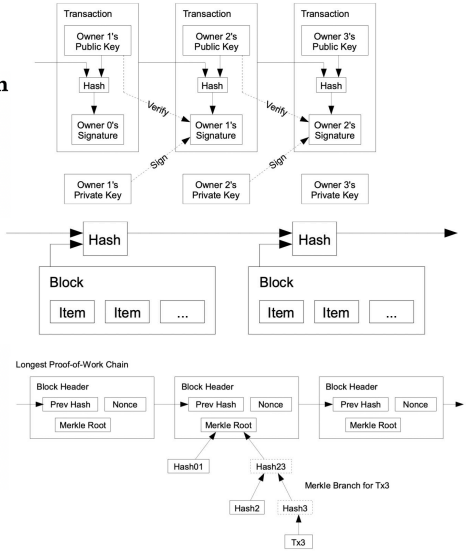
Bitcoin – Basics, Fights and Evolution

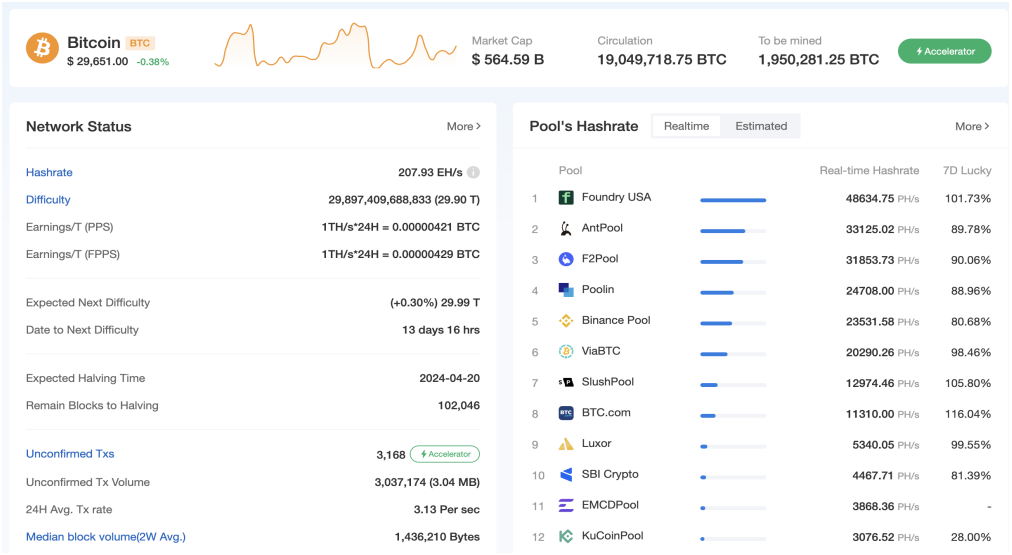
Long Wen
longwen6@gmail.com
20250714 @ Qingdao, SDU

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org



Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

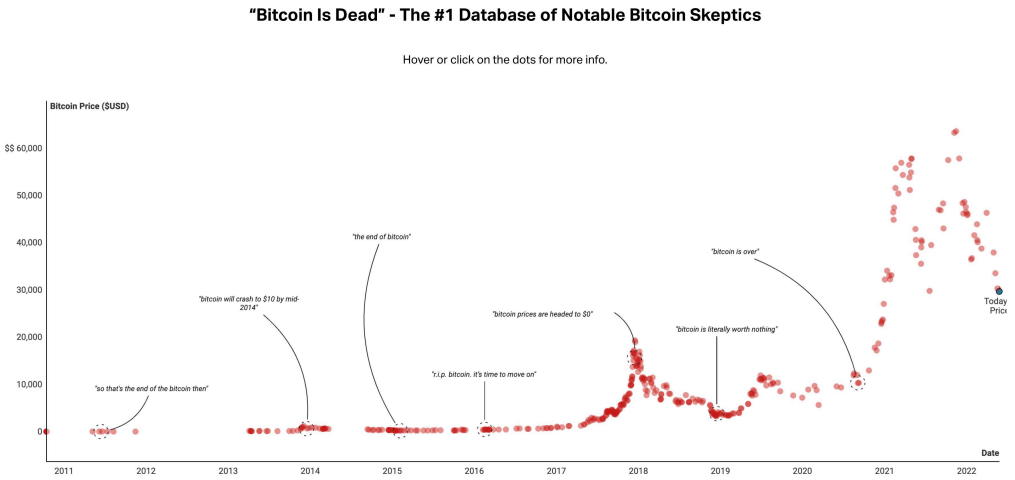


FIRST INTERNATIONAL HOTEL GROUP IN THE WORLD TO ACCEPT CRYPTOCURRENCY FOR HOTEL BOOKINGS

Published July 2, 2021

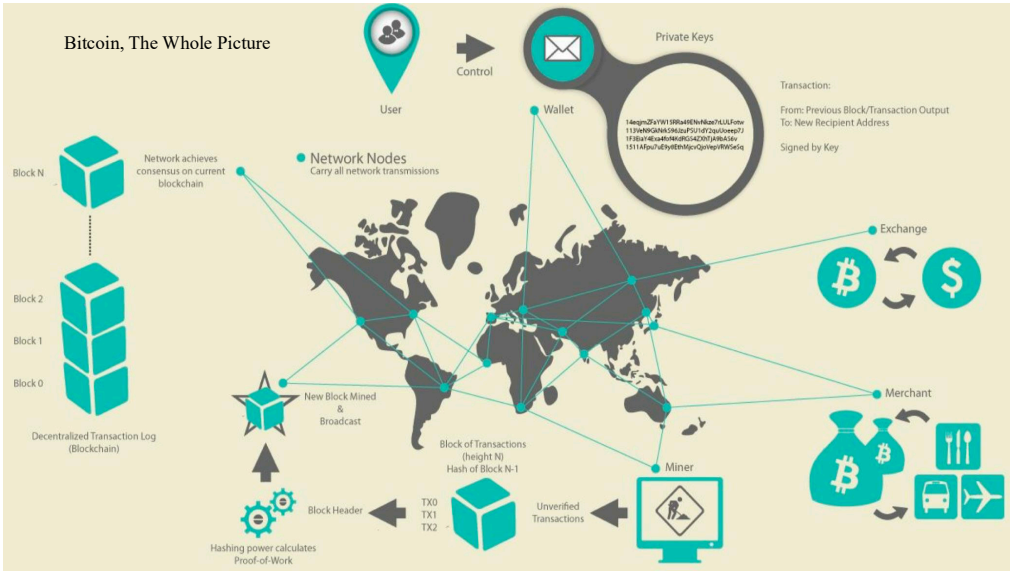
In the ever-evolving world of Crypto, The Pavilions Hotels & Resorts enhances its guest experiences by becoming the first global boutique hotel group to accept Cryptocurrency from any country for hotel bookings, leading the tourism industry into the future of secure online payments.

The Pavilions Hotels & Resorts has partnered with leading global Crypto-payment gateway, Coindirect, enabling customers to pay with Bitcoin, Ethereum and 40 other virtual currencies confidently and securely, 24/7.

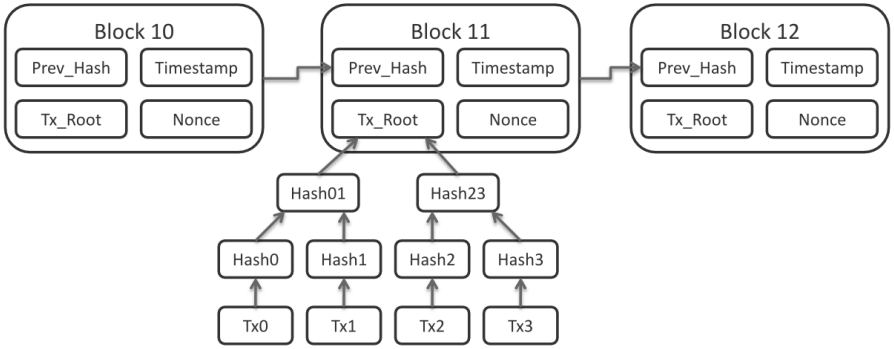


Bitcoin, The Basics

- Overview: The Whole Ecosystem
- Transactions in UTXO Model, The Flesh
- Authenticated Data Structure, The Backbone
- Bitcoin Script Engine, The Property Right
- Proof-of-Work Mining, The Shield
- Nakamoto Consensus, The Mind
- Basic Architecture of Bitcoin Client

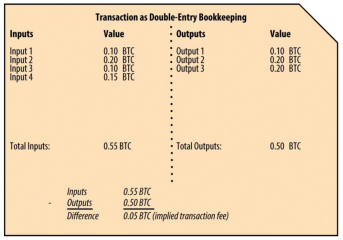


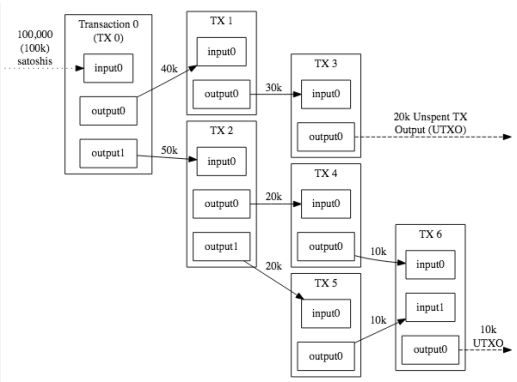
Authenticated Data Structure, The Backbone – aka “BLOCKCHAIN”



