {
                                nLogTime = GetTime();
                                LogPrintf("hashmeter %6.0f khash/s\n",
dHashesPerSec/1000.0);
                            }
                        }
                    }
                }
                // Check for stop or if block needs to be rebuilt
                boost::this_thread::interruption_point();
                // Regtest mode doesn't require peers
                if (vNodes.empty() && Params().MiningRequiresPeers())
                    break;
                if (pblock->nNonce >= 0xffff0000)
                    break;
                if (mempool.GetTransactionsUpdated() != nTransactionsUpdatedLast
&& GetTime() - nStart > 60)
                    break;
                if (pindexPrev != chainActive.Tip())
```

```
                    break;
                // Update nTime every few seconds
                UpdateTime(pblock, pindexPrev);
                if (Params().AllowMinDifficultyBlocks())
                {
                    // Changing pblock->nTime can change work required on
testnet:
                    hashTarget.SetCompact(pblock->nBits);
                }
            }
        }
    }
    catch (boost::thread_interrupted)
    {
        LogPrintf("FactorMiner terminated\n");
        throw;
    }
    catch (const std::runtime_error &e)
    {
        LogPrintf("BitcoinMiner runtime error: %s\n", e.what());
        return;
    }
}
void GenerateBitcoins(bool fGenerate, CWallet* pwallet, int nThreads)
{
    static boost::thread_group* minerThreads = NULL;
    if (nThreads < 0) {
        // In regtest threads defaults to 1
        if (Params().DefaultMinerThreads())
            nThreads = Params().DefaultMinerThreads();
        else
            nThreads = boost::thread::hardware_concurrency();
    }
    if (minerThreads != NULL)
    {
        minerThreads->interrupt_all();
        delete minerThreads;
        minerThreads = NULL;
    }
    if (nThreads == 0 || !fGenerate)
        return;
    minerThreads = new boost::thread_group();
    for (int i = 0; i < nThreads; i++)
        minerThreads->create_thread(boost::bind(&BitcoinMiner, pwallet));
}
#endif // ENABLE_WALLET
```

4. Nomal Node

一般节点(Nomal node)是网络的基本要素—连接到地区网络的电脑和其所属装备统称为一个节点。 例如,在本地网络A中有20台电脑,2个集散中心和2个路由器,还有路由器。 将这些细节全部结合在一起,属于网络A的装备叫做一个节点。 物理性网络节点是安装在网络上的能动性电子设备,可以在通信渠道上制造,接收和传输信息,在数据通信中物理性网络节点有数据通信装置(DCE)或是数据线路终端装置(DTE)不管距离通信网(LAN)还是广域通信网(WAN),至少数据链接层的所有LAN或WAN 节点都有网络地址并运行。即,拥有上述网络环境的一般用户,想要用于传输和接收,需经过同步化(Syncronizing)过程,并因使用 Seednode技术与Spread喷射连接方式, 于是形成网络速度快的normal node也可以在全体网络作出贡献。假设前面说明的1. SeedNode 2.Masternode 3.MiningNode 4.Nomal Node 整体节点构建的话, 网络将形成下图一样。



节点 1 号是 SeedNode ,节点 2 号是 Nomal Node,若具有最快的速度的话,会在节点 2Normal node 带来分散账簿信息。 这意味着使用钱包或 Client 的使用者也能在全体网络速度作出贡献。Nomal Node 的算法代码请参考下列内容。

```
#if
defined(HAVE_CONFIG_H)
                        #include "config/factor-config.h"
                        #endif


                        #include "net.h"


                        #include "addrman.h"
                        #include "chainparams.h"
                        #include "clientversion.h"
```

```
#include "primitives/transaction.h"
#include "ui_interface.h"
#include "darksend.h"
#include "wallet.h"


#ifdef WIN32
#include <string.h>
#else
#include <fcntl.h>
#endif


#ifdef USE_UPNP
#include <miniupnpc/miniupnpc.h>
#include <miniupnpc/miniwget.h>
#include <miniupnpc/upnpcommands.h>
#include <miniupnpc/upnperrors.h>
#endif


#include <boost/filesystem.hpp>
#include <boost/thread.hpp>


// Dump addresses to peers.dat every 15 minutes (900s)
#define DUMP_ADDRESSES_INTERVAL 900


#if !defined(HAVE_MSG_NOSIGNAL) && !defined(MSG_NOSIGNAL)
#define MSG_NOSIGNAL 0
#endif


// Fix for ancient MinGW versions, that don't have defined
these in ws2tcpip.h.
// Todo: Can be removed when our pull-tester is upgraded to a
modern MinGW version.
#ifdef WIN32
#ifndef PROTECTION_LEVEL_UNRESTRICTED
#define PROTECTION_LEVEL_UNRESTRICTED 10
#endif
#ifndef IPV6_PROTECTION_LEVEL
#define IPV6_PROTECTION_LEVEL 23
#endif
#endif
```

```
using namespace boost;
using namespace std;


namespace {
    const int MAX_OUTBOUND_CONNECTIONS = 8;


    struct ListenSocket {
        SOCKET socket;
        bool whitelisted;


        ListenSocket(SOCKET socket, bool whitelisted) :
socket(socket), whitelisted(whitelisted) {}
    };
}


//
// Global state variables
//
bool fDiscover = true;
bool fListen = true;
uint64_t nLocalServices = NODE_NETWORK;
CCriticalSection cs_mapLocalHost;
map<CNetAddr, LocalServiceInfo> mapLocalHost;
static bool vfReachable[NET_MAX] = {};
static bool vfLimited[NET_MAX] = {};
static CNode* pnodeLocalHost = NULL;
uint64_t nLocalHostNonce = 0;
static std::vector<ListenSocket> vhListenSocket;
CAddrMan addrman;
int nMaxConnections = 125;
bool fAddressesInitialized = false;


vector<CNode*> vNodes;
CCriticalSection cs_vNodes;
map<CInv, CDataStream> mapRelay;
deque<pair<int64_t, CInv> > vRelayExpiration;
CCriticalSection cs_mapRelay;
limitedmap<CInv, int64_t> mapAlreadyAskedFor(MAX_INV_SZ);
```

```
static deque<string> vOneShots;
CCriticalSection cs_vOneShots;


set<CNetAddr> setservAddNodeAddresses;
CCriticalSection cs_setservAddNodeAddresses;


vector<std::string> vAddedNodes;
CCriticalSection cs_vAddedNodes;


NodeId nLastNodeId = 0;
CCriticalSection cs_nLastNodeId;


static CSemaphore *semOutbound = NULL;
boost::condition_variable messageHandlerCondition;


// Signals for message handling
static CNodeSignals g_signals;
CNodeSignals& GetNodeSignals() { return g_signals; }


void AddOneShot(string strDest)
{
    LOCK(cs_vOneShots);
    vOneShots.push_back(strDest);
}


unsigned short GetListenPort()
{
    return (unsigned short)(GetArg("-port",
Params().GetDefaultPort()));
}


// find 'best' local address for a particular peer
bool GetLocal(CService& addr, const CNetAddr *paddrPeer)
{
    if (!fListen)
        return false;
```

```
    int nBestScore = -1;
    int nBestReachability = -1;
    {
        LOCK(cs_mapLocalHost);
        for (map<CNetAddr, LocalServiceInfo>::iterator it =
mapLocalHost.begin(); it != mapLocalHost.end(); it++)
        {
            int nScore = (*it).second.nScore;
            int nReachability =
(*it).first.GetReachabilityFrom(paddrPeer);
            if (nReachability > nBestReachability ||
(nReachability == nBestReachability && nScore > nBestScore))
            {
                addr = CService((*it).first, (*it).second.nPort);
                nBestReachability = nReachability;
                nBestScore = nScore;
            }
        }
    }
    return nBestScore >= 0;
}


// get best local address for a particular peer as a CAddress
// Otherwise, return the unroutable 0.0.0.0 but filled in with
// the normal parameters, since the IP may be changed to a useful
// one by discovery.
CAddress GetLocalAddress(const CNetAddr *paddrPeer)
{
    CAddress ret(CService("0.0.0.0",GetListenPort()),0);
    CService addr;
    if (GetLocal(addr, paddrPeer))
    {
        ret = CAddress(addr);
    }
    ret.nServices = nLocalServices;
    ret.nTime = GetAdjustedTime();
    return ret;
}


bool RecvLine(SOCKET hSocket, string& strLine)
{
    strLine = "";
    while (true)
```

```
    {
        char c;
        int nBytes = recv(hSocket, &c, 1, 0);
        if (nBytes > 0)
        {
            if (c == '\n')
                continue;
            if (c == '\r')
                return true;
            strLine += c;
            if (strLine.size() >= 9000)
                return true;
        }
        else if (nBytes <= 0)
        {
            boost::this_thread::interruption_point();
            if (nBytes < 0)
            {
                int nErr = WSAGetLastError();
                if (nErr == WSAEMSGSIZE)
                    continue;
                if (nErr == WSAEWOULDBLOCK || nErr == WSAEINTR ||
nErr == WSAEINPROGRESS)
                {
                    MilliSleep(10);
                    continue;
                }
            }
            if (!strLine.empty())
                return true;
            if (nBytes == 0)
            {
                // socket closed
                LogPrint("net", "socket closed\n");
                return false;
            }
            else
            {
                // socket error
                int nErr = WSAGetLastError();
                LogPrint("net", "recv failed: %s\n",
NetworkErrorString(nErr));
                return false;
            }
        }
    }
```

```
}


int GetnScore(const CService& addr)
{
    LOCK(cs_mapLocalHost);
    if (mapLocalHost.count(addr) == LOCAL_NONE)
        return 0;
    return mapLocalHost[addr].nScore;
}


// Is our peer's addrLocal potentially useful as an external
IP source?
bool IsPeerAddrLocalGood(CNode *pnode)
{
    return fDiscover && pnode->addr.IsRoutable() && pnode-
>addrLocal.IsRoutable() &&
           !IsLimited(pnode->addrLocal.GetNetwork());
}


// pushes our own address to a peer
void AdvertizeLocal(CNode *pnode)
{
    if (fListen && pnode->fSuccessfullyConnected)
    {
        CAddress addrLocal = GetLocalAddress(&pnode->addr);
        // If discovery is enabled, sometimes give our peer the
address it
        // tells us that it sees us as in case it has a better
idea of our
        // address than we do.
        if (IsPeerAddrLocalGood(pnode) && (!addrLocal.IsRoutable()
||
             GetRand((GetnScore(addrLocal) > LOCAL_MANUAL) ? 8:2)
== 0))
        {
            addrLocal.SetIP(pnode->addrLocal);
        }
        if (addrLocal.IsRoutable())
        {
            pnode->PushAddress(addrLocal);
        }
    }
}
```

```
void SetReachable(enum Network net, bool fFlag)
{
    LOCK(cs_mapLocalHost);
    vfReachable[net] = fFlag;
    if (net == NET_IPV6 && fFlag)
        vfReachable[NET_IPV4] = true;
}


// learn a new local address
bool AddLocal(const CService& addr, int nScore)
{
    if (!addr.IsRoutable())
        return false;


    if (!fDiscover && nScore < LOCAL_MANUAL)
        return false;


    if (IsLimited(addr))
        return false;


    LogPrintf("AddLocal(%s,%i)\n", addr.ToString(), nScore);


    {
        LOCK(cs_mapLocalHost);
        bool fAlready = mapLocalHost.count(addr) > 0;
        LocalServiceInfo &info = mapLocalHost[addr];
        if (!fAlready || nScore >= info.nScore) {
            info.nScore = nScore + (fAlready ? 1 : 0);
            info.nPort = addr.GetPort();
        }
        SetReachable(addr.GetNetwork());
    }


    return true;
}


bool AddLocal(const CNetAddr &addr, int nScore)
```

```
{
    return AddLocal(CService(addr, GetListenPort()), nScore);
}


/** Make a particular network entirely off-limits (no
automatic connects to it) */
void SetLimited(enum Network net, bool fLimited)
{
    if (net == NET_UNROUTABLE)
        return;
    LOCK(cs_mapLocalHost);
    vfLimited[net] = fLimited;
}


bool IsLimited(enum Network net)
{
    LOCK(cs_mapLocalHost);
    return vfLimited[net];
}


bool IsLimited(const CNetAddr &addr)
{
    return IsLimited(addr.GetNetwork());
}


/** vote for a local address */
bool SeenLocal(const CService& addr)
{
    {
        LOCK(cs_mapLocalHost);
        if (mapLocalHost.count(addr) == 0)
            return false;
        mapLocalHost[addr].nScore++;
    }
    return true;
}



/** check whether a given address is potentially local */
bool IsLocal(const CService& addr)
```

```
{
    LOCK(cs_mapLocalHost);
    return mapLocalHost.count(addr) > 0;
}


/** check whether a given network is one we can probably
connect to */
bool IsReachable(enum Network net)
{
    LOCK(cs_mapLocalHost);
    return vfReachable[net] && !vfLimited[net];
}


/** check whether a given address is in a network we can
probably connect to */
bool IsReachable(const CNetAddr& addr)
{
    enum Network net = addr.GetNetwork();
    return IsReachable(net);
}


void AddressCurrentlyConnected(const CService& addr)
{
    addrman.Connected(addr);
}




uint64_t CNode::nTotalBytesRecv = 0;
uint64_t CNode::nTotalBytesSent = 0;
CCriticalSection CNode::cs_totalBytesRecv;
CCriticalSection CNode::cs_totalBytesSent;


CNode* FindNode(const CNetAddr& ip)
{
    LOCK(cs_vNodes);
    BOOST_FOREACH(CNode* pnode, vNodes)
        if ((CNetAddr)pnode->addr == ip)
            return (pnode);
    return NULL;
}
```

```
CNode* FindNode(const std::string& addrName)
{
    LOCK(cs_vNodes);
    BOOST_FOREACH(CNode* pnode, vNodes)
        if (pnode->addrName == addrName)
            return (pnode);
    return NULL;
}


CNode* FindNode(const CService& addr)
{
    LOCK(cs_vNodes);
    BOOST_FOREACH(CNode* pnode, vNodes){
        if(Params().NetworkID() == CBaseChainParams::REGTEST){
            //if using regtest, just check the IP
            if((CNetAddr)pnode->addr == (CNetAddr)addr)
                return (pnode);
        } else {
            if(pnode->addr == addr)
                return (pnode);
        }
    }
    return NULL;
}


CNode* ConnectNode(CAddress addrConnect, const char *pszDest,
bool darkSendMaster)
{
    if (pszDest == NULL) {
        // we clean masternode connections in
CMasternodeMan::ProcessMasternodeConnections()
        // so should be safe to skip this and connect to local Hot
MN on CActiveMasternode::ManageStatus()
        if (IsLocal(addrConnect) && !darkSendMaster)
            return NULL;


        // Look for an existing connection
        CNode* pnode = FindNode((CService)addrConnect);
        if (pnode)
        {
            pnode->fDarkSendMaster = darkSendMaster;
```

```
            pnode->AddRef();
            return pnode;
        }
    }


    /// debug print
    LogPrint("net", "trying connection %s lastseen=%.1fhrs\n",
        pszDest ? pszDest : addrConnect.ToString(),
        pszDest ? 0.0 : (double)(GetAdjustedTime() -
addrConnect.nTime)/3600.0);


    // Connect
    SOCKET hSocket;
    bool proxyConnectionFailed = false;
    if (pszDest ? ConnectSocketByName(addrConnect, hSocket,
pszDest, Params().GetDefaultPort(), nConnectTimeout,
&proxyConnectionFailed) :
                  ConnectSocket(addrConnect, hSocket,
nConnectTimeout, &proxyConnectionFailed))
    {
        if (!IsSelectableSocket(hSocket)) {
            LogPrintf("Cannot create connection: non-selectable
socket created (fd >= FD_SETSIZE ?)\n");
            CloseSocket(hSocket);
            return NULL;
        }


        addrman.Attempt(addrConnect);


        // Add node
        CNode* pnode = new CNode(hSocket, addrConnect, pszDest ?
pszDest : "", false);
        pnode->AddRef();


        {
            LOCK(cs_vNodes);
            vNodes.push_back(pnode);
        }
```

```
        pnode->nTimeConnected = GetTime();
        if(darkSendMaster) pnode->fDarkSendMaster = true;


        return pnode;
    } else if (!proxyConnectionFailed) {
        // If connecting to the node failed, and failure is not
caused by a problem connecting to
        // the proxy, mark this as an attempt.
        addrman.Attempt(addrConnect);
    }


    return NULL;
}


void CNode::CloseSocketDisconnect()
{
    fDisconnect = true;
    if (hSocket != INVALID_SOCKET)
    {
        LogPrint("net", "disconnecting peer=%d\n", id);
        CloseSocket(hSocket);
    }


    // in case this fails, we'll empty the recv buffer when
the CNode is deleted
    TRY_LOCK(cs_vRecvMsg, lockRecv);
    if (lockRecv)
        vRecvMsg.clear();
}


void CNode::PushVersion()
{
    int nBestHeight = g_signals.GetHeight().get_value_or(0);


    /// when NTP implemented, change to just nTime
= GetAdjustedTime()
    int64_t nTime = (fInbound ? GetAdjustedTime() : GetTime());
    CAddress addrYou = (addr.IsRoutable() && !IsProxy(addr) ?
addr : CAddress(CService("0.0.0.0",0)));
```

```
    CAddress addrMe = GetLocalAddress(&addr);
    GetRandBytes((unsigned char*)&nLocalHostNonce,
sizeof(nLocalHostNonce));
    if (fLogIPs)
        LogPrint("net", "send version message: version %d,
blocks=%d, us=%s, them=%s, peer=%d\n", PROTOCOL_VERSION,
nBestHeight, addrMe.ToString(), addrYou.ToString(), id);
    else
        LogPrint("net", "send version message: version %d,
blocks=%d, us=%s, peer=%d\n", PROTOCOL_VERSION,
nBestHeight, addrMe.ToString(), id);
    PushMessage("version", PROTOCOL_VERSION, nLocalServices,
nTime, addrYou, addrMe,
                nLocalHostNonce, FormatSubVersion(CLIENT_NAME,
CLIENT_VERSION, std::vector<string>()), nBestHeight, true);
}
```

```
std::map<CNetAddr, int64_t> CNode::setBanned;
CCriticalSection CNode::cs_setBanned;


void CNode::ClearBanned()
{
    setBanned.clear();
}


bool CNode::IsBanned(CNetAddr ip)
{
    bool fResult = false;
    {
        LOCK(cs_setBanned);
        std::map<CNetAddr, int64_t>::iterator i =
setBanned.find(ip);
        if (i != setBanned.end())
        {
            int64_t t = (*i).second;
```

