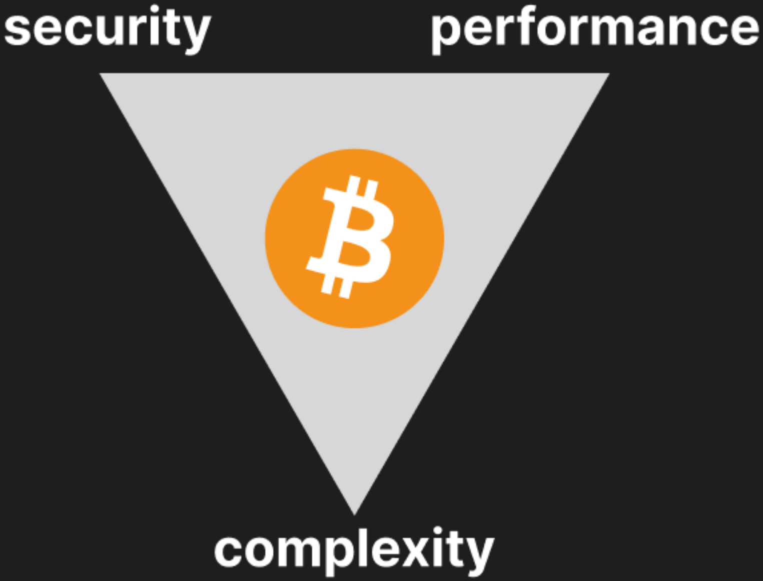
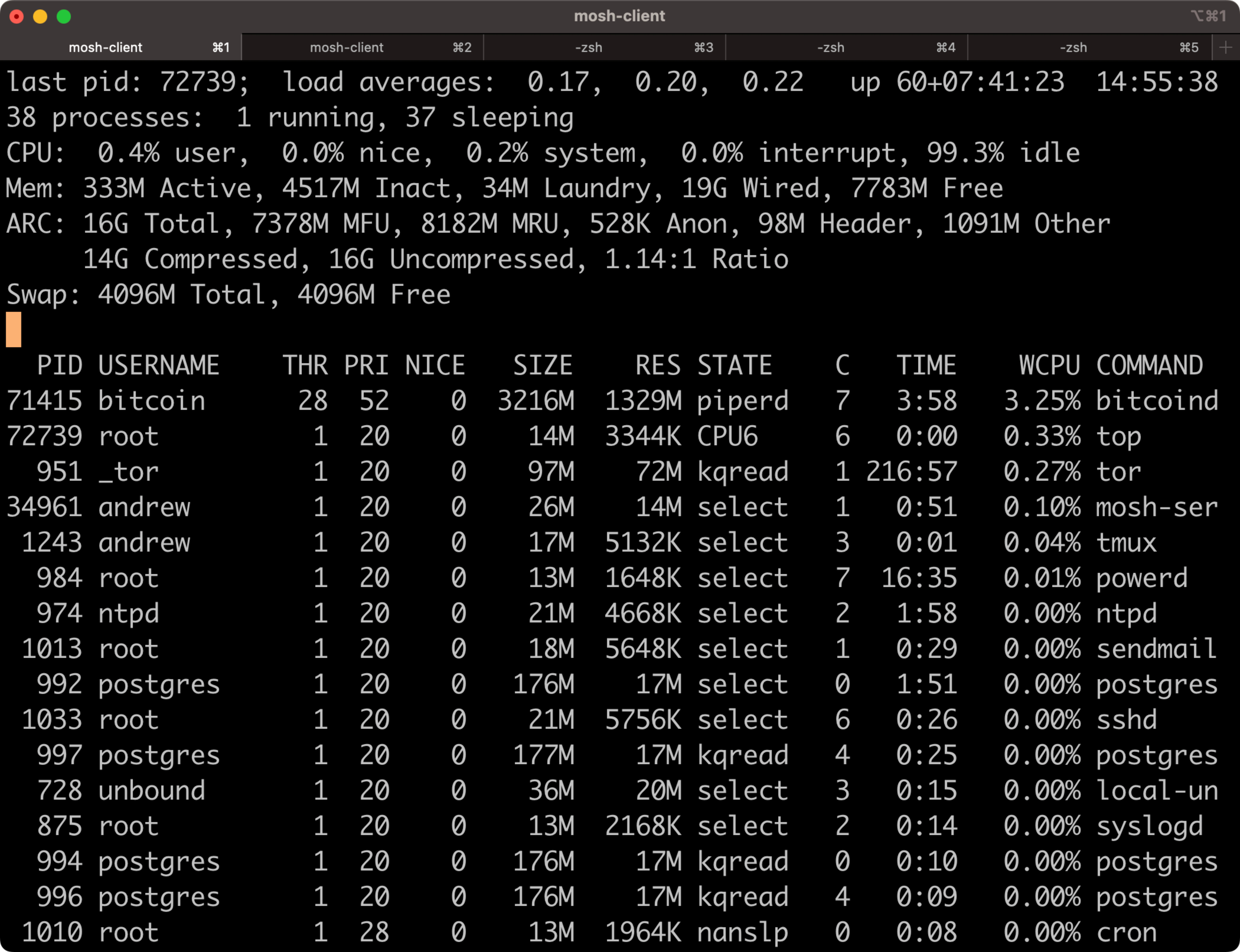


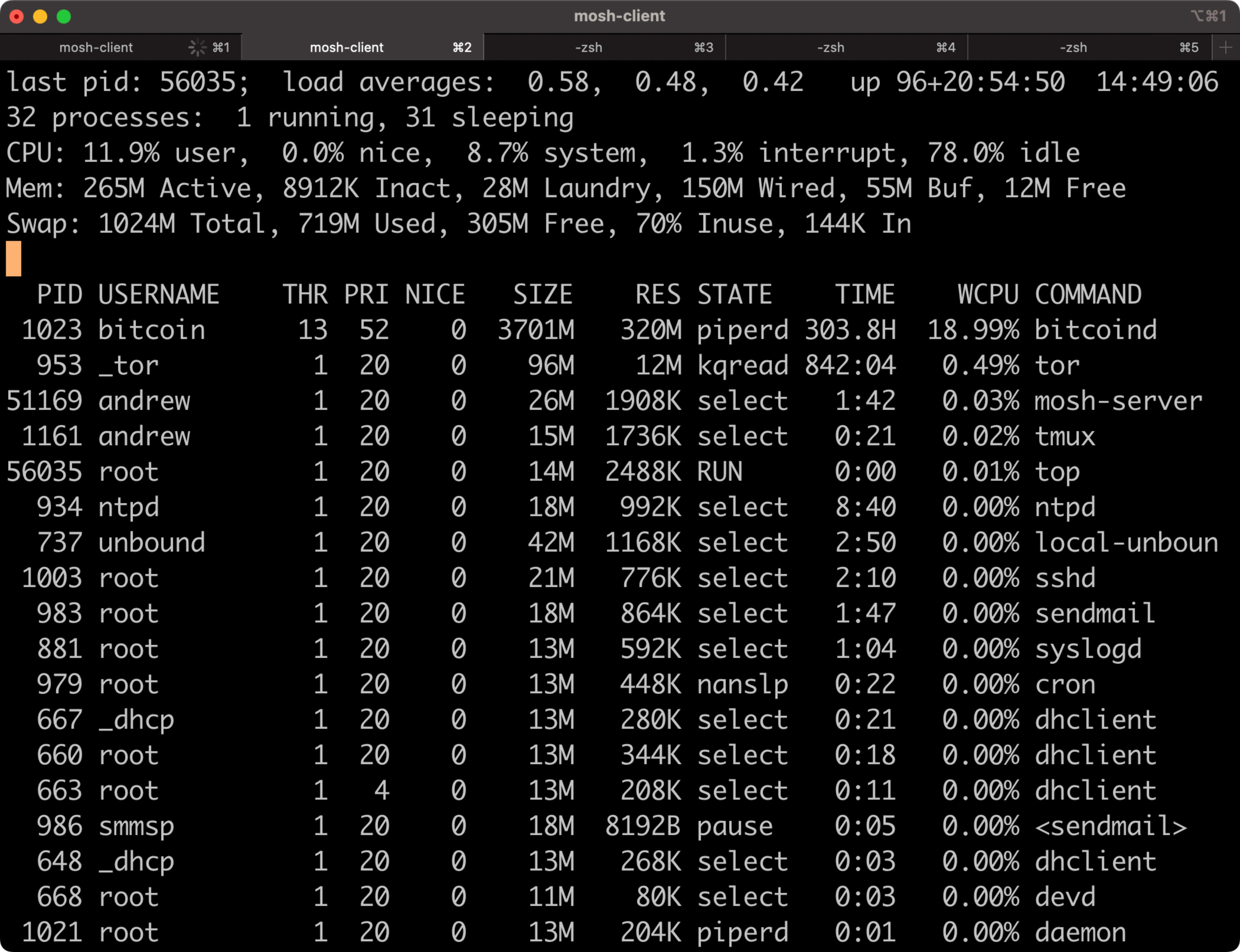
security

performance

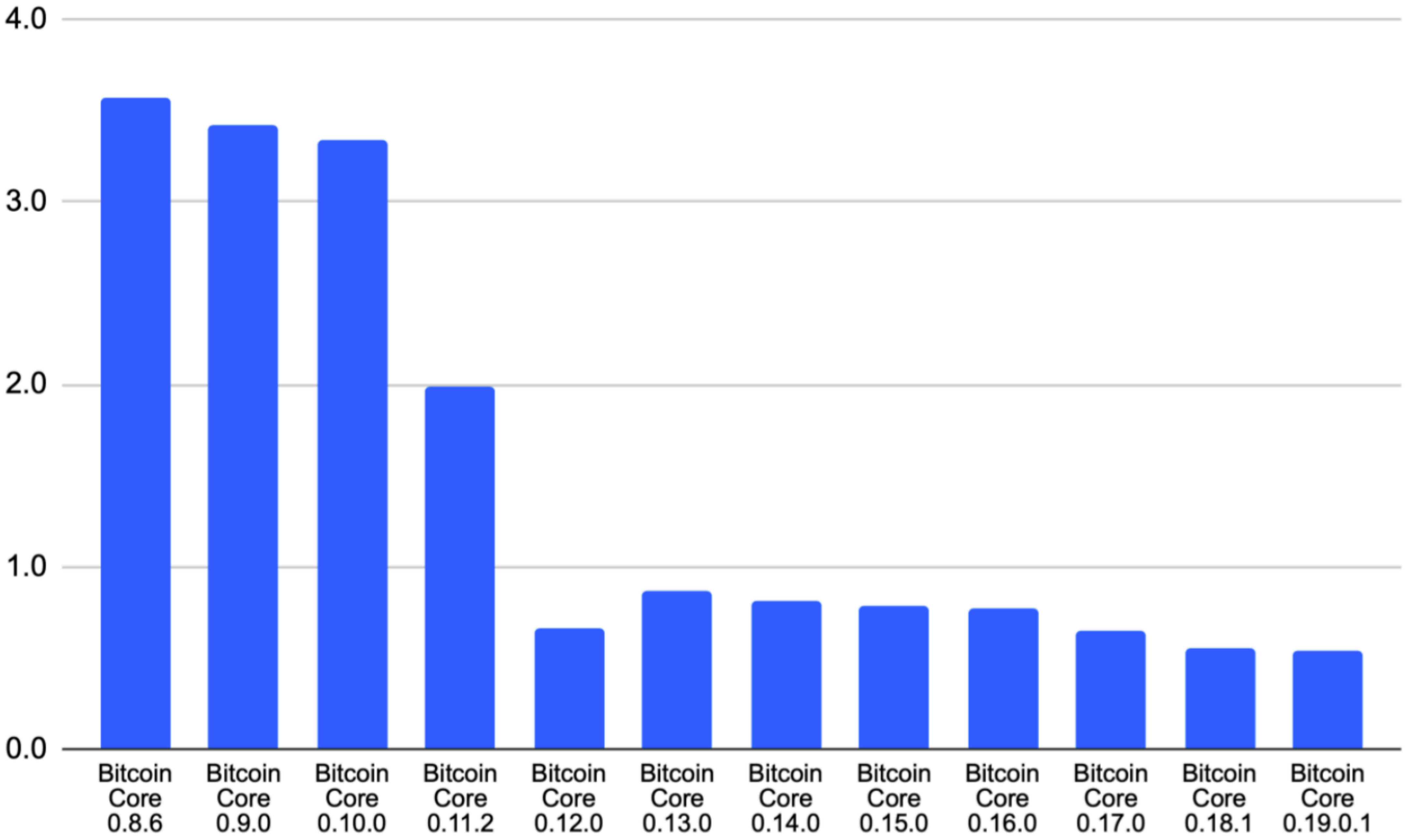
complexity



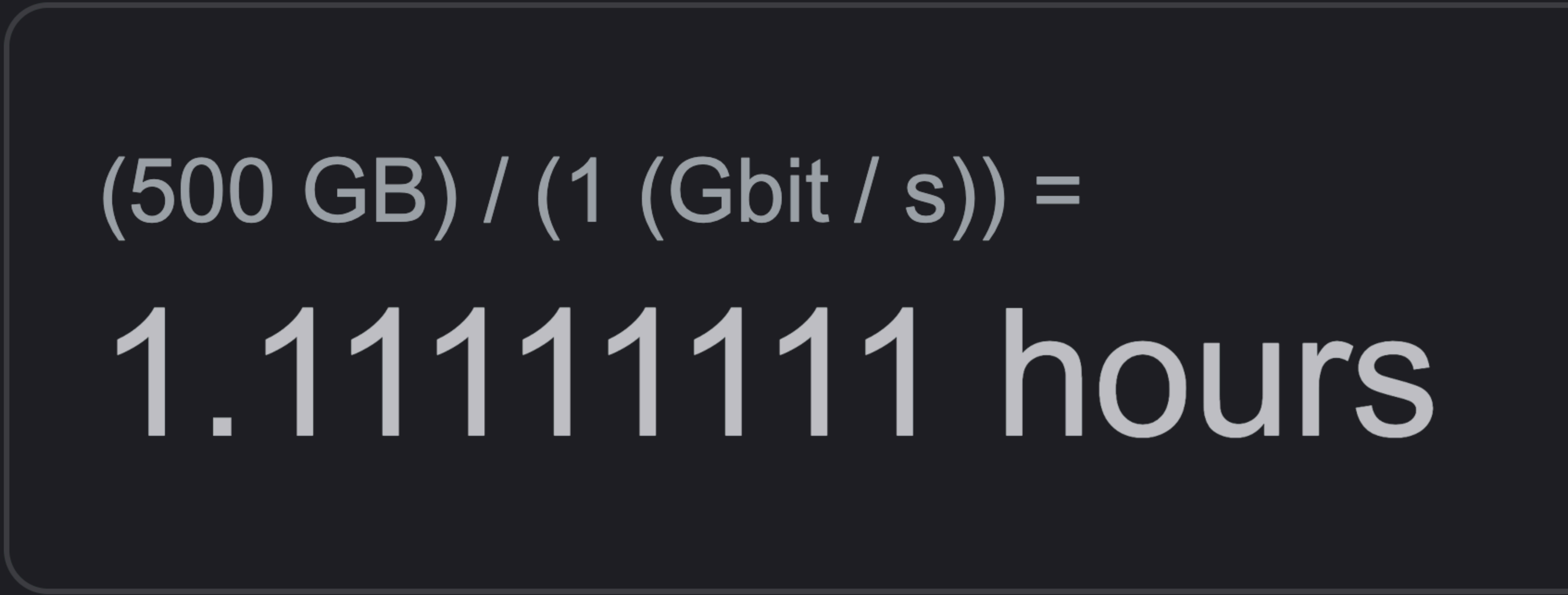
```
last pid: 72739;  load averages:  0.17,  0.20,  0.22   up 60+07:41:23  14:55:38
38 processes:  1 running, 37 sleeping
CPU:  0.4% user,  0.0% nice,  0.2% system,  0.0% interrupt, 99.3% idle
Mem: 333M Active, 4517M Inact, 34M Laundry, 19G Wired, 7783M Free
ARC: 16G Total, 7378M MFU, 8182M MRU, 528K Anon, 98M Header, 1091M Other
     14G Compressed, 16G Uncompressed, 1.14:1 Ratio
Swap: 4096M Total, 4096M Free


  PID USERNAME    THR PRI NICE   SIZE    RES STATE    C   TIME    WCPU COMMAND
71415 bitcoin      28  52    0  3216M  1329M piperd   7   3:58   3.25% bitcoind
72739 root          1  20    0    14M  3344K CPU6     6   0:00   0.33% top
  951 _tor          1  20    0    97M    72M kqread   1 216:57   0.27% tor
34961 andrew        1  20    0    26M    14M select   1   0:51   0.10% mosh-ser
 1243 andrew        1  20    0    17M  5132K select   3   0:01   0.04% tmux
  984 root          1  20    0    13M  1648K select   7  16:35   0.01% powerd
  974 ntpd          1  20    0    21M  4668K select   2   1:58   0.00% ntpd
 1013 root          1  20    0    18M  5648K select   1   0:29   0.00% sendmail
  992 postgres      1  20    0   176M    17M select   0   1:51   0.00% postgres
 1033 root          1  20    0    21M  5756K select   6   0:26   0.00% sshd
  997 postgres      1  20    0   177M    17M kqread   4   0:25   0.00% postgres
  728 unbound       1  20    0    36M    20M select   3   0:15   0.00% local-un
  875 root          1  20    0    13M  2168K select   2   0:14   0.00% syslogd
  994 postgres      1  20    0   176M    17M kqread   0   0:10   0.00% postgres
  996 postgres      1  20    0   176M    17M kqread   4   0:09   0.00% postgres
 1010 root          1  28    0    13M  1964K nanslp   0   0:08   0.00% cron
```



```
last pid: 56035;  load averages:  0.58,  0.48,  0.42   up 96+20:54:50  14:49:06
32 processes:  1 running, 31 sleeping
CPU: 11.9% user,  0.0% nice,  8.7% system,  1.3% interrupt, 78.0% idle
Mem: 265M Active, 8912K Inact, 28M Laundry, 150M Wired, 55M Buf, 12M Free
Swap: 1024M Total, 719M Used, 305M Free, 70% Inuse, 144K In

  PID USERNAME    THR PRI NICE   SIZE    RES STATE    TIME    WCPU COMMAND
 1023 bitcoin      13  52    0  3701M   320M piperd 303.8H  18.99% bitcoind
  953 _tor          1  20    0    96M    12M kqread 842:04   0.49% tor
51169 andrew        1  20    0    26M  1908K select   1:42   0.03% mosh-server
 1161 andrew        1  20    0    15M  1736K select   0:21   0.02% tmux
56035 root          1  20    0    14M  2488K RUN      0:00   0.01% top
  934 ntpd          1  20    0    18M   992K select   8:40   0.00% ntpd
  737 unbound       1  20    0    42M  1168K select   2:50   0.00% local-unboun
 1003 root          1  20    0    21M   776K select   2:10   0.00% sshd
  983 root          1  20    0    18M   864K select   1:47   0.00% sendmail
  881 root          1  20    0    13M   592K select   1:04   0.00% syslogd
  979 root          1  20    0    13M   448K nanslp   0:22   0.00% cron
  667 _dhcp         1  20    0    13M   280K select   0:21   0.00% dhclient
  660 root          1  20    0    13M   344K select   0:18   0.00% dhclient
  663 root          1   4    0    13M   208K select   0:11   0.00% dhclient
  986 smmsp         1  20    0    18M  8192B pause    0:05   0.00% <sendmail>
  648 _dhcp         1  20    0    13M   268K select   0:03   0.00% dhclient
  668 root          1  20    0    11M    80K select   0:03   0.00% devd
 1021 root          1  20    0    13M   204K piperd   0:01   0.00% daemon
```