Transactions in UTXO Model, The Flesh

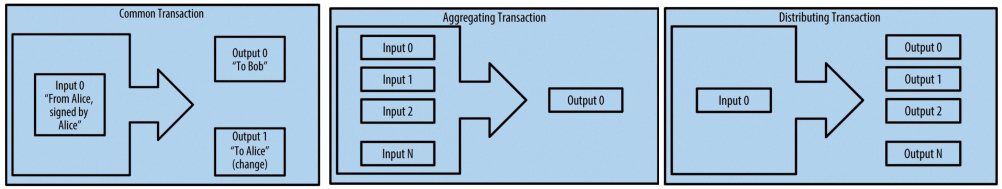
- There are no “accounts” in Bitcoin
- Transactions destroy old “coins”
- Transactions create new “coins”
- Sum of new coins ≤ Sum of old coins
- outputs ≤ inputs
- Diff is regarded as transaction fee





Triple-Entry Bookkeeping (Transaction-To-Transaction Payments) As Used By Bitcoin

Transactions in UTXO Model, The Flesh



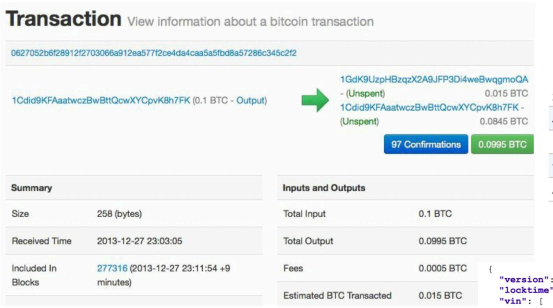


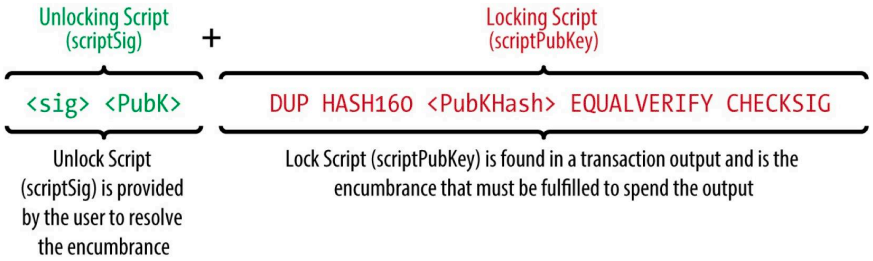
Figure 6-1. Alice’s transaction to Bob’s Cafe

Size	Field	Description
8 bytes (little-endian)	Amount	Bitcoin value in satoshis (10^{-8} bitcoin)
1–9 bytes (VarInt)	Locking-Script Size	Locking-Script length in bytes, to follow
Variable	Locking-Script	A script defining the conditions needed to spend the output

32 bytes	Transaction Hash	Pointer to the transaction containing the UTXO to be spent
4 bytes	Output Index	The index number of the UTXO to be spent; first one is 0
1–9 bytes (VarInt)	Unlocking-Script Size	Unlocking-Script length in bytes, to follow
Variable	Unlocking-Script	A script that fulfills the conditions of the UTXO locking script
4 bytes	Sequence Number	Used for locktime or disabled (0xFFFFFFFF)

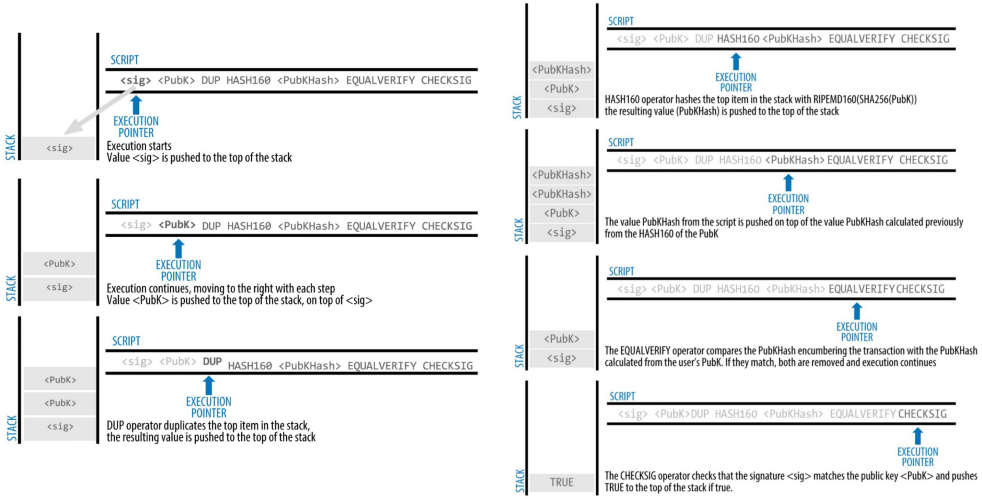

```
{
  "version": 1,
  "locktime": 0,
  "vin": [
    {
      "txid": "7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18",
      "vout": 0,
      "scriptSig" :
"3045022100884d142d86652a3f47ba4746ec719bbfbd040a570b1deccbb6498c75c4ae24cb02204b9f039f
f08df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e3813[ALL]
0484ecc0d46f1918b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376789d172787ec
3457eee41c04f4938de5cc17b4a10fa336a8d752adf",
      "sequence": 4294967295
    }
  ],
  "vout": [
    {
      "value": 0.01500000,
      "scriptPubKey": "OP_DUP OP_HASH160 ab68025513c3dbd2f7b92a94e0581f5d50f654e7
OP_EQUALVERIFY OP_CHECKSIG"
    },
    {
      "value": 0.08450000,
      "scriptPubKey": "OP_DUP OP_HASH160 7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a8
OP_EQUALVERIFY OP_CHECKSIG",
    }
  ]
}
```

Bitcoin Script Engine, The Property Right



Pay-to-Public-Key-Hash (P2PKH)

Stack Based Execution of Bitcoin Script Engine



More on Bitcoin Script, MultiSignature

- Locking script setting an M-of-N multisignature condition
  `M <Public Key 1> <Public Key 2> ... <Public Key N> N CHECKMULTISIG`
  - Demo locking script of 2-of-3 multisignature condition
  - Locking script: `2 <Public Key A> <Public Key B> <Public Key C> 3 CHECKMULTISIG`
  - Unlocking script: `<Signature B> <Signature C>`
  - Combined: `<Signature B> <Signature C> 2 <Public Key A> <Public Key B> <Public Key C> 3 CHECKMULTISIG`
- A bug in CHECKMULTISIG execution (Bitcoin client is also buggy)
  - Opcode CHECKMULTISIG should pop N then N public keys, then M and then M signatures
  - Yet, the impl would pop M+1 signatures
  - Pop on empty stack would cause stack error and script failure (marking the tx as invalid)
  - The bug becomes part of consensus rules and cannot be changed easily
  - Work around: use `0 <Signature B> <Signature C>` as unlocking script

More on Bitcoin Script, Pay-to-Script-Hash (P2SH)

- P2SH: pay to a script matching this hash, a script that will be presented later when this output is spent."
- Complex scripts are replaced by shorter fingerprints in the transaction output, making the transaction smaller.
- Scripts can be coded as an address, so the sender and the sender's wallet don't need complex engineering to implement P2SH.
- P2SH shifts the burden of constructing the script to the recipient, not the sender.
- P2SH shifts the burden in data storage for the long script from the output (which is in the UTXO set) to the input (stored on the blockchain).
- P2SH shifts the burden in data storage for the long script from the present time (payment) to a future time (when it is spent).
- P2SH shifts the transaction fee cost of a long script from the sender to the recipient, who has to include the long redeem script to spend it.

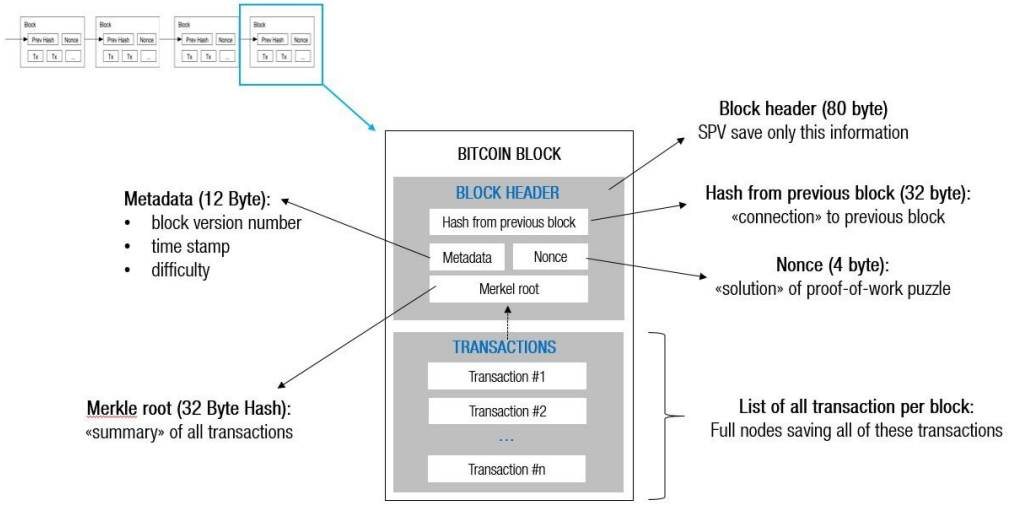
Table 7-1. Complex script without P2SH

Locking Script	2 PubKey1 PubKey2 PubKey3 PubKey4 PubKey5 5 CHECKMULTISIG
Unlocking Script	Sig1 Sig2