```
            if (GetTime() < t)
                fResult = true;
        }
    }
    return fResult;
}


bool CNode::Ban(const CNetAddr &addr) {
    int64_t banTime = GetTime()+GetArg("-bantime", 60*60*24);  //
Default 24-hour ban
    {
        LOCK(cs_setBanned);
        if (setBanned[addr] < banTime)
            setBanned[addr] = banTime;
    }
    return true;
}



std::vector<CSubNet> CNode::vWhitelistedRange;
CCriticalSection CNode::cs_vWhitelistedRange;


bool CNode::IsWhitelistedRange(const CNetAddr &addr) {
    LOCK(cs_vWhitelistedRange);
    BOOST_FOREACH(const CSubNet& subnet, vWhitelistedRange) {
        if (subnet.Match(addr))
            return true;
    }
    return false;
}


void CNode::AddWhitelistedRange(const CSubNet &subnet) {
    LOCK(cs_vWhitelistedRange);
    vWhitelistedRange.push_back(subnet);
}


#undef X
#define X(name) stats.name = name
void CNode::copyStats(CNodeStats &stats)
{
```

```
    stats.nodeid = this->GetId();
    X(nServices);
    X(nLastSend);
    X(nLastRecv);
    X(nTimeConnected);
    X(addrName);
    X(nVersion);
    X(cleanSubVer);
    X(fInbound);
    X(nStartingHeight);
    X(nSendBytes);
    X(nRecvBytes);
    X(fWhitelisted);


    // It is common for nodes with good ping times to
suddenly become lagged,
    // due to a new block arriving or other large transfer.
    // Merely reporting pingtime might fool the caller into
thinking the node was still responsive,
    // since pingtime does not update until the ping is complete,
which might take a while.
    // So, if a ping is taking an unusually long time in flight,
    // the caller can immediately detect that this is happening.
    int64_t nPingUsecWait = 0;
    if ((0 != nPingNonceSent) && (0 != nPingUsecStart)) {
        nPingUsecWait = GetTimeMicros() - nPingUsecStart;
    }


    // Raw ping time is in microseconds, but show it to user as
whole seconds (Factor users should be well used to small
numbers with many decimal places by now :)
    stats.dPingTime = (((double)nPingUsecTime) / 1e6);
    stats.dPingWait = (((double)nPingUsecWait) / 1e6);


    // Leave string empty if addrLocal invalid (not filled in yet)
    stats.addrLocal = addrLocal.IsValid() ? addrLocal.ToString() :
"";
}
#undef X


// requires LOCK(cs_vRecvMsg)
bool CNode::ReceiveMsgBytes(const char *pch, unsigned int nBytes)
```

```
{
    while (nBytes > 0) {

        // get current incomplete message, or create a new one
        if (vRecvMsg.empty() ||
            vRecvMsg.back().complete())
            vRecvMsg.push_back(CNetMessage(SER_NETWORK,
nRecvVersion));

        CNetMessage& msg = vRecvMsg.back();

        // absorb network data
        int handled;
        if (!msg.in_data)
            handled = msg.readHeader(pch, nBytes);
        else
            handled = msg.readData(pch, nBytes);

        if (handled < 0)
                return false;

        if (msg.in_data && msg.hdr.nMessageSize >
MAX_PROTOCOL_MESSAGE_LENGTH) {
            LogPrint("net", "Oversized message from peer=%i,
disconnecting", GetId());
            return false;
        }

        pch += handled;
        nBytes -= handled;

        if (msg.complete()) {
            msg.nTime = GetTimeMicros();
            messageHandlerCondition.notify_one();
        }
    }

    return true;
```

```
}


int CNetMessage::readHeader(const char *pch, unsigned int nBytes)
{
    // copy data to temporary parsing buffer
    unsigned int nRemaining = 24 - nHdrPos;
    unsigned int nCopy = std::min(nRemaining, nBytes);


    memcpy(&hdrbuf[nHdrPos], pch, nCopy);
    nHdrPos += nCopy;


    // if header incomplete, exit
    if (nHdrPos < 24)
        return nCopy;


    // deserialize to CMessageHeader
    try {
        hdrbuf >> hdr;
    }
    catch (const std::exception &) {
        return -1;
    }


    // reject messages larger than MAX_SIZE
    if (hdr.nMessageSize > MAX_SIZE)
            return -1;


    // switch state to reading message data
    in_data = true;


    return nCopy;
}


int CNetMessage::readData(const char *pch, unsigned int nBytes)
{
    unsigned int nRemaining = hdr.nMessageSize - nDataPos;
    unsigned int nCopy = std::min(nRemaining, nBytes);
```

```
    if (vRecv.size() < nDataPos + nCopy) {
        // Allocate up to 256 KiB ahead, but never more than the
total message size.
        vRecv.resize(std::min(hdr.nMessageSize, nDataPos + nCopy +
256 * 1024));
    }


    memcpy(&vRecv[nDataPos], pch, nCopy);
    nDataPos += nCopy;


    return nCopy;
}
```

```
// requires LOCK(cs_vSend)
void SocketSendData(CNode *pnode)
{
    std::deque<CSerializeData>::iterator it = pnode-
>vSendMsg.begin();


    while (it != pnode->vSendMsg.end()) {
        const CSerializeData &data = *it;
        assert(data.size() > pnode->nSendOffset);
        int nBytes = send(pnode->hSocket, &data[pnode-
>nSendOffset], data.size() - pnode->nSendOffset, MSG_NOSIGNAL
| MSG_DONTWAIT);
```

```
            if (nBytes > 0) {
                pnode->nLastSend = GetTime();
                pnode->nSendBytes += nBytes;
                pnode->nSendOffset += nBytes;
                pnode->RecordBytesSent(nBytes);
                if (pnode->nSendOffset == data.size()) {
                    pnode->nSendOffset = 0;
                    pnode->nSendSize -= data.size();
                    it++;
                } else {
                    // could not send full message; stop sending more
                    break;
                }
            } else {
                if (nBytes < 0) {
                    // error
                    int nErr = WSAGetLastError();
                    if (nErr != WSAEWOULDBLOCK && nErr != WSAEMSGSIZE
&& nErr != WSAEINTR && nErr != WSAEINPROGRESS)
                    {
                        LogPrintf("socket send error %s\n",
NetworkErrorString(nErr));
                        pnode->CloseSocketDisconnect();
                    }
                }
                // couldn't send anything at all
                break;
            }
        }


        if (it == pnode->vSendMsg.end()) {
            assert(pnode->nSendOffset == 0);
            assert(pnode->nSendSize == 0);
        }
        pnode->vSendMsg.erase(pnode->vSendMsg.begin(), it);
    }


    static list<CNode*> vNodesDisconnected;


    void ThreadSocketHandler()
    {
        unsigned int nPrevNodeCount = 0;
        while (true)
```

```
    {
        //
        // Disconnect nodes
        //
        {
            LOCK(cs_vNodes);
            // Disconnect unused nodes
            vector<CNode*> vNodesCopy = vNodes;
            BOOST_FOREACH(CNode* pnode, vNodesCopy)
            {
                if (pnode->fDisconnect ||
                    (pnode->GetRefCount() <= 0 && pnode-
>vRecvMsg.empty() && pnode->nSendSize == 0 && pnode-
>ssSend.empty()))
                {
                    // remove from vNodes
                    vNodes.erase(remove(vNodes.begin(),
vNodes.end(), pnode), vNodes.end());


                    // release outbound grant (if any)
                    pnode->grantOutbound.Release();


                    // close socket and cleanup
                    pnode->CloseSocketDisconnect();


                    // hold in disconnected pool until all refs
are released
                    if (pnode->fNetworkNode || pnode->fInbound)
                        pnode->Release();
                    vNodesDisconnected.push_back(pnode);
                }
            }
        }
        {
            // Delete disconnected nodes
            list<CNode*> vNodesDisconnectedCopy =
vNodesDisconnected;
            BOOST_FOREACH(CNode* pnode, vNodesDisconnectedCopy)
            {
                // wait until threads are done using it
                if (pnode->GetRefCount() <= 0)
                {
                    bool fDelete = false;
```

```
                    {
                        TRY_LOCK(pnode->cs_vSend, lockSend);
                        if (lockSend)
                        {
                            TRY_LOCK(pnode->cs_vRecvMsg,
lockRecv);
                            if (lockRecv)
                            {
                                TRY_LOCK(pnode->cs_inventory,
lockInv);
                                if (lockInv)
                                    fDelete = true;
                            }
                        }
                    }
                    if (fDelete)
                    {
                        vNodesDisconnected.remove(pnode);
                        delete pnode;
                    }
                }
            }
        }
        if(vNodes.size() != nPrevNodeCount) {
            nPrevNodeCount = vNodes.size();

uiInterface.NotifyNumConnectionsChanged(nPrevNodeCount);
        }


        //
        // Find which sockets have data to receive
        //
        struct timeval timeout;
        timeout.tv_sec  = 0;
        timeout.tv_usec = 50000; // frequency to poll pnode->vSend


        fd_set fdsetRecv;
        fd_set fdsetSend;
        fd_set fdsetError;
        FD_ZERO(&fdsetRecv);
        FD_ZERO(&fdsetSend);
        FD_ZERO(&fdsetError);
        SOCKET hSocketMax = 0;
        bool have_fds = false;
```

```
        BOOST_FOREACH(const ListenSocket& hListenSocket,
vhListenSocket) {
            FD_SET(hListenSocket.socket, &fdsetRecv);
            hSocketMax = max(hSocketMax, hListenSocket.socket);
            have_fds = true;
        }


        {
            LOCK(cs_vNodes);
            BOOST_FOREACH(CNode* pnode, vNodes)
            {
                if (pnode->hSocket == INVALID_SOCKET)
                    continue;
                FD_SET(pnode->hSocket, &fdsetError);
                hSocketMax = max(hSocketMax, pnode->hSocket);
                have_fds = true;


                // Implement the following logic:
                // * If there is data to send, select() for
sending data. As this only
                //   happens when optimistic write failed, we
choose to first drain the
                //   write buffer in this case before receiving
more. This avoids
                //   needlessly queueing received data, if the
remote peer is not themselves
                //   receiving data. This means properly utilizing
TCP flow control signalling.
                // * Otherwise, if there is no (complete) message
in the receive buffer,
                //   or there is space left in the buffer,
select() for receiving data.
                // * (if neither of the above applies, there is
certainly one message
                //   in the receiver buffer ready to be
processed).
                // Together, that means that at least one of the
following is always possible,
                // so we don't deadlock:
                // * We send some data.
                // * We wait for data to be received (and
disconnect after timeout).
```

```
                // * We process a message in the buffer (message
handler thread).
                {
                    TRY_LOCK(pnode->cs_vSend, lockSend);
                    if (lockSend && !pnode->vSendMsg.empty()) {
                        FD_SET(pnode->hSocket, &fdsetSend);
                        continue;
                    }
                }
                {
                    TRY_LOCK(pnode->cs_vRecvMsg, lockRecv);
                    if (lockRecv && (
                        pnode->vRecvMsg.empty() || !pnode-
>vRecvMsg.front().complete() ||
                        pnode->GetTotalRecvSize() <=
ReceiveFloodSize()))
                        FD_SET(pnode->hSocket, &fdsetRecv);
                }
            }
        }


        int nSelect = select(have_fds ? hSocketMax + 1 : 0,
                             &fdsetRecv, &fdsetSend, &fdsetError,
&timeout);
        boost::this_thread::interruption_point();


        if (nSelect == SOCKET_ERROR)
        {
            if (have_fds)
            {
                int nErr = WSAGetLastError();
                LogPrintf("socket select error %s\n",
NetworkErrorString(nErr));
                for (unsigned int i = 0; i <= hSocketMax; i++)
                    FD_SET(i, &fdsetRecv);
            }
            FD_ZERO(&fdsetSend);
            FD_ZERO(&fdsetError);
            MilliSleep(timeout.tv_usec/1000);
        }


        //
        // Accept new connections
```

```
        //
        BOOST_FOREACH(const ListenSocket& hListenSocket,
vhListenSocket)
        {
            if (hListenSocket.socket != INVALID_SOCKET &&
FD_ISSET(hListenSocket.socket, &fdsetRecv))
            {
                struct sockaddr_storage sockaddr;
                socklen_t len = sizeof(sockaddr);
                SOCKET hSocket = accept(hListenSocket.socket,
(struct sockaddr*)&sockaddr, &len);
                CAddress addr;
                int nInbound = 0;


                if (hSocket != INVALID_SOCKET)
                    if (!addr.SetSockAddr((const struct
sockaddr*)&sockaddr))
                        LogPrintf("Warning: Unknown socket
family\n");


                bool whitelisted = hListenSocket.whitelisted ||
CNode::IsWhitelistedRange(addr);
                {
                    LOCK(cs_vNodes);
                    BOOST_FOREACH(CNode* pnode, vNodes)
                        if (pnode->fInbound)
                            nInbound++;
                }


                if (hSocket == INVALID_SOCKET)
                {
                    int nErr = WSAGetLastError();
                    if (nErr != WSAEWOULDBLOCK)
                        LogPrintf("socket error accept
failed: %s\n", NetworkErrorString(nErr));
                }
                else if (!IsSelectableSocket(hSocket))
                {
                    LogPrintf("connection from %s dropped: non-
selectable socket\n", addr.ToString());
                    CloseSocket(hSocket);
                }
                else if (nInbound >= nMaxConnections -
```

```
MAX_OUTBOUND_CONNECTIONS)
                {
                    LogPrint("net", "connection from %s dropped
(full)\n", addr.ToString());
                    CloseSocket(hSocket);
                }
                else if (CNode::IsBanned(addr) && !whitelisted)
                {
                    LogPrintf("connection from %s dropped
(banned)\n", addr.ToString());
                    CloseSocket(hSocket);
                }
                else
                {
                    CNode* pnode = new CNode(hSocket, addr, "",
true);
                    pnode->AddRef();
                    pnode->fWhitelisted = whitelisted;


                    {
                        LOCK(cs_vNodes);
                        vNodes.push_back(pnode);
                    }
                }
            }
        }


        //
        // Service each socket
        //
        vector<CNode*> vNodesCopy;
        {
            LOCK(cs_vNodes);
            vNodesCopy = vNodes;
            BOOST_FOREACH(CNode* pnode, vNodesCopy)
                pnode->AddRef();
        }
        BOOST_FOREACH(CNode* pnode, vNodesCopy)
        {
            boost::this_thread::interruption_point();


            //
            // Receive
```

```
            //
            if (pnode->hSocket == INVALID_SOCKET)
                continue;
            if (FD_ISSET(pnode->hSocket, &fdsetRecv) ||
FD_ISSET(pnode->hSocket, &fdsetError))
            {
                TRY_LOCK(pnode->cs_vRecvMsg, lockRecv);
                if (lockRecv)
                {
                    {
                        // typical socket buffer is 8K-64K
                        char pchBuf[0x10000];
                        int nBytes = recv(pnode->hSocket, pchBuf,
sizeof(pchBuf), MSG_DONTWAIT);
                        if (nBytes > 0)
                        {
                            if (!pnode->ReceiveMsgBytes(pchBuf,
nBytes))
                                pnode->CloseSocketDisconnect();
                            pnode->nLastRecv = GetTime();
                            pnode->nRecvBytes += nBytes;
                            pnode->RecordBytesRecv(nBytes);
                        }
                        else if (nBytes == 0)
                        {
                            // socket closed gracefully
                            if (!pnode->fDisconnect)
                                LogPrint("net", "socket
closed\n");
                            pnode->CloseSocketDisconnect();
                        }
                        else if (nBytes < 0)
                        {
                            // error
                            int nErr = WSAGetLastError();
                            if (nErr != WSAEWOULDBLOCK && nErr !=
WSAEMSGSIZE && nErr != WSAEINTR && nErr != WSAEINPROGRESS)
                            {
                                if (!pnode->fDisconnect)
                                    LogPrintf("socket recv
error %s\n", NetworkErrorString(nErr));
                                pnode->CloseSocketDisconnect();
                            }
                        }
                    }
                }
```

```
            }


            //
            // Send
            //
            if (pnode->hSocket == INVALID_SOCKET)
                continue;
            if (FD_ISSET(pnode->hSocket, &fdsetSend))
            {
                TRY_LOCK(pnode->cs_vSend, lockSend);
                if (lockSend)
                    SocketSendData(pnode);
            }


            //
            // Inactivity checking
            //
            int64_t nTime = GetTime();
            if (nTime - pnode->nTimeConnected > 60)
            {
                if (pnode->nLastRecv == 0 || pnode->nLastSend ==
0)
                {
                    LogPrint("net", "socket no message in first 60
seconds, %d %d from %d\n", pnode->nLastRecv != 0, pnode-
>nLastSend != 0, pnode->id);
                    pnode->fDisconnect = true;
                }
                else if (nTime - pnode->nLastSend >
TIMEOUT_INTERVAL)
                {
                    LogPrintf("socket sending timeout: %is\n",
nTime - pnode->nLastSend);
                    pnode->fDisconnect = true;
                }
                else if (nTime - pnode->nLastRecv > (pnode-
>nVersion > BIP0031_VERSION ? TIMEOUT_INTERVAL : 90*60))
                {
                    LogPrintf("socket receive timeout: %is\n",
nTime - pnode->nLastRecv);
                    pnode->fDisconnect = true;
                }
                else if (pnode->nPingNonceSent && pnode-
>nPingUsecStart + TIMEOUT_INTERVAL * 1000000 < GetTimeMicros())
```

```
                {
                    LogPrintf("ping timeout: %fs\n", 0.000001 *
(GetTimeMicros() - pnode->nPingUsecStart));
                    pnode->fDisconnect = true;
                }
            }
        }
        {
            LOCK(cs_vNodes);
            BOOST_FOREACH(CNode* pnode, vNodesCopy)
                pnode->Release();
        }
    }
}
```

```
#ifdef USE_UPNP
void ThreadMapPort()
{
    std::string port = strprintf("%u", GetListenPort());
    const char * multicastif = 0;
    const char * minissdpdpath = 0;
    struct UPNPDev * devlist = 0;
    char lanaddr[64];


#ifndef UPNPDISCOVER_SUCCESS
    /* miniupnpc 1.5 */
    devlist = upnpDiscover(2000, multicastif, minissdpdpath, 0);
#elif MINIUPNPC_API_VERSION < 14
```