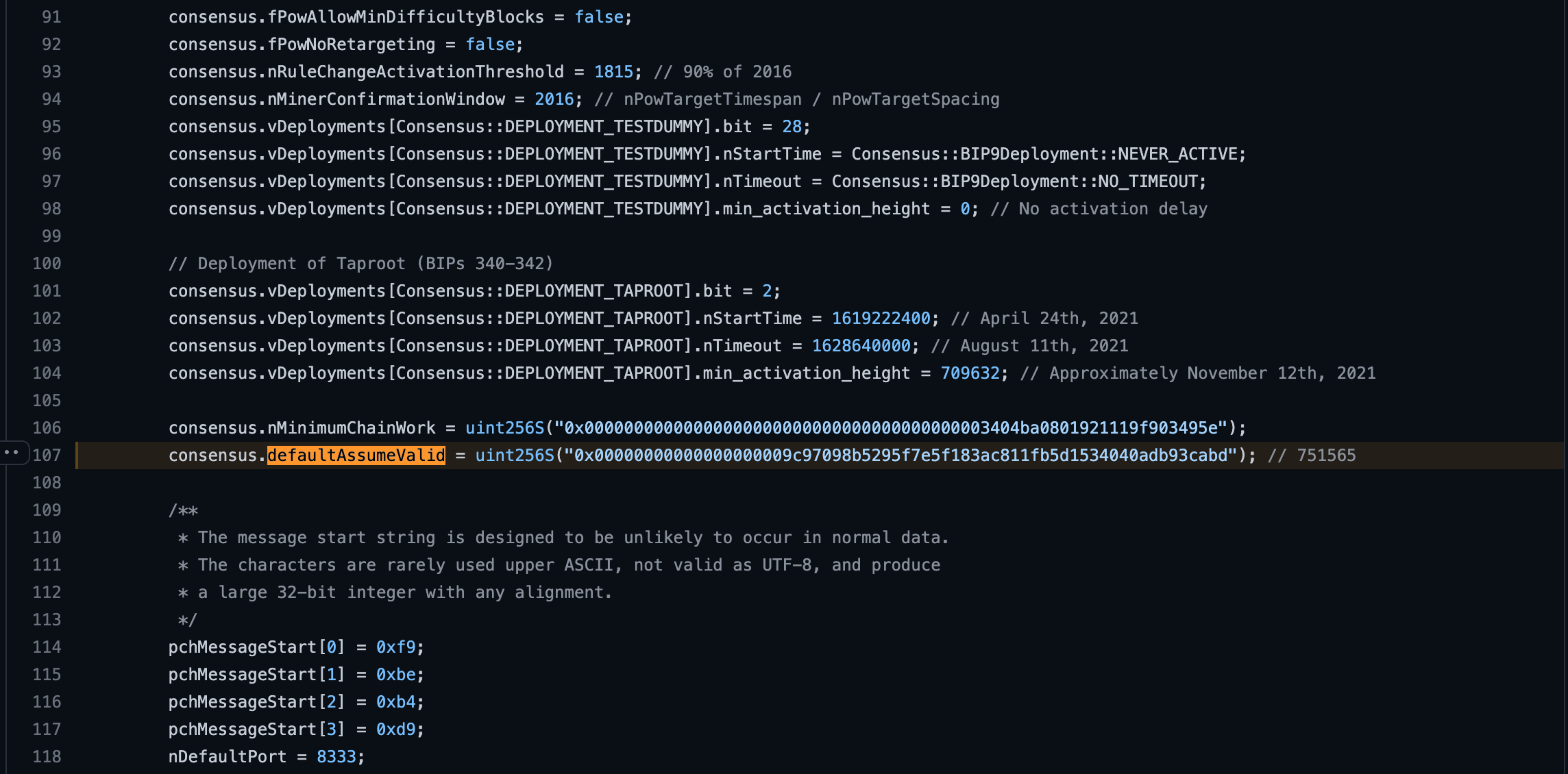


Bitcoin initial block download time in days — an average of three attempts. Source: BitMex

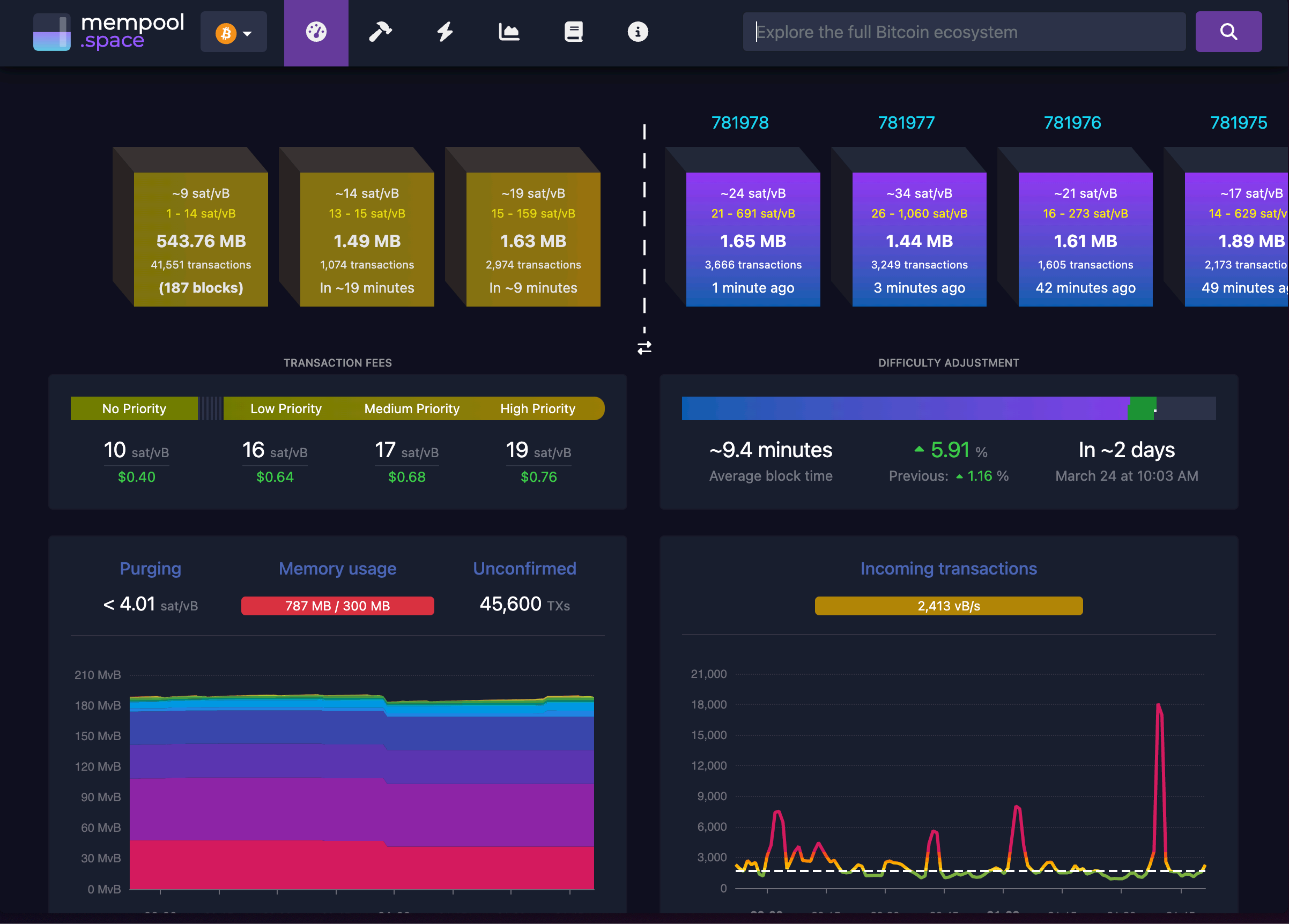


(500 GB) / (1 (Gbit / s)) =

1.11111111 hours



```
        consensus.fPowAllowMinDifficultyBlocks = false;
        consensus.fPowNoRetargeting = false;
        consensus.nRuleChangeActivationThreshold = 1815; // 90% of 2016
        consensus.nMinerConfirmationWindow = 2016; // nPowTargetTimespan / nPowTargetSpacing
        consensus.vDeployments[Consensus::DEPLOYMENT_TESTDUMMY].bit = 28;
        consensus.vDeployments[Consensus::DEPLOYMENT_TESTDUMMY].nStartTime = Consensus::BIP9Deployment::NEVER_ACTIVE;
        consensus.vDeployments[Consensus::DEPLOYMENT_TESTDUMMY].nTimeout = Consensus::BIP9Deployment::NO_TIMEOUT;
        consensus.vDeployments[Consensus::DEPLOYMENT_TESTDUMMY].min_activation_height = 0; // No activation delay

        // Deployment of Taproot (BIPs 340-342)
        consensus.vDeployments[Consensus::DEPLOYMENT_TAPROOT].bit = 2;
        consensus.vDeployments[Consensus::DEPLOYMENT_TAPROOT].nStartTime = 1619222400; // April 24th, 2021
        consensus.vDeployments[Consensus::DEPLOYMENT_TAPROOT].nTimeout = 1628640000; // August 11th, 2021
        consensus.vDeployments[Consensus::DEPLOYMENT_TAPROOT].min_activation_height = 709632; // Approximately November 12th, 2021

        consensus.nMinimumChainWork = uint256S("0x00000000000000000000000000000000000000003404ba0801921119f903495e");
        consensus.defaultAssumeValid = uint256S("0x00000000000000000009c97098b5295f7e5f183ac811fb5d1534040adb93cabd"); // 751565

        /**
         * The message start string is designed to be unlikely to occur in normal data.
         * The characters are rarely used upper ASCII, not valid as UTF-8, and produce
         * a large 32-bit integer with any alignment.
         */
        pchMessageStart[0] = 0xf9;
        pchMessageStart[1] = 0xbe;
        pchMessageStart[2] = 0xb4;
        pchMessageStart[3] = 0xd9;
        nDefaultPort = 8333;
```



				781978	781977	781976	781975
~9 sat/vB	~14 sat/vB	~19 sat/vB		~24 sat/vB	~34 sat/vB	~21 sat/vB	~17 sat/vB
1 - 14 sat/vB	13 - 15 sat/vB	15 - 159 sat/vB		21 - 691 sat/vB	26 - 1,060 sat/vB	16 - 273 sat/vB	14 - 629 sat/v
543.76 MB	1.49 MB	1.63 MB		1.65 MB	1.44 MB	1.61 MB	1.89 MB
41,551 transactions	1,074 transactions	2,974 transactions		3,666 transactions	3,249 transactions	1,605 transactions	2,173 transactio
(187 blocks)	In ~19 minutes	In ~9 minutes		1 minute ago	3 minutes ago	42 minutes ago	49 minutes a

TRANSACTION FEES

No Priority	Low Priority	Medium Priority	High Priority
10 sat/vB	16 sat/vB	17 sat/vB	19 sat/vB
$0.40	$0.64	$0.68	$0.76

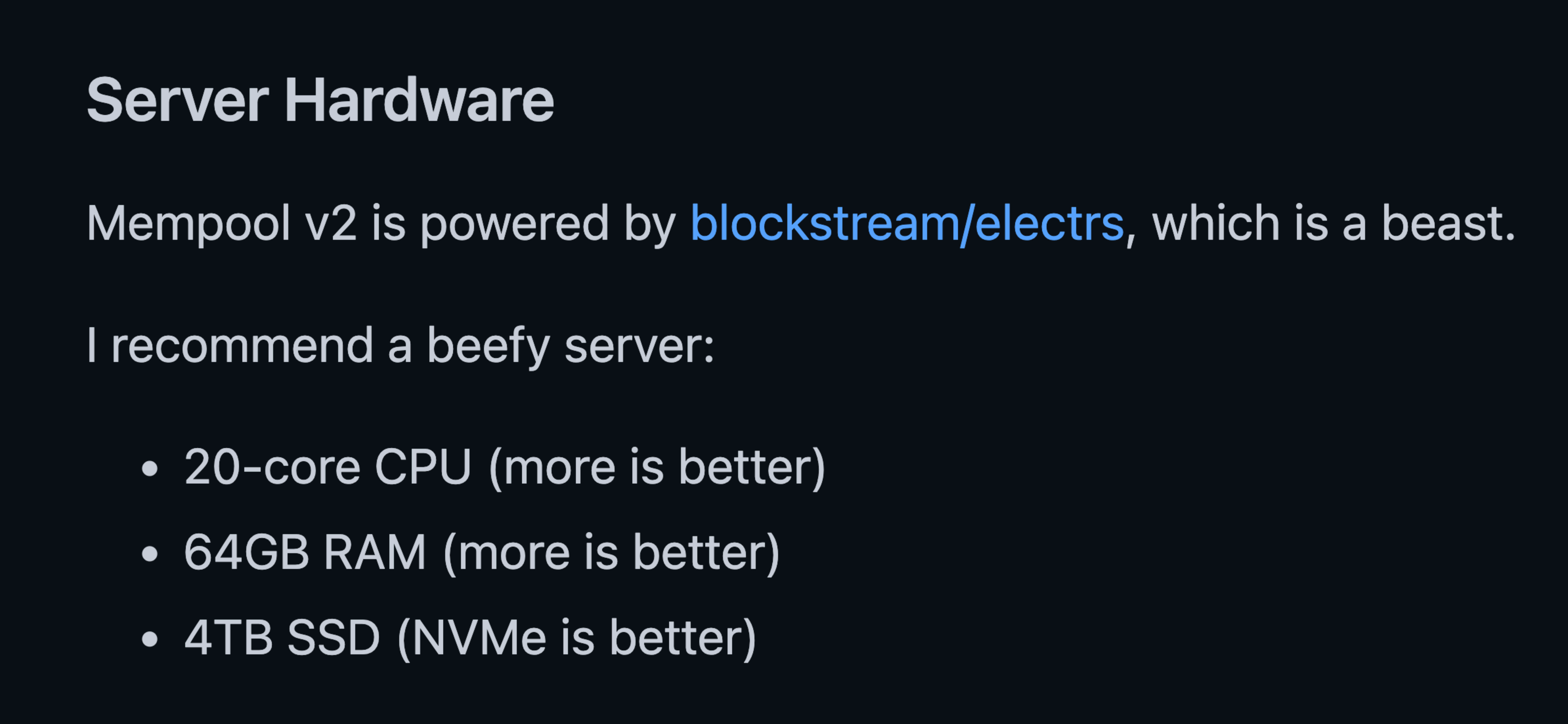
DIFFICULTY ADJUSTMENT

~9.4 minutes	▲ 5.91 %	In ~2 days
Average block time	Previous: ▲ 1.16 %	March 24 at 10:03 AM