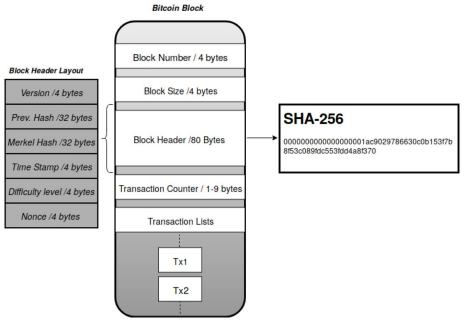
Table 7-2. Complex script as P2SH

Redeem Script	2 PubKey1 PubKey2 PubKey3 PubKey4 PubKey5 5 CHECKMULTISIG
Locking Script	HASH160 <20-byte hash of redeem script> EQUAL
Unlocking Script	Sig1 Sig2 <redeem script>

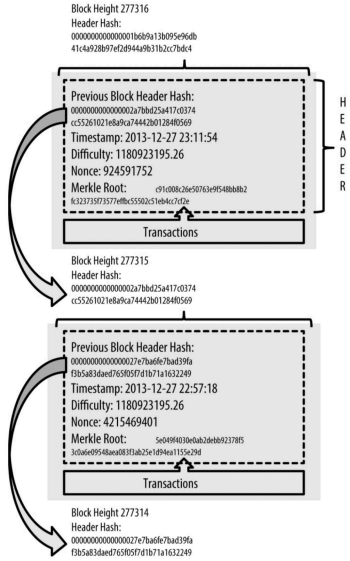
Bitcoin Block



Bitcoin Block



Size	Field	Description
4 bytes	Block Size	The size of the block, in bytes, following this field
80 bytes	Block Header	Several fields form the block header
1–9 bytes (VarInt)	Transaction Counter	How many transactions follow
Variable	Transactions	The transactions recorded in this block

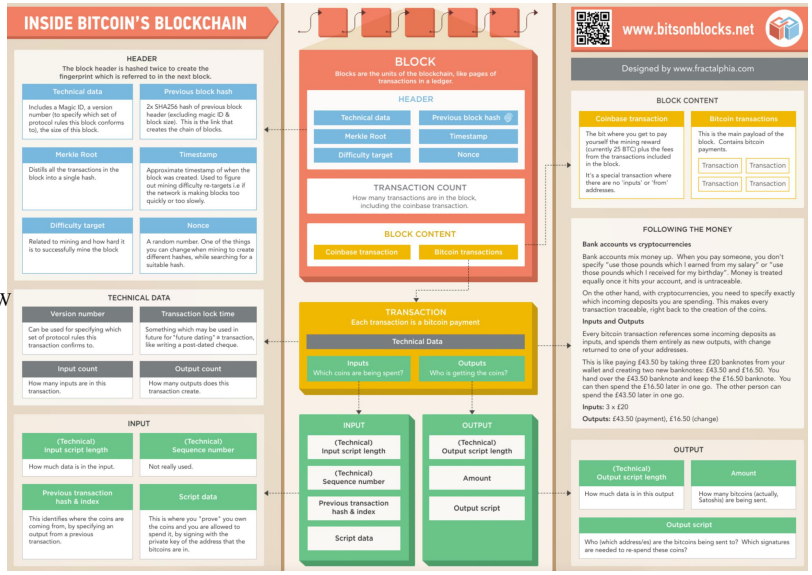


Put All Things Together

Blockchain
Block
Tx / Coinbase tx
Input
Output
Locking Script
Unlocking Script

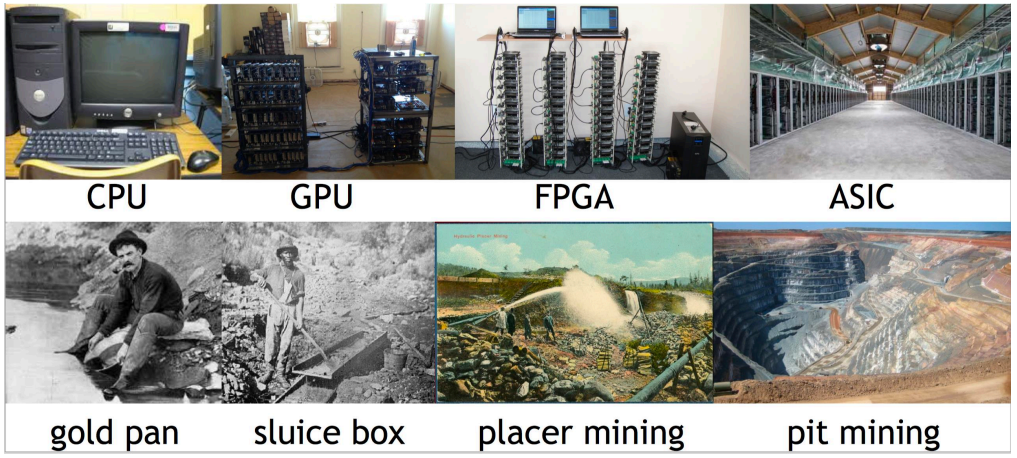
And now we can talk PoW

*Project: send a tx on Bitcoin testnet, and parse the tx data down to every bit, better write script yourself



Proof-of-Work Mining, The Shield

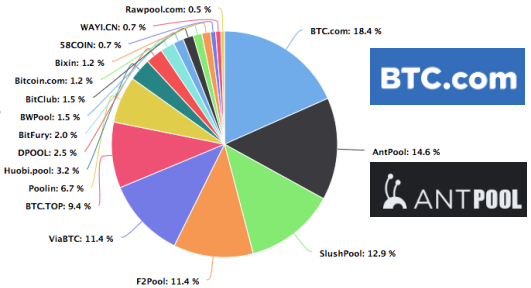
From Joseph Bonneau's "Mining, network, economics" 密码货币与区块链技术高端培训-SJTU-2017



Proof-of-Work Mining, The Shield

But, how hard

- BTC block height, 545443, 2018-10-12, mined by BTC.com
- 256-bit target
- 0000000000000000000169d3b1f409f7257663ebd0c44ab6fd2482a1d498e63b8
- 75 leading zero bits
- Number of hashes tried per 10 min
- 2^{75} = 37,778,931,862,957,161,709,568
- Network hash rate: 51.16 EH/s
- 蚂蚁矿机S9 Hydro,18TH/s,



Proof-of-Work Mining, The Shield

Date of data, 2018-10-12

KH/s：1KH/s=1000H/s 通常 K=千的意思，每秒1,000次哈希
MH/s：1MH/s=1000KH/s M是兆，1M=1000千，1MH/s=每秒1,000,000次哈希
GH/s：1GH/s =1000MH/s 每秒1,000,000,000次哈希。
TH/s：1 TH/s =1000GH/s 每秒1,000,000,000,000次哈希。
PH/s：1 PH/s =1000TH/s 每秒1,000,000,000,000,000次哈希。
EH/s：1 EH/s =1000PH/s 每秒1,000,000,000,000,000,000次哈希。

- Network hash rate: 51.16 EH/s
- 蚂蚁矿机S9 Hydro,18TH/s, 5300￥

Bitcoin Supply and Mining Reward Distribution Via CoinBase Tx

```
$ bitcoin-cli getrawtransaction
d5ada064c6417ca25c4308bd158c34b77e1c0eca2a73cda16c737e7424afba2f 1
{
    "hex" :
"0100000001000000000000000000000000000000000000000000000000000000000000000000ffffffff0f0344
3b0403858402062f503253482fffffffff0110c08d95000000000232102aa970c592640d19de03ff6f329d6fd2
eecb023263b9ba5d1b81c29b523da8b21ac00000000",
    "txid" : "d5ada064c6417ca25c4308bd158c34b77e1c0eca2a73cda16c737e7424afba2f",
    "version" : 1,
    "locktime" : 0,
    "vin" : [
        {
            "coinbase" : "03443b0403858402062f503253482f",
            "sequence" : 4294967295
        }
    ],
    "vout" : [
        {
            "value" : 25.09094928,
            "n" : 0,
            "scriptPubKey" : {
                "asm" :
"02aa970c592640d19de03ff6f329d6fd2eecb023263b9ba5d1b81c29b523da8b210P_CHECKSIG",
                "hex" :
"2102aa970c592640d19de03ff6f329d6fd2eecb023263b9ba5d1b81c29b523da8b21ac",
                "reqSigs" : 1,
                "type" : "pubkey",
                "addresses" : [
                    "1MxTkeEP2PmHSMze5tUZ1hAV3YTKu2Gh1N"
                ]
            }
        }
    ]
}
```

More on Bitcoin's Mining Reward Implementation

```
1053  int64 static GetBlockValue(int nHeight, int64 nFees)
1054  {
1055      int64 nSubsidy = 50 * COIN;
1056
1057      // Subsidy is cut in half every 210000 blocks, which will occur approximately every 4 years
1058      nSubsidy >>= (nHeight / 210000);
1059
1060      return nSubsidy + nFees;
1061  }
```

Do you believe CODE IS LAW ?

5.8 Shift operators [expr.shift]

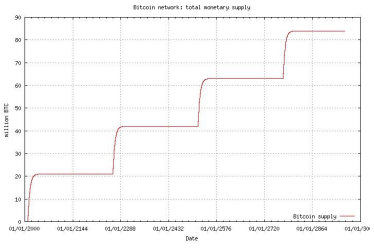
1 The shift operators << and >> group left-to-right.

shift-expression:
additive-expression
shift-expression << *additive-expression*
shift-expression >> *additive-expression*

The operands shall be of integral or unscoped enumeration type and integral promotions are performed. The type of the result is that of the promoted left operand. The behavior is undefined if the right operand is negative, or greater than or equal to the length in bits of the promoted left operand.

2 The value of E1 << E2 is E1 left-shifted E2 bit positions; vacated bits are zero-filled. If E1 has an unsigned type, the value of the result is $E1 \times 2^{E2}$, reduced modulo one more than the maximum value representable in the result type. Otherwise, if E1 has a signed type and non-negative value, and $E1 \times 2^{E2}$ is representable in the result type, then that is the resulting value; otherwise, the behavior is undefined.