```
    /* miniupnpc 1.6 */
    int error = 0;
    devlist = upnpDiscover(2000, multicastif, minissdpdpath, 0, 0,
&error);
#else
    /* miniupnpc 1.9.20150730 */
    int error = 0;
    devlist = upnpDiscover(2000, multicastif, minissdpdpath, 0, 0,
2, &error);
#endif


    struct UPNPUrls urls;
    struct IGDdatas data;
    int r;


    r = UPNP_GetValidIGD(devlist, &urls, &data, lanaddr,
sizeof(lanaddr));
    if (r == 1)
    {
        if (fDiscover) {
            char externalIPAddress[40];
            r = UPNP_GetExternalIPAddress(urls.controlURL,
data.first.servicetype, externalIPAddress);
            if(r != UPNPCOMMAND_SUCCESS)
                LogPrintf("UPnP: GetExternalIPAddress()
returned %d\n", r);
            else
            {
                if(externalIPAddress[0])
                {
                    LogPrintf("UPnP: ExternalIPAddress = %s\n",
externalIPAddress);
                    AddLocal(CNetAddr(externalIPAddress),
LOCAL_UPNP);
                }
                else
                    LogPrintf("UPnP: GetExternalIPAddress
failed.\n");
            }
        }


        string strDesc = "Factor " + FormatFullVersion();
```

```
        try {
            while (true) {
#ifndef UPNPDISCOVER_SUCCESS
                /* miniupnpc 1.5 */
                r = UPNP_AddPortMapping(urls.controlURL,
data.first.servicetype,
                                        port.c_str(), port.c_str(),
lanaddr, strDesc.c_str(), "TCP", 0);
#else
                /* miniupnpc 1.6 */
                r = UPNP_AddPortMapping(urls.controlURL,
data.first.servicetype,
                                        port.c_str(), port.c_str(),
lanaddr, strDesc.c_str(), "TCP", 0, "0");
#endif


                if(r!=UPNPCOMMAND_SUCCESS)
                    LogPrintf("AddPortMapping(%s, %s, %s) failed
with code %d (%s)\n",
                         port, port, lanaddr, r, strupnperror(r));
                else
                    LogPrintf("UPnP Port Mapping successful.\n");;


                MilliSleep(20*60*1000); // Refresh every 20
minutes
            }
        }
        catch (boost::thread_interrupted)
        {
            r = UPNP_DeletePortMapping(urls.controlURL,
data.first.servicetype, port.c_str(), "TCP", 0);
            LogPrintf("UPNP_DeletePortMapping() returned : %d\n",
r);
            freeUPNPDevlist(devlist); devlist = 0;
            FreeUPNPUrls(&urls);
            throw;
        }
    } else {
        LogPrintf("No valid UPnP IGDs found\n");
        freeUPNPDevlist(devlist); devlist = 0;
        if (r != 0)
            FreeUPNPUrls(&urls);
    }
```

```
}


void MapPort(bool fUseUPnP)
{
    static boost::thread* upnp_thread = NULL;


    if (fUseUPnP)
    {
        if (upnp_thread) {
            upnp_thread->interrupt();
            upnp_thread->join();
            delete upnp_thread;
        }
        upnp_thread = new
boost::thread(boost::bind(&TraceThread<void (*)()>, "upnp",
&ThreadMapPort));
    }
    else if (upnp_thread) {
        upnp_thread->interrupt();
        upnp_thread->join();
        delete upnp_thread;
        upnp_thread = NULL;
    }
}


#else
void MapPort(bool)
{
    // Intentionally left blank.
}
#endif
```

```
void ThreadDNSAddressSeed()
{
    // goal: only query DNS seeds if address need is acute
    if ((addrman.size() > 0) &&
        (!GetBoolArg("-forcednsseed", false))) {
        MilliSleep(11 * 1000);


        LOCK(cs_vNodes);
        if (vNodes.size() >= 2) {
            LogPrintf("P2P peers available. Skipped DNS
seeding.\n");
            return;
        }
    }


    const vector<CDNSSeedData> &vSeeds = Params().DNSSeeds();
    int found = 0;


    LogPrintf("Loading addresses from DNS seeds (could take
a while)\n");


    BOOST_FOREACH(const CDNSSeedData &seed, vSeeds) {
        if (HaveNameProxy()) {
            AddOneShot(seed.host);
        } else {
            vector<CNetAddr> vIPs;
            vector<CAddress> vAdd;
            if (LookupHost(seed.host.c_str(), vIPs))
            {
                BOOST_FOREACH(CNetAddr& ip, vIPs)
                {
                    int nOneDay = 24*3600;
                    CAddress addr = CAddress(CService(ip,
Params().GetDefaultPort()));
                    addr.nTime = GetTime() - 3*nOneDay -
GetRand(4*nOneDay); // use a random age between 3 and 7 days old
                    vAdd.push_back(addr);
                    found++;
                }
            }
            addrman.Add(vAdd, CNetAddr(seed.name, true));
        }
```

```
    }

    LogPrintf("%d addresses found from DNS seeds\n", found);
}
```

```
void DumpAddresses()
{
    int64_t nStart = GetTimeMillis();

    CAddrDB adb;
    adb.Write(addrman);

    LogPrint("net", "Flushed %d addresses to peers.dat  %dms\n",
           addrman.size(), GetTimeMillis() - nStart);
}

void static ProcessOneShot()
{
    string strDest;
```

```
    {
        LOCK(cs_vOneShots);
        if (vOneShots.empty())
            return;
        strDest = vOneShots.front();
        vOneShots.pop_front();
    }
    CAddress addr;
    CSemaphoreGrant grant(*semOutbound, true);
    if (grant) {
        if (!OpenNetworkConnection(addr, &grant, strDest.c_str(),
true))
            AddOneShot(strDest);
    }
}


void ThreadOpenConnections()
{
    // Connect to specific addresses
    if (mapArgs.count("-connect") && mapMultiArgs["-
connect"].size() > 0)
    {
        for (int64_t nLoop = 0;; nLoop++)
        {
            ProcessOneShot();
            BOOST_FOREACH(string strAddr, mapMultiArgs["-
connect"])
            {
                CAddress addr;
                OpenNetworkConnection(addr, NULL,
strAddr.c_str());
                for (int i = 0; i < 10 && i < nLoop; i++)
                {
                    MilliSleep(500);
                }
            }
            MilliSleep(500);
        }
    }


    // Initiate network connections
    int64_t nStart = GetTime();
    while (true)
    {
```

```
        ProcessOneShot();

        MilliSleep(500);

        CSemaphoreGrant grant(*semOutbound);
        boost::this_thread::interruption_point();

        // Add seed nodes if DNS seeds are all down
(an infrastructure attack?).
        if (addrman.size() == 0 && (GetTime() - nStart > 60)) {
            static bool done = false;
            if (!done) {
                LogPrintf("Adding fixed seed nodes as DNS doesn't
seem to be available.\n");
                addrman.Add(Params().FixedSeeds(),
CNetAddr("127.0.0.1"));
                done = true;
            }
        }

        //
        // Choose an address to connect to based on most recently
seen
        //
        CAddress addrConnect;

        // Only connect out to one peer per network group (/16 for
IPv4).
        // Do this here so we don't have to critsect vNodes inside
mapAddresses critsect.
        int nOutbound = 0;
        set<vector<unsigned char> > setConnected;
        {
            LOCK(cs_vNodes);
            BOOST_FOREACH(CNode* pnode, vNodes) {
                if (!pnode->fInbound) {
                    setConnected.insert(pnode->addr.GetGroup());
                    nOutbound++;
                }
            }
        }
```

```
        int64_t nANow = GetAdjustedTime();


        int nTries = 0;
        while (true)
        {
            CAddress addr = addrman.Select();


            // if we selected an invalid address, restart
            if (!addr.IsValid() ||
setConnected.count(addr.GetGroup()) || IsLocal(addr))
                break;


            // If we didn't find an appropriate destination
after trying 100 addresses fetched from addrman,
            // stop this loop, and let the outer loop run again
(which sleeps, adds seed nodes, recalculates
            // already-connected network ranges, ...) before
trying new addrman addresses.
            nTries++;
            if (nTries > 100)
                break;


            if (IsLimited(addr))
                continue;


            // only consider very recently tried nodes after 30
failed attempts
            if (nANow - addr.nLastTry < 600 && nTries < 30)
                continue;


            // do not allow non-default ports, unless after
50 invalid addresses selected already
            if (addr.GetPort() != Params().GetDefaultPort() &&
nTries < 50)
                continue;


            addrConnect = addr;
```

```
            break;
        }


        if (addrConnect.IsValid())
            OpenNetworkConnection(addrConnect, &grant);
    }
}


void ThreadOpenAddedConnections()
{
    {
        LOCK(cs_vAddedNodes);
        vAddedNodes = mapMultiArgs["-addnode"];
    }


    if (HaveNameProxy()) {
        while(true) {
            list<string> lAddresses(0);
            {
                LOCK(cs_vAddedNodes);
                BOOST_FOREACH(string& strAddNode, vAddedNodes)
                    lAddresses.push_back(strAddNode);
            }
            BOOST_FOREACH(string& strAddNode, lAddresses) {
                CAddress addr;
                CSemaphoreGrant grant(*semOutbound);
                OpenNetworkConnection(addr, &grant,
strAddNode.c_str());
                MilliSleep(500);
            }
            MilliSleep(120000); // Retry every 2 minutes
        }
    }


    for (unsigned int i = 0; true; i++)
    {
        list<string> lAddresses(0);
        {
            LOCK(cs_vAddedNodes);
            BOOST_FOREACH(string& strAddNode, vAddedNodes)
                lAddresses.push_back(strAddNode);
        }
```

```
        list<vector<CService> > lservAddressesToAdd(0);
        BOOST_FOREACH(string& strAddNode, lAddresses)
        {
            vector<CService> vservNode(0);
            if(Lookup(strAddNode.c_str(), vservNode,
Params().GetDefaultPort(), fNameLookup, 0))
            {
                lservAddressesToAdd.push_back(vservNode);
                {
                    LOCK(cs_setservAddNodeAddresses);
                    BOOST_FOREACH(CService& serv, vservNode)
                        setservAddNodeAddresses.insert(serv);
                }
            }
        }
        // Attempt to connect to each IP for each addnode entry
until at least one is successful per addnode entry
        // (keeping in mind that addnode entries can have many IPs
if fNameLookup)
        {
            LOCK(cs_vNodes);
            BOOST_FOREACH(CNode* pnode, vNodes)
                for (list<vector<CService> >::iterator it =
lservAddressesToAdd.begin(); it !=
lservAddressesToAdd.end(); it++)
                    BOOST_FOREACH(CService& addrNode, *(it))
                        if (pnode->addr == addrNode)
                        {
                            it = lservAddressesToAdd.erase(it);
                            it--;
                            break;
                        }
        }
        BOOST_FOREACH(vector<CService>& vserv,
lservAddressesToAdd)
        {
            CSemaphoreGrant grant(*semOutbound);
            OpenNetworkConnection(CAddress(vserv[i %
vserv.size()]), &grant);
            MilliSleep(500);
        }
        MilliSleep(120000); // Retry every 2 minutes
    }
}
```

```
// if successful, this moves the passed grant to the constructed
node
bool OpenNetworkConnection(const CAddress& addrConnect,
CSemaphoreGrant *grantOutbound, const char *pszDest,
bool fOneShot)
{
    //
    // Initiate outbound network connection
    //
    boost::this_thread::interruption_point();
    if (!pszDest) {
        if (IsLocal(addrConnect) ||
            FindNode((CNetAddr)addrConnect) ||
CNode::IsBanned(addrConnect) ||
            FindNode(addrConnect.ToStringIPPort()))
            return false;
    } else if (FindNode(pszDest))
        return false;


    CNode* pnode = ConnectNode(addrConnect, pszDest);
    boost::this_thread::interruption_point();


    if (!pnode)
        return false;
    if (grantOutbound)
        grantOutbound->MoveTo(pnode->grantOutbound);
    pnode->fNetworkNode = true;
    if (fOneShot)
        pnode->fOneShot = true;


    return true;
}



void ThreadMessageHandler()
{
    boost::mutex condition_mutex;
    boost::unique_lock<boost::mutex> lock(condition_mutex);
```

```
    SetThreadPriority(THREAD_PRIORITY_BELOW_NORMAL);
    while (true)
    {
        vector<CNode*> vNodesCopy;
        {
            LOCK(cs_vNodes);
            vNodesCopy = vNodes;
            BOOST_FOREACH(CNode* pnode, vNodesCopy) {
                pnode->AddRef();
            }
        }


        // Poll the connected nodes for messages
        CNode* pnodeTrickle = NULL;
        if (!vNodesCopy.empty())
            pnodeTrickle = vNodesCopy[GetRand(vNodesCopy.size())];


        bool fSleep = true;


        BOOST_FOREACH(CNode* pnode, vNodesCopy)
        {
            if (pnode->fDisconnect)
                continue;


            // Receive messages
            {
                TRY_LOCK(pnode->cs_vRecvMsg, lockRecv);
                if (lockRecv)
                {
                    if (!g_signals.ProcessMessages(pnode))
                        pnode->CloseSocketDisconnect();


                    if (pnode->nSendSize < SendBufferSize())
                    {
                        if (!pnode->vRecvGetData.empty() ||
(!pnode->vRecvMsg.empty() && pnode->vRecvMsg[0].complete()))
                        {
                            fSleep = false;
                        }
                    }
                }
```

```
            }
            boost::this_thread::interruption_point();


            // Send messages
            {
                TRY_LOCK(pnode->cs_vSend, lockSend);
                if (lockSend)
                    g_signals.SendMessages(pnode, pnode ==
pnodeTrickle || pnode->fWhitelisted);
            }
            boost::this_thread::interruption_point();
        }




        {
            LOCK(cs_vNodes);
            BOOST_FOREACH(CNode* pnode, vNodesCopy)
                pnode->Release();
        }


        if (fSleep)
            messageHandlerCondition.timed_wait(lock,
boost::posix_time::microsec_clock::universal_time() +
boost::posix_time::milliseconds(100));
    }
}
```

```
bool BindListenPort(const CService &addrBind, string&
strError, bool fWhitelisted)
{
    strError = "";
```

```
    int nOne = 1;


    // Create socket for listening for incoming connections
    struct sockaddr_storage sockaddr;
    socklen_t len = sizeof(sockaddr);
    if (!addrBind.GetSockAddr((struct sockaddr*)&sockaddr, &len))
    {
        strError = strprintf("Error: Bind address family for %s
not supported", addrBind.ToString());
        LogPrintf("%s\n", strError);
        return false;
    }


    SOCKET hListenSocket = socket(((struct sockaddr*)&sockaddr)-
>sa_family, SOCK_STREAM, IPPROTO_TCP);
    if (hListenSocket == INVALID_SOCKET)
    {
        strError = strprintf("Error: Couldn't open socket for
incoming connections (socket returned error %s)",
NetworkErrorString(WSAGetLastError()));
        LogPrintf("%s\n", strError);
        return false;
    }
    if (!IsSelectableSocket(hListenSocket))
    {
        strError = "Error: Couldn't create a listenable socket for
incoming connections";
        LogPrintf("%s\n", strError);
        return false;
    }




#ifndef WIN32
#ifdef SO_NOSIGPIPE
    // Different way of disabling SIGPIPE on BSD
    setsockopt(hListenSocket, SOL_SOCKET, SO_NOSIGPIPE,
(void*)&nOne, sizeof(int));
#endif
    // Allow binding if the port is still in TIME_WAIT state after
    // the program was closed and restarted. Not an issue on
windows!
    setsockopt(hListenSocket, SOL_SOCKET, SO_REUSEADDR,
```

```
(void*)&nOne, sizeof(int));
#endif


    // Set to non-blocking, incoming connections will also
inherit this
    if (!SetSocketNonBlocking(hListenSocket, true)) {
        strError = strprintf("BindListenPort: Setting listening
socket to non-blocking failed, error %s\n",
NetworkErrorString(WSAGetLastError()));
        LogPrintf("%s\n", strError);
        return false;
    }


    // some systems don't have IPV6_V6ONLY but are always
v6only; others do have the option
    // and enable it by default or not. Try to enable it, if
possible.
    if (addrBind.IsIPv6()) {
#ifdef IPV6_V6ONLY
#ifdef WIN32
        setsockopt(hListenSocket, IPPROTO_IPV6, IPV6_V6ONLY,
(const char*)&nOne, sizeof(int));
#else
        setsockopt(hListenSocket, IPPROTO_IPV6, IPV6_V6ONLY,
(void*)&nOne, sizeof(int));
#endif
#endif
#ifdef WIN32
        int nProtLevel = PROTECTION_LEVEL_UNRESTRICTED;
        setsockopt(hListenSocket, IPPROTO_IPV6,
IPV6_PROTECTION_LEVEL, (const char*)&nProtLevel, sizeof(int));
#endif
    }


    if (::bind(hListenSocket, (struct sockaddr*)&sockaddr, len)
== SOCKET_ERROR)
    {
        int nErr = WSAGetLastError();
        if (nErr == WSAEADDRINUSE)
            strError = strprintf(_("Unable to bind to %s on this
computer. Factor Core is probably already
running."), addrBind.ToString());
        else
```

```
            strError = strprintf(_("Unable to bind to %s on this
computer (bind returned error %s)"), addrBind.ToString(),
NetworkErrorString(nErr));
        LogPrintf("%s\n", strError);
        CloseSocket(hListenSocket);
        return false;
    }
    LogPrintf("Bound to %s\n", addrBind.ToString());


    // Listen for incoming connections
    if (listen(hListenSocket, SOMAXCONN) == SOCKET_ERROR)
    {
        strError = strprintf(_("Error: Listening for incoming
connections failed (listen returned error %s)"),
NetworkErrorString(WSAGetLastError()));
        LogPrintf("%s\n", strError);
        CloseSocket(hListenSocket);
        return false;
    }


    vhListenSocket.push_back(ListenSocket(hListenSocket,
fWhitelisted));


    if (addrBind.IsRoutable() && fDiscover && !fWhitelisted)
        AddLocal(addrBind, LOCAL_BIND);


    return true;
}


void static Discover(boost::thread_group& threadGroup)
{
    if (!fDiscover)
        return;


#ifdef WIN32
    // Get local host IP
    char pszHostName[256] = "";
    if (gethostname(pszHostName, sizeof(pszHostName)) !=
SOCKET_ERROR)
    {
```