Purging	Memory usage	Unconfirmed
< 4.01 sat/vB	787 MB / 300 MB	45,600 TXs

Incoming transactions

2,413 vB/s

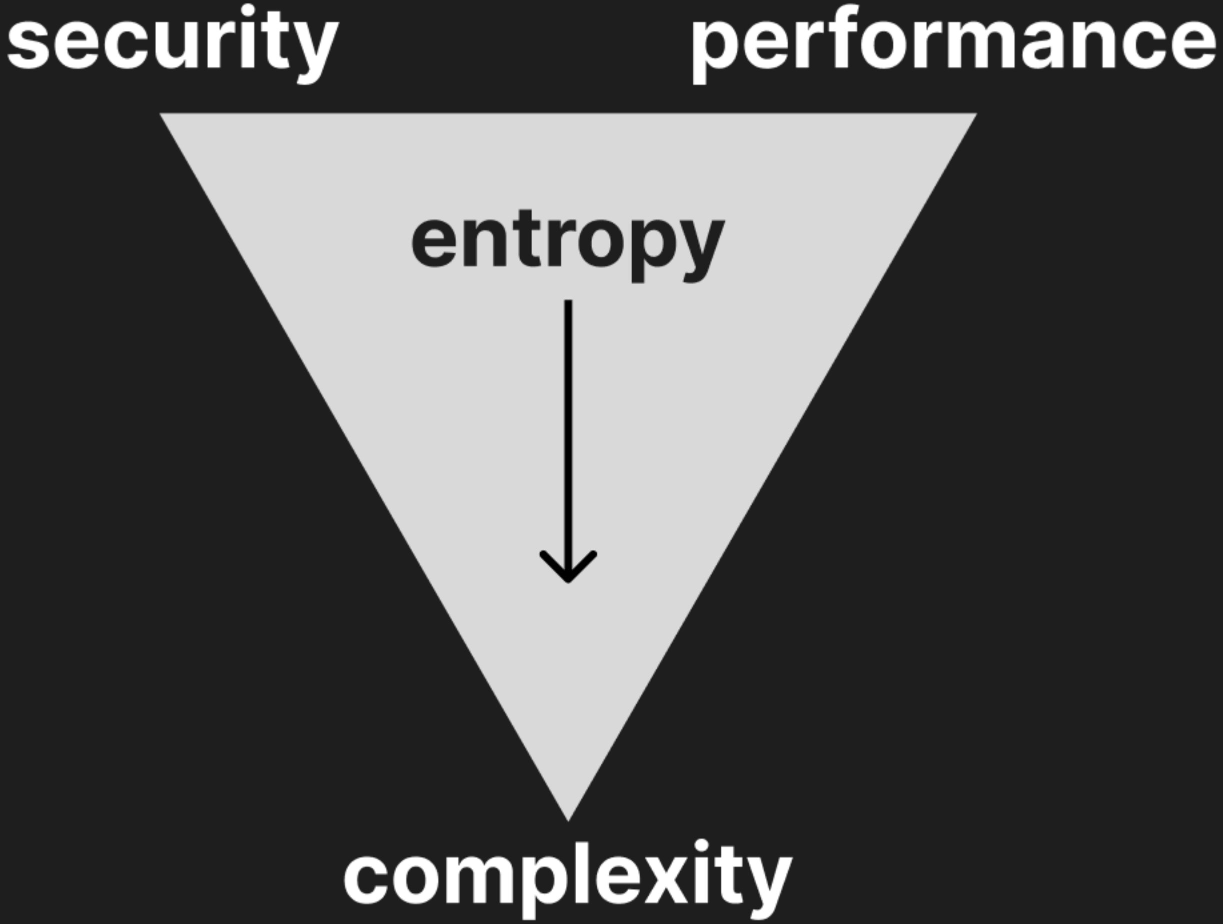


Server Hardware

Mempool v2 is powered by blockstream/electrs, which is a beast.

I recommend a beefy server:

- 20-core CPU (more is better)
- 64GB RAM (more is better)
- 4TB SSD (NVMe is better)



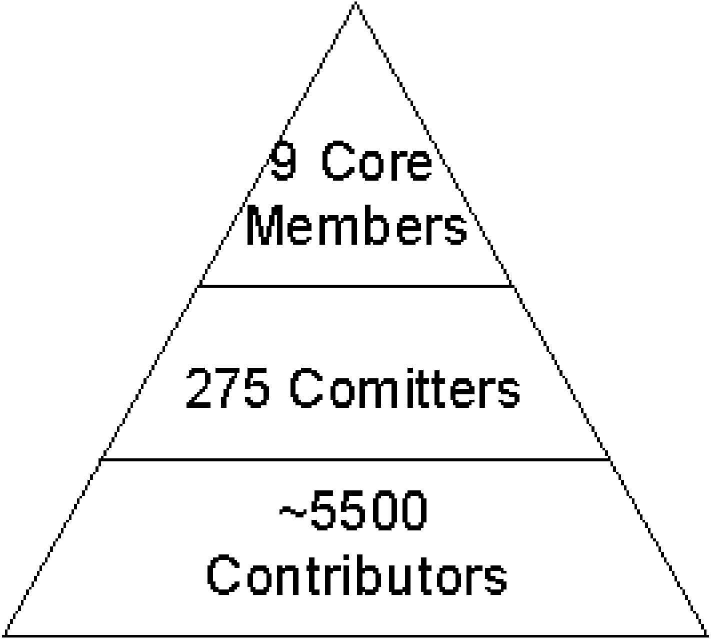
security

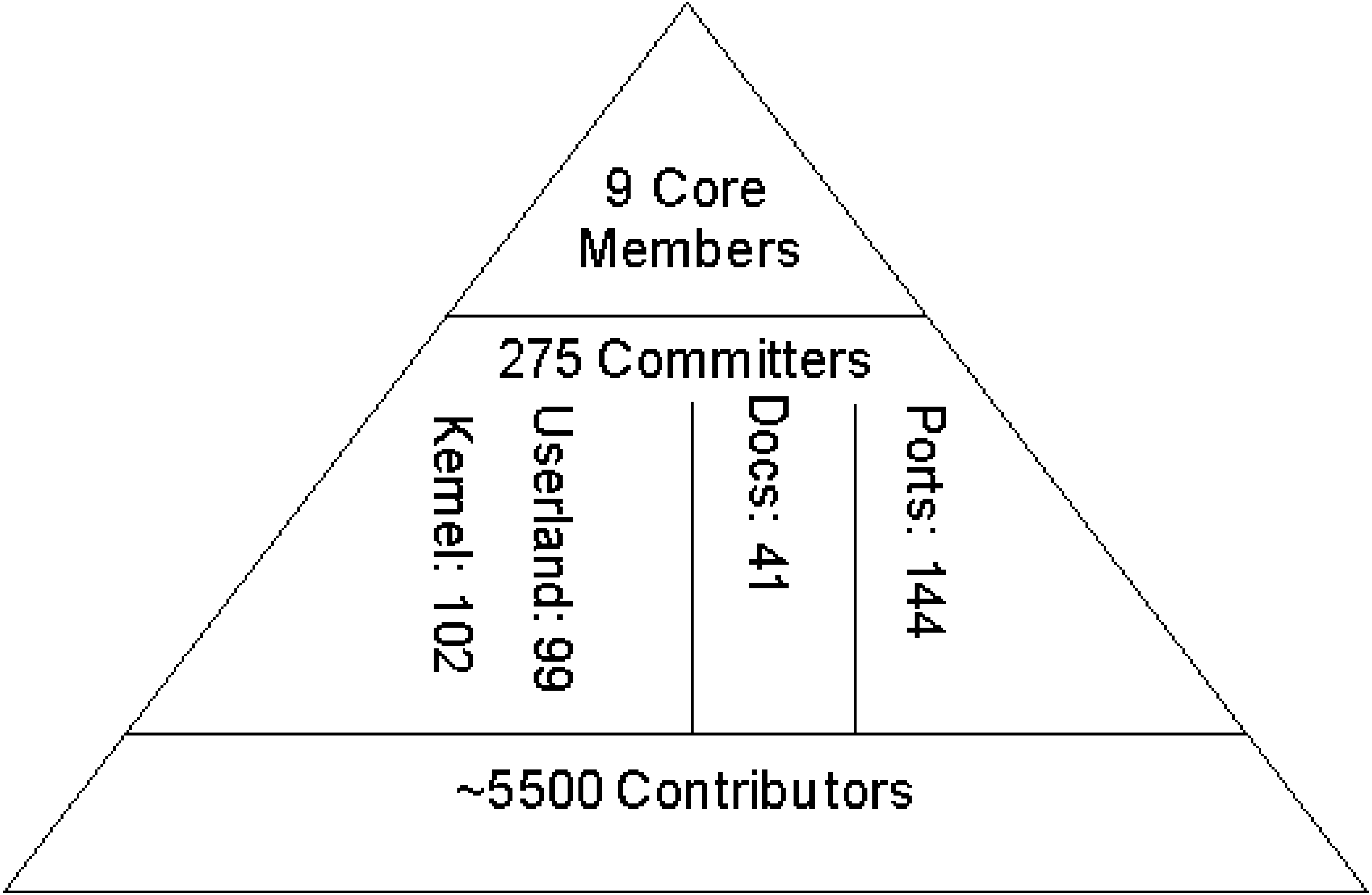
performance

entropy

complexity

1970
1980
1990
2000
2010
2020
Time
FreeBSD 12.2
DragonFly BSD 6.0
Matthew Dillon
BSD family
NetBSD 9.2
OpenBSD 7.0
Theo de Raadt
BSD (Berkeley Software Distribution) 4.4
Bill Joy
SunOS 4.1.4
Darwin 21.0
NextStep 3.3
macOS 11.6
Apple
Xenix OS
Microsoft/SCO
GNU/Hurd 0.9
GNU
Richard Stallman
Linux 5.15
Linus Torvalds
Minix 3.4
Andrew S. Tanenbaum
Research UNIX 10.5
Bell Labs: Ken Thompson, Dennis Ritchie, et al.
Commercial UNIX
AT&T
UnixWare 7
Univel/SCO/Xinuos
Solaris 11.4
Sun/Oracle
System III & V family
HP-UX 11i v3
AIX 7.2
IBM
IRIX 6.5.30
SGI


9 Core Members
275 Comitters
~5500 Contributors


9 Core
Members
275 Committers
Kemel: 102
Userland: 99
Docs: 41
Ports: 144
~5500 Contributors

VuXML

VuXML is the data format used to document security vulnerabilities in the FreeBSD Ports Collection.

Adding entries

Note: this process is tentative. Feel free to discuss and contribute. A reworded version of this section might fit into a PHB chapter.