3 The value of E1 >> E2 is E1 right-shifted E2 bit positions. If E1 has an unsigned type or if E1 has a signed type and a non-negative value, the value of the result is the integral part of the quotient of $E1/2^{E2}$. If E1 has a signed type and a negative value, the resulting value is implementation-defined.



Bug In Bitcoin's Mining Reward Design or Not ?

Fix for GetBlockValue() after block 13,440,000 #3842

https://github.com/bitcoin/bitcoin/pull/3842/commits/c5a9d2ca9e3234db9687c8cbec4b5b93ec161190

Merged gavinandresen merged 2 commits into bitcoin:master from ditto-b:master on Apr 3, 2014

Conversation 21 Commits 2 Checks 0 Files changed 2 +7 −2

Changes from 1 commit File filter Conversations Jump to Review changes

Fix for GetBlockValue() after block 13,440,000

Forces the block reward to zero when right shift in GetBlockValue() is undefined, after 64 reward halvings (block height 13,440,000).

master (#4161)
v23.0 v0.9-final

ditto-b committed on Mar 11, 2014 commit c5a9d2ca9e3234db9687c8cbec4b5b93ec161190

7 src/main.cpp

```
@@ -1176,9 +1176,14 @@ void static PruneOrphanBlocks()
 int64_t GetBlockValue(int nHeight, int64_t nFees)
 {
     int64_t nSubsidy = 50 * COIN;
+    int halvings = nHeight / Params().SubsidyHalvingInterval();
+
+    // Force block reward to zero when right shift is undefined.
+    if (halvings >= 64)
+        return nFees;

     // Subsidy is cut in half every 210,000 blocks which will occur approximately every 4 years.
-    nSubsidy >>= (nHeight / Params().SubsidyHalvingInterval());
+    nSubsidy >>= halvings;

     return nSubsidy + nFees;
 }
```

Mining and Forks

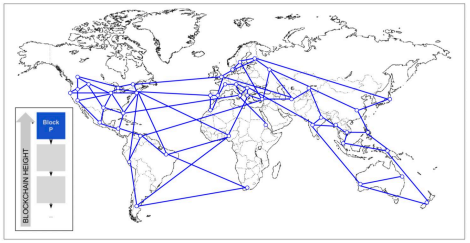


Figure 8-2. Visualization of a blockchain fork event—before the fork

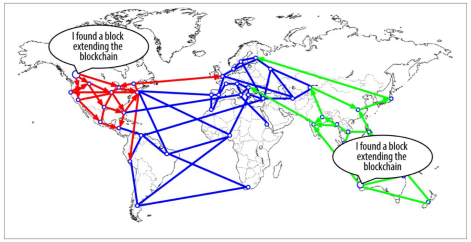


Figure 8-3. Visualization of a blockchain fork event: two blocks found simultaneously

Mining and Forks

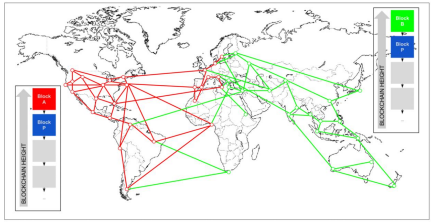


Figure 8-4. Visualization of a blockchain fork event: two blocks propagate, splitting the network

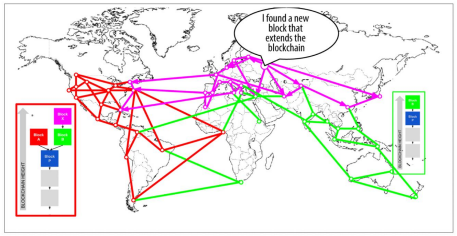


Figure 8-5. Visualization of a blockchain fork event: a new block extends one fork

Mining and Forks and Converge, Nakamoto's Consensus

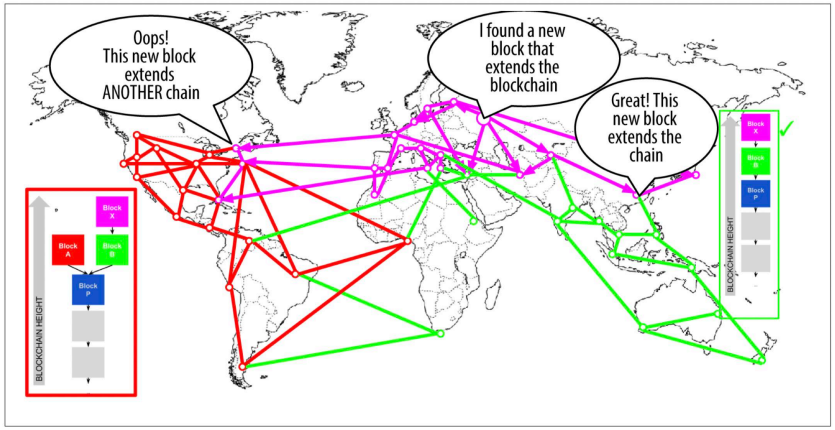
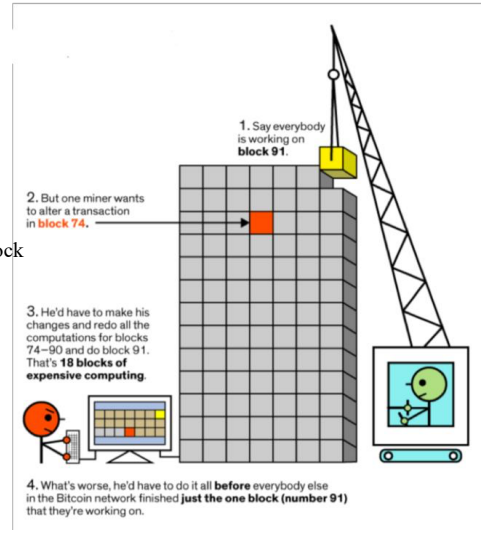


Figure 8-6. Visualization of a blockchain fork event: the network reconverges on a new longest chain

Nakamoto Consensus, The Mind

- New transactions are broadcasted to all nodes
- Each node collects new transactions into a block
- Each round, a random node gets to broadcast its block
- Other nodes accept the block if valid (PoW)
- Node express acceptance of block
- By including its hash in the next block they create
- Incentive for miners: constructing new block
- the only way to forge new coins



ECDSA, The Evil Cornerstone of Blockchain

- Key Gen: $P = dG$, n is order
- Sign(m)
  - $k \leftarrow Z_n^*, R = kG$
  - $r = R_x \bmod n, r \neq 0$
  - $e = hash(m)$
  - $s = k^{-1}(e + dr) \bmod n$
  - Signature is (r, s)
- Verify (r, s) of m with P
  - $e = hash(m)$
  - $w = s^{-1} \bmod n$
  - $(r', s') = e \cdot wG + r \cdot wP$
  - Check if $r' == r$
  - Holds for correct sig since
  - $es^{-1}G + rs^{-1}P = s^{-1}(eG + rP) =$
  - $k(e + dr)^{-1}(e + dr)G = kG = R$

1. Leaking k leads to leaking of d
2. Reusing k leads to leaking of d
3. Two users, using k leads to leaking of d, that is they can deduce each other's d
4. Malleability of ECDSA, e.g. (r, s) and $(r, -s)$ are both valid signatures, lead to blockchain network split
5. Ambiguity of DER encode could lead to blockchain network split
6. One can forge signature if the verification does not check m
7. Same d and k used in ECDSA & Schnorr signature, leads to leaking of d

Malleability of ECDSA

While it might be fine in other circumstances, malleability is a serious problem in blockchain world

- Malleability: modify a valid sig to produce another valid sig without knowing the private key
- ECDSA malleability: If (r, s) is a valid ECDSA signature, then is $(r, -s)$
  - Verification on (r, s) :
    - $e \cdot s^{-1}G + r \cdot s^{-1}P = (x', y'), r = x' mod\ p$
  - Verification on $(r, -s)$ also succeed:
    - $e \cdot (-s)^{-1}G + r \cdot (-s)^{-1}P = -(e \cdot s^{-1}G + r \cdot s^{-1}P) = (x', -y'),\ r = x' mod\ p$
- DER encoding related malleability:
  - Correct encoding of ECDSA sig:
    - 0x30 [total-length] 0x02 [R-length] [R] 0x02 [S-length] [S] [sighash-type]
    - R value: cannot start with any 0x00 bytes, unless the first byte that follows is 0x80 or higher
  - DER variant of encoding, such as R value is prepended 0x00 byte, may also pass the verification
- Defense with BIP62
  - Limiting the value of s, require $1 \leq s \leq n/2$
  - Exhaustively check the DER encoding of ECDSA signature, why DER?