```
        vector<CNetAddr> vaddr;
        if (LookupHost(pszHostName, vaddr))
        {
            BOOST_FOREACH (const CNetAddr &addr, vaddr)
            {
                if (AddLocal(addr, LOCAL_IF))
                    LogPrintf("%s: %s - %s\n",__func__,
pszHostName, addr.ToString());
            }
        }
    }
#else
    // Get local host ip
    struct ifaddrs* myaddrs;
    if (getifaddrs(&myaddrs) == 0)
    {
        for (struct ifaddrs* ifa = myaddrs; ifa != NULL; ifa =
ifa->ifa_next)
        {
            if (ifa->ifa_addr == NULL) continue;
            if ((ifa->ifa_flags & IFF_UP) == 0) continue;
            if (strcmp(ifa->ifa_name, "lo") == 0) continue;
            if (strcmp(ifa->ifa_name, "lo0") == 0) continue;
            if (ifa->ifa_addr->sa_family == AF_INET)
            {
                struct sockaddr_in* s4 = (struct
sockaddr_in*)(ifa->ifa_addr);
                CNetAddr addr(s4->sin_addr);
                if (AddLocal(addr, LOCAL_IF))
                    LogPrintf("%s: IPv4 %s: %s\n",__func__, ifa-
>ifa_name, addr.ToString());
            }
            else if (ifa->ifa_addr->sa_family == AF_INET6)
            {
                struct sockaddr_in6* s6 = (struct
sockaddr_in6*)(ifa->ifa_addr);
                CNetAddr addr(s6->sin6_addr);
                if (AddLocal(addr, LOCAL_IF))
                    LogPrintf("%s: IPv6 %s: %s\n",__func__, ifa-
>ifa_name, addr.ToString());
            }
        }
        freeifaddrs(myaddrs);
    }
#endif
}
```

```
void StartNode(boost::thread_group& threadGroup)
{
    uiInterface.InitMessage(_("Loading addresses..."));
    // Load addresses for peers.dat
    int64_t nStart = GetTimeMillis();
    {
        CAddrDB adb;
        if (!adb.Read(addrman))
            LogPrintf("Invalid or missing peers.dat;
recreating\n");
    }
    LogPrintf("Loaded %i addresses from peers.dat  %dms\n",
           addrman.size(), GetTimeMillis() - nStart);
    fAddressesInitialized = true;


    if (semOutbound == NULL) {
        // initialize semaphore
        int nMaxOutbound = min(MAX_OUTBOUND_CONNECTIONS,
nMaxConnections);
        semOutbound = new CSemaphore(nMaxOutbound);
    }


    if (pnodeLocalHost == NULL)
        pnodeLocalHost = new CNode(INVALID_SOCKET,
CAddress(CService("127.0.0.1", 0), nLocalServices));


    Discover(threadGroup);


    //
    // Start threads
    //


    if (!GetBoolArg("-dnsseed", true))
        LogPrintf("DNS seeding disabled\n");
    else
        threadGroup.create_thread(boost::bind(&TraceThread<void
(*)()>, "dnsseed", &ThreadDNSAddressSeed));
```

```
    // Map ports with UPnP
    MapPort(GetBoolArg("-upnp", DEFAULT_UPNP));


    // Send and receive from sockets, accept connections
    threadGroup.create_thread(boost::bind(&TraceThread<void
(*)()>, "net", &ThreadSocketHandler));


    // Initiate outbound connections from -addnode
    threadGroup.create_thread(boost::bind(&TraceThread<void
(*)()>, "addcon", &ThreadOpenAddedConnections));


    // Initiate outbound connections
    threadGroup.create_thread(boost::bind(&TraceThread<void
(*)()>, "opencon", &ThreadOpenConnections));


    // Process messages
    threadGroup.create_thread(boost::bind(&TraceThread<void
(*)()>, "msghand", &ThreadMessageHandler));


    // Dump network addresses
    threadGroup.create_thread(boost::bind(&LoopForever<void
(*)()>, "dumpaddr", &DumpAddresses, DUMP_ADDRESSES_INTERVAL
* 1000));


}


bool StopNode()
{
    LogPrintf("StopNode()\n");
    MapPort(false);
    if (semOutbound)
        for (int i=0; i<MAX_OUTBOUND_CONNECTIONS; i++)
            semOutbound->post();


    if (fAddressesInitialized)
    {
        DumpAddresses();
        fAddressesInitialized = false;
```

```
    }


    return true;
}


class CNetCleanup
{
public:
    CNetCleanup() {}


    ~CNetCleanup()
    {
        // Close sockets
        BOOST_FOREACH(CNode* pnode, vNodes)
            if (pnode->hSocket != INVALID_SOCKET)
                CloseSocket(pnode->hSocket);
        BOOST_FOREACH(ListenSocket& hListenSocket, vhListenSocket)
            if (hListenSocket.socket != INVALID_SOCKET)
                if (!CloseSocket(hListenSocket.socket))
                    LogPrintf("CloseSocket(hListenSocket) failed
with error %s\n", NetworkErrorString(WSAGetLastError()));


        // clean up some globals (to help leak detection)
        BOOST_FOREACH(CNode *pnode, vNodes)
            delete pnode;
        BOOST_FOREACH(CNode *pnode, vNodesDisconnected)
            delete pnode;
        vNodes.clear();
        vNodesDisconnected.clear();
        vhListenSocket.clear();
        delete semOutbound;
        semOutbound = NULL;
        delete pnodeLocalHost;
        pnodeLocalHost = NULL;


#ifdef WIN32
        // Shutdown Windows Sockets
        WSACleanup();
#endif
    }
}
```

```
instance_of_cnetcleanup;


void CExplicitNetCleanup::callCleanup()
{
    // Explicit call to destructor of CNetCleanup because it's not
implicitly called
    // when the wallet is restarted from within the wallet itself.
    CNetCleanup *tmp = new CNetCleanup();
    delete tmp; // Stroustrup's gonna kill me for that
}


void RelayTransaction(const CTransaction& tx)
{
    CDataStream ss(SER_NETWORK, PROTOCOL_VERSION);
    ss.reserve(10000);
    ss << tx;
    RelayTransaction(tx, ss);
}


void RelayTransaction(const CTransaction& tx, const
CDataStream& ss)
{
    CInv inv(MSG_TX, tx.GetHash());
    {
        LOCK(cs_mapRelay);
        // Expire old relay messages
        while (!vRelayExpiration.empty() &&
vRelayExpiration.front().first < GetTime())
        {
            mapRelay.erase(vRelayExpiration.front().second);
            vRelayExpiration.pop_front();
        }


        // Save original serialized message so newer versions
are preserved
        mapRelay.insert(std::make_pair(inv, ss));
        vRelayExpiration.push_back(std::make_pair(GetTime() + 15 *
60, inv));
    }
    LOCK(cs_vNodes);
    BOOST_FOREACH(CNode* pnode, vNodes)
    {
```

```
        if(!pnode->fRelayTxes)
            continue;
        LOCK(pnode->cs_filter);
        if (pnode->pfilter)
        {
            if (pnode->pfilter->IsRelevantAndUpdate(tx))
                pnode->PushInventory(inv);
        } else
            pnode->PushInventory(inv);
    }
}


void RelayTransactionLockReq(const CTransaction& tx, bool
relayToAll)
{
    CInv inv(MSG_TXLOCK_REQUEST, tx.GetHash());


    //broadcast the new lock
    LOCK(cs_vNodes);
    BOOST_FOREACH(CNode* pnode, vNodes)
    {
        if(!relayToAll && !pnode->fRelayTxes)
            continue;


        pnode->PushMessage("ix", tx);
    }
}


void RelayInv(CInv &inv, const int minProtoVersion) {
    LOCK(cs_vNodes);
    BOOST_FOREACH(CNode* pnode, vNodes)
        if(pnode->nVersion >= minProtoVersion)
            pnode->PushInventory(inv);
}


void CNode::RecordBytesRecv(uint64_t bytes)
{
    LOCK(cs_totalBytesRecv);
    nTotalBytesRecv += bytes;
}
```

```
void CNode::RecordBytesSent(uint64_t bytes)
{
    LOCK(cs_totalBytesSent);
    nTotalBytesSent += bytes;
}


uint64_t CNode::GetTotalBytesRecv()
{
    LOCK(cs_totalBytesRecv);
    return nTotalBytesRecv;
}


uint64_t CNode::GetTotalBytesSent()
{
    LOCK(cs_totalBytesSent);
    return nTotalBytesSent;
}


void CNode::Fuzz(int nChance)
{
    if (!fSuccessfullyConnected) return; // Don't fuzz initial
handshake
    if (GetRand(nChance) != 0) return; // Fuzz 1 of every nChance
messages


    switch (GetRand(3))
    {
    case 0:
        // xor a random byte with a random value:
        if (!ssSend.empty()) {
            CDataStream::size_type pos = GetRand(ssSend.size());
            ssSend[pos] ^= (unsigned char)(GetRand(256));
        }
        break;
    case 1:
        // delete a random byte:
        if (!ssSend.empty()) {
            CDataStream::size_type pos = GetRand(ssSend.size());
            ssSend.erase(ssSend.begin()+pos);
        }
        break;
```

```
        case 2:
            // insert a random byte at a random position
            {
                CDataStream::size_type pos = GetRand(ssSend.size());
                char ch = (char)GetRand(256);
                ssSend.insert(ssSend.begin()+pos, ch);
            }
            break;
        }
        // Chance of more than one change half the time:
        // (more changes exponentially less likely):
        Fuzz(2);
    }


    //
    // CAddrDB
    //


    CAddrDB::CAddrDB()
    {
        pathAddr = GetDataDir() / "peers.dat";
    }


    bool CAddrDB::Write(const CAddrMan& addr)
    {
        // Generate random temporary filename
        unsigned short randv = 0;
        GetRandBytes((unsigned char*)&randv, sizeof(randv));
        std::string tmpfn = strprintf("peers.dat.%04x", randv);


        // serialize addresses, checksum data up to that point,
    then append csum
        CDataStream ssPeers(SER_DISK, CLIENT_VERSION);
        ssPeers << FLATDATA(Params().MessageStart());
        ssPeers << addr;
        uint256 hash = Hash(ssPeers.begin(), ssPeers.end());
        ssPeers << hash;


        // open output file, and associate with CAutoFile
        boost::filesystem::path pathAddr = GetDataDir() / "peers.dat";
        FILE *file = fopen(pathAddr.string().c_str(), "wb");
```

```
    CAutoFile fileout(file, SER_DISK, CLIENT_VERSION);
    if (fileout.IsNull())
        return error("%s : Failed to open file %s",___func__,
pathAddr.string());


    // Write and commit header, data
    try {
        fileout << ssPeers;
    }
    catch (std::exception &e) {
        return error("%s : Serialize or I/O error - %s",___func__,
e.what());
    }
    FileCommit(fileout.Get());
    fileout.fclose();


    return true;
}


bool CAddrDB::Read(CAddrMan& addr)
{
    // open input file, and associate with CAutoFile
    FILE *file = fopen(pathAddr.string().c_str(), "rb");
    CAutoFile filein(file, SER_DISK, CLIENT_VERSION);
    if (filein.IsNull())
        return error("%s : Failed to open file %s",___func__,
pathAddr.string());


    // use file size to size memory buffer
    int fileSize = boost::filesystem::file_size(pathAddr);
    int dataSize = fileSize - sizeof(uint256);
    // Don't try to resize to a negative number if file is small
    if (dataSize < 0)
        dataSize = 0;
    vector<unsigned char> vchData;
    vchData.resize(dataSize);
    uint256 hashIn;


    // read data and checksum from file
    try {
        filein.read((char *)&vchData[0], dataSize);
```

```
        filein >> hashIn;
    }
    catch (std::exception &e) {
        return error("%s : Deserialize or I/O error - %s",
__func__, e.what());
    }
    filein.fclose();


    CDataStream ssPeers(vchData, SER_DISK, CLIENT_VERSION);


    // verify stored checksum matches input data
    uint256 hashTmp = Hash(ssPeers.begin(), ssPeers.end());
    if (hashIn != hashTmp)
        return error("%s : Checksum mismatch, data corrupted",
__func__);


    unsigned char pchMsgTmp[4];
    try {
        // de-serialize file header (network specific magic
number) and ..
        ssPeers >> FLATDATA(pchMsgTmp);


        // ... verify the network matches ours
        if (memcmp(pchMsgTmp, Params().MessageStart(),
sizeof(pchMsgTmp)))
            return error("%s : Invalid network magic number",
__func__);


        // de-serialize address data into one CAddrMan object
        ssPeers >> addr;
    }
    catch (std::exception &e) {
        return error("%s : Deserialize or I/O error - %s",
__func__, e.what());
    }


    return true;
}
```

```
unsigned int ReceiveFloodSize() { return 1000*GetArg("-
maxreceivebuffer", 5*1000); }
unsigned int SendBufferSize() { return 1000*GetArg("-
maxsendbuffer", 1*1000); }


CNode::CNode(SOCKET hSocketIn, CAddress addrIn,
std::string addrNameIn, bool fInboundIn) :
ssSend(SER_NETWORK,  INIT_PROTO_VERSION),
setAddrKnown(5000)
{
    nServices = 0;
    hSocket = hSocketIn;
    nRecvVersion = INIT_PROTO_VERSION;
    nLastSend = 0;
    nLastRecv = 0;
    nSendBytes = 0;
    nRecvBytes = 0;
    nTimeConnected = GetTime();
    addr = addrIn;
    addrName = addrNameIn == "" ? addr.ToStringIPPort() :
addrNameIn;
    nVersion = 0;
    strSubVer = "";
    fWhitelisted = false;
    fOneShot = false;
    fClient = false; // set by version message
    fInbound = fInboundIn;
    fNetworkNode = false;
    fSuccessfullyConnected = false;
    fDisconnect = false;
    nRefCount = 0;
    nSendSize = 0;
    nSendOffset = 0;
    hashContinue = 0;
    nStartingHeight = -1;
    fGetAddr = false;
    fRelayTxes = false;
    setInventoryKnown.max_size(SendBufferSize() / 1000);
    pfilter = new CBloomFilter();
    nPingNonceSent = 0;
    nPingUsecStart = 0;
    nPingUsecTime = 0;
    fPingQueued = false;
    fDarkSendMaster = false;
```

```
    {
        LOCK(cs_nLastNodeId);
        id = nLastNodeId++;
    }


    if (fLogIPs)
        LogPrint("net", "Added connection to %s peer=%d\n",
addrName, id);
    else
        LogPrint("net", "Added connection peer=%d\n", id);


    // Be shy and don't send version until we hear
    if (hSocket != INVALID_SOCKET && !fInbound)
        PushVersion();


    GetNodeSignals().InitializeNode(GetId(), this);
}


CNode::~CNode()
{
    CloseSocket(hSocket);


    if (pfilter)
        delete pfilter;


    GetNodeSignals().FinalizeNode(GetId());
}


void CNode::AskFor(const CInv& inv)
{
    if (mapAskFor.size() > MAPASKFOR_MAX_SZ)
        return;
    // We're using mapAskFor as a priority queue,
    // the key is the earliest time the request can be sent
    int64_t nRequestTime;
    limitedmap<CInv, int64_t>::const_iterator it =
mapAlreadyAskedFor.find(inv);
    if (it != mapAlreadyAskedFor.end())
        nRequestTime = it->second;
```

```
    else
        nRequestTime = 0;
    LogPrint("net", "askfor %s  %d (%s) peer=%d\n",
inv.ToString(), nRequestTime, DateTimeStrFormat("%H:%M:%S",
nRequestTime/1000000), id);


    // Make sure not to reuse time indexes to keep things in
the same order
    int64_t nNow = GetTimeMicros() - 1000000;
    static int64_t nLastTime;
    ++nLastTime;
    nNow = std::max(nNow, nLastTime);
    nLastTime = nNow;


    // Each retry is 2 minutes after the last
    nRequestTime = std::max(nRequestTime + 2 * 60 * 1000000,
nNow);
    if (it != mapAlreadyAskedFor.end())
        mapAlreadyAskedFor.update(it, nRequestTime);
    else
        mapAlreadyAskedFor.insert(std::make_pair(inv,
nRequestTime));
    mapAskFor.insert(std::make_pair(nRequestTime, inv));
}


void CNode::BeginMessage(const char* pszCommand)
EXCLUSIVE_LOCK_FUNCTION(cs_vSend)
{
    ENTER_CRITICAL_SECTION(cs_vSend);
    assert(ssSend.size() == 0);
    ssSend << CMessageHeader(pszCommand, 0);
    LogPrint("net", "sending: %s ", SanitizeString(pszCommand));
}


void CNode::AbortMessage() UNLOCK_FUNCTION(cs_vSend)
{
    ssSend.clear();


    LEAVE_CRITICAL_SECTION(cs_vSend);
```

```
    LogPrint("net", "(aborted)\n");
}


void CNode::EndMessage() UNLOCK_FUNCTION(cs_vSend)
{
    // The -*messagestest options are intentionally not documented
in the help message,
    // since they are only used during development to debug the
networking code and are
    // not intended for end-users.
    if (mapArgs.count("-dropmessagestest") && GetRand(GetArg("-
dropmessagestest", 2)) == 0)
    {
        LogPrint("net", "dropmessages DROPPING SEND MESSAGE\n");
        AbortMessage();
        return;
    }
    if (mapArgs.count("-fuzzmessagestest"))
        Fuzz(GetArg("-fuzzmessagestest", 10));


    if (ssSend.size() == 0)
        return;


    // Set the size
    unsigned int nSize = ssSend.size() -
CMessageHeader::HEADER_SIZE;
    memcpy((char*)&ssSend[CMessageHeader::MESSAGE_SIZE_OFFSET],
&nSize, sizeof(nSize));


    // Set the checksum
    uint256 hash = Hash(ssSend.begin() +
CMessageHeader::HEADER_SIZE, ssSend.end());
    unsigned int nChecksum = 0;
    memcpy(&nChecksum, &hash, sizeof(nChecksum));
    assert(ssSend.size () >= CMessageHeader::CHECKSUM_OFFSET +
sizeof(nChecksum));
    memcpy((char*)&ssSend[CMessageHeader::CHECKSUM_OFFSET],
&nChecksum, sizeof(nChecksum));


    LogPrint("net", "(%d bytes) peer=%d\n", nSize, id);
```

```
    std::deque<CSerializeData>::iterator it =
vSendMsg.insert(vSendMsg.end(), CSerializeData());
    ssSend.GetAndClear(*it);
    nSendSize += (*it).size();


    // If write queue empty, attempt "optimistic write"
    if (it == vSendMsg.begin())
        SocketSendData(this);


    LEAVE_CRITICAL_SECTION(cs_vSend);
}
```

IV Protocole

1. Common structures

1. Message structure

Field Size	Description	Data type	Comments
4	magic	uint32_t	Magic value indicating message origin network, and used to seek to next message when stream state is unknown
12	command	char[12]	ASCII string identifying the packet content, NULL padded (non-NULL padding results in packet rejected)
4	length	uint32_t	Length of payload in number of bytes
4	checksum	uint32_t	First 4 bytes of sha256(sha256(payload))
?	payload	uchar[]	The actual data

实际使用的代码示例

Network	Magic value	Sent over wire as
main	0xD9B4BEF9	F9 BE B4 D9
testnet	0xDAB5BFFA	FA BF B5 DA
testnet3	0x0709110B	0B 11 09 07
namecoin	0xFEB4BEF9	F9 BE B4 FE

1.2 Variable length integer

为了节约空间，按照表现的值可以 incoding 定数。
可变长度定数是位置在长度会不同的数据类型排列/矢量。

Value	Storage length	Format
< 0xFD	1	uint8_t
<= 0xFFFF	3	0xFD followed by the length as uint16_t
<= 0xFFFF FFFF	5	0xFE followed by the length as uint32_t
-	9	0xFF followed by the length as uint64_t

把 incoding 叫做 "CompactSize"。而且以 local 储存所 (不互换与 "CompactSize")的用途，提供实现更小定数的 CVarInt class。 CVarInt 不是协议的一部分。

1.3 Variable length string

可变长度文字列是在可变长度定数后使用文字列可进行储存

Field Size	Description	Data type	Comments
1+	length	var_int	Length of the string
?	string	char[]	The string itself (can be empty)

1.4 Network address

需要网络地址时使用下列构造。
在网络地址中短信的时间戳不使用为接头词。

Size	Description	Data type	Comments
4	time	uint32	the Time (version >= 31402). **Not present in version message.**
8	services	uint64_t	same service(s) listed in version
16	IPv6/4	char[16]	IPv6 address. Network byte order. The original client only supported IPv4 and only read the last 4 bytes to get the IPv4 address. However, the IPv4 address is written into the message as a 16 byte IPv4-mapped IPv6 address (12 bytes *00 00 00 00 00 00 00 00 00 00 FF FF*, followed by the 4 bytes of the IPv4 address).
2	port	uint16_t	port number, network byte order

网络地址构造的 Hexdump 示例

```
0000   01 00 00 00 00 00 00 00  00 00 00 00 00 00 00
00  ...............
0010   00 00 FF FF 0A 00 00 01  20 8D                    ........ .