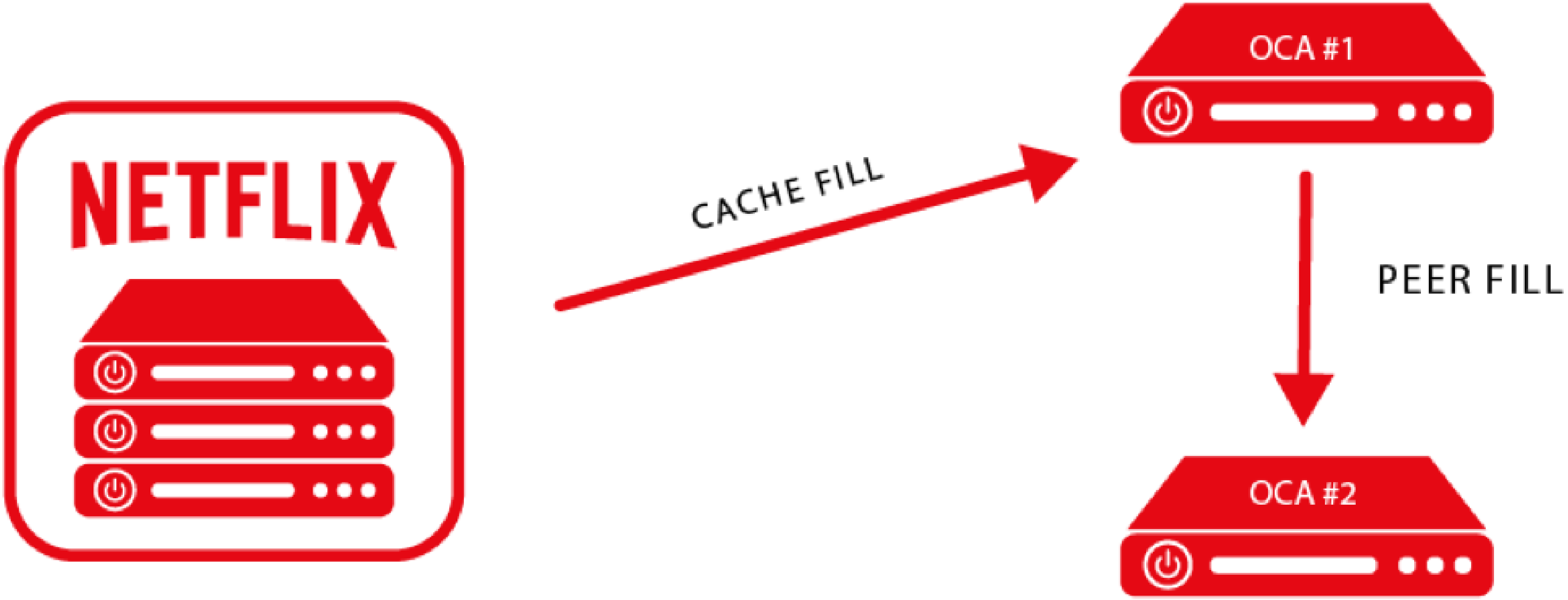
Theory

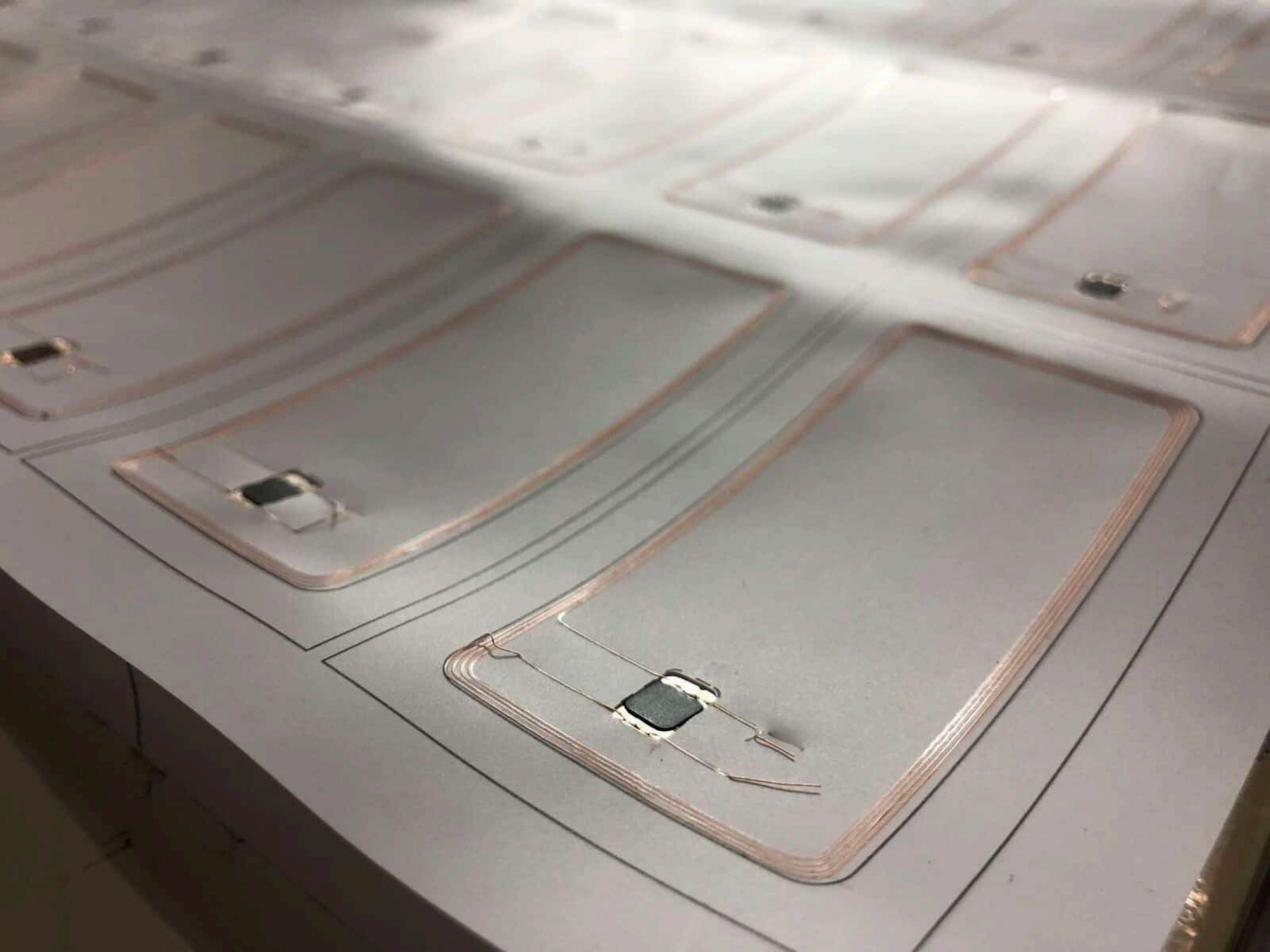
- Don't panic!
  - Rushing with security advisories can bring more damage than delaying them.
  - Experience shows quite a few entries are added in a hurry, contain incomplete information and are unlikely to be corrected due to a "problem closed" kind of syndrome.
- Read up a bit.
  - A security advisory made by someone who doesn't understand the first thing about it is a security hole in itself.
- Search VuXML for 2-3 (the more the better) previous entries affecting the package in question.
  - Chances are the previous entries affected other packages as well and you need to include them in the new entry.
  - Firefox is a good example where every other entry lacks half of affected packages.
- Don't ignore recently deleted packages.
  - If a package has been deleted within a few months ago, we can't let down all the users who still have it installed and trust portaudit.
- Respect format
  - If paragraphs in the entire vuln.xml file are wrapped at column 80 (or less) and your terminal has 160 columns, it doesn't mean we should all switch to your standards.

Practice

- Make sure you have security/vuxml installed.
- Check out security/vuxml into a working dir and cd to it.
- % make newentry
- % make validate
- submit a diff for review or commit it right away if you running low on pointyhats

NGINX
N


NETFLIX
CACHE FILL
OCA #1
PEER FILL
OCA #2

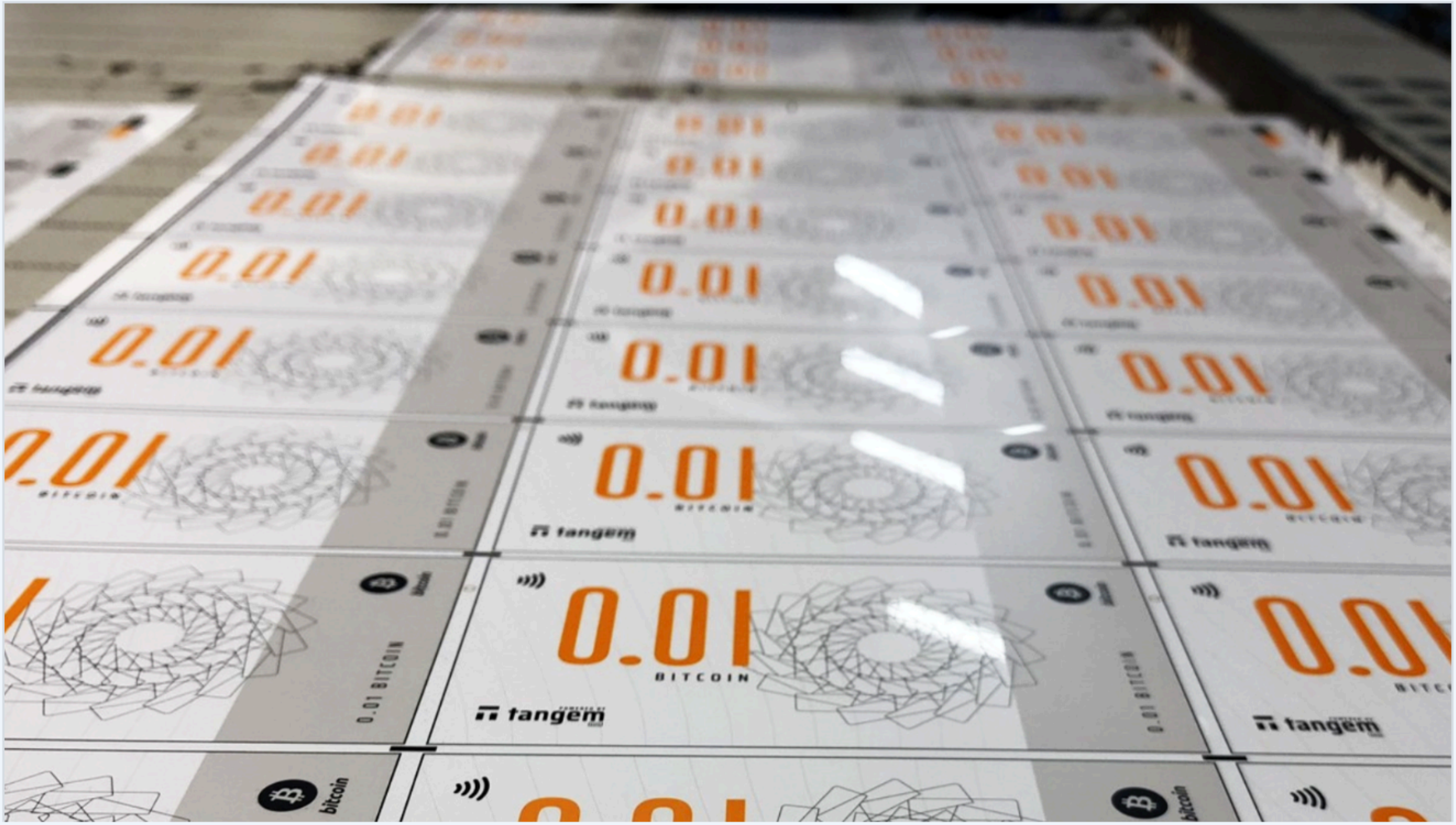


TECHNOLOGY
by **Kevin Helms**

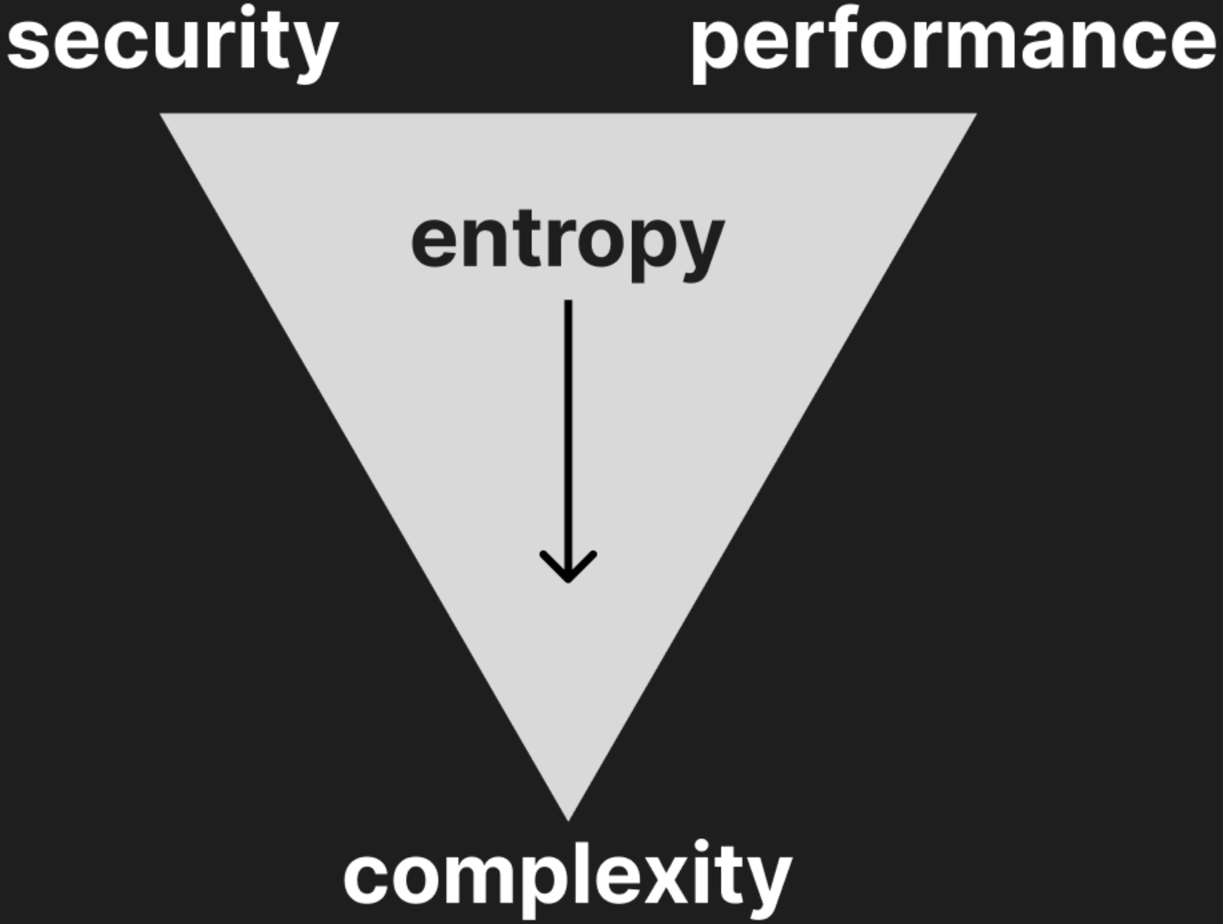
May 4, 2018

72776

Bitcoin Smart Banknotes Launched in Singapore



A digital asset smart banknote manufacturer has launched bitcoin banknotes at a store in Singapore. Designed to make owning and circulating cryptocurrencies as easy as using paper money, they are currently available in denominations of 0.01 and 0.05 BTC.