Why

ECDSA – Forge signature when the signed message is not checked

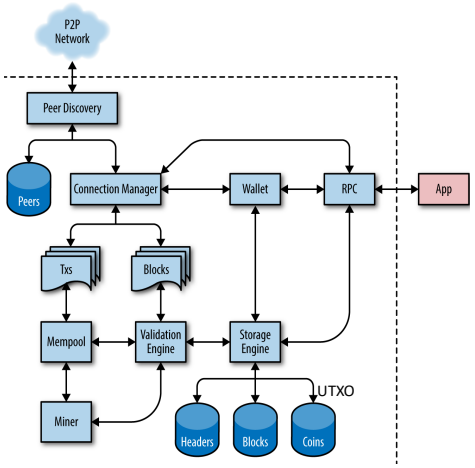
- Key Gen: $P = dG$, n is order
- Sign(m)
  - $k \leftarrow Z_n^*, R = kG$
  - $r = R_x \bmod n, r \neq 0$
  - $e = hash(m)$
  - $s = k^{-1}(e + dr) \bmod n$
  - Signature is (r, s)
- Verify (r, s) of m with P
  - $e = hash(m)$
  - $w = s^{-1} \bmod n$
  - $(r', s') = e \cdot wG + r \cdot wP$
  - Check if $r' == r$
  - Holds for correct sig since
  - $es^{-1}G + rs^{-1}P = s^{-1}(eG + rP) =$
  - $k(e + dr)^{-1}(e + dr)G = kG = R$

- $\sigma = (r, s)$ is valid signature of m with secret key d
- If only the hash of the signed message is required
- Then anyone can forge signature $\sigma' = (r', s')$ for d
- (Anyone can pretend to be someone else)
- Ecdsa verification is to verify:
- $s^{-1}(eG + rP) = (x', y') = R', r' = x' mod\ n == r$?
- To forge, choose $u, v \in \mathbb{F}_n^*$
- Compute $R' = (x', y') = uG + vP$
- Choose $r' = x' \bmod n$, to pass verification, we need
- $s'^{-1}(e'G + r'P) = uG + vP$
  - $s'^{-1}e' = u \bmod n \rightarrow e' = r'uv^{-1} mod\ n$
  - $s'^{-1}r' = v \bmod n \rightarrow s' = r'v^{-1} \bmod n$
- $\sigma' = (r', s')$ is a valid signature of e' with secret key d

*Project: forge a signature to pretend that you are Satoshi

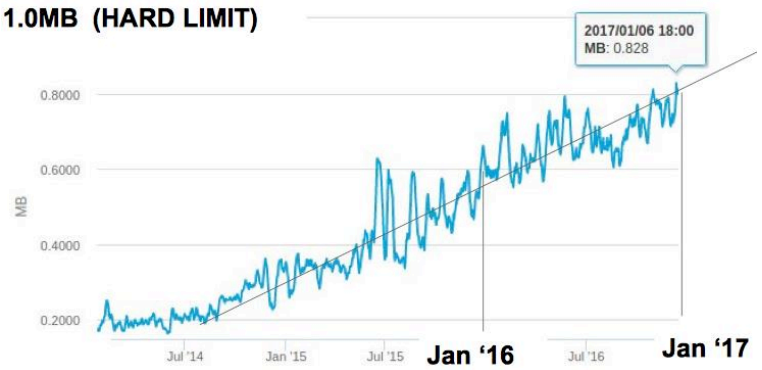
Engineering Bitcoin, How?

- Peer discovery to find network nodes
- Connection manager
  - Process all P2P & RPC commands
- Mempool holds all txs to be included
  - Unconfirmed txs received from network
- UTXO: unspent transaction output
  - All spendable coins
- Blocks: all blocks of the chain
- Headers: metadata of all known blocks



The 1MB Block Size Is Not Enough

Bitcoin 3~4 TPS with 1MB block size



The 1MB Block Size Limit & Proposals Raising Block Size

The natural way

- There was no 1MB limit in Satoshi's first cut of the Bitcoin code
- 2010-07-14 A 'max_block_size' of 1MB was set on the bitcoin client
- 2010-10-24 1MB default block size imposed to reduce potential DoS attack
  - Originally Hal Finney's idea, agreed by Satoshi and user 'Cryddit' on Bitcointalk
  - To remove limit, Satoshi wrote: if (blocknumber > 115000) maxblocksize=largerlimit
- 2011-06-14 Gavin Andresen presented Bitcoin to CIA
  - May have precipitated Bitcoin's creator, Satoshi Nakamoto, going dark
  - The block size limit has no authoritative decision-maker without Satoshi
- 2015-05-08 Gavin Andreesen: resize to 20MB, personal blog "UTXO uh-oh..."
- 2015-06-11 BIP100 by Garzik et al. "Dynamic maximum block size by miner vote", 5% max
- 2015-06-12 Chinese miners signed a statement that they want 8MB blocks
- 2015-06-22 BIP101 by Andresen, "Increase maximum block size", double every 2y, start with 8MB
- **2015-12-21BIP141 by Lombrozo et al. "Segregated Witness (Consensus layer)"**
- 2016-06-04 Andresen rejected the idea of set an explicit and unchangeable limit on block size


五家矿池对于"为何扩容到 8MB，而不是 20MB？"问题考虑到以下因素：

1. 中国大陆地区网络（带宽）基础设施并没有达到欧美发达地区的先进水平；
2. 中国大陆出口带宽有限，与美国、欧洲的通信延时较大；
3. 国内矿池系统的架构仍然需要为 8（20）MB 以上的区块做出调整和适应。

综上所述，较大的区块大小势必导致区块孤立概率大幅上升，更容易造成比特币区块链分叉，降低比特币网络的安全性。在目前的情况下 8MB 我们认为是一个稳妥的方案，此次扩容也是为未来进一步扩容打下基础。据统计，世界范围内，中国五家矿池共占全网总算力 60%以上。

F2pool　Antpool　BW
BPCChina　Huobi
2015 年 6 月 10 日

The Fight, The Bitcoin Community Is Not Paradise

Bigger blocks vs SegWit, the origin of Bitcoin Cash

- 2016-02-21 "Hong Kong Agreement"
  - SegWit(soft fork, to be released in 2016.04) + 2MB resize(hard fork, code be available by 2016.07)
- 2017-03-12 BIP148 "Mandatory activation of SegWit deployment"
  - Orphan Bitcoin blocks not signaling Bit 1 at its UASF forking point

Bitcoin Core

- 2017-05-23 "New York Agreement"(without Bitcoin Core) **SegWit2x (SegWit2Mb)**
  - Activate SegWit at 80% threshold, signaling at bit 4
  - Activate a 2MB hard fork within six months, with 80% threshold hash rate
- **2017-08-01 Bitcoin Cash forked away from Bitcoin with 8MB blocks**
  - Bitcoin Unlimited, Bitcoin ABC and Bitcoin XT

Bitcoin ABC

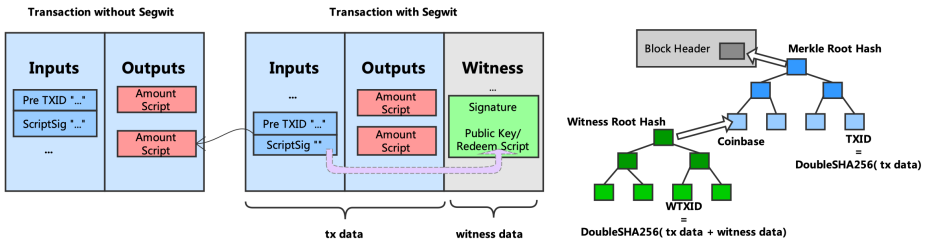
- 2017-08-23 Segwit activated via soft fork (as of 2018.11 ~40% SegWit txs)
- 2017-11-09 SegWit2x hard fork suspended "Onwards — all in on SegWit"
- **2018-11-15: BSV forked away from BCH with 128MB blocks**

BITCOINSV SATOSHI VISION

SegWit (Segregated Witness BIP141)

To carry more, shrink each tx by moving scriptSig data to another place (or, just pretend)

1MB (1,000,000 bytes) block size → 4MB (4,000,000 bytes) block weight
block weight = base_size * 4 + witness_size; virtual tx size = tx weight / 4



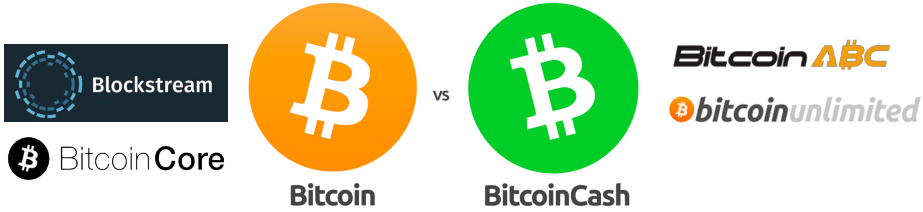
Effects: **1.6x~2x block size increase**, rough estimate 0.4x * 4 + 0.6x * 1 = 4MB, x≈1.8MB

For blocks with only P2PKH ~1.6MB, for blocks with only 2-2 P2MS ~2MB

The Fork & The Origin Of Bitcoin Cash

Chain fork on 2017-08-01, block height 478558, 8MB block size

Bitcoin is an innovative payment network and a new kind of money.



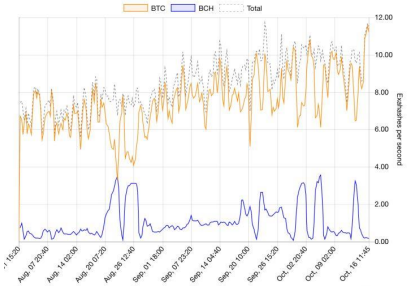
Bitcoin Cash brings sound money to the world, fulfilling the original promise of Bitcoin as "Peer-to-Peer Electronic Cash".