Network address:
 01 00 00 00 00 00 00 00                          - 1 (NODE_NETWORK: see
services listed under version command)
 00 00 00 00 00 00 00 00 00 00 FF FF 0A 00 00 01 - IPv6: ::ffff:a00:1
or IPv4: 10.0.0.1
 20 8D                                            - Port 8333
```

1.5 Inventory Vectors

Inventory vectors 是使用为对拥有的客体或要请的数据宣传到其它节点。

Inventory vectors 构成为下列数据形式。

Field Size	Description	Data type	Comments
4	type	uint32_t	Identifies the object type linked to this inventory
32	hash	char[32]	Hash of the object

目标程序类型是在下列可能性中定义为一个。

Value	Name	Description
0	ERROR	Any data of with this number may be ignored
1	MSG_TX	Hash is related to a transaction
2	MSG_BLOCK	Hash is related to a data block
3	MSG_FILTERED_BLOCK	Hash of a block header; identical to MSG_BLOCK. Only to be used in getdata message. Indicates the reply should be a merkleblock message rather than a block message; this only works if a bloom filter has been set.
4	MSG_CMPCT_BLOCK	Hash of a block header; identical to MSG_BLOCK. Only to be used in getdata message. Indicates the reply should be a cmpctblock message. See BIP 152 for more info.

其它数据类型值是为了向后实现视为预约。

1.6 Block Headers

区块 header 是对 getheaders 短信的应答，传送header packin。

Field Size	Description	Data type	Comments
4	version	int32_t	Block version information (note, this is signed)
32	prev_block	char[32]	The hash value of the previous block this particular block references
32	merkle_root	char[32]	The reference to a Merkle tree collection which is a hash of all transactions related to this block
4	timestamp	uint32_t	A timestamp recording when this block was created (Will overflow in 2106)
4	bits	uint32_t	The calculated difficulty target being used for this block
4	nonce	uint32_t	The nonce used to generate this block… to allow variations of the header and compute different hashes
1+	txn_count	var _ int	Number of transaction entries, this value is always 0

1.7 PrefilledTransaction

PrefilledTransaction 构造体是被使用为在 HeaderAndShortIDs 为了提供几种交易目录。

Field Name	Type	Size	Encoding	Purpose
index	CompactSize	1, 3 bytes	Compact Size, differentially encoded since the last PrefilledTransaction in a list	The index into the block at which this transaction is
tx	Transaction	variable	As encoded in tx messages	The transaction which is in the block at index index.

1.8 HeaderAndShortIDs

HeaderAndShortIDs 构造是使用为延迟在一致区块 header，可使用的交易等上使用的简单交易 ID 及部分交易。

Field Name	Type	Size	Encoding	Purpose
header	Block header	80 bytes	First 80 bytes of the block as defined by the encoding used by "block" messages	The header of the block being provided
nonce	uint64_t	8 bytes	Little Endian	A nonce for use in short transaction ID calculations
shortids_length	CompactSize	1 or 3 bytes	As used to encode array lengths elsewhere	The number of short transaction IDs in shortids (ie block tx count - prefilledtxn_length)
shortids	List of 6-byte integers	6*shortids_length bytes	Little Endian	The short transaction IDs calculated from the transactions which were not provided explicitly in prefilledtxn
prefilledtxn_length	CompactSize	1 or 3 bytes	As used to encode array lengths elsewhere	The number of prefilled transactions in prefilledtxn (ie block tx count - shortids_length)
prefilledtxn	List of PrefilledTransactions	variable size*prefilledtxn_length	As defined by PrefilledTransactionfinition, above	Used to provide the coinbase transaction and a select few which we expect a peer may be missing

1.9 BlockTransactionsRequest

BlockTransactionsRequest 构造体是使用为罗列要请的区块交易 index。

Field Name	Type	Size	Encoding	Purpose
blockhash	Binary blob	32 bytes	The output from a double-SHA256 of the block header, as used elsewhere	The blockhash of the block which the transactions being requested are in
indexes_length	CompactSize	1 or 3 bytes	As used to encode array lengths elsewhere	The number of transactions being requested
indexes	List of CompactSize	1 or 3 bytes*indexes_length	Differentially encoded	The indexes of the transactions being requested in the block

1.10 BlockTransactions

BlockTransactions 构造是使用为提供区块的部分交易。

Field Name	Type	Size	Encoding	Purpose
blockhash	Binary blob	32 bytes	The output from a double-SHA256 of the block header, as used elsewhere	The blockhash of the block which the transactions being provided are in
transactions_length	CompactSize	1 or 3 bytes	As used to encode array lengths elsewhere	The number of transactions provided
transactions	List of Transactions	variable	As encoded in tx messages	The transactions provided

1.11 Short transaction ID

短的交易 ID 是不传送全体 256bit 散列，使用为显示交易。依据下列内容来进行计算：

1. 在区块 header 添加 nonce 的 Hash
2. 输入为交易 ID，键 (k0 / k1)在实行上面散列的两个 little edian 64bit 定数的 SipHash-2-4。
3. 在 SipHash 输出删除 2 个最上位 bytes，制造为 6bytes。

1.12 Private Send (Dark Send)

private send 功能的原理很简单。通过许多人发送交易，于是发送人不分明，具有提前保障匿名性的功能。 用户为了进行交易,将汇款金额输入到钱包,按 private send 键的话,您的钱包会将输入金额分为小单位金额。(单位:0.01,0.1,1) 然后,用户的钱包会发送给具备网络上特殊环境设定的节点(Master Nord)。 每个 Masternode 在将用户输入的信息混合后,在 Masternode 用户的钱包中传送内容,并将分隔的汇款金额寄给自己,地址是以随机形式形成，发送到形成的地址中。 为了让你的资金完全无法追踪,你的钱包会重复这些过程,比如金额可以分割的任意数量,这样的过程一次结束,叫做"round"。 因此,经过尽可能多的 round 数量后,形成交易。 此外,mixing 过程在用户看不见的地方进行,在任何情况下,接收到的 Masternode 都不会接收个人信息。 因此, mixing 的 masternode 不可追击。

1.13 InstantX

该交易与标准交易不同,不是通过区块生成确认,而是 Masternode 立即确认交易。 但由于 Masternode 不会生成 Block,因此使用 InstantX 功能的交易明细也必须进入 Block。 取而代之的是,在装入积木时以"确认交易"之名进行。 一般来说,该交易方法需要 1~4 秒,比一般的传送速度要快很多。 收件人的钱包仅需几秒钟即可确认交易。 在交易确认过程中,为了防止双重支付,InstantX 在网络生成交易后,将相应交易"暂停"。 使用 InstantX 功能,具有更进一步提升交易速度的功能。 可根据使用者的意向使用。

2 Message types

1. version

制作节点连接后会立即告知版本。 远程节点应答为相应版本。

直到两个同事交换版本为止，不可进行通讯。

Payload

Field Size	Description	Data type	Comments
4	version	int32_t	Identifies protocol version being used by the node
8	services	uint64_t	bitfield of features to be enabled for this connection
8	timestamp	int64_t	standard UNIX timestamp in seconds
26	addr_recv	net_addr	The network address of the node receiving this message
Fields below require version ≥ 106			
26	addr_from	net_addr	The network address of the node emitting this message
8	nonce	uint64_t	Node random nonce, randomly generated every time a version packet is sent. This nonce is used to detect connections to self.
?	user_agent	Var _ str	User Agent (0x00 if string is 0 bytes long)
4	start_height	int32_t	The last block received by the emitting node
Fields below require version ≥ 70001			
1	relay	bool	Whether the remote peer should announce relayed transactions or not,

版本 packin 接收后，需要传送 "verack"

Value	Name	Description
1	NODE_NETWORK	This node can be asked for full blocks instead of just headers.
2	NODE_GETUTXO	See BIP 0064
4	NODE_BLOOM	See BIP 0111
8	NODE_WITNESS	See BIP 0144
1024	NODE_NETWORK_LIMITED	See BIP 0159

版本短信的 Hexdump 示例(OBSOLETE EXAMPLE : 在这示例上没有 check sum 和使用者 agent)

```
0000   F9 BE B4 D9 76 65 72 73  69 6F 6E 00 00 00 00
00   ....version.....
0010   55 00 00 00 9C 7C 00 00  01 00 00 00 00 00 00 00
U....|..........
0020   E6 15 10 4D 00 00 00 00  01 00 00 00 00 00 00
00   ...M............
0030   00 00 00 00 00 00 00 00  00 00 FF FF 0A 00 00
01   ................
0040   20 8D 01 00 00 00 00 00  00 00 00 00 00 00 00
00   ................
0050   00 00 00 00 FF FF 0A 00  00 02 20 8D DD 9D 20
2C   .......... ... ,
0060   3A B4 57 13 00 55 81 01  00                    :.W..U...

Message header:
 F9 BE B4 D9
- Main network magic bytes
 76 65 72 73 69 6F 6E 00 00 00 00 00
-"version" command
 55 00 00 00
- Payload is 85 bytes long
```

```
-No checksum in version message until 20 February 2012. See
https://Factortalk.org/index.php?topic=55852.0
Version message:
 9C 7C 00 00
 - 31900 (version 0.3.19)
  01 00 00 00 00 00 00 00
-1 (NODE_NETWORK services)
 E6 15 10 4D 00 00 00 00
- Mon Dec 20 21:50:14 EST 2010
 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF 0A 00 00
1. 20 8D - Recipient address info - see Network Address
 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF 0A 00 00
2.20 8D - Sender address info - see Network Address
 DD 9D 20 2C 3A B4 57 13
-Node random unique ID
 00
-"" sub-version string (string is 0 bytes long)
 55 81 01 00
- Last block sending node has is block #98645
```

```
0000    f9 be b4 d9 76 65 72 73 69 6f 6e 00 00 00 00
00  ....version.....
0010    64 00 00 00 35 8d 49 32 62 ea 00 00 01 00 00 00
d...5.I2b.......
0020    00 00 00 00 11 b2 d0 50 00 00 00 00 01 00 00
00  .......P........
0030    00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff
ff  ...............
0040    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00  ...............
0050    00 00 00 00 00 00 00 00 ff ff 00 00 00 00 00
00  ...............
0060    3b 2e b3 5d 8c e6 17 65 0f 2f 53 61 74 6f 73
68  ;..]...e./Satosh
0070    69 3a 30 2e 37 2e 32 2f c0 3e 03 00             i:0.7.2/.>..

Message Header:
 F9 BE B4 D9
- Main network magic bytes
```

```
 76 65 72 73 69 6F 6E 00 00 00 00 00
-"version" command
 64 00 00 00
-Payload is 100 bytes long
 35 8d 49 32
- payload checksum (little endian)

Version message:
 62 EA 00 00
- 60002 (protocol version 60002)
 01 00 00 00 00 00 00 00
-1 (NODE_NETWORK services)
 11 B2 D0 50 00 00 00 00
- Tue Dec 18 10:12:33 PST 2012
 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF 00 00 00
00 00 00 - Recipient address info - see Network Address
 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF 00 00 00
00 00 00 - Sender address info - see Network Address
 3B 2E B3 5D 8C E6 17 65
- Node ID
 0F 2F 53 61 74 6F 73 68 69 3A 30 2E 37 2E 32 2F
-"/Satoshi:0.7.2/" sub-version string (string is 15 bytes long) C
 0 3E 03 00
- Last block sending node has is block #212672
```

2.2 verack

是对 verack 短信的应答来传送的版本。这短信是构成为 "verack"命令具有文字列的短信

Verack 短信的 Hexdump :

```
0000   F9 BE B4 D9 76 65 72 61   63 6B 00 00 00 00 00
00   ....verack......
0010   00 00 00 00 5D F6 E0 E2                           ........

Message header:
 F9 BE B4 D9                          - Main network magic bytes
 76 65 72 61  63 6B 00 00 00 00 00 00 - "verack" command
 00 00 00 00                          - Payload is 0 bytes long
 5D F6 E0 E2                          - Checksum (little endian)
```

2.3 address

地址是与时间戳一起 prefixed。万一地址未确认的话，不可 relay 到其他 pear。

Field Size	Description	Data type	Comments
1+	count	var_int	Number of address entries (max: 1000)
30x?	addr_list	(uint32_t + net_addr)[]	Address of other nodes on the network. version < 209 will only read the first one. The uint32_t is a timestamp (see note below).

addr 短信的 Hexdump 示例 :

```
0000   F9 BE B4 D9 61 64 64 72  00 00 00 00 00 00 00 00   ....addr........
0010   1F 00 00 00 ED 52 39 9B  01 E2 15 10 4D 01 00 00   .....R9.....M...
0020   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 FF   ................
0030   FF 0A 00 00 01 20 8D                              ..... .

Message Header:
 F9 BE B4 D9                                          - Main network magic bytes
 61 64 64 72  00 00 00 00 00 00 00 00                 - "addr"
 1F 00 00 00                                          - payload is 31 bytes long
 ED 52 39 9B                                          - payload checksum (little
endian)

Payload:
 01                                                   - 1 address in this
message

Address:
 E2 15 10 4D                                          - Mon Dec 20 21:50:10 EST
2010 (only when version is >= 31402)
 01 00 00 00 00 00 00 00                              - 1 (NODE_NETWORK service
- see version message)
 00 00 00 00 00 00 00 00 00 00 FF FF 0A 00 00 01 - IPv4: 10.0.0.1,
IPv6: ::ffff:10.0.0.1 (IPv4-mapped IPv6 address)
 20 8D                                                - port 8333
```

2.4 inv

节点可广告对一个以上客体的知识。
可自发接受或应答 getblocks 。

Payload（最大 50,000 个项目，超过 1.8 mega bytes）：

Field Size	Description	Data type	Comments
1+	count	Var _ int	Number of inventory entries
36x?	inventory	inv_vect[]	Inventory vectors

2.5 getdata

getdata 是回应 inv 使用在搜索特定客体内容，一般是过滤掉已知的要素后收到 inv 分组后传送。 对链条交易的任意 access 是不许可 client 依赖具有全体交易 index 的节点。

Payload （最大 50,000 个项目，超过 1.8 mega bytes）：

Field Size	Description	Data type	Comments
1+	count	var_int	Number of inventory entries
36x?	inventory	inv_vect[]	Inventory vectors

2.6 notfound

Notfound 是对 getdata 的应答。举例，在 memory pool 或 relay 集合中不存在要请的交易，因此不可 relay 要请的数据项目时传送。

Field Size	Description	Data type	Comments
1+	count	var_int	Number of inventory entries

36x?	inventory	inv_vect[]	Inventory vectors

2.7 getblocks

包括从区块 locator 客体中最后得知的散列后开始的区块目录在内的 inv 部分,返还至 hash_stop 或 500 区块中首先发生的时刻。 散列是按照在短信上表示的顺序，依据节点来进行处理。 在节点的主要链子上发现区块散列的话,不管是否达到要求的界限,适当下位目录通过 inv 目录再被返还。 要想获得下一个区块散列，,必须重新发行 getblocks。

Payload:

Field Size	Description	Data type	Comments
4	version	uint32_t	the protocol version
1+	hash count	var_int	number of block locator hash entries
32+	block locator hashes	char[32]	block locator object; newest back to genesis block (dense to start, but then sparse)
32	hash_stop	char[32]	hash of the last desired block; set to zero to get as many blocks as possible (500)

为了制造区块 locater 散列，直到返还到键区块为止，需要持续 push 散列。
Push10 个散列后，所有 roof 增加两倍。

```
// From libFactor which is under AGPL
std::vector<size_t> block_locator_indexes(size_t top_height)
{
    std::vector<size_t> indexes;

    // Modify the step in the iteration.
    int64_t step = 1;

    // Start at the top of the chain and work backwards.
    for (auto index = (int64_t)top_height; index > 0; index -= step)
    {
```