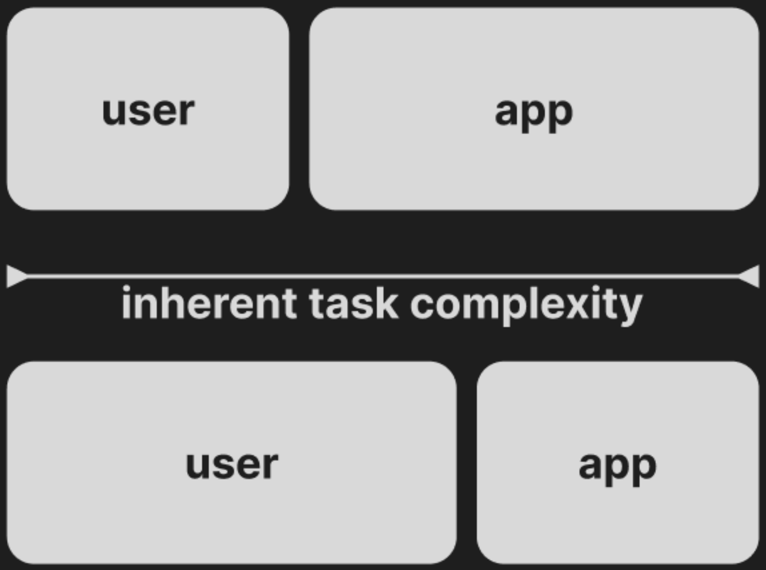
security

performance

entropy

complexity

Tesler’s Law



user

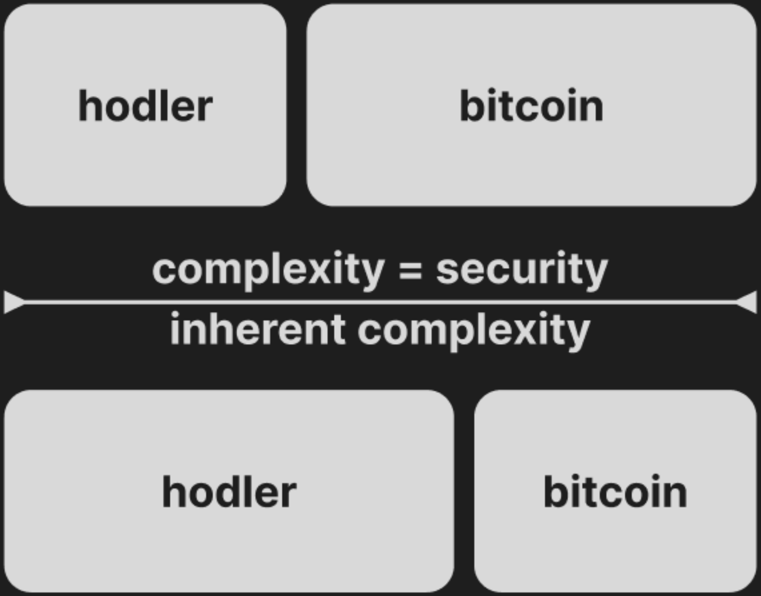
app

inherent task complexity

user

app

Tesler’s Law for Bitcoin



hodler

bitcoin

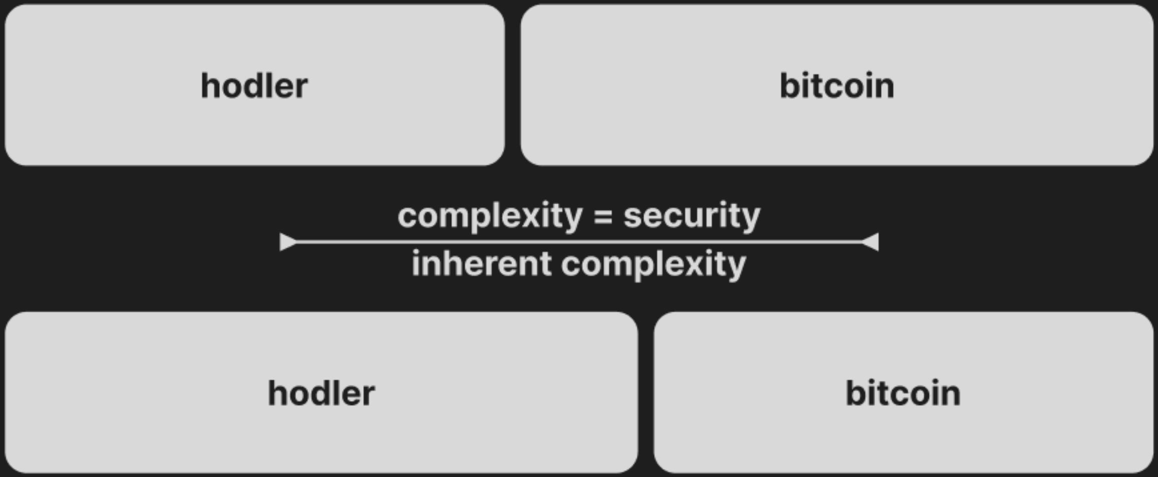
complexity = security

inherent complexity

hodler

bitcoin

Reality for Bitcoin



hodler

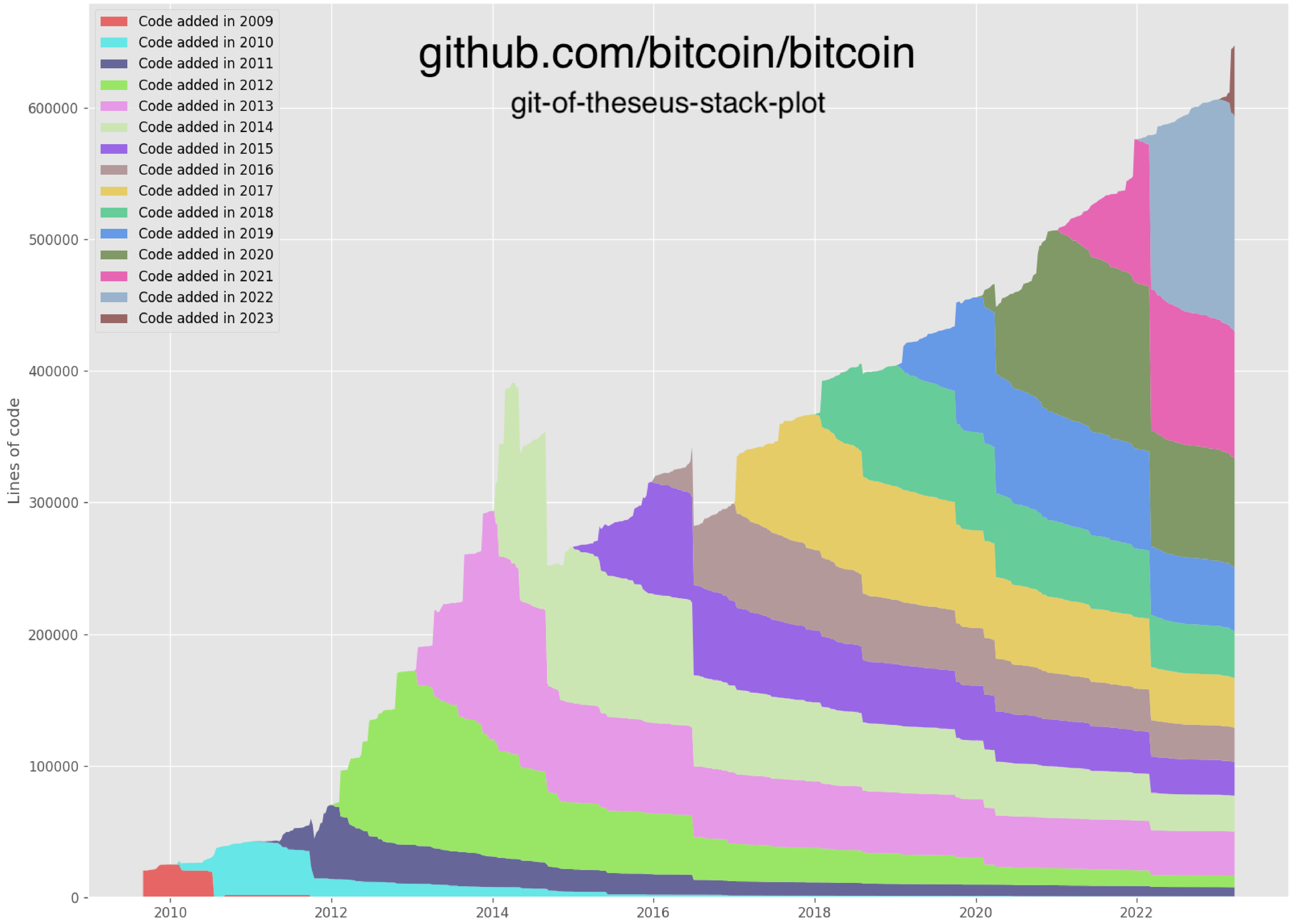
bitcoin

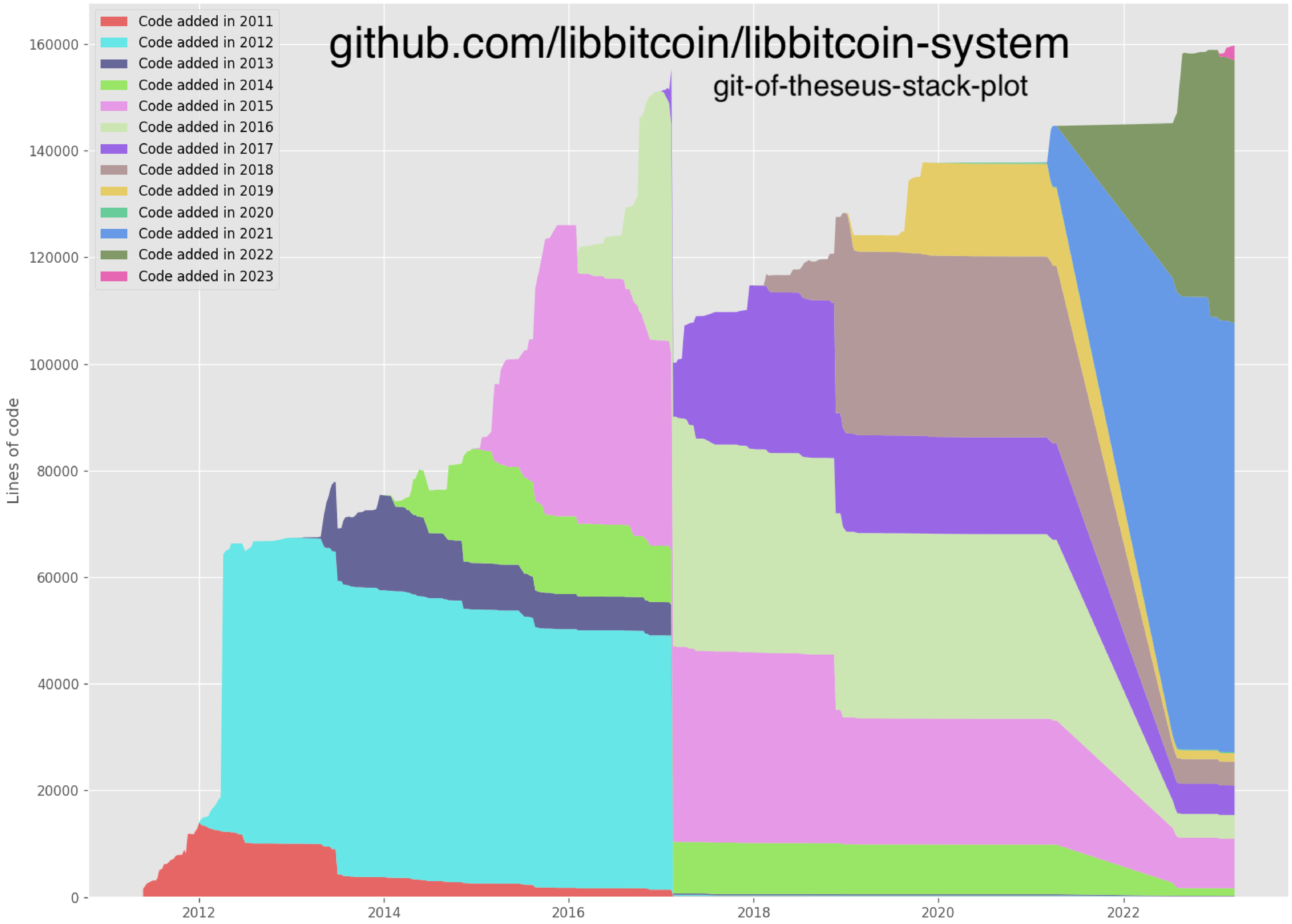
complexity = security

inherent complexity

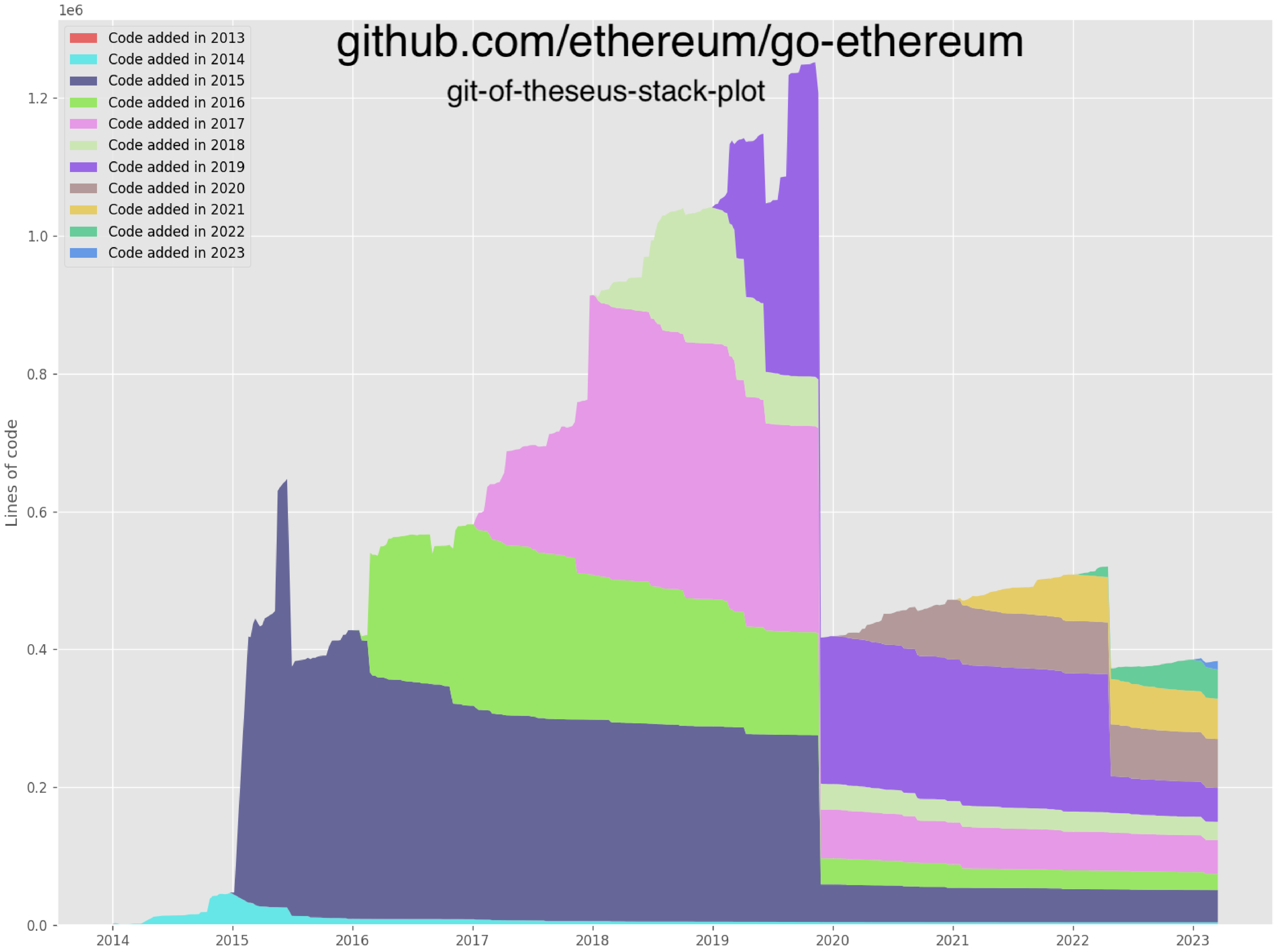
hodler

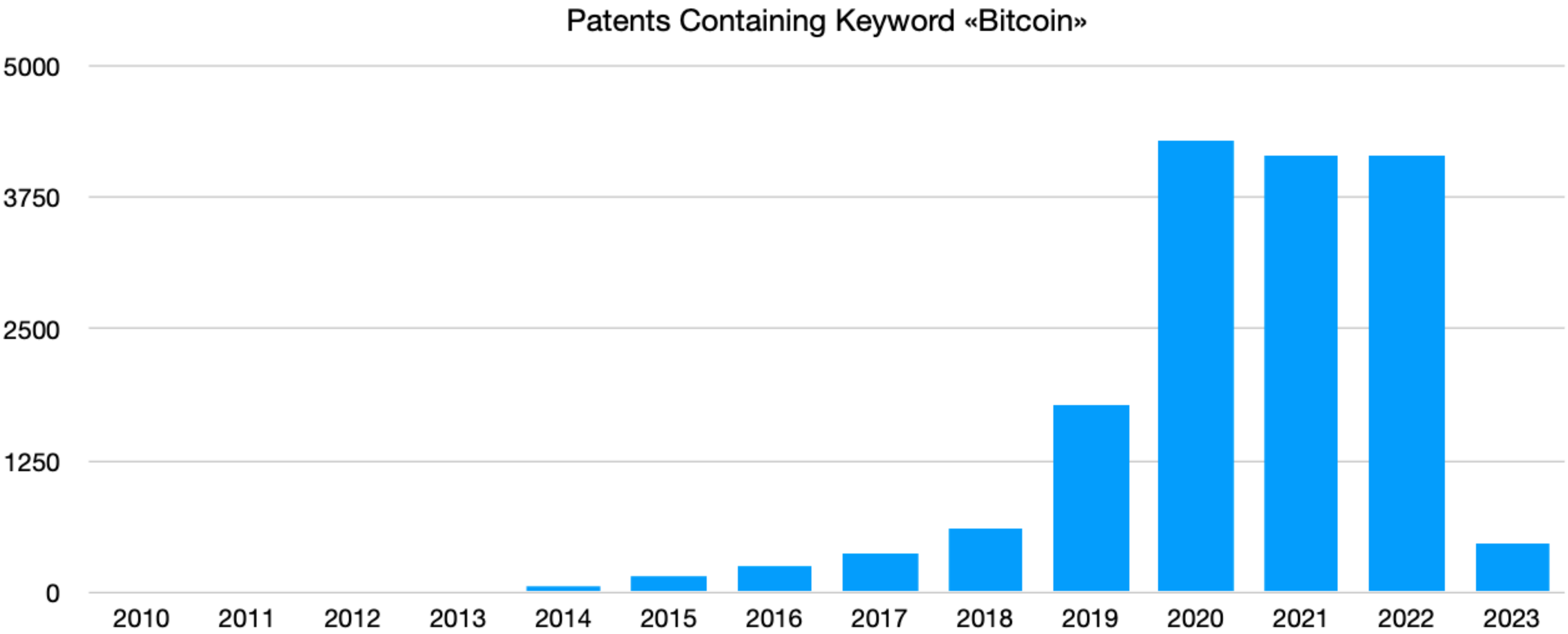
bitcoin


github.com/bitcoin/bitcoin
git-of-theseus-stack-plot
Code added in 2009
Code added in 2010
Code added in 2011
Code added in 2012
Code added in 2013
Code added in 2014
Code added in 2015
Code added in 2016
Code added in 2017
Code added in 2018
Code added in 2019
Code added in 2020
Code added in 2021
Code added in 2022
Code added in 2023
Lines of code
0
100000
200000
300000
400000
500000
600000
2010
2012
2014
2016
2018
2020
2022

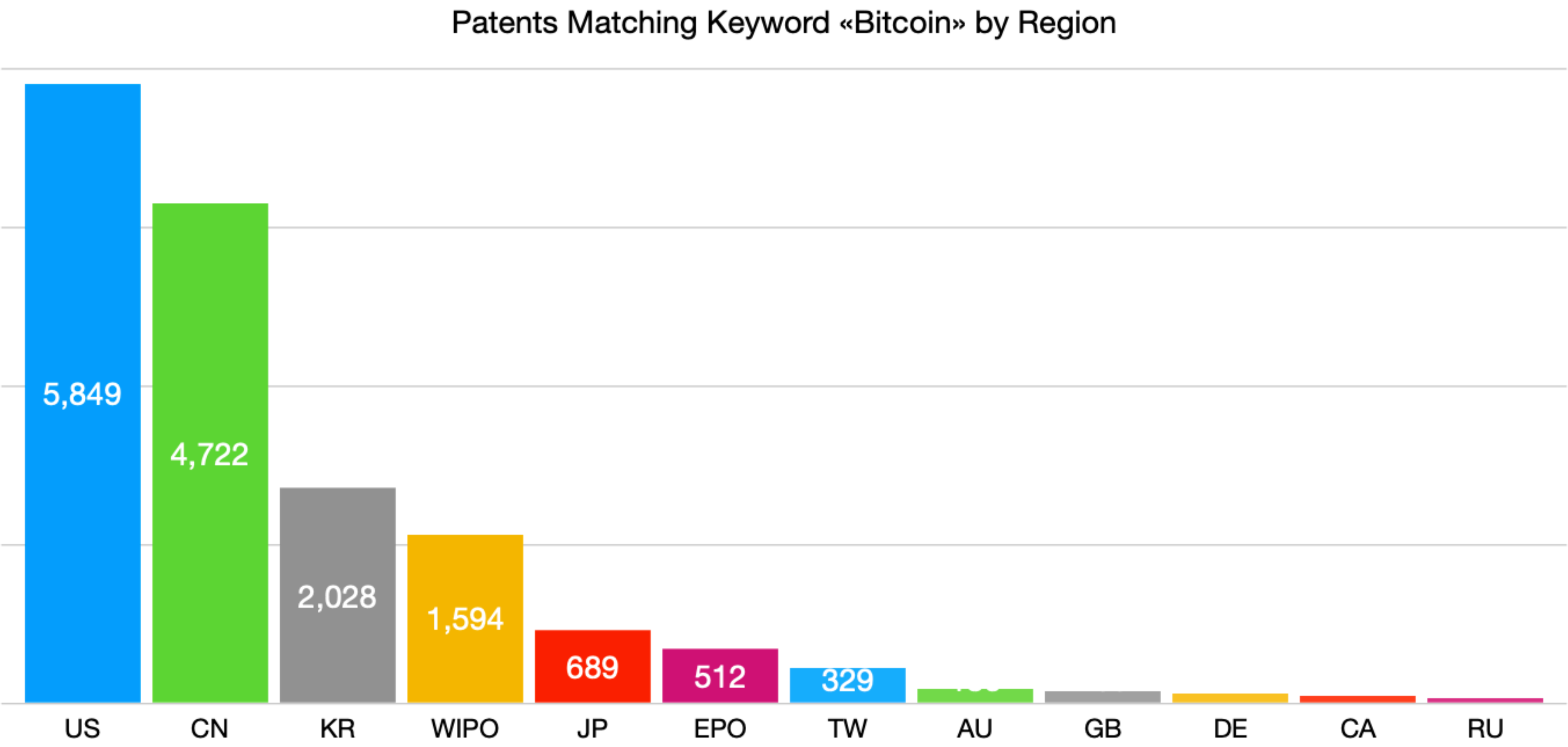


github.com/libbitcoin/libbitcoin-system
git-of-theseus-stack-plot
Code added in 2011
Code added in 2012
Code added in 2013
Code added in 2014
Code added in 2015
Code added in 2016
Code added in 2017
Code added in 2018
Code added in 2019
Code added in 2020
Code added in 2021
Code added in 2022
Code added in 2023
Lines of code
160000
140000
120000
100000
80000
60000
40000
20000
0
2012
2014
2016
2018
2020
2022