EDA - Emergency Difficulty Adjustment

Hash rate splits following the chain fork & hash power migration

2016 blocks adjust [4, ¼] &
6 blocks emergence adjust

"obviously hope that this feature will not need to be used, and it would not kick in if the hashrate stays above 8% of the global hashrate" - BitcoinABC 0.14.3 release notes



BitcoinABC 0.14.3 2017-07-14

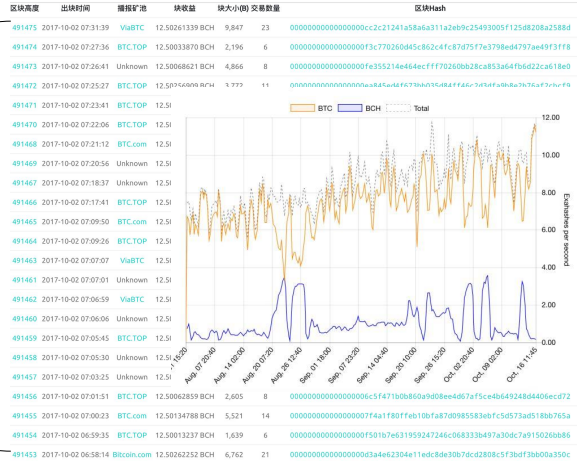
483853	2017-08-28 09:46:45	BTC.TOP	12.55804658 BCH	66,669	158	00000
483852	2017-08-28 09:10:56	Unknown	12.52707335 BCH	113,740	56	00000
483851	2017-08-28 09:03:15	BTC.TOP	12.51116718 BCH	16,657	40	00000
483850	2017-08-28 08:57:17	Unknown	12.56825753 BCH	128,981	139	00000
483849	2017-08-28 08:39:10	BTC.TOP	13.09998872 BCH	1,148,250	1,274	00000
483848	2017-08-28 04:10:55	BTC.TOP	13.02[illegible]152 BCH	805,190	1,292	00000
483847	2017-08-28 00:22:24	BTC.TOP	13.04[illegible]816 BCH	523,938	968	00000
483846	2017-08-27 21:59:54	Unknown	12.68[illegible]848 BCH	223,030	497	00000
483845	2017-08-27 20:40:25	Unknown	12.53[illegible]256 BCH	90,081	63	00000
483844	2017-08-27 20:33:42	Unknown	12.59944092 BCH	158,577	254	00000
483843	2017-08-27 19:47:24	ViaBTC	13.09046535 BCH	785,505	975	00000
483842	2017-08-27 17:01:36	AntPool	13.13467591 BCH	748,494	1,470	00000
483841	2017-08-27 11:00:00	BTC.com	12.52257341 BCH	25,979	30	00000

降20%
降20%
12.5 hours
13 hours

EDA - Emergency Difficulty Adjustment

No countermeasure against hashing rate's suddenly rise

More than 20 blocks mined within JUST 30 minutes



EDA - Emergency Difficulty Adjustment

Hash rate split following the chain fork & hash power migration

BCH	高度	503,815	版本	0x20000000	
	确认数	54,712	难度	451.31 G / 130.62 G	
	大小	202 Bytes	Bits	0x18086af5	
	数量	1	Nonce	0x1d1daeed	
			播报方	AntPool	EDA 100 days
			时间	2017-11-12 23:59:44	
BTC	高度	494,079	版本	0x20000000	
	确认数	57,732	难度	3.83 T / 1.36 T	
	大小	998,217 Bytes	Bits	0x1800ce4b	
	Stripped Size	998,181 Bytes	Nonce	0xb4a4e262	
	Weight	3,992,760	播报方	1Hash	
	数量	2,116	时间	2017-11-12 23:41:07	

gap 10000

Better Difficulty Adjust Algorithm, DAA

Better defense against hash power migration by quick response

BitcoinABC 0.116.0 2017-11-13

- Idea: quick response by adjust difficulty per block
  - Based on the worksum & elapsed time of previous 144 blocks

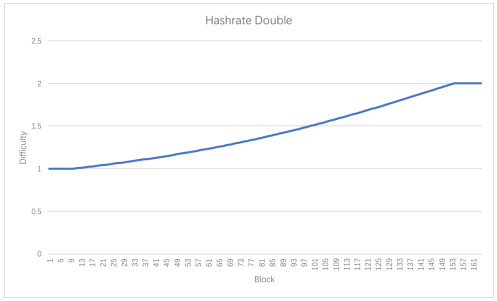
$$work = 10 \times 60 \times \frac{worksum\ of\ previous\ 144\ blocks}{actual\ time\ span\ of\ previous\ 144\ blocks}$$

- Adjust difficulty to hash rate to target a mean block interval of 600 seconds
  - Block height 504031, 2017-11-14 04:58, first block mined with DAA enabled
  - Block height 558636, 2018-11-29 14:29, data from **BTC.com**
  - Actual averaged block interval: 10.06 mins, pretty close to the expected 10min block interval
- Avoid sudden changes in difficulty when hash rate is fairly stable
- Adjust difficulty rapidly when hash rate changes rapidly
- Avoid oscillations from feedback between hash rate and difficulty
- Be resilient to attacks such as timestamp manipulation

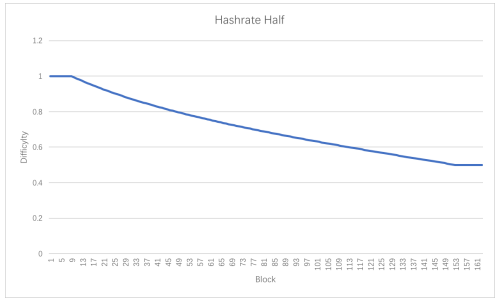
Better Difficulty Adjust Algorithm, DAA

When hash power suddenly doubles or halves

When hash power doubles



When hash power halves



Replace By Fee (aka RBF)

Transaction replacement in mempool

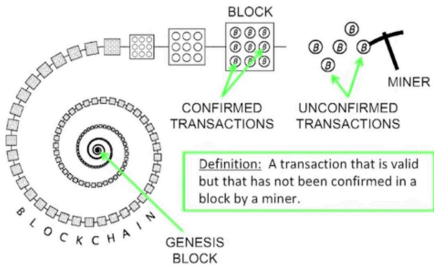
- Replace transaction in mempool spending part/all of the same UTXO
  - Enabled in the 1st version of Bitcoin
  - There is no need to pay higher fees, which may result to overuse of relay node bandwidth
  - Disabled in 0.3.12 with comment "Disable replacement feature for now" by Satoshi
- Opt-in Replace-by-fee transactions
  - Allow users to deal with unexpected confirmation delays due to block size limit
  - BIP125 Opt-in Full Replace-by-Fee Signaling, 2015.12
  - Bitcoin Core 0.12.0, Opt-in Replace-by-fee transactions feature implemented, 2016-02-23
  - Bitcoin Core 0.15.0, Replace-by-fee control in GUI, 2017-09-14
  - Bitcoin Core 0.16.0, Replace-By-Fee by default in GUI, 2018-02-26
- Bitcoin Cash: BIP 125 Replace-by-Fee was removed in Bitcoin ABC

https://github.com/Bitcoin-ABC/bitcoin-abc/blob/f5f1555c14e75f65e09c55f78e099d116868568b/doc/bips.md

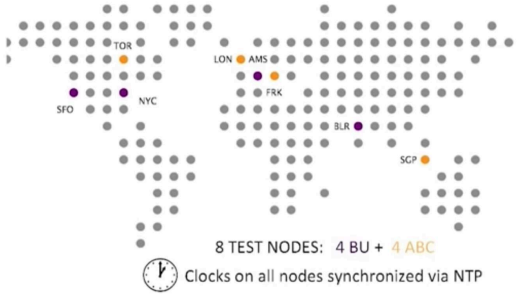
Security Of 0-Conf Transactions

Peter Rizun @ Instant Transaction Workshop, 2018-10-16

What is an unconfirmed transaction?



Definition: A transaction that is valid but that has not been confirmed in a block by a miner.



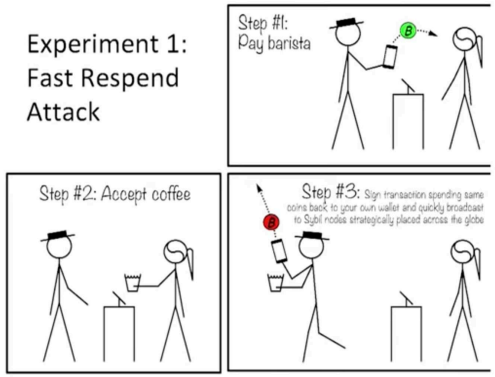
8 TEST NODES: 4 BU + 4 ABC

Clocks on all nodes synchronized via NTP

Empirical Double Spend Probabilities For Unconfirmed Tx

Peter Rizun @ Instant Transaction Workshop, 2018-10-16

Experiment 1: Fast Respend Attack

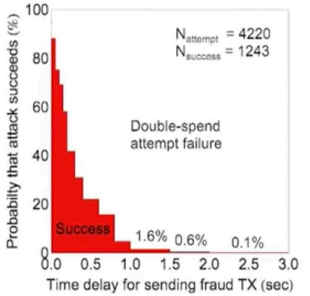


Delay 0.0 s – 0.05 s:
$N_{attempt}$ = 274
$N_{success}$ = 242
$P_{doublespend}$ = 88%

Delay 0.05 s – 0.10 s:
$N_{attempt}$ = 329
$N_{success}$ = 248
$P_{doublespend}$ = 75%

Delay 0.10 s – 0.15 s:
$N_{attempt}$ = 296
$N_{success}$ = 205
$P_{doublespend}$ = 69%

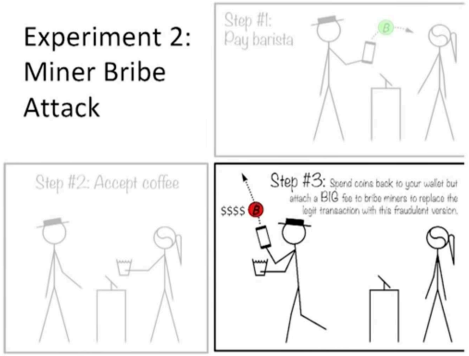
...4220 attempts in total...