```
        // Push top 10 indexes first, then back off exponentially.
        if (indexes.size() >= 10)
            step *= 2;

        indexes.push_back((size_t)index);
    }

    //  Push the genesis block index.
    indexes.push_back(0);
    return indexes;
}
```

所知道的散列值可把最低的散列值发送到最少一个散列值。
但是,区块转接器个体的目的是从呼叫人的主链子上感知错误的分岔。
若 pear 感知脱出主要链条的话，传送到最后之前的一个区块散列。Pear 从区块#1 开始。

2.8 getheaders

如果想接收下区块 header,需要重新发行 getheaders 作为新的模块位置指定者。 如果在区块 locator 工程上包含无效的散列时，部分客户会提供无效的区块 header。

Payload:

Field Size	Description	Data type	Comments
4	version	uint32_t	the protocol version
1+	hash count	var_int	number of block locator hash entries
32+	block locator hashes	char[32]	block locator object; newest back to genesis block (dense to start, but then sparse)
32	hash_stop	char[32]	hash of the last desired block header; set to zero to get as many blocks as possible (2000

2.9 tx

tx 是对 getdata 的回应，说明交易。适用 bloom filter 时，tx 客体是为了下列交易自动传送。

Field Size	Description	Data type	Comments
4	version	int32_t	Transaction data format version (note, this is signed)
0 or 2	flag	optional uint8_t[2]	If present, always 0001, and indicates the presence of witness data
1+	tx_in count	var_int	Number of Transaction inputs (never zero)
41+	tx_in	tx_in[]	A list of 1 or more transaction inputs or sources for coins
1+	tx_out count	var_int	Number of Transaction outputs
9+	tx_out	tx_out[]	A list of 1 or more transaction outputs or destinations for coins
0+	tx_witnesses	tx_witness[]	A list of witnesses, one for each input; omitted if *flag* is omitted above
4	lock_time	uint32_t	The block number or timestamp at which this transaction is unlocked: If all TxIn inputs have final (0xffffffff) sequence numbers then lock_time is irrelevant. Otherwise, the transaction may not be added to a block until after lock_time (see NLockTime).

TxIn 是构成为下列 field。

Field Size	Description	Data type	Comments
36	previous_output	outpoint	The previous output transaction reference, as an OutPoint structure
1+	script length	var_int	The length of the signature script
?	signature script	uchar[]	Computational Script for confirming transaction authorization
4	sequence	uint32_t	Transaction version as defined by the sender. Intended for "replacement" of transactions when information is updated before inclusion into a block.

OutPoint 构造体是构成为下列 field。

Field Size	Description	Data type	Comments
32	hash	char[32]	The hash of the referenced transaction.
4	index	uint32_t	The index of the specific output in the transaction. The first output is 0, etc.

Script 构造是构成为交易值和关联的信息及连算。

TxOut 构造体是构成为下列 field。

Field Size	Description	Data type	Comments
8	value	int64_t	Transaction Value
1+	pk_script length	var_int	Length of the pk_script
?	pk_script	uchar[]	Usually contains the public key as a Factor script setting up conditions to claim this output.

TxWitness 构造是构成为各数据构成要素的 var_int 个数和各构成数据要素的 var_int 长度及原始构成要素信息。

示例 *tx* 短信：

```
000000  F9 BE B4 D9 74 78 00 00   00 00 00 00 00 00 00
00    ....tx.........
000010  02 01 00 00 E2 93 CD BE   01 00 00 00 01 6D BD
DB    .............m..
000020  08 5B 1D 8A F7 51 84 F0   BC 01 FA D5 8D 12 66
E9    .[...Q........f.
000030  B6 3B 50 88 19 90 E4 B4   0D 6A EE 36 29 00 00
00    .;P......j.6)...
000040  00 8B 48 30 45 02 21 00   F3 58 1E 19 72 AE 8A
C7    ..H0E.!..X..r...
000050  C7 36 7A 7A 25 3B C1 13   52 23 AD B9 A4 68 BB
3A    .6zz%;..R#...h.:
000060  59 23 3F 45 BC 57 83 80   02 20 59 AF 01 CA 17 D0    Y#?E.W...
Y.....
000070  0E 41 83 7A 1D 58 E9 7A   A3 1B AE 58 4E DE C2
8D    .A.z.X.z...XN...
000080  35 BD 96 92 36 90 91 3B   AE 9A 01 41 04 9C 02 BF
5...6..;...A....
000090  C9 7E F2 36 CE 6D 8F E5   D9 40 13 C7 21 E9 15
98    .~.6.m...@..!...
0000A0  2A CD 2B 12 B6 5D 9B 7D   59 E2 0A 84 20 05 F8 FC
*.+..].}Y... ...
0000B0  4E 02 53 2E 87 3D 37 B9   6F 09 D6 D4 51 1A DA 8F
N.S..=7.o...Q...
0000C0  14 04 2F 46 61 4A 4C 70   C0 F1 4B EF F5 FF FF
FF    ../FaJLp..K.....
0000D0  FF 02 40 4B 4C 00 00 00   00 00 19 76 A9 14 1A
A0    ..@KL......v....
0000E0  CD 1C BE A6 E7 45 8A 7A   BA D5 12 A9 D9 EA 1A
FB    .....E.z........
0000F0  22 5E 88 AC 80 FA E9 C7   00 00 00 00 19 76 A9 14
"^............v..
000100  0E AB 5B EA 43 6A 04 84   CF AB 12 48 5E FD A0
B7    ..[.Cj.....H^...
000110  8B 4E CC 52 88 AC 00 00   00 00                    .N.R......


Message header:
 F9 BE B4 D9                                        - main network magic
bytes
```

```
 74 78 00 00 00 00 00 00 00 00 00 00               - "tx" command
 02 01 00 00                                       - payload is 258
bytes long
 E2 93 CD BE                                       - payload checksum
(little endian)

Transaction:
 01 00 00 00                                       - version

Inputs:
 01                                                - number of
transaction inputs

Input 1:
 6D BD DB 08 5B 1D 8A F7  51 84 F0 BC 01 FA D5 8D  - previous output
(outpoint)
 12 66 E9 B6 3B 50 88 19  90 E4 B4 0D 6A EE 36 29
 00 00 00 00

 8B                                                - script is 139
bytes long

 48 30 45 02 21 00 F3 58  1E 19 72 AE 8A C7 C7 36  - signature script
(scriptSig)
 7A 7A 25 3B C1 13 52 23  AD B9 A4 68 BB 3A 59 23
 3F 45 BC 57 83 80 02 20  59 AF 01 CA 17 D0 0E 41
 83 7A 1D 58 E9 7A A3 1B  AE 58 4E DE C2 8D 35 BD
 96 92 36 90 91 3B AE 9A  01 41 04 9C 02 BF C9 7E
 F2 36 CE 6D 8F E5 D9 40  13 C7 21 E9 15 98 2A CD
 2B 12 B6 5D 9B 7D 59 E2  0A 84 20 05 F8 FC 4E 02
 53 2E 87 3D 37 B9 6F 09  D6 D4 51 1A DA 8F 14 04
 2F 46 61 4A 4C 70 C0 F1  4B EF F5

 FF FF FF FF                                       - sequence

Outputs:
 02                                                - 2 Output
Transactions

Output 1:
 40 4B 4C 00 00 00 00 00                           - 0.05 BTC (5000000)
 19                                                - pk_script is 25
bytes long
```

```
 76 A9 14 1A A0 CD 1C BE   A6 E7 45 8A 7A BA D5 12   - pk_script
 A9 D9 EA 1A FB 22 5E 88   AC

Output 2:
 80 FA E9 C7 00 00 00 00                            - 33.54 BTC
(3354000000)
 19                                                 - pk_script is 25
bytes long

 76 A9 14 0E AB 5B EA 43   6A 04 84 CF AB 12 48 5E   - pk_script
 FD A0 B7 8B 4E CC 52 88   AC

Locktime:  0
 0 00 00 00                                         - lock time
```

2.10 block

区块短信是传送为在区块散列上要请交易信息的 GetData 短信的应答。

Field Size	Description	Data type	Comments
4	version	int32_t	Block version information (note, this is signed)
32	prev_block	char[32]	The hash value of the previous block this particular block references
32	merkle_root	char[32]	The reference to a Merkle tree collection which is a hash of all transactions related to this block
4	timestamp	uint32_t	A Unix timestamp recording when this block was created (Currently limited to dates before the year 2106!)
4	bits	uint32_t	The calculated difficulty target being used for this block

4	nonce	uint32_t	The nonce used to generate this block… to allow variations of the header and compute different hashes
1+	txn_count	var_int	Number of transaction entries
?	txns	tx[]	Block transactions, in format of "tx" command

这构造的 6 个 field （版本, prev_block, merkle_root, timestamp, bits, nonce 及标准 MX-Algorithm 上计算识别各区块的 MX-Algorithm 散列。在全体区块需要计算散列的话，在 MX- Algorithm 算法只处理第二个 chunk 就可以。

2.11 headers

Header packin 是回应，对 block header 返还 getHeaders 的 packin

Payload:

Field Size	Description	Data type	Comments
1+	count	var_int	Number of block headers
81x?	headers	block_header[]	Block headers

在这 packin 的区块 header 上包含交易数 (var_int, 每 header 会有 81 bytes 以上)

12. getaddr

Getaddr 信息是给节点发送助于在网络上寻找潜在性节点相关的信息要请。对接收该信息的回应是,在已知的活性 pear 数据库中,与一个以上的 pear 一起传送一个以上的地址信息。 一般家庭在过去 3 个小时内发送信息时节点被激活的可能性很高。 在此信息不传送添加信息。

13. mempool

mempool 短信是对还未确认的交易要请信息的节点传送要请。收信这短信的回应是在节点 mempool 中对所有交易包含交易散列的 inv 短信。

14. ping

Ping 短信是为了确认 TCP / IP 连接是否依然有效而传送。传送错误是看作被关闭的连接，地址是切除为现在的 pear。

Payload:

Field Size	Description	Data type	Comments
8	nonce	uint64_t	random nonce

2.15 pong

在协议中使用在 ping 包含的 nonce 形成 *pong* 应答。

Payload:

Field Size	Description	Data type	Comments
8	nonce	uint64_t	nonce from ping

2.16 reject

拒绝短信被拒绝时传送信息。

Payload:

Field Size	Description	Data type	Comments
1+	message	var_str	type of message rejected
1	ccode	char	code relating to rejected message

1+	reason	var_str	text version of reason for rejection
0+	data	char	Optional extra data provided by some errors. Currently, all errors which provide this field fill it with the TXID or block header hash of the object being rejected, so the field is 32 bytes.

CCodes

Value	Name	Description
0x01	REJECT_MALFORMED	
0x10	REJECT_INVALID	
0x11	REJECT_OBSOLETE	
0x12	REJECT_DUPLICATE	
0x40	REJECT_NONSTANDARD	
0x41	REJECT_DUST	
0x42	REJECT_INSUFFICIENTFEE	
0x43	REJECT_CHECKPOINT	

2.17 filterload, filteradd, filterclear, merkleblock

Filterload 是如下定义命令。

Field Size	Description	Data type	Comments
?	filter	uint8_t[]	The filter itself is simply a bit field of arbitrary byte-aligned size. The maximum size is 36,000 bytes.
4	nHashFuncs	uint32_t	The number of hash functions to use in this filter. The maximum value allowed in this field is 50.
4	nTweak	uint32_t	A random value to add to the seed value in the hash function used by the bloom filter.
1	nFlags	uint8_t	A set of flags that control how matched items are added to the filter.

Filterload 是如下定义命令。

Field Size	Description	Data type	Comments
?	data	uint8_t[]	The data element to add to the current filter.

Data Field 是比 520 bytes 大小更小或一样。

Field Size	Description	Data type	Comments
4	version	int32_t	Block version information, based upon the software version creating this block (note, this is signed)
32	prev_block	char[32]	The hash value of the previous block this particular block references
32	merkle_root	char[32]	The reference to a Merkle tree collection which is a hash of all transactions related to this block

4	timestamp	uint32_t	A timestamp recording when this block was created (Limited to 2106!)
4	bits	uint32_t	The calculated difficulty target being used for this block
4	nonce	uint32_t	The nonce used to generate this block… to allow variations of the header and compute different hashes
4	total_transactions	uint32_t	Number of transactions in the block (including unmatched ones)
1+	hash_count	var_int	The number of hashes to follow
32x?	hashes	char[32]	Hashes in depth-first order
1+	flag_bytes	var_int	The size of flags (in bytes) to follow
?	flags	byte[]	Flag bits, packed per 8 in a byte, least significant bit first. Extra 0 bits are padded on to reach full byte size.

2.18 alert

通过网络传送一般通知信息的节点之间。 警告如能通过签名确认,则建议信息要显示给最终用户。 通过客户端的交易(特别是自动化的 Tranjacksion)尝试被终止。 短信文字列的文本需通过登录文件及所有用户界面进行转播。

Alert format:

Field Size	Description	Data type	Comments
?	payload	uchar[]	Serialized alert payload
?	signature	uchar[]	An ECDSA signature of the message

客户是在警告签名上使用公开键。

例如）

```
04fc9702847840aaf195de8442ebecedf5b095cdbb9bc716bda9110971b28a49e0ead85
64ff0db22209e0374782c093bb899692d524e9d6a6956e7c5ecbcd68284
(hash) 1AGRxqDa5WjUKBwHB9XYEjmkv1ucoUUy1s
```

现在 payload 形式如下。

Field Size	Description	Data type	Comments
4	Version	int32_t	Alert format version
8	RelayUntil	int64_t	The timestamp beyond which nodes should stop relaying this alert
8	Expiration	int64_t	The timestamp beyond which this alert is no longer in effect and should be ignored
4	ID	int32_t	A unique ID number for this alert
4	Cancel	int32_t	All alerts with an ID number less than or equal to this number should be cancelled: deleted and not accepted in the future
?	setCancel	set<int32_t>	All alert IDs contained in this set should be cancelled as above
4	MinVer	int32_t	This alert only applies to versions greater than or equal to this version. Other versions should still relay it.
4	MaxVer	int32_t	This alert only applies to versions less than or equal to this version. Other versions should still relay it.
?	setSubVer	set<string>	If this set contains any elements, then only nodes that have their subVer contained in this set are affected by the alert. Other versions should still relay it.
4	Priority	int32_t	Relative priority compared to other alerts

?	Comment	string	A comment on the alert that is not displayed
?	StatusBar	string	The alert message that is displayed to the user
?	Reserved	string	Reserved

```
730100000037664040f00000000b305434f00000000f2030000f10300000010270000048e
e0000
00640000000046536565206269746 36f696e2e6f72672f6665623230206962079 6f752
06861
76652074726f75626c6520636f6e6e656374696e6720616674657220323020466562727
56172
790047304502210083 89df45f0703f39ec8c1cc42c13810ffcae14995bb648340219e35
3b63b
53eb022009ec65e1c1aaeec1fd334c6b684bde2b3f573060d5b70c3a46723326e4e8a4f
1