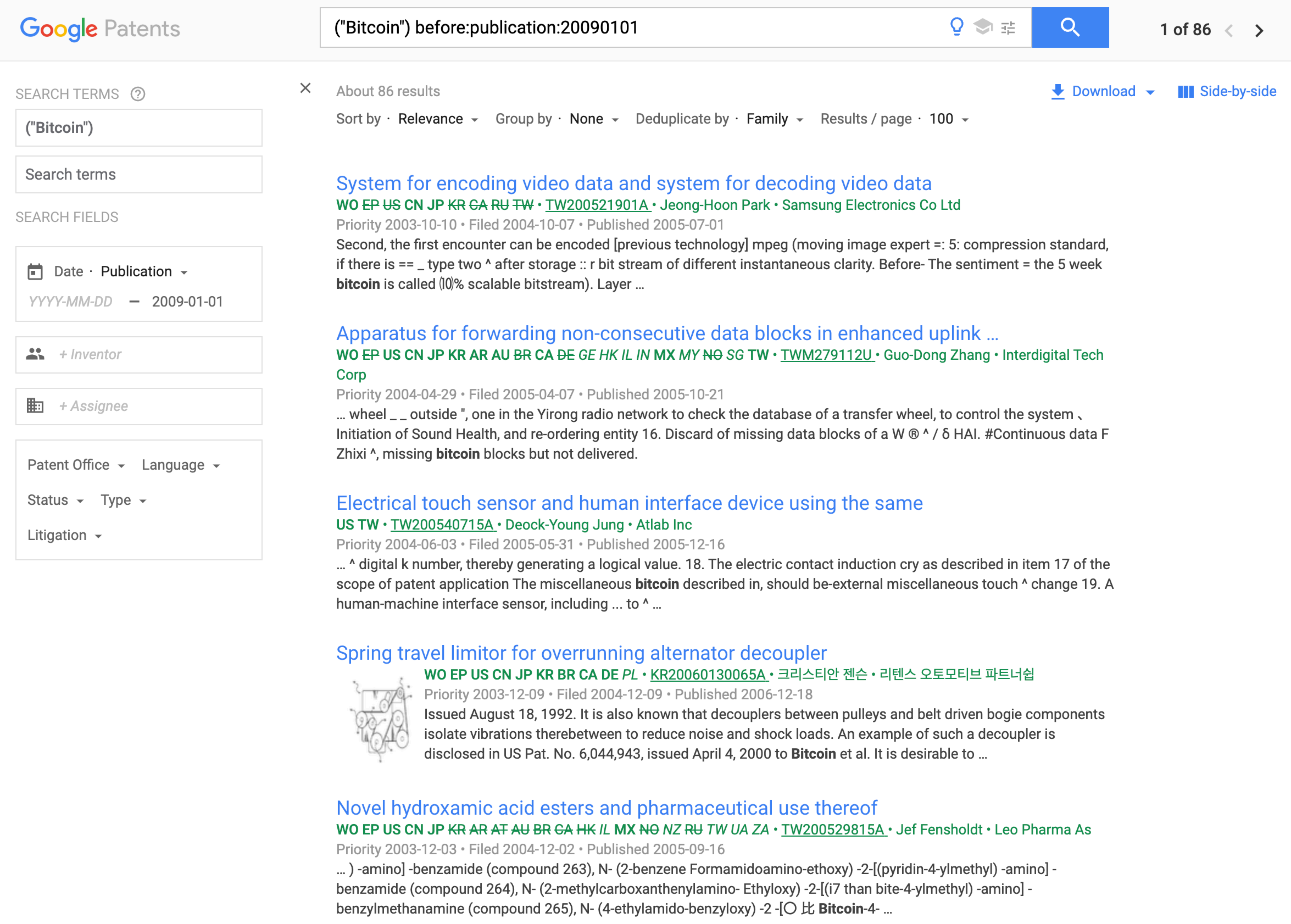

github.com/ethereum/go-ethereum
git-of-theseus-stack-plot
1e6
Lines of code
Code added in 2013
Code added in 2014
Code added in 2015
Code added in 2016
Code added in 2017
Code added in 2018
Code added in 2019
Code added in 2020
Code added in 2021
Code added in 2022
Code added in 2023
0.0
0.2
0.4
0.6
0.8
1.0
1.2
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023



Patents Containing Keyword «Bitcoin»



Patents Matching Keyword «Bitcoin» by Region



("Bitcoin") before:publication:20090101

SEARCH TERMS

("Bitcoin")

Search terms

SEARCH FIELDS

Date · Publication

YYYY-MM-DD — 2009-01-01

+ Inventor

+ Assignee

About 86 results

Sort by · Relevance   Group by · None   Deduplicate by · Family   Results / page · 100

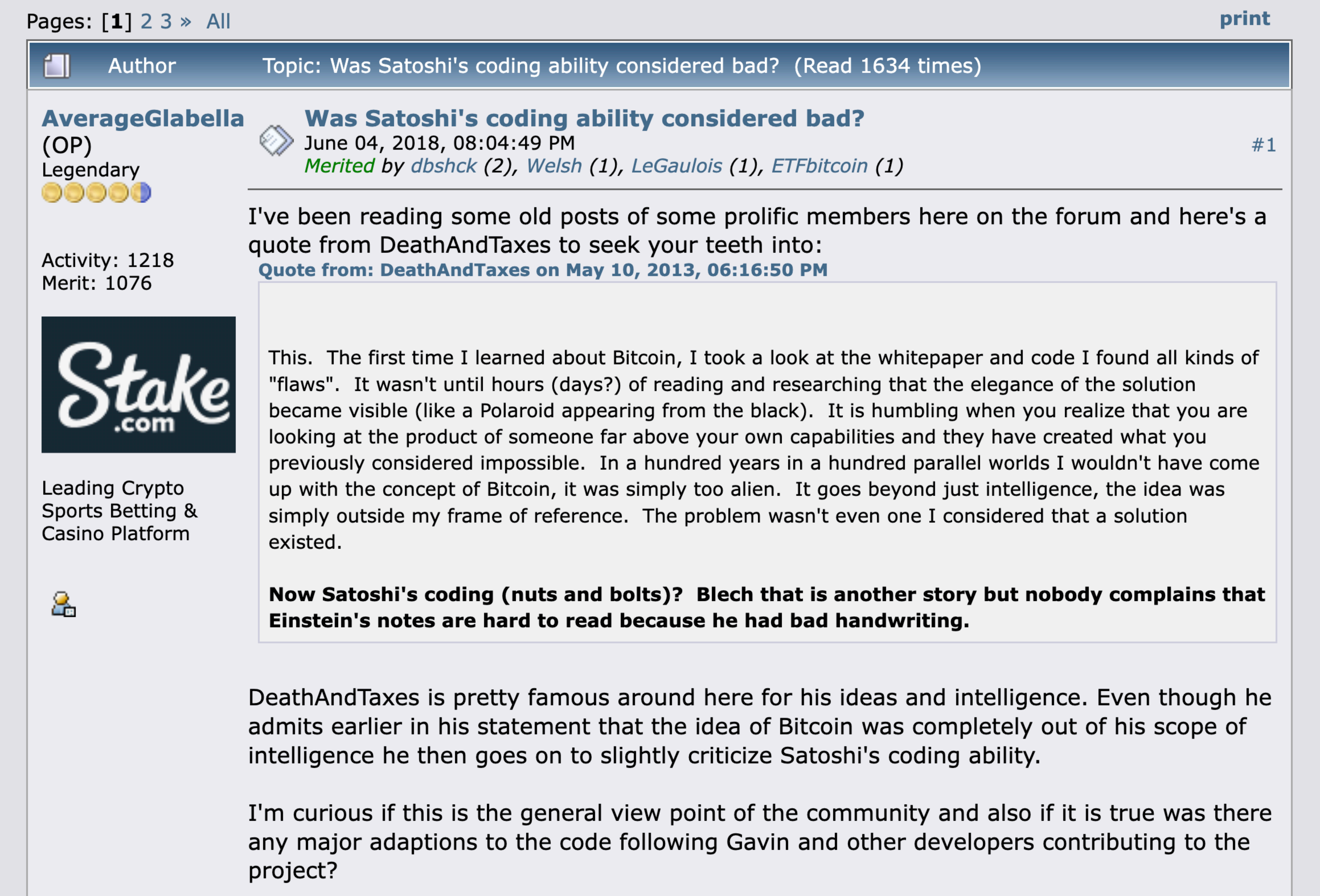
System for encoding video data and system for decoding video data
WO ~~EP~~ ~~US~~ CN JP ~~KR~~ ~~CA~~ ~~RU~~ ~~TW~~ • TW200521901A • Jeong-Hoon Park • Samsung Electronics Co Ltd
Priority 2003-10-10 • Filed 2004-10-07 • Published 2005-07-01
Second, the first encounter can be encoded [previous technology] mpeg (moving image expert =: 5: compression standard, if there is == _ type two ^ after storage :: r bit stream of different instantaneous clarity. Before- The sentiment = the 5 week **bitcoin** is called ⑽% scalable bitstream). Layer ...

Apparatus for forwarding non-consecutive data blocks in enhanced uplink ...
WO ~~EP~~ US CN JP KR AR AU ~~BR~~ CA ~~DE~~ *GE HK IL IN* MX *MY* ~~NO~~ *SG* TW • TWM279112U • Guo-Dong Zhang • Interdigital Tech Corp
Priority 2004-04-29 • Filed 2005-04-07 • Published 2005-10-21
... wheel _ _ outside ", one in the Yirong radio network to check the database of a transfer wheel, to control the system 、 Initiation of Sound Health, and re-ordering entity 16. Discard of missing data blocks of a W ® ^ / δ HAI. #Continuous data F Zhixi ^, missing **bitcoin** blocks but not delivered.

Electrical touch sensor and human interface device using the same
US TW • TW200540715A • Deock-Young Jung • Atlab Inc
Priority 2004-06-03 • Filed 2005-05-31 • Published 2005-12-16
... ^ digital k number, thereby generating a logical value. 18. The electric contact induction cry as described in item 17 of the scope of patent application The miscellaneous **bitcoin** described in, should be-external miscellaneous touch ^ change 19. A human-machine interface sensor, including ... to ^ ...

Spring travel limitor for overrunning alternator decoupler
WO EP US CN JP KR BR CA DE *PL* • KR20060130065A • 크리스티안 젠슨 • 리텐스 오토모티브 파트너쉽
Priority 2003-12-09 • Filed 2004-12-09 • Published 2006-12-18
Issued August 18, 1992. It is also known that decouplers between pulleys and belt driven bogie components isolate vibrations therebetween to reduce noise and shock loads. An example of such a decoupler is disclosed in US Pat. No. 6,044,943, issued April 4, 2000 to **Bitcoin** et al. It is desirable to ...

Novel hydroxamic acid esters and pharmaceutical use thereof
WO EP US CN JP ~~KR~~ ~~AR~~ ~~AT~~ ~~AU~~ ~~BR~~ ~~CA~~ ~~HK~~ *IL* MX ~~NO~~ *NZ* ~~RU~~ *TW UA ZA* • TW200529815A • Jef Fensholdt • Leo Pharma As
Priority 2003-12-03 • Filed 2004-12-02 • Published 2005-09-16
...) -amino] -benzamide (compound 263), N- (2-benzene Formamidoamino-ethoxy) -2-[(pyridin-4-ylmethyl) -amino] -benzamide (compound 264), N- (2-methylcarboxanthenylamino- Ethyloxy) -2-[(i7 than bite-4-ylmethyl) -amino] -benzylmethanamine (compound 265), N- (4-ethylamido-benzyloxy) -2 -[O 比 **Bitcoin**-4- ...



Pages: [**1**] 2 3 » All

print

Author | Topic: Was Satoshi's coding ability considered bad? (Read 1634 times)

AverageGlabella (OP)
Legendary

Activity: 1218
Merit: 1076

Was Satoshi's coding ability considered bad?
June 04, 2018, 08:04:49 PM
Merited by dbshck (2), Welsh (1), LeGaulois (1), ETFbitcoin (1)

#1

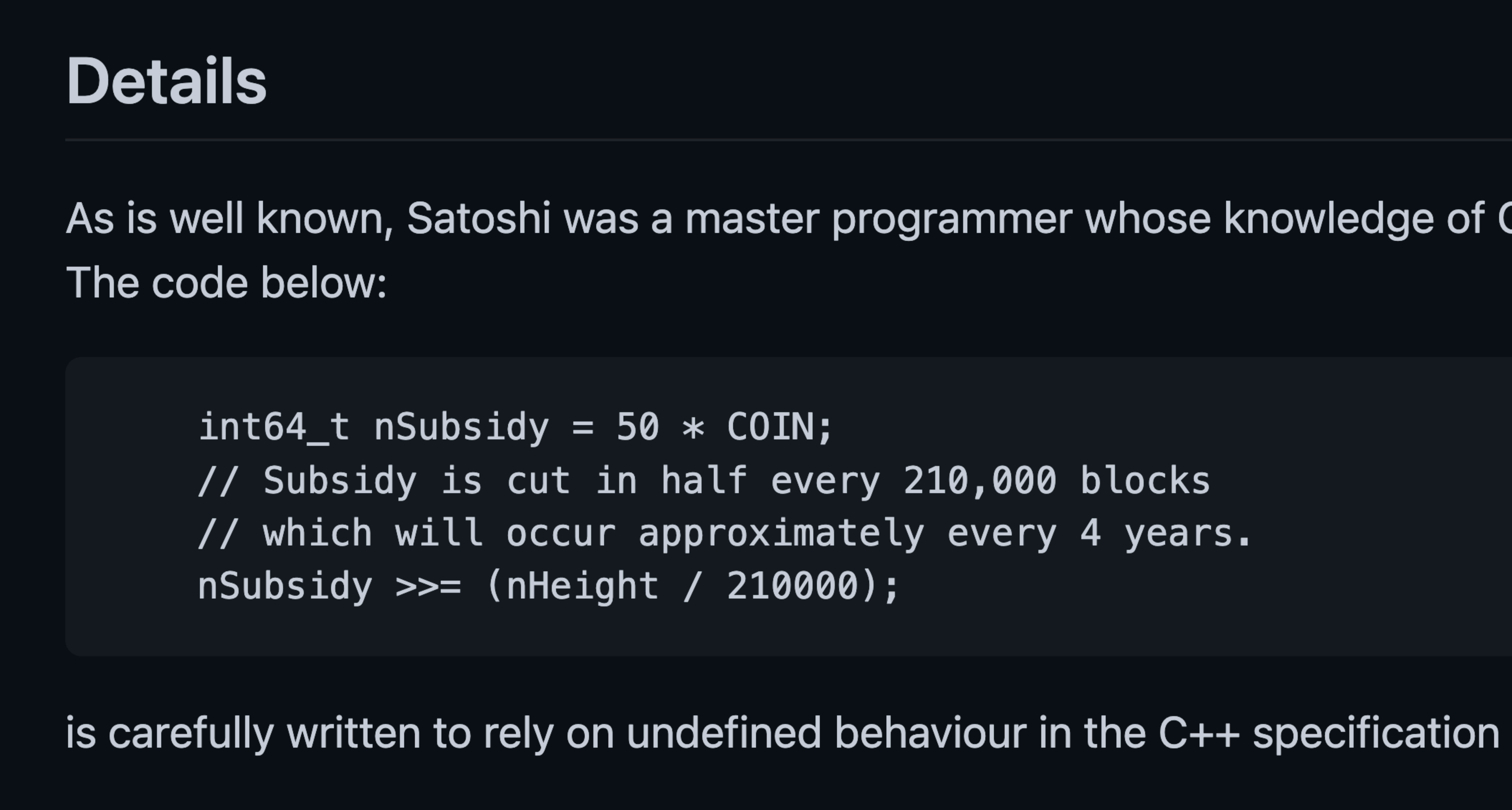
I've been reading some old posts of some prolific members here on the forum and here's a quote from DeathAndTaxes to seek your teeth into:

Quote from: DeathAndTaxes on May 10, 2013, 06:16:50 PM

> This. The first time I learned about Bitcoin, I took a look at the whitepaper and code I found all kinds of "flaws". It wasn't until hours (days?) of reading and researching that the elegance of the solution became visible (like a Polaroid appearing from the black). It is humbling when you realize that you are looking at the product of someone far above your own capabilities and they have created what you previously considered impossible. In a hundred years in a hundred parallel worlds I wouldn't have come up with the concept of Bitcoin, it was simply too alien. It goes beyond just intelligence, the idea was simply outside my frame of reference. The problem wasn't even one I considered that a solution existed.
>
> **Now Satoshi's coding (nuts and bolts)? Blech that is another story but nobody complains that Einstein's notes are hard to read because he had bad handwriting.**

DeathAndTaxes is pretty famous around here for his ideas and intelligence. Even though he admits earlier in his statement that the idea of Bitcoin was completely out of his scope of intelligence he then goes on to slightly criticize Satoshi's coding ability.