Empirical Double Spend Probabilities For Unconfirmed Tx

Peter Rizun @ Instant Transaction Workshop, 2018-10-16

Experiment 2: Miner Bribe Attack



Experiment 2: Miner Bribe Attack

		10 sat/B 0.5 s delay	100 sat/B 0.5 s delay	100 sat/B 1.0 s delay	100 sat/B 5.0 s delay
Bribe attack	$N_{attempt}$	868	863	815	729
	$N_{success}$	209	248	142	40
	$P_{double\text{-}spend}$	24.1%	28.7%	17.5%	5.5%
Control (1 sat/B)	$N_{attempt}$	870	862	814	730
	$N_{success}$	201	204	2	0
	$P_{double\text{-}spend}$	23.1%	23.7%	0.3%	0%

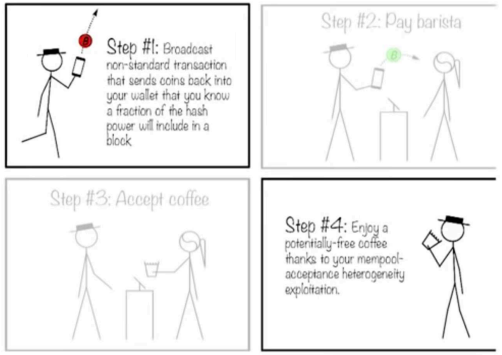
±1%: 50%; ±2%: 17%; ±3%: 5%; ±4%: 1%; ±5%: 0.1%

Attaching a fee of 100 sat/byte increased the probability of succeeding in a double-spend attempt in some cases

Empirical Double Spend Probabilities For Unconfirmed Tx

Peter Rizun @ Instant Transaction Workshop, 2018-10-16

Experiment 3: Reverse Respend Attack



Experiment 3: Reverse Respend Attack

$N_{attempts}$	3646
$N_{success}$	406
$P_{double\text{-}spend}$	11%

Every successful double-spend occurred when bitcoin.com (mining with Bitcoin Unlimited) found a block.

Topological Tx Order vs. Canonical Tx Order

The controversial modification of core consensus rules

BitcoinABC 0.18.0 2018-08-27

Topological Tx Order, TTOR

- Mempool, the most complex part of a BCH client
  - Mental picture: a forest
  - Each tree is a series of txs, A→B→C
- Required to order parent txs before children
- Prob. of short TXID collisions increase
  - As blocks get larger, more txs in one block
- Slow block propagation
  - Order information is the majority data transferred

Canonical Tx Order, CTOR

- CTOR is simpler to implement than TTOR
- No need to discover a valid ordering for miners
- Enable parallel block validation
- Eliminating an entire class of attack vectors
  - Mine a block that is as slow to validate as possible

✓Makes encoding & transmitting blocks a lot easier

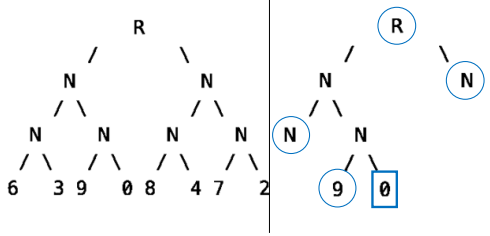
✓Enhance the security of SPV wallet
- Merkle exclusive proof is possible with CTOR
- PoC code by @SiyuHE

- Horizontally Scale (sharding) with Merklix tree

CTOR - Merkle Exclusion Proof

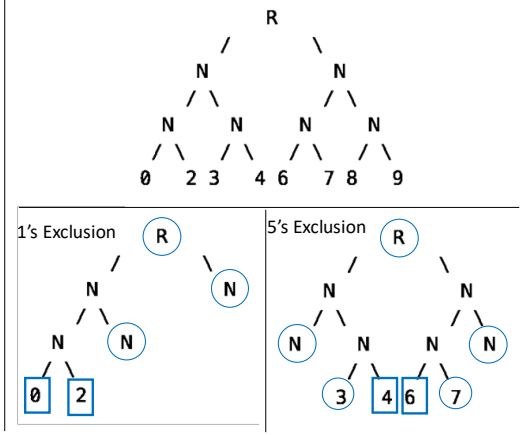
Enhance security of SPV wallet

Inclusion Proof For Unsorted Merkle tree



https://gist.github.com/chris-belcher/eb9abe417d74a7b5f20aabe6bff10de0

Exclusion Proof For Sorted Merkle Tree



BCH vs BSV (Satoshi's Vision)

Bitcoin ABC

The drama of Dr. Craig S Wright (aka CSW, aka Fake Satoshi)

- Hard fork on Nov. 15 as planned by Bitcoin ABC team
  - Remove topological ordering constraint from blocks starting Nov, 15 2018
  - Implement **canonical transaction order**, enforced Nov, 15 2018
  - Add **OP_CHECKDATASIG & OP_CHECKDATASIGVERIFY**, activates Nov, 15 2018
  - Enforce minimum transaction size of 100 bytes, starting Nov, 15 2018
- CSW and nChain had different thoughts
  - The opcodes **OP_MUL, OP_INVERT, OP_LSHIFT, & OP_RSHIFT** are re-enabled
  - The limit on the number of opcodes per script is increased to 500
  - The default maximum size of accepted **blocks = 128MB**

BITCOINSV SATOSHI VISION

The nature of Bitcoin is such that once version 0.1 was released, the core design was set in stone for the rest of its lifetime.

-- Satoshi Nakamoto @ 2010-06-17 18:46:08 UTC

https://satoshi.nakamotoinstitute.org/posts/bitcointalk/126/#selection-29.0-45.380

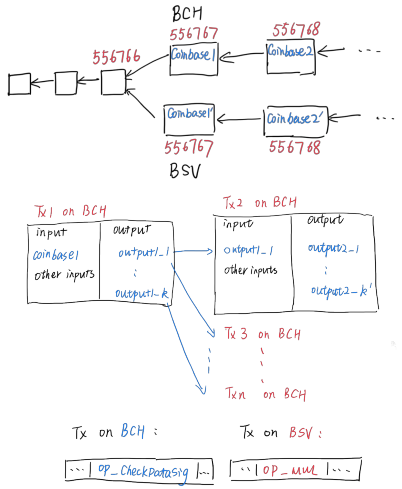
Protecting Your Assets Post-Fork

When forking without replay protection

- When there is no replay protection
  - A tx spending history UTXOs will be valid on both chains

How to make a tx only valid on one chain

- Reference an output exists only on one chain
  - Coinbase txs' output after the hard fork
  - The way Copernicus helped several exchanges after the fork
- Construct tx with chain-specific opcode
  - BCH: OP_CHECKDATASIG
  - BSV: OP_MUL, OP_INVERT, OP_LSHIFT, & OP_RSHIFT



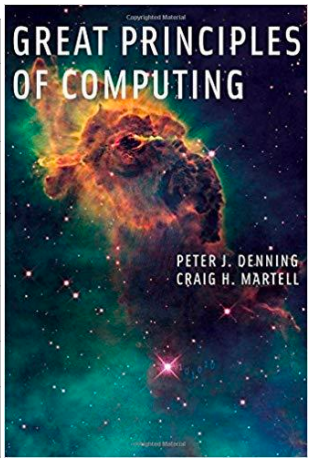
All kinds of improvement ideas

- Graphene
- UTXO Commitment
- Avalanche Protocol
- Pre-Consensus with Avalanche
- Schnorr Signature

Scaling Via Big Blocks, The Bitcoin Cash Way

The scaling dream, 1tx/pp/day, the bigger challenge

类别	关注点	相关技术点
通信	信息的可靠传输	P2P, Tor, Compact Block Relay, Graphene etc.
计算	可计算性	Parallel, Sharding Script scheme
记忆	信息表示、存放与读取	ADS design, UTXO Commitment
协作	利用多个自主计算实体	Pre-consensus, Sharding, etc.
评估	系统是否表现出预期行为	(Pressure) test
设计	设计特性结构的软件系统	Protocol Impl. Redesign

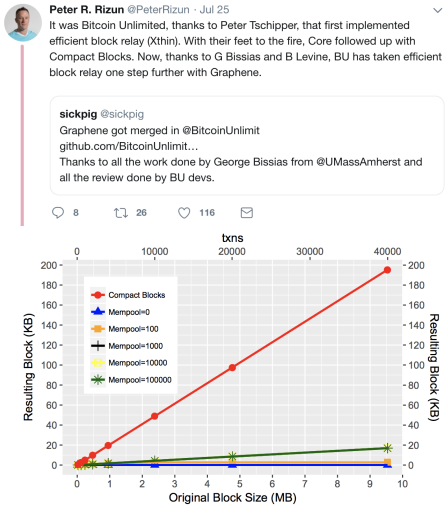


Graphene, Reduce Bandwidth

Marching towards huge block size

- Help nodes to run more efficiently
  - Ensure txs processed only once
  - Smaller block propagate faster
  - Fast propagation means less orphaning
  - Do not allow more txs per block
  - Not a scaling solution, but helps scaling
- Use bloom filter & IBLT from set reconciliation literature
  - bloom filter (BF) check 2 lists for identical entries
  - BF to reduce the diff between block & mempool
  - BF have False Positives (FP), propagation failure
  - IBLT: invertible bloom lookup tables
  - IBLT eliminates false positive of bloom filters
- Deployment status
  - Already merged into Bitcoin Unlimited
  - Also adopted by EOS etc.