Version: 1
RelayUntil: 1329620535
Expiration: 1329792435
ID: 1010
Cancel: 1009
setCancel: <empty>
MinVer: 10000
MaxVer: 61000
setSubVer: <empty>
Priority: 100
Comment: <empty>
StatusBar: "See Factor.org/feb20 if you have trouble connecting after
20 February"
Reserved: <empty>
```

2.19 feefilter

payload 通常是 8bytes 的长度,并 incoding feerate 的 64bytes 定数值(LSB/little endian)。 ″收到 feelfilter 信息后,节点是 kilobyte,与 feefilliter 信息中提供的活动相比。 因此 SPV 客户导入 Bloom 过滤器,发送 Pypetter 信息时,只有在两个过滤器全部通过时才转播交易。 在 mempool 信息中生成的 Inv 也成为收费过滤器的对象。

2.20 sendcmpct

1. sendcmpct 信息定义为 1bytes 定数下面跟来的 8bytes 定数。这里 pchCommand = "sendcmpct"。

2. 第一个整数应解释为 Bowl(必须要拥有 1 或 0 值)

3. 第二个整数应解释为 little Endian 版本的号码。 发送 sendcmpct 信息的节点必须设定为 1。

4. 接收第一和第二整数设定为 1 的"sendcmpct"信息时， 节点必须发送 cmpctblock 信息,发布新的区块。

5. 收到第一个定数设定为 0 的"sendcmpct"短信后,节点必须发送 cmpctblock 信息,发布新的区块,但如 BIP130 所定义的一样发送 invs 或 header 来发表新区块。

6. 若收信第二个定数不是 1 而是设定为其它值的‘sendcmpct’短信时， 节点是 pear 未收到短信的一样要处理 pear。

7. cmpctblock 及/或其他信息)。 通过这个方法,在今后的版本中,作为今后版本的手写的一部分,可以重复发送其他版本的 sendcmpct 信息。

8. 节点在发送 sendcmpct 信息之前必须确认协议版本

9. 在从 Pear 收到 sendcmpct 信息之前,节店

不能向 Pear 发送对 MSG_CMPCT_BLOCK 客体的请求。

2.21 cmpctblock

1. cmpctblock 信息是定义为包括一系列的 HeaderAndroids 信息和 pchCommand ==="cmpctblock"信息。

2. 传送 sendcmpct 信息后， 如果接收 cmpctblock 信息， 节点就会计算出简短的交易 ID, 对各个未确认的交易(即自己的 mempool)和 cmpctblock 短信的短交易 ID 进行比较。

3. 寻找可使用的交易后，没有交易的节点需要使用 getblocktxn 短信要请缺漏的交易。

4. 如果节点不能回应 Blocketxn 提出的所有信息， 则节点不能传送 cmpctblock 信息。

5. 节点对模块的各个交易，在不确认 Header 是否正确切割的情况下,不能传送 cmpctblock 信息,必须适当地在有有效作业证明的现有链条的最上面。 节点是验证区块内的各交易是否有效使用原有 UTXO 集合 ntree 之前可发送 cmpctblock。

2.22 getblocktxn

1. getblocktxn 短信是定义为直列化的 BlockTransactionsRequest 短信和包含 pchCommand == "getblocktxn"的短信。

2. 收信适合形式化的 getblocktxnmessage，提供
cmpctblock 的节点需要用适合的 blocktxn 短信来回应。这样的 blocktxn
短信是按照要请顺序正确包含在 getblocktxn index 目录指定的 index 交易。

2.23 blocktxn

1. blocktxn 短信是定义为直列化的 BlockTransactions 短信和包含 pchCommand == "blocktxn"
的短信。
2. 若收到适合形式化的 blocktxn 短信，节点通过下列方法再次构成全体区块。
3. 把交易带到原本 cmpctblock 排列在表示的位置。
4. 原本每个 cmpctblock 短的交易 ID 按顺序在模块或信息或其他源代码中找到相应的交易，
放在模块的第一个可使用的位置。
5. 一旦区块再次构成，考虑短的交易 ID 会冲突，要正确处理，节点是每当发生这样的冲突时不可
发生处罚。

V 结论

下列整理了在本技术书内容中所查看的FACTOR mx区块链的技术。

1)Factor mxblockchein采用主区块的secp256r1方式,开发并搭载了26个以上的新散列算法功能,此后如果出现新技术量子计算机的话，可追加更新。

2)Factor mxblockchain的技术不仅与现有的硬币连接,还可以互换。

3)Factor mx 区块链也是与比特币相同的采掘方式,虽然具有半衰期功能,但高的散列功能的电力消耗量却减少为比特币的100分之1左右。

4) 可以互换pow,pos功能,并设计成可以采掘master node,任何人都可以开采。

5)mx区块链的节点选择方式是获得专利的spread喷射系统,设计成优先顺位节点选择方式，比原有节点选择方式在速度方面更加卓越。

6)在区块链的分散账簿上装载量子力学保安功能，在Oracle区间也具有优秀的保安性。

7)mx 区块链是自体开发的主要区块链，有转换器(dapp)功能,因此可展现许多的扩展性。

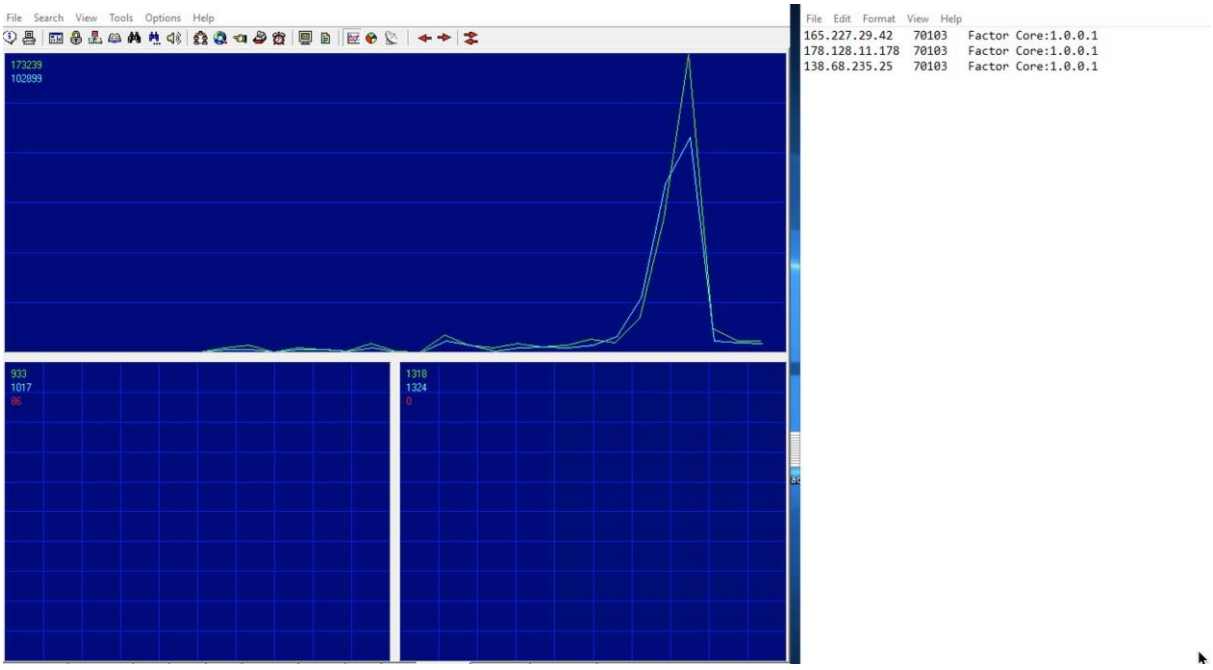
8) mx区块链是利用转换器开发工具,不仅是新开发者,任何人都可以容易接近并使用。

< 试验信息及结果>

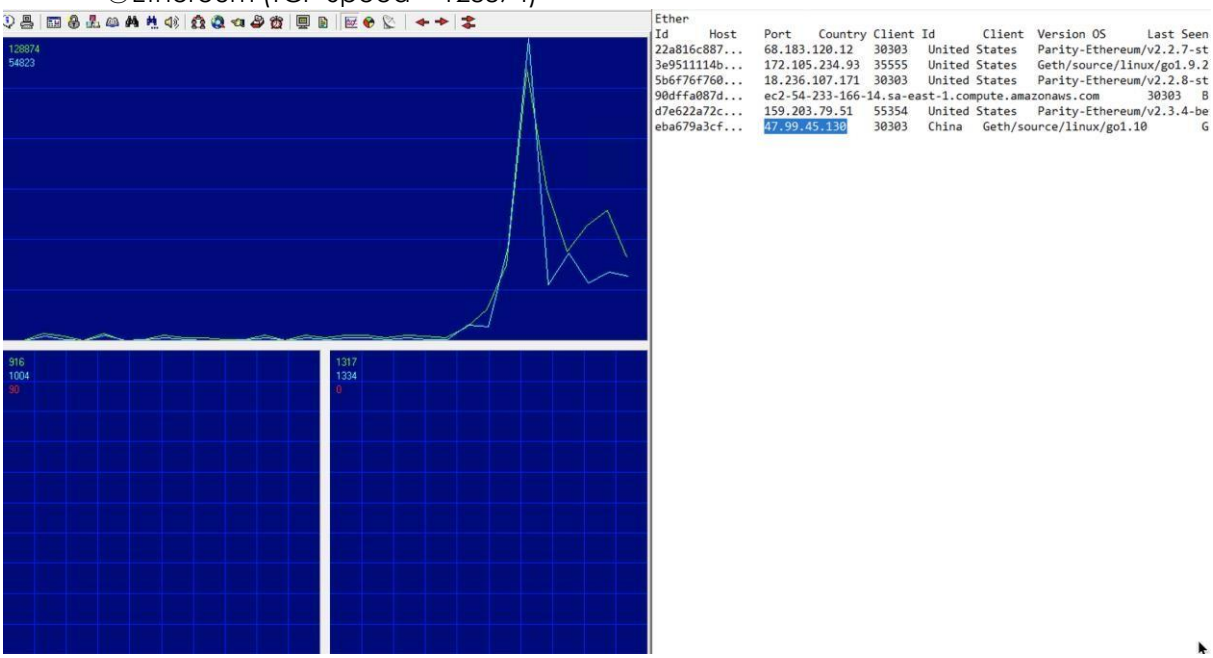
1. 原有区块链的节点速度比较测试

在 Factor(MX) BlockChain 的全体网络中有 3 个节点，假设全体网络的通信量没有过负荷现象时，比较网络 TCP 的速度。结果值算出方法是按照时间测定速度，成为基准的连接 IP-address 是记录在右侧记录栏里。

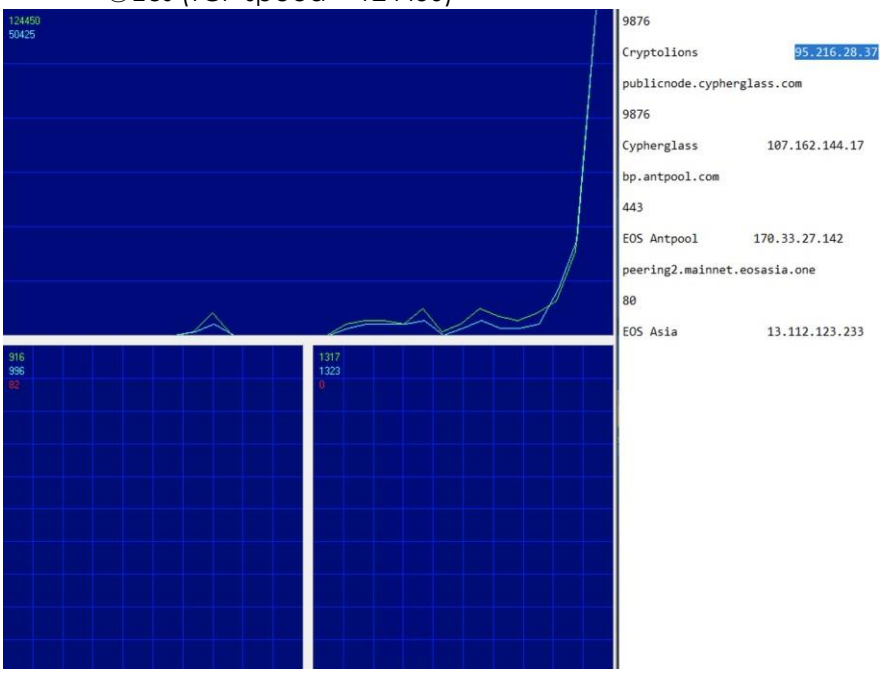
①Factor (TCP Speed = 173239)



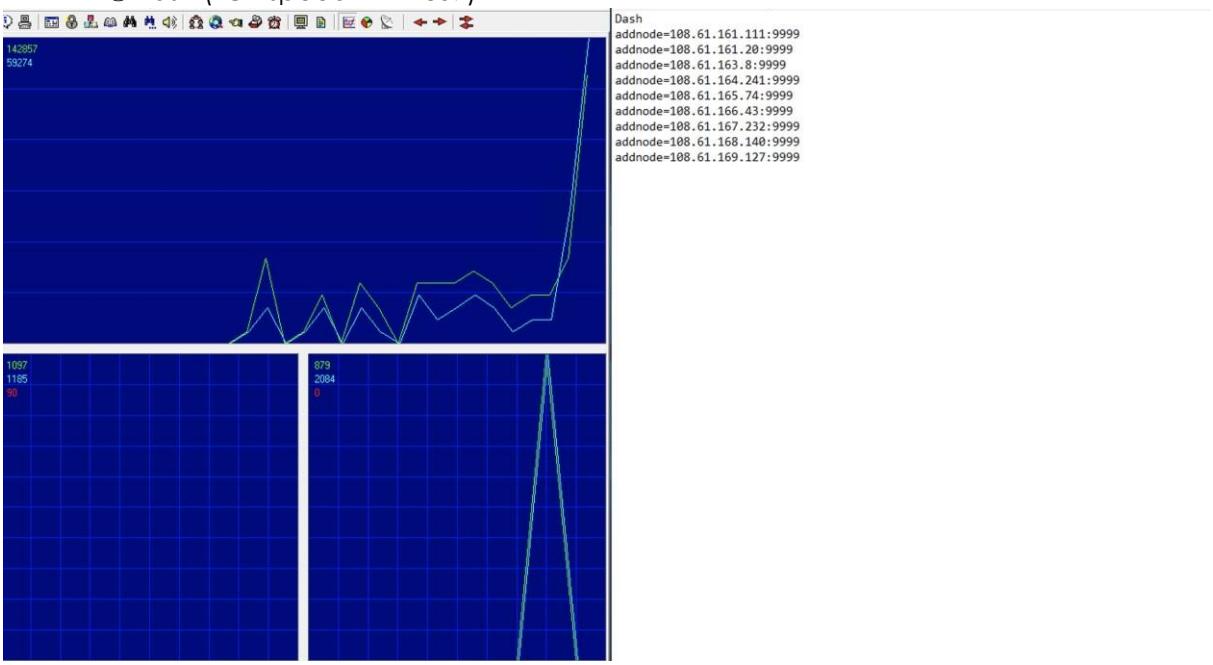
②Ethereum (TCP Speed = 128874)



③Eos (TCP speed = 124450)



④Dash (TCP speed = 142857)



各区块链的节点速度的算出值和速度顺位如下。

Factor (TCP Speed = 173239) = 第一
Dash (TCP speed = 142857) = 第二 Et
hereum (TCP Speed = 128874) 第三 E
os (TCP speed = 124450) =第四

2. 交易速度(Transection Speed)

在 Factor(MX) BlockChain 的全体网络上，假设有三个节点，且没有全体网络的交易量过负荷现象时，交易速度如下算出为 1 秒内的速度。

27th Apr 2019 17:44:29	44afd6c7a4c235363704ae7493c7626fddb4b65ca41ba832c42f1104e5e51515	1019.999958
27th Apr 2019 17:44:29	cb4ad6548a38514ab7968666fcbf5c938965956793c84788cfed29462e8fd367	1019.9999581
27th Apr 2019 17:44:29	157da64ca1d381855ad563e1cb45d5cebb98545f5f1f830e002968f009a101ad	1019.9999581
27th Apr 2019 17:44:29	75c0ba15173f18609aeb799108692e01fe2e13a29633129705b4a74b4584e5c4	1019.9999581
27th Apr 2019 17:44:29	827abdc1f264a3473cd381f14333850374351 44f726a3c9b8e400002a5cf7d6b	1019.999958
27th Apr 2019 17:44:29	7b5699cb76a6b06d69a0808670d8be28c362e963bea58ae1bad6dcdb7f4602e0	1019.9999582
27th Apr 2019 17:44:29	e255da7fdc023b69279098b887f82c9bcbe2a87d45d8c8c66e88f049a4b7e9b4	100019.9977101
27th Apr 2019 17:44:29	327acbf647ec6768eb8f5dc1099e58016ba876ba2c8a18f604e329bc9e349f5a	100019.9977103

<참고문헌>

- 참고문件

Ken Shirriff’s blog https://www.sec
g.org/SEC2-Ver-1.0.pdf Wikipedia
TTM Agency - Medium, Differences between Bitcoin-type and masternode networks
火熊, Tistory
Genier, brunch
查看区块链p.209
etotheipi@gmail.com
Mastering Cryptocurrency 2nd Edition, O’Reillly http://www.righto.com/2014/02/Factor-mining-hard-way-algorithms.html 区块链构造与理论p.120
https://blog-archive.bitgo.com/the-challenges-of-block-chain-indexing

- **Factor(Multi-X) Algorithm 参考代码**

src/compat \ sr
c/config \ src/c
rypto \ src/json
\ src/obj \ src/
primitives \ src/
qt \ src/script \
src/test \ src/uni
value \ src/level
db/db \
src/leveldb/issues \
src/leveldb/port \
src/leveldb/table \ src/le
veldb/util \ src/qt/forms
\ src/qt/locale \ src/qt
/test \ src/secp256k1/inc
lude \ src/secp256k1/src
\ src/test/data \ src/lev
eldb/doc/bench \
src/leveldb/helpers/memenv \
src/leveldb/include/leveldb \
src/leveldb/port/win \ src/se
cp256k1/src/java src/config
\ src/leveldb/port/win \ src/
crypto \
src/primitives \
src/script \
src/obj \

```
src/univalue \
src/json \  src/
qt \  src/qt/fo
rms \ src/com
pat \
src/secp256k1/include \
src/leveldb/helpers/memenv \
src/test/data \
src/test \
src/qt/test \  src/secp256k
1/src \  src/secp256k1/src
/java  src/activemasterno
de.h \ src/addrman.h \  s
rc/alert.h \  src/allocators.
h \  src/amount.h \  src/b
ase58.h \  src/bloom.h \
src/chain.h \
src/chainparams.h \  src/
chainparamsbase.h \  src
/chainparamsseeds.h \ sr
c/checkpoints.h \  src/ch
eckqueue.h \  src/clientv
ersion.h \  src/coincontrol.
h \  src/coins.h  \  src/com
pat.h \  src/compressor.h
\  src/core_io.h \  src/cryp
ter.h \  src/darksend-relay.
h \  src/darksend.h \  src/f
actor-config.h \  src/db.h
\  src/eccryptoverify.h \  s
rc/ecwrapper.h \  src/has
h.h \
src/init.h \  src/
instantx.h \
src/keepass.h \  src/key
.h \  src/keystore.h \  sr
c/leveldbwrapper.h \ s
rc/limitedmap.h \  src/
main.h \
src/masternode-budget.h \  sr
c/masternode-payments.h \ s
rc/masternode-sync.h \  src/
masternode.h \  src/masterno
deconfig.h \  src/masternode
man.h \  src/merkleblock.h \
src/miner.h \
src/mruset.h \
src/net.h \
src/netbase.h \
  rc/noui.h \  src/p
ow.h \  src/protoc
ol.h \  src/pubkey.
h \  src/random.h \
src/rpcclient.h \  sr
c/rpcprotocol.h \ s
rc/rpcserver.h \  sr
c/serialize.h \
```