I'm curious if this is the general view point of the community and also if it is true was there any major adaptions to the code following Gavin and other developers contributing to the project?



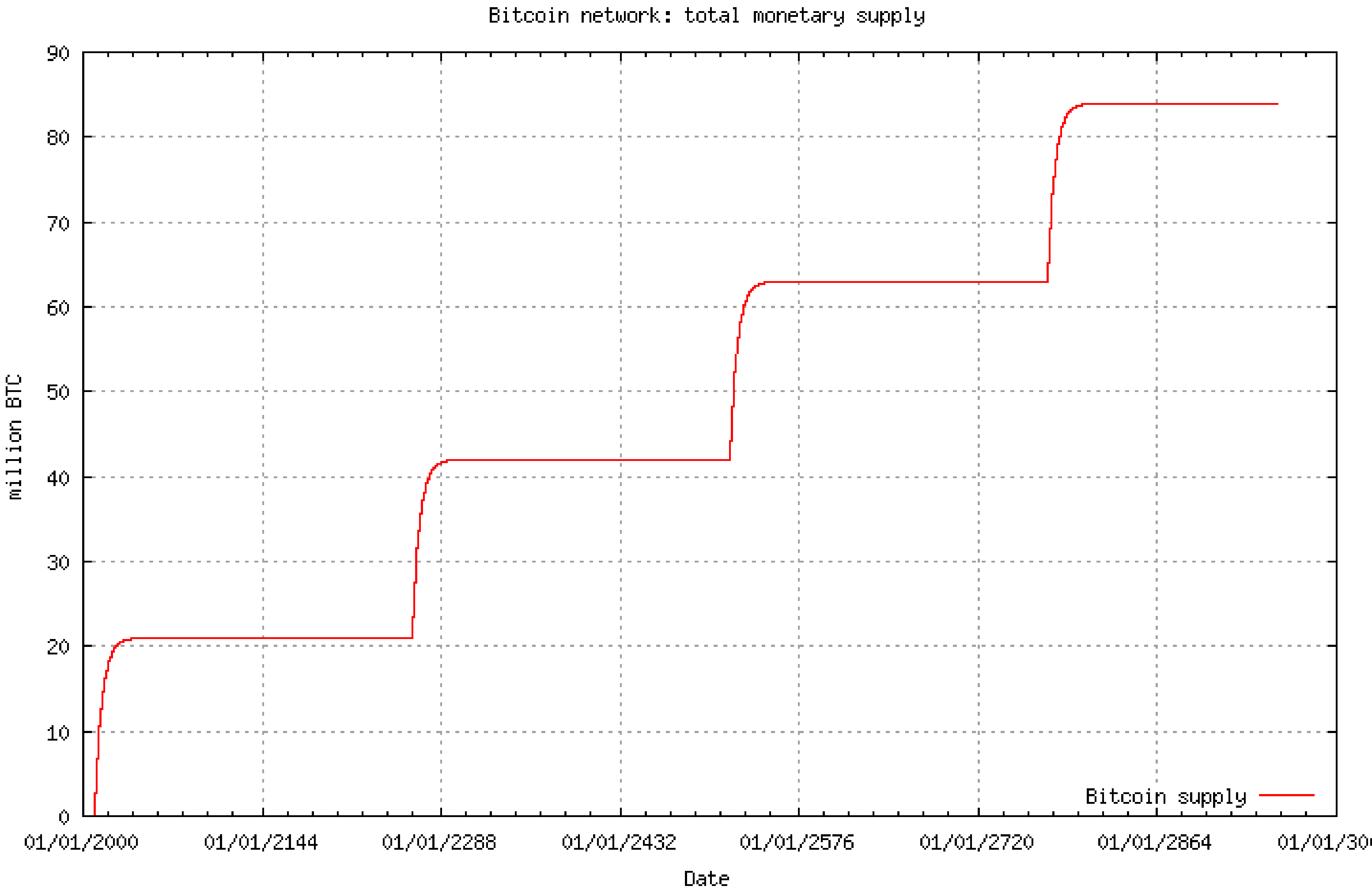
Details

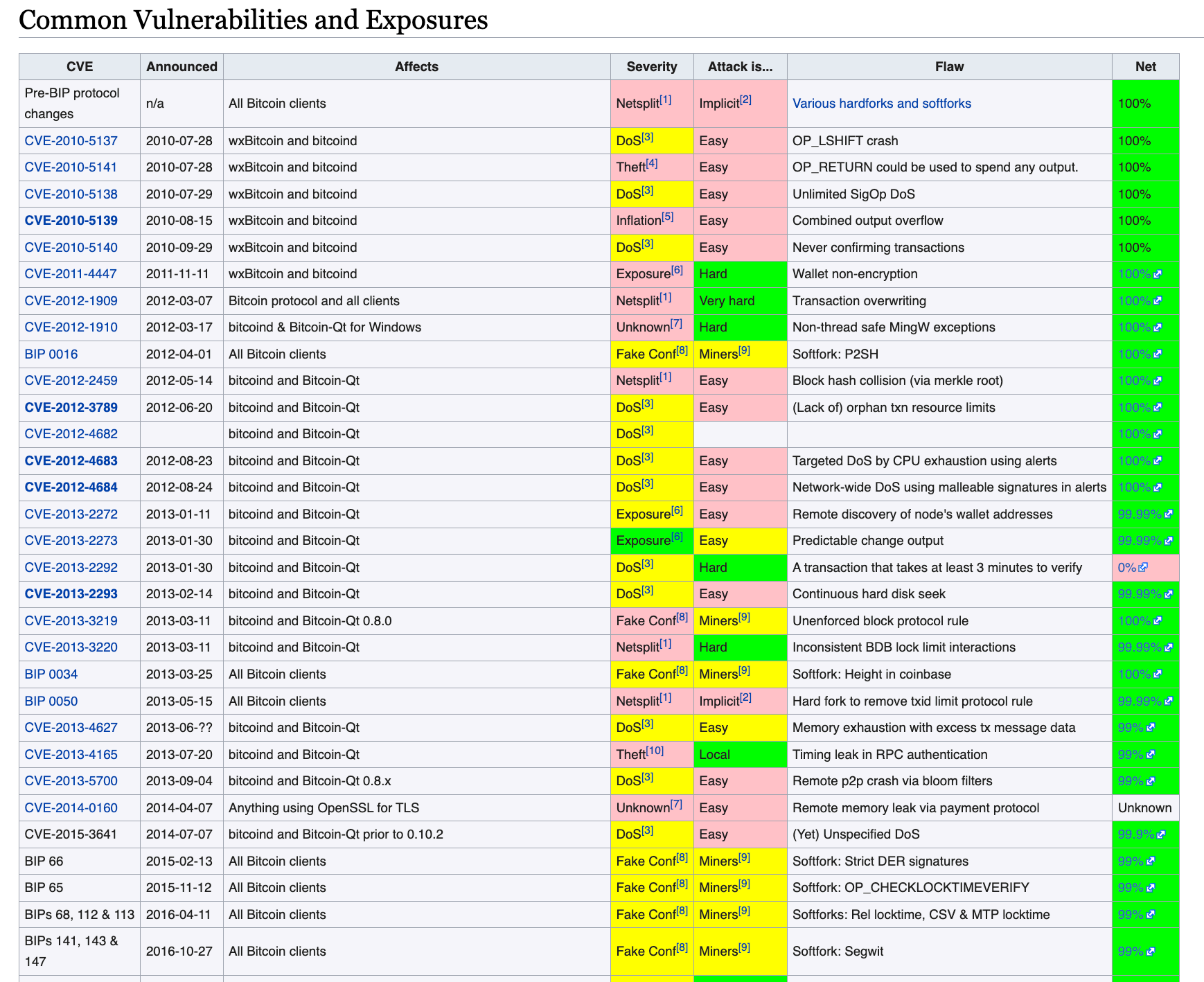
As is well known, Satoshi was a master programmer whose knowledge of C

The code below:

```
    int64_t nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks
    // which will occur approximately every 4 years.
    nSubsidy >>= (nHeight / 210000);
```

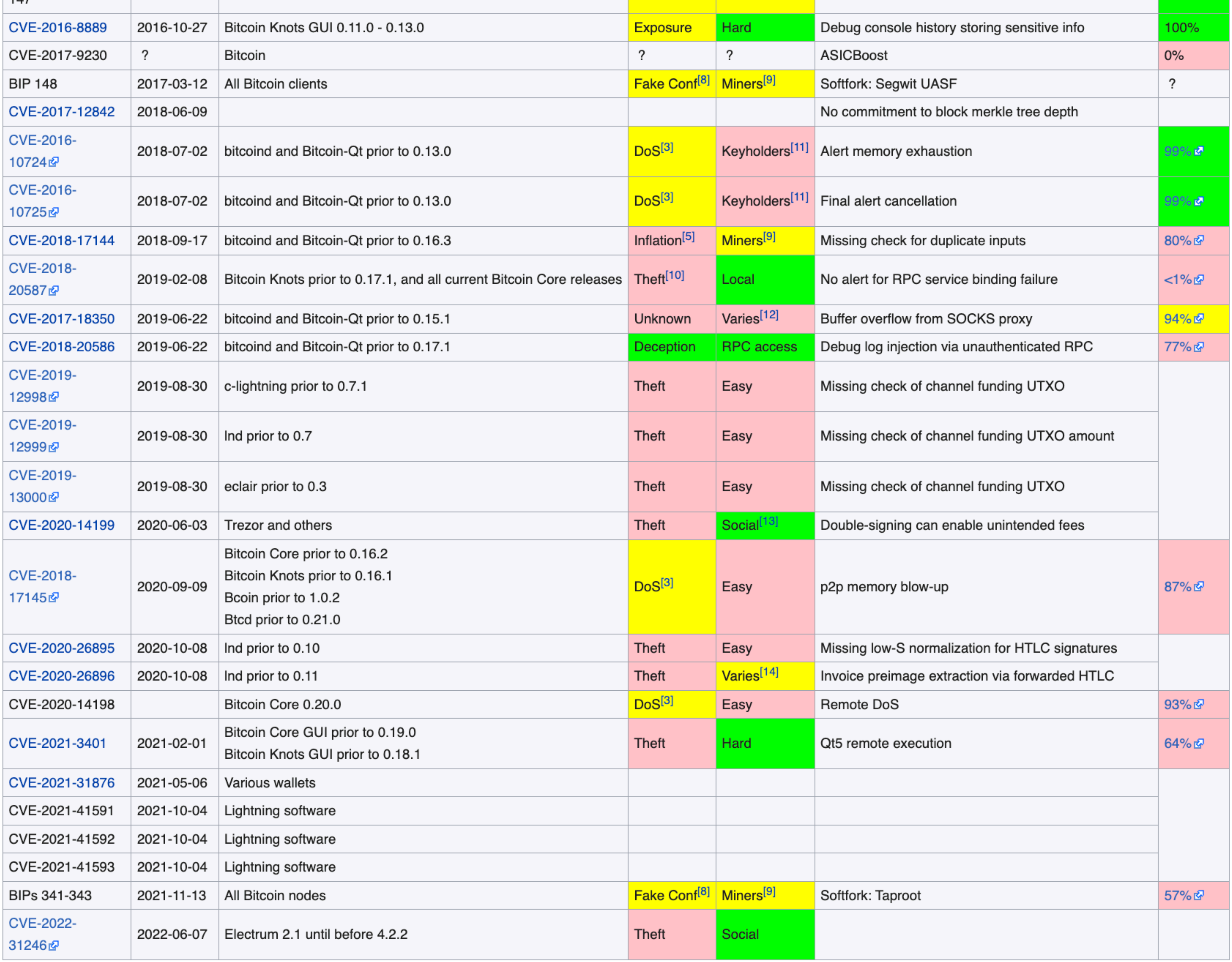
is carefully written to rely on undefined behaviour in the C++ specification


Bitcoin network: total monetary supply
million BTC
0
10
20
30
40
50
60
70
80
90
01/01/2000
01/01/2144
01/01/2288
01/01/2432
01/01/2576
01/01/2720
01/01/2864
01/01/30
Date
Bitcoin supply