bitcoinunlimited



UTXO Commitment: Motivation

Growing blockchain is a problem to every part of the ecosystem

- Painful Initial Block Download
  - New full node needs to download the entire historical blockchain
  - Annoying delay in the initial setup
  - Burden on existing node: large portion of bandwidth providing old blocks
- Large storage required: more than 100GB
- Why do nodes need this: VALIDATION
  - A full ledger to verify new blocks (txs)
- Does node verify each tx directly against the whole blockchain
  - **NO**. Tx. is currently verified according to UTXO
- Why do nodes need this: construct the current UTXO state
- **Full ledger is there to vouch for UTXO**
  - **What if we have another way to vouch (commit) UTXO?**

UTXO Commitment: The Challenge

Special Thanks to Shuai Qi, https://github.com/qshuai/utxo-stats

UTXO Status, 2018-06-14

- BCH mainnet: block height 534637, spendable outputs 39922839
- BTC mainnet: block height 527687, spendable outputs 51283037
- UTXO changes with each newly mined block, old consumed and new created

Commitment → hash function

- Working on sets, not individual element
- Quick: each block contains multiple txs., multiple outputs
- Iterative: difference between UTXOs of adjacent blocks is not huge
- Small proof: small constant commitment (proof) is preferred
- Safe: collision resistant

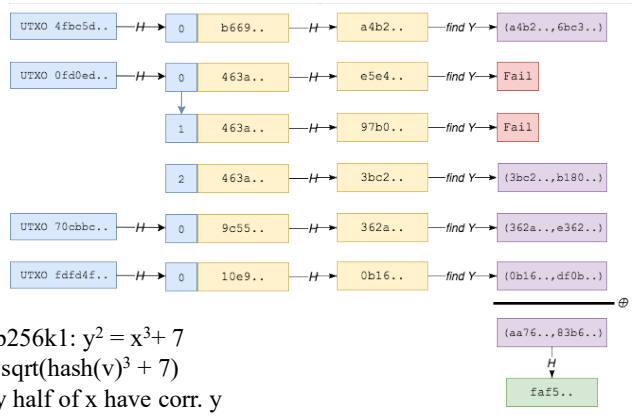
UTXO Commitment: Elliptic Curve Multiset Hash

Takeaway: another kind of collision resistant hash function for multiset

- Homomorphic, or incremental, multiset hash function
  - hash({a}) + hash({b}) = hash({a,b})
- Basic idea: hash each element to an EC point (try and increment)
  - An empty set maps to the infinity point of EC
- Combine/add/remove elements → Points Add of corr. EC Point
- The order of the elements in the multiset does not matter
- Duplicate elements are possible, {a} and {a, a} have different digest
- To update the digest of a multiset, only needs to compute the difference
- Can be constructed on secp256k1 elliptic curve
- Collision resistant relies on hardness of ECDLP
  - The same security assumption as ECDSA sign/verify
  - Need more eyes to look into the security proof of ECMH (maybe worry too much)

UTXO Commitment: Hashing To Elliptic Curve Point

Takeaway: another kind of collision resistant hash function for multiset



secp256k1: $y^2 = x^3 + 7$
$y = \text{sqrt}(\text{hash}(v)^3 + 7)$
only half of x have corr. y

*Project: ECMH PoC

UTXO Commitment: ECMH Demo & PoC on Testnet

Based on https://github.com/tomasvdw/secp256k1

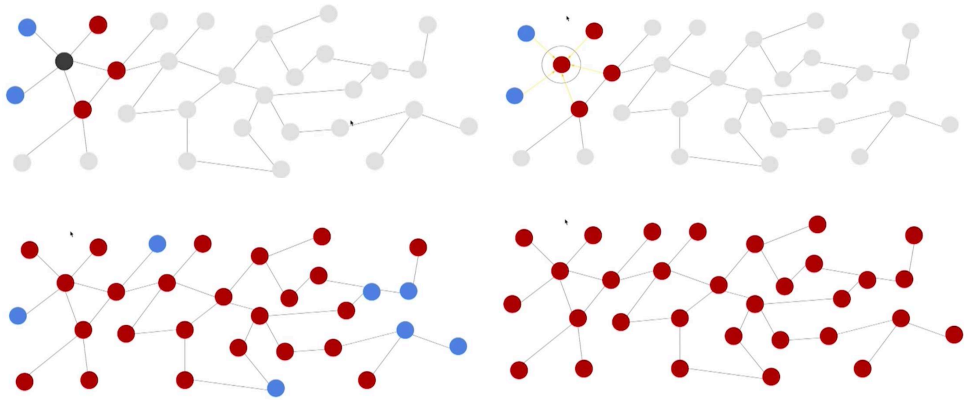
- ECMH demo:

```
[bitmain@LongMac ]ecmh $ ./ecmu_test.exe
     {}: : 8667e718294e9e0df1d30600ba3eeb201f764aad2dad72748643e4a285e1d1f7
     12: : 26418d1239938da4f661f4bffc9e87b74d0c25571f37a5da3a306163ee4511af
     23: : 2799ed5d63d2dcef80f17682faa9772496cff59ac194e353283897243c269d97
    123: : e3fc681313888143471cef461382afbf1b6cf82e1ed98e85d36a953429742031
12+23-2: : e3fc681313888143471cef461382afbf1b6cf82e1ed98e85d36a953429742031
```

- First UTXO Commitment on Testnet, 2018-06-05
  - https://www.yours.org/content/first-utxo-commitment-on-testnet-db7bf45bf83d
  - BCH testnet, block 1237565, commitment in coinbase's OP_RETURN output
  - [5554583011007bc4426b03824ccca5912bb147bd9f6847b670a08f24b79a4b5ed0b36393]

Avalanche Protocol From Team Rocket

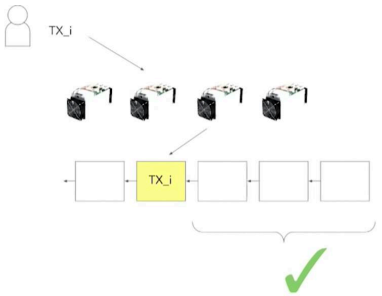
A novel metastable consensus protocol family for cryptocurrencies



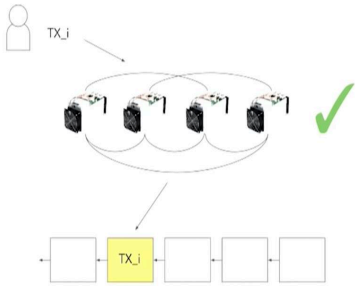
Better 0-conf: Pre-Consensus with Avalanche

Faster & safer transaction confirmation

Nakamoto transaction model



Faster & safer transaction confirmation



Schnorr Signature – Two Variants & The Trend

Schnorr signature is more suitable for usage in blockchain

- Key Generation
  - $P = dG$
- Sign on given message M
  - randomly k, let $R = kG$
  - $e = hash(R||M)$
  - $s = k + ed \ mod \ n$
  - Signature is : (R, s)
- Verify (R, s) of M with P
  - Check sG vs $R + eP$
  - $sG = (k + ed)G = kG + edG = R + eP$

- Key Generation
  - $P = dG$
- Sign on given message M
  - randomly k, let $R = kG$
  - $e = hash(R||M)$
  - $s = r - ex \ mod \ n$
  - Signature is : (e, s)
- Verify (e, s) of M with P
  - Compute $R' = sG + eP$
  - Check $hash(R'||M)$ vs e
  - $R' = (r - ed)G + eP = rG$

Which variant to use if you are constructing a blockchain?

Schnorr Signature – Batch Verification

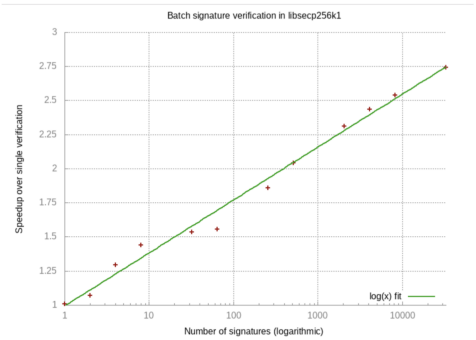
Utilize the linear property of Schnorr signature's verification process

- Recall Schnorr signature's verification: $sG = (k + ed)G = kG + edG = R + eP$
- Batch verification equation is :
  - $(\sum_{i=1}^{n} s_i) * G = (\sum_{i=1}^{n} R_i) + (\sum_{i=1}^{n} e_i * P_i)$
  - Attacker can forge signature to pass the batch verification
- Suppose attacker's public key $P_1 = x_1 * G$, to forge signature for public key $P_2 = x_2 * G$
  - x_2 is not known to attacker
  - Attacker randomly choose $r_2, s_2, R_2 = r_2 * G$, and computes $e_2 = h(P_2||R_2||m_2)$
  - Attacker set $R_1 = -(e_2 * P_2)$, and computes $e_1 = h(P_1||R_1||m_1)$
  - Then he derive $s_1 = r_2 + e_1 x_1 - s_2 \ mod \ p$
  - It can be verified that signatures $(R_1, s_1), (R_2, s_2)$ pass the batch verification:
  - $(s_1 + s_2) * G = R_1 + R_2 + e_1 P_1 + e_2 P_2$
- Defense: randomly choose $a_i \in [0, p-1]$, $i \in [2, n]$ and verifies the following equation:
  - $(s_1 + \sum_{\{i=2\}}^{\{n\}} a_i \, s_i) * G = (R_1 + \sum_{\{i=2\}}^{n} a_i * R_i) + (e_1 * P_1 + \sum_{\{i=2\}}^{n} (e_i a_i) * P_i)$

*Project: Schnorr Bacth

Schnorr Signature – Batch Verification

Utilize the linear property of Schnorr signature's verification process



# batch	verify	verify/s
1	78.9us	12674
4	56.7us	17637
16	48.7us	20534
64	45.6us	21930
256	37.6us	26596
1024	32.7us	30581
4096	27.9us	35842

THANKS
Q&A