```
src/spork.h \
```

src/streams.h \ src/
sync.h \ src/thread
safety.h \ src/timed
ata.h \ src/tinyform
at.h \ src/txdb.h \
src/txmempool.h \
src/ui_interface.h \
src/uint256.h \ src/u
ndo.h \ src/util.h \
src/utilmoneystr.h \
src/utilstrencodings.h \
src/utiltime.h \ src/ver
sion.h \ src/wallet.h \
src/wallet_ismine.h \
src/walletdb.h \ src/c
ompat/sanity.h \
src/config/factor-config.h \
src/crypto src/json/js
on_spirit.h \
src/json/json_spirit_error_position.h \
src/json/json_spirit_reader.h \ src/json/js
on_spirit_reader_template.h \ src/json/j
son_spirit_stream_reader.h \ src/json/js
on_spirit_utils.h \ src/json/json_spirit_val
ue.h \ src/json/json_spirit_writer.h \ src
/json/json_spirit_writer_template.h \ src
/obj/build.h \
src/primitives/block.h \ src/p
rimitives/transaction.h \ src/
qt/addressbookpage.h \
src/qt/addresstablemodel.h \ src
/qt/askpassphrasedialog.h \ src/
qt/Factoraddressvalidator.h \ src
/qt/Factoramountfield.h \ src/qt
/Factorgui.h \ src/qt/Factorunits.
h \ src/qt/clientmodel.h \ src/qt
/coincontroldialog.h \ src/qt/coi
ncontroltreewidget.h \ src/qt/cs
vmodelwriter.h \ src/qt/darksend
config.h \ src/qt/editaddressdial
og.h \ src/qt/guiconstants.h \ sr
c/qt/guiutil.h \
src/qt/intro.h \ src/qt/macdockic
onhandler.h \ src/qt/macnotifica
tionhandler.h \
src/qt/networkstyle.h \ src/qt/notific
ator.h \ src/qt/openuridialog.h \ src
/qt/optionsdialog.h \ src/qt/options
model.h \ src/qt/overviewpage.h \
src/qt/paymentrequest.pb.h \ src/qt
/paymentrequestplus.h \ src/qt/pay
mentserver.h \ src/qt/peertablemod
el.h \ src/qt/qvalidatedlineedit.h \ s
rc/qt/qvaluecombobox.h \ src/qt/r
eceivecoinsdialog.h \ src/qt/receiv
erequestdialog.h \ src/qt/recentreq
ueststablemodel.h \ src/qt/rpcconso
le.h \

src/qt/sendcoinsdialog.h \

src/qt/sendcoinsentry.h \ src/qt/si
gnverifymessagedialog.h \ src/qt/s
plashscreen.h \ src/qt/trafficgrap
hwidget.h \ src/qt/transactiondes
c.h \ src/qt/transactiondescdialo
g.h \ src/qt/transactionfilterproxy.
h \ src/qt/transactionrecord.h \ sr
c/qt/transactiontablemodel.h \ sr
c/qt/transactionview.h \ src/qt/uti
litydialog.h \ src/qt/walletframe.h
\ src/qt/walletmodel.h \ src/qt/w
alletmodeltransaction.h \ src/qt/w
alletview.h \ src/qt/winshutdown
monitor.h \ src/script/Factorconse
nsus.h \ src/script/interpreter.h \ sr
c/script/script.h \ src/script/script_
error.h \ src/script/sigcache.h \ sr
c/script/sign.h \ src/script/standar
d.h \ src/test/bignum.h \ src/univ
alue/univalue.h \ src/univalue/uni
value_escapes.h \ src/leveldb/db/
builder.h \ src/leveldb/db/db_imp
l.h \ src/leveldb/db/db_iter.h \ sr
c/leveldb/db/dbformat.h \ src/lev
eldb/db/filename.h \ src/leveldb/
db/log_format.h \ src/leveldb/db/
log_reader.h \ src/leveldb/db/log
_writer.h \ src/leveldb/db/memta
ble.h \ src/leveldb/db/skiplist.h \
src/leveldb/db/snapshot.h \ src/le
veldb/db/table_cache.h \ src/lev
eldb/db/version_edit.h \ src/level
db/db/version_set.h \
src/leveldb/db/write_batch_internal.h \
src/leveldb/port/atomic_pointer.h \ src/
leveldb/port/port.h \ src/leveldb/port/p
ort_example.h \ src/leveldb/port/port_
posix.h \ src/leveldb/port/port_win.h \ s
rc/leveldb/port/thread_annotations.h \
src/leveldb/table/block.h \ src/leveldb/
table/block_builder.h \ src/leveldb/tabl
e/filter_block.h \ src/leveldb/table/form
at.h \ src/leveldb/table/iterator_wrapp
er.h \ src/leveldb/table/merger.h \ src/
leveldb/table/two_level_iterator.h \ src/
leveldb/util/arena.h \ src/leveldb/util/c
oding.h \ src/leveldb/util/crc32c.h \ src
/leveldb/util/hash.h \ src/leveldb/util/his
togram.h \ src/leveldb/util/logging.h \
src/leveldb/util/mutexlock.h \ src/leveld
b/util/posix_logger.h \ src/leveldb/util/r
andom.h \ src/leveldb/util/testharness.h
\ src/leveldb/util/testutil.h \ src/qt/form
s/ui_aboutdialog.h \ src/qt/test/payme
ntrequestdata.h \

src/qt/test/paymentservertests.h \

src/qt/test/uritests.h \ src/secp256k1/include/se
cp256k1.h \ src/secp256k1/src/ecdsa.h \ src/s
ecp256k1/src/ecdsa_impl.h \ src/secp256k1/sr
c/eckey.h \ src/secp256k1/src/eckey_impl.h \
src/secp256k1/src/ecmult.h \ src/secp256k1/sr
c/ecmult_gen.h \ src/secp256k1/src/ecmult_g
en_impl.h \ src/secp256k1/src/ecmult_impl.h \
src/secp256k1/src/field.h \ src/secp256k1/src/fi
eld_10x26.h \ src/secp256k1/src/field_10x26_im
pl.h \ src/secp256k1/src/field_5x52.h \ src/sec
p256k1/src/field_5x52_asm_impl.h \ src/secp25
6k1/src/field_5x52_impl.h \ src/secp256k1/src/fi
eld_5x52_int128_impl.h \ src/secp256k1/src/fiel
d_gmp.h \ src/secp256k1/src/field_gmp_impl.h
\ src/secp256k1/src/field_impl.h \ src/secp256
k1/src/group.h \ src/secp256k1/src/group_impl.
h \ src/secp256k1/src/libsecp256k1-config.h \
src/secp256k1/src/num.h \ src/secp256k1/src/n
um_gmp.h \ src/secp256k1/src/num_gmp_impl
.h \ src/secp256k1/src/num_impl.h \ src/secp2
56k1/src/scalar.h \ src/secp256k1/src/scalar_4x
64.h \ src/secp256k1/src/scalar_4x64_impl.h \ s
rc/secp256k1/src/scalar_8x32.h \ src/secp256k
1/src/scalar_8x32_impl.h \ src/secp256k1/src/sc
alar_impl.h \ src/secp256k1/src/testrand.h \ sr
c/secp256k1/src/testrand_impl.h \ src/secp256
k1/src/util.h \ src/test/data/alertTests.raw.h \ sr
c/test/data/base58_encode_decode.json.h \ s
rc/test/data/base58_keys_invalid.json.h \ src/te
st/data/base58_keys_valid.json.h \ src/test/dat
a/script_invalid.json.h \ src/test/data/script_vali
d.json.h \ src/test/data/sig_canonical.json.h \
src/test/data/sig_noncanonical.json.h \ src/test
/data/sighash.json.h \ src/test/data/tx_invalid.j
son.h \ src/test/data/tx_valid.json.h \ src/level
db/helpers/memenv/memenv.h \ src/leveldb/i
nclude/leveldb/c.h \ src/leveldb/include/level
db/cache.h \ src/leveldb/include/leveldb/co
mparator.h \ src/leveldb/include/leveldb/db.h
\ src/leveldb/include/leveldb/dumpfile.h \ src
/leveldb/include/leveldb/env.h \ src/leveldb/i
nclude/leveldb/filter_policy.h \ src/leveldb/incl
ude/leveldb/iterator.h \ src/leveldb/include/le
veldb/options.h \ src/leveldb/include/leveldb/
slice.h \ src/leveldb/include/leveldb/status.h \
src/leveldb/include/leveldb/table.h \ src/level
db/include/leveldb/table_builder.h \ src/level
db/include/leveldb/write_batch.h \ src/leveld
b/port/win/stdint.h \
src/secp256k1/src/java/org_Factor_NativeSecp256k1.h \
src/crypto/aes_helper.c \ src/q
t/Factoramountfield.moc \ src/
qt/factor.moc \

src/qt/intro.moc \

```
src/qt/overviewpage.moc \ src/qt/rpcc
onsole.moc \ src/secp256k1/src/secp256
k1.c src/qt/forms/addressbookpage.ui \
src/qt/forms/askpassphrasedialog.ui \ src
/qt/forms/coincontroldialog.ui \ src/qt/fo
rms/darksendconfig.ui \ src/qt/forms/edit
addressdialog.ui \ src/qt/forms/helpmess
agedialog.ui \ src/qt/forms/intro.ui \ src/
qt/forms/openuridialog.ui \ src/qt/forms/
optionsdialog.ui \ src/qt/forms/overview
page.ui \ src/qt/forms/receivecoinsdialo
g.ui \ src/qt/forms/receiverequestdialog.
ui \ src/qt/forms/rpcconsole.ui \ src/qt/f
orms/sendcoinsdialog.ui \ src/qt/forms/s
endcoinsentry.ui \ src/qt/forms/signverify
messagedialog.ui \ src/qt/forms/transacti
ondescdialog.ui src/activemasternode.c
pp \ src/addrman.cpp \
src/alert.cpp \
src/allocators.cpp \ src/am
ount.cpp \ src/base58.cpp
\ src/bloom.cpp \ src/chai
n.cpp \ src/chainparams.c
pp \ src/chainparamsbase.
cpp \ src/checkpoints.cpp
\ src/clientversion.cpp \ sr
c/coins.cpp \ src/compress
or.cpp \ src/core_read.cpp
\ src/core_write.cpp \ src/
crypter.cpp \ src/darksend-
relay.cpp \ src/darksend.c
pp \ src/factor-cli.cpp \ src
/factor-tx.cpp \ src/factord.
cpp \ src/db.cpp \ src/ec
cryptoverify.cpp \ src/ecwr
apper.cpp \ src/editaddres
sdialog.cpp \ src/hash.cpp
\
src/init.cpp \ src/instantx.
cpp \ src/keepass.cpp \
src/key.cpp \ src/keystor
e.cpp \ src/leveldbwrap
per.cpp \ src/main.cpp \
src/masternode-budget.cpp \ sr
c/masternode-payments.cpp \ s
rc/masternode-sync.cpp \ src/
masternode.cpp \ src/masterno
deconfig.cpp \ src/masternode
man.cpp \ src/merkleblock.cpp
\ src/miner.cpp \
src/net.cpp \ src/
netbase.cpp \
src/noui.cpp \ src
/pow.cpp \ src/pr
otocol.cpp \
```

```
src/pubkey.cpp \
```

```
src/random.cpp \ src/re
st.cpp \ src/rpcblockch
ain.cpp \ src/rpcclient.c
pp \ src/rpcdump.cpp
\
src/rpcmasternode-budget.cpp \
src/rpcmasternode.cpp \ sr
c/rpcmining.cpp \ src/rpc
misc.cpp \ rc/rpcnet.cpp \
src/rpcprotocol.cpp \ src/rp
crawtransaction.cpp \ src/r
pcserver.cpp \ src/rpcwalle
t.cpp \ src/spork.cpp \ src
/sync.cpp \ src/timedata.c
pp \ src/txdb.cpp \ src/txm
empool.cpp \ src/uint256.c
pp \ src/util.cpp \ src/utilm
oneystr.cpp \ src/utilstrenco
dings.cpp \ src/utiltime.cpp
\ src/wallet.cpp \ src/walle
t_ismine.cpp \ src/walletdb.
cpp \
src/compat/glibc_compat.cpp \ sr
c/compat/glibc_sanity.cpp \ src/c
ompat/glibcxx_compat.cpp \ src/c
ompat/glibcxx_sanity.cpp \ src/co
mpat/strnlen.cpp \ src/json/json_spi
rit_reader.cpp \ src/json/json_spirit_
value.cpp \ src/json/json_spirit_write
r.cpp \ src/primitives/block.cpp \ sr
c/primitives/transaction.cpp \ src/q
t/addressbookpage.cpp \ src/qt/a
ddresstablemodel.cpp \ src/qt/ask
passphrasedialog.cpp \ src/qt/Fact
oraddressvalidator.cpp \ src/qt/Fac
toramountfield.cpp \ src/qt/Factor
gui.cpp \ src/qt/Factorunits.cpp \
src/qt/clientmodel.cpp \ src/qt/coi
ncontroldialog.cpp \ src/qt/coinco
ntroltreewidget.cpp \ src/qt/csvmo
delwriter.cpp \ src/qt/darksendcon
fig.cpp \ src/qt/factor.cpp \ src/qt
/factorstrings.cpp \ src/qt/editaddr
essdialog.cpp \ src/qt/guiutil.cpp \
src/qt/intro.cpp \
src/qt/networkstyle.cpp \ src/qt/
notificator.cpp \ src/qt/openuridi
alog.cpp \ src/qt/optionsdialog.
cpp \ src/qt/optionsmodel.cpp
\ src/qt/overviewpage.cpp \ src
/qt/paymentrequest.pb.cc \ src/
qt/paymentrequestplus.cpp \ src
/qt/paymentserver.cpp \ src/qt/
peertablemodel.cpp \ src/qt/qv
alidatedlineedit.cpp \ src/qt/qv
aluecombobox.cpp \ src/qt/rec
eivecoinsdialog.cpp \
```

```
src/qt/receiverequestdialog.cpp \
```

```
src/qt/recentrequeststablemodel.cpp \
src/qt/rpcconsole.cpp \ src/qt/sendcoi
nsdialog.cpp \ src/qt/sendcoinsentry.c
pp \ src/qt/signverifymessagedialog.c
pp \ src/qt/splashscreen.cpp \ src/qt/
trafficgraphwidget.cpp \ src/qt/transa
ctiondesc.cpp \ src/qt/transactiondes
cdialog.cpp \ src/qt/transactionfilterpr
oxy.cpp \ src/qt/transactionrecord.cp
p \ src/qt/transactiontablemodel.cpp
\ src/qt/transactionview.cpp \ src/qt/
utilitydialog.cpp \ src/qt/walletframe.c
pp \ src/qt/walletmodel.cpp \ src/qt/
walletmodeltransaction.cpp \ src/qt/w
alletview.cpp \ src/qt/winshutdownmo
nitor.cpp \ src/script/Factorconsensus.
cpp \ src/script/interpreter.cpp \ src/s
cript/script.cpp \ src/script/script_error.
cpp \ src/script/sigcache.cpp \ src/sc
ript/sign.cpp \ src/script/standard.cpp
\ src/test/accounting_tests.cpp \ src/t
est/alert_tests.cpp \ src/test/allocator_t
ests.cpp \ src/test/base32_tests.cpp \
src/test/base58_tests.cpp \ src/test/bas
e64_tests.cpp \ src/test/bip32_tests.cp
p \ src/test/bloom_tests.cpp \ src/test/
checkblock_tests.cpp \ src/test/Check
points_tests.cpp \ src/test/coins_tests.c
pp \ src/test/compress_tests.cpp \ src/
test/crypto_tests.cpp \ src/test/DoS_tes
ts.cpp \ src/test/getarg_tests.cpp \ src
/test/hash_tests.cpp \ src/test/key_tests
.cpp \ src/test/main_tests.cpp \ src/te
st/mempool_tests.cpp \ src/test/miner_
tests.cpp \ src/test/mruset_tests.cpp \
src/test/multisig_tests.cpp \ src/test/net
base_tests.cpp \ src/test/pmt_tests.cpp
\ src/test/rpc_tests.cpp \ src/test/rpc_
wallet_tests.cpp \ src/test/sanity_tests.c
pp \ src/test/script_P2SH_tests.cpp \ sr
c/test/script_tests.cpp \ src/test/scriptn
um_tests.cpp \ src/test/serialize_tests.c
pp \ src/test/sighash_tests.cpp \ src/te
st/sigopcount_tests.cpp \ src/test/skiplis
t_tests.cpp \ src/test/test_factor.cpp \
src/test/timedata_tests.cpp \ src/test/tr
ansaction_tests.cpp \ src/test/uint256_t
ests.cpp \ src/test/univalue_tests.cpp \
src/test/util_tests.cpp \ src/test/wallet_t
ests.cpp \ src/univalue/gen.cpp \
```

```
src/univalue/univalue.cpp \ src/univalu
e/univalue_read.cpp \ src/univalue/uni
value_write.cpp \ src/leveldb/db/autoc
ompact_test.cc \ src/leveldb/db/builde
r.cc \ src/leveldb/db/c.cc \ src/leveldb
/db/c_test.c \ src/leveldb/db/corruptio
n_test.cc \ src/leveldb/db/db_bench.c
c \ src/leveldb/db/db_impl.cc \ src/lev
eldb/db/db_iter.cc \ src/leveldb/db/db
_test.cc \ src/leveldb/db/dbformat.cc \
src/leveldb/db/dbformat_test.cc \ src/l
eveldb/db/dumpfile.cc \ src/leveldb/d
b/filename.cc \ src/leveldb/db/filenam
e_test.cc \ src/leveldb/db/leveldb_mai
n.cc \ src/leveldb/db/log_reader.cc \ s
rc/leveldb/db/log_test.cc \ src/leveldb/
db/log_writer.cc \ src/leveldb/db/memt
able.cc \ src/leveldb/db/repair.cc \ sr
c/leveldb/db/skiplist_test.cc \ src/leveld
b/db/table_cache.cc \ src/leveldb/db/
version_edit.cc \ src/leveldb/db/version
_edit_test.cc \ src/leveldb/db/version_s
et.cc \ src/leveldb/db/version_set_test.c
c \ src/leveldb/db/write_batch.cc \ src
/leveldb/db/write_batch_test.cc \ src/le
veldb/issues/issue178_test.cc \ src/level
db/issues/issue200_test.cc \ src/leveldb/
port/port_posix.cc \ src/leveldb/port/po
rt_win.cc \ src/leveldb/table/block.cc \
src/leveldb/table/block_builder.cc \ src
/leveldb/table/filter_block.cc \ src/level
db/table/filter_block_test.cc \ src/leveld
b/table/format.cc \ src/leveldb/table/it
erator.cc \ src/leveldb/table/merger.cc
\ src/leveldb/table/table.cc \ src/level
db/table/table_builder.cc \ src/leveldb
/table/table_test.cc \ src/leveldb/table/
two_level_iterator.cc \ src/leveldb/util/ar
ena.cc \ src/leveldb/util/arena_test.cc
\ src/leveldb/util/bloom.cc \ src/leveld
b/util/bloom_test.cc \ src/leveldb/util/c
ache.cc \ src/leveldb/util/cache_test.c
c \ src/leveldb/util/coding.cc \ src/leve
ldb/util/coding_test.cc \ src/leveldb/util
/comparator.cc \ src/leveldb/util/crc32
c.cc \ src/leveldb/util/crc32c_test.cc \
src/leveldb/util/env.cc \ src/leveldb/util/
env_posix.cc \ src/leveldb/util/env_test.
cc \ src/leveldb/util/env_win.cc \ src/le
veldb/util/filter_policy.cc \ src/leveldb/u
til/hash.cc \ src/leveldb/util/hash_test.c
c \ src/leveldb/util/histogram.cc \ src/l
eveldb/util/logging.cc \ src/leveldb/util/
options.cc \ src/leveldb/util/status.cc \
```

```
src/leveldb/util/testharness.cc \ src/leveldb/util/t
estutil.cc \ src/qt/test/paymentservertests.cpp \
src/qt/test/test_main.cpp \ src/qt/test/uritests.cp
p \ src/secp256k1/src/bench_inv.c \ src/secp25
6k1/src/bench_sign.c \ src/secp256k1/src/benc
h_verify.c \ src/secp256k1/src/secp256k1.c \ src
/secp256k1/src/tests.c \ src/leveldb/doc/bench
/db_bench_sqlite3.cc \ src/leveldb/doc/bench/
db_bench_tree_db.cc \ src/leveldb/helpers/me
menv/memenv.cc \ src/leveldb/helpers/meme
nv/memenv_test.cc \
src/secp256k1/src/java/org_Factor_NativeSecp256k1.c
src/qt/factor.qrc src/qt/factor_locale.qrc  src/qt/locale
/factor_bg.ts \
src/qt/locale/factor_de.ts \ src
/qt/locale/factor_en.ts \ src/qt
/locale/factor_es.ts \ src/qt/lo
cale/factor_fi.ts \ src/qt/locale
/factor_fr.ts \ src/qt/locale/fac
tor_it.ts \ src/qt/locale/factor_j
a.ts \ src/qt/locale/factor_pl.ts
\ src/qt/locale/factor_pt.ts \ s
rc/qt/locale/factor_ru.ts \ src/
qt/locale/factor_sk.ts \ src/qt/l
ocale/factor_sv.ts \ src/qt/loc
ale/factor_vi.ts \ src/qt/locale/
factor_zh_CN.ts \
src/qt/locale/factor_zh_TW.ts
```

免责条款

在本技术说明书提出的内容仅以提供信息为目的,本文件不得依赖陈述。我们对不承担在本技术说明书明示的所有信息所发生的任何法律责任。尤其是在本说明书明示的‘开发技术’是会有所变更，而且与硬币性能及收益等没有任何关联。在本技术书上的信息是没有承认的规制机关。 因此,根据管制要求或管辖规则,不得接受任何必要的法律措施。 在本技术书的发行,分发或推广方面,准据法和限制条件也不受规则约束。在技术书中的信息会加以变更。 Factor Flock 锁链不断研究,因此此后在"修改版本"部分上可确认变更事项。