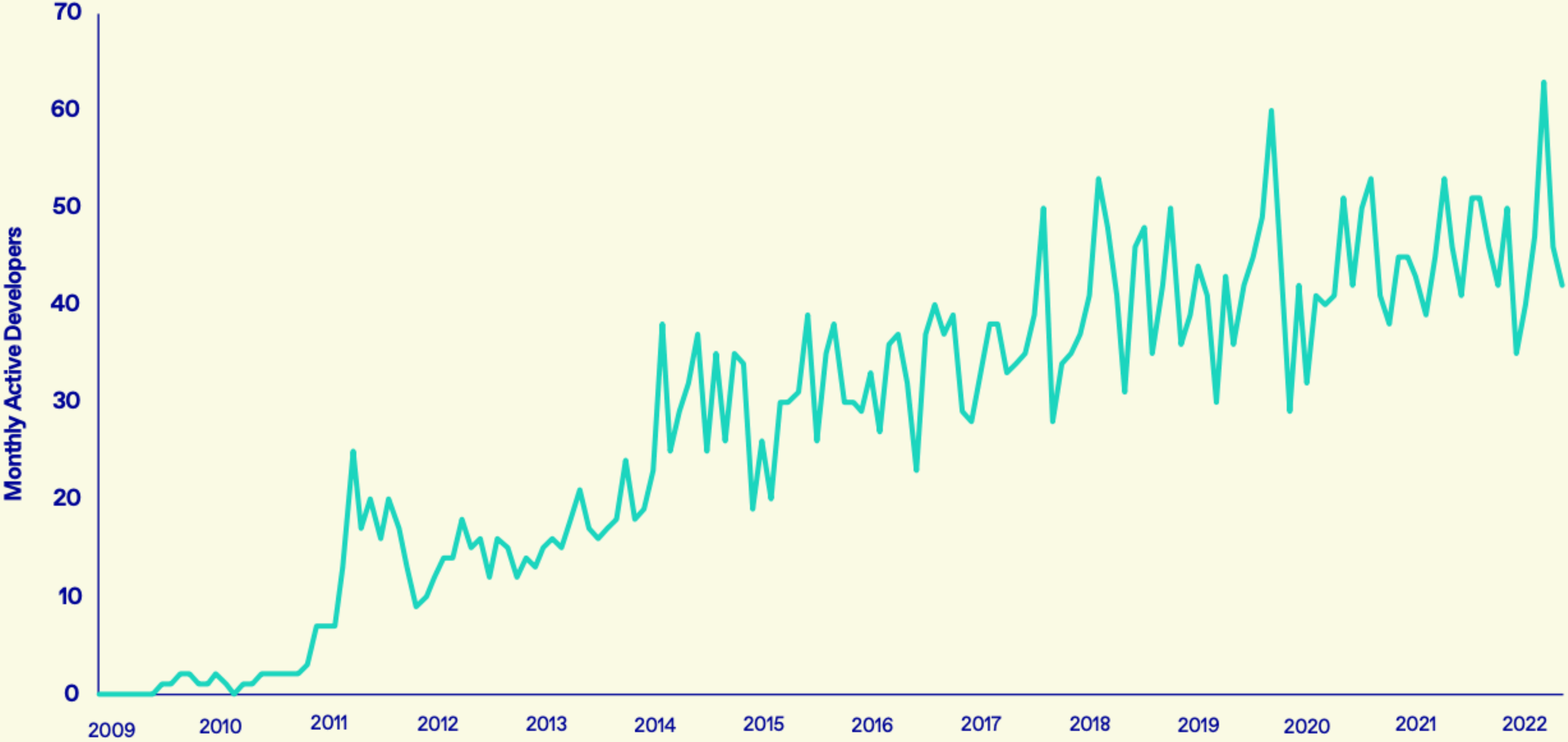
Common Vulnerabilities and Exposures

CVE	Announced	Affects	Severity	Attack is...	Flaw	Net
Pre-BIP protocol changes	n/a	All Bitcoin clients	Netsplit[1]	Implicit[2]	Various hardforks and softforks	100%
CVE-2010-5137	2010-07-28	wxBitcoin and bitcoind	DoS[3]	Easy	OP_LSHIFT crash	100%
CVE-2010-5141	2010-07-28	wxBitcoin and bitcoind	Theft[4]	Easy	OP_RETURN could be used to spend any output.	100%
CVE-2010-5138	2010-07-29	wxBitcoin and bitcoind	DoS[3]	Easy	Unlimited SigOp DoS	100%
CVE-2010-5139	2010-08-15	wxBitcoin and bitcoind	Inflation[5]	Easy	Combined output overflow	100%
CVE-2010-5140	2010-09-29	wxBitcoin and bitcoind	DoS[3]	Easy	Never confirming transactions	100%
CVE-2011-4447	2011-11-11	wxBitcoin and bitcoind	Exposure[6]	Hard	Wallet non-encryption	100%
CVE-2012-1909	2012-03-07	Bitcoin protocol and all clients	Netsplit[1]	Very hard	Transaction overwriting	100%
CVE-2012-1910	2012-03-17	bitcoind & Bitcoin-Qt for Windows	Unknown[7]	Hard	Non-thread safe MingW exceptions	100%
BIP 0016	2012-04-01	All Bitcoin clients	Fake Conf[8]	Miners[9]	Softfork: P2SH	100%
CVE-2012-2459	2012-05-14	bitcoind and Bitcoin-Qt	Netsplit[1]	Easy	Block hash collision (via merkle root)	100%
CVE-2012-3789	2012-06-20	bitcoind and Bitcoin-Qt	DoS[3]	Easy	(Lack of) orphan txn resource limits	100%
CVE-2012-4682		bitcoind and Bitcoin-Qt	DoS[3]			100%
CVE-2012-4683	2012-08-23	bitcoind and Bitcoin-Qt	DoS[3]	Easy	Targeted DoS by CPU exhaustion using alerts	100%
CVE-2012-4684	2012-08-24	bitcoind and Bitcoin-Qt	DoS[3]	Easy	Network-wide DoS using malleable signatures in alerts	100%
CVE-2013-2272	2013-01-11	bitcoind and Bitcoin-Qt	Exposure[6]	Easy	Remote discovery of node's wallet addresses	99.99%
CVE-2013-2273	2013-01-30	bitcoind and Bitcoin-Qt	Exposure[6]	Easy	Predictable change output	99.99%
CVE-2013-2292	2013-01-30	bitcoind and Bitcoin-Qt	DoS[3]	Hard	A transaction that takes at least 3 minutes to verify	0%
CVE-2013-2293	2013-02-14	bitcoind and Bitcoin-Qt	DoS[3]	Easy	Continuous hard disk seek	99.99%
CVE-2013-3219	2013-03-11	bitcoind and Bitcoin-Qt 0.8.0	Fake Conf[8]	Miners[9]	Unenforced block protocol rule	100%
CVE-2013-3220	2013-03-11	bitcoind and Bitcoin-Qt	Netsplit[1]	Hard	Inconsistent BDB lock limit interactions	99.99%
BIP 0034	2013-03-25	All Bitcoin clients	Fake Conf[8]	Miners[9]	Softfork: Height in coinbase	100%
BIP 0050	2013-05-15	All Bitcoin clients	Netsplit[1]	Implicit[2]	Hard fork to remove txid limit protocol rule	99.99%
CVE-2013-4627	2013-06-??	bitcoind and Bitcoin-Qt	DoS[3]	Easy	Memory exhaustion with excess tx message data	99%
CVE-2013-4165	2013-07-20	bitcoind and Bitcoin-Qt	Theft[10]	Local	Timing leak in RPC authentication	99%
CVE-2013-5700	2013-09-04	bitcoind and Bitcoin-Qt 0.8.x	DoS[3]	Easy	Remote p2p crash via bloom filters	99%
CVE-2014-0160	2014-04-07	Anything using OpenSSL for TLS	Unknown[7]	Easy	Remote memory leak via payment protocol	Unknown
CVE-2015-3641	2014-07-07	bitcoind and Bitcoin-Qt prior to 0.10.2	DoS[3]	Easy	(Yet) Unspecified DoS	99.9%
BIP 66	2015-02-13	All Bitcoin clients	Fake Conf[8]	Miners[9]	Softfork: Strict DER signatures	99%
BIP 65	2015-11-12	All Bitcoin clients	Fake Conf[8]	Miners[9]	Softfork: OP_CHECKLOCKTIMEVERIFY	99%
BIPs 68, 112 & 113	2016-04-11	All Bitcoin clients	Fake Conf[8]	Miners[9]	Softforks: Rel locktime, CSV & MTP locktime	99%
BIPs 141, 143 & 147	2016-10-27	All Bitcoin clients	Fake Conf[8]	Miners[9]	Softfork: Segwit	99%

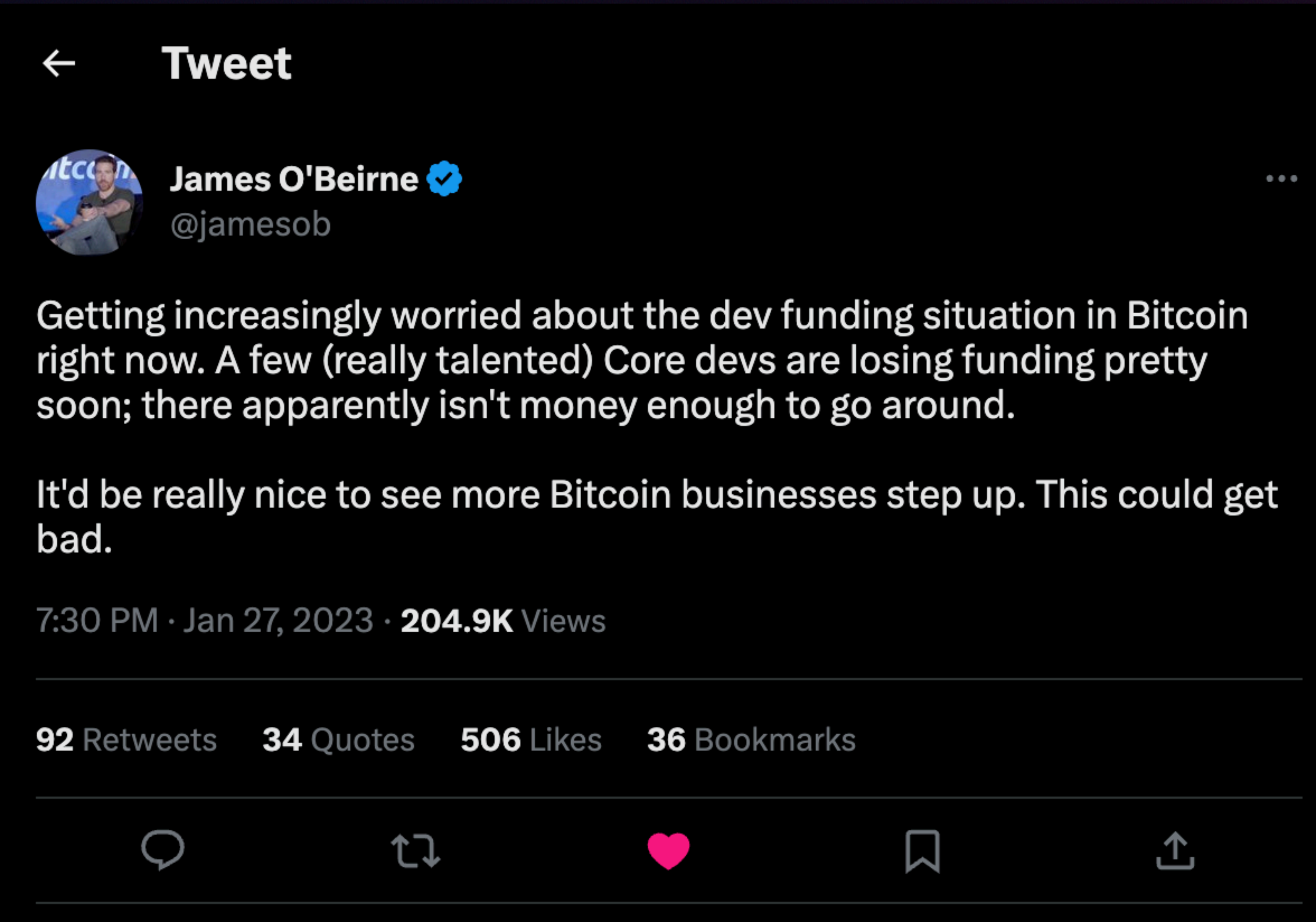


147						
CVE-2016-8889	2016-10-27	Bitcoin Knots GUI 0.11.0 - 0.13.0	Exposure	Hard	Debug console history storing sensitive info	100%
CVE-2017-9230	?	Bitcoin	?	?	ASICBoost	0%
BIP 148	2017-03-12	All Bitcoin clients	Fake Conf[8]	Miners[9]	Softfork: Segwit UASF	?
CVE-2017-12842	2018-06-09				No commitment to block merkle tree depth	
CVE-2016-10724	2018-07-02	bitcoind and Bitcoin-Qt prior to 0.13.0	DoS[3]	Keyholders[11]	Alert memory exhaustion	99%
CVE-2016-10725	2018-07-02	bitcoind and Bitcoin-Qt prior to 0.13.0	DoS[3]	Keyholders[11]	Final alert cancellation	99%
CVE-2018-17144	2018-09-17	bitcoind and Bitcoin-Qt prior to 0.16.3	Inflation[5]	Miners[9]	Missing check for duplicate inputs	80%
CVE-2018-20587	2019-02-08	Bitcoin Knots prior to 0.17.1, and all current Bitcoin Core releases	Theft[10]	Local	No alert for RPC service binding failure	<1%
CVE-2017-18350	2019-06-22	bitcoind and Bitcoin-Qt prior to 0.15.1	Unknown	Varies[12]	Buffer overflow from SOCKS proxy	94%
CVE-2018-20586	2019-06-22	bitcoind and Bitcoin-Qt prior to 0.17.1	Deception	RPC access	Debug log injection via unauthenticated RPC	77%
CVE-2019-12998	2019-08-30	c-lightning prior to 0.7.1	Theft	Easy	Missing check of channel funding UTXO	
CVE-2019-12999	2019-08-30	lnd prior to 0.7	Theft	Easy	Missing check of channel funding UTXO amount	
CVE-2019-13000	2019-08-30	eclair prior to 0.3	Theft	Easy	Missing check of channel funding UTXO	
CVE-2020-14199	2020-06-03	Trezor and others	Theft	Social[13]	Double-signing can enable unintended fees	
CVE-2018-17145	2020-09-09	Bitcoin Core prior to 0.16.2 Bitcoin Knots prior to 0.16.1 Bcoin prior to 1.0.2 Btcd prior to 0.21.0	DoS[3]	Easy	p2p memory blow-up	87%
CVE-2020-26895	2020-10-08	lnd prior to 0.10	Theft	Easy	Missing low-S normalization for HTLC signatures	
CVE-2020-26896	2020-10-08	lnd prior to 0.11	Theft	Varies[14]	Invoice preimage extraction via forwarded HTLC	
CVE-2020-14198		Bitcoin Core 0.20.0	DoS[3]	Easy	Remote DoS	93%
CVE-2021-3401	2021-02-01	Bitcoin Core GUI prior to 0.19.0 Bitcoin Knots GUI prior to 0.18.1	Theft	Hard	Qt5 remote execution	64%
CVE-2021-31876	2021-05-06	Various wallets				
CVE-2021-41591	2021-10-04	Lightning software				
CVE-2021-41592	2021-10-04	Lightning software				
CVE-2021-41593	2021-10-04	Lightning software				
BIPs 341-343	2021-11-13	All Bitcoin nodes	Fake Conf[8]	Miners[9]	Softfork: Taproot	57%
CVE-2022-31246	2022-06-07	Electrum 2.1 until before 4.2.2	Theft	Social		

MONTHLY ACTIVE DEVELOPERS IN CORE PROTOCOL



Source: Github



James O'Beirne
@jamesob

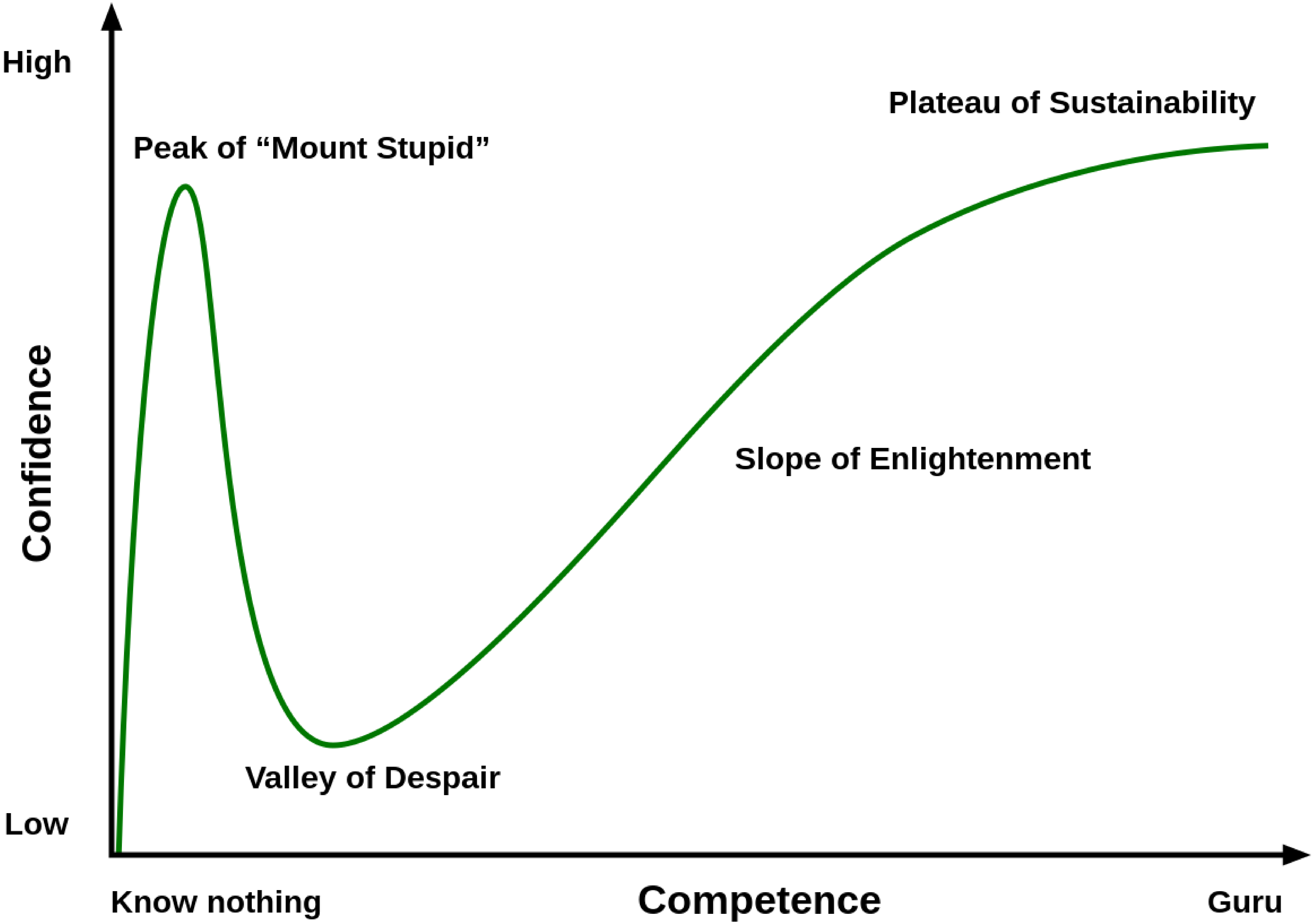
Getting increasingly worried about the dev funding situation in Bitcoin right now. A few (really talented) Core devs are losing funding pretty soon; there apparently isn't money enough to go around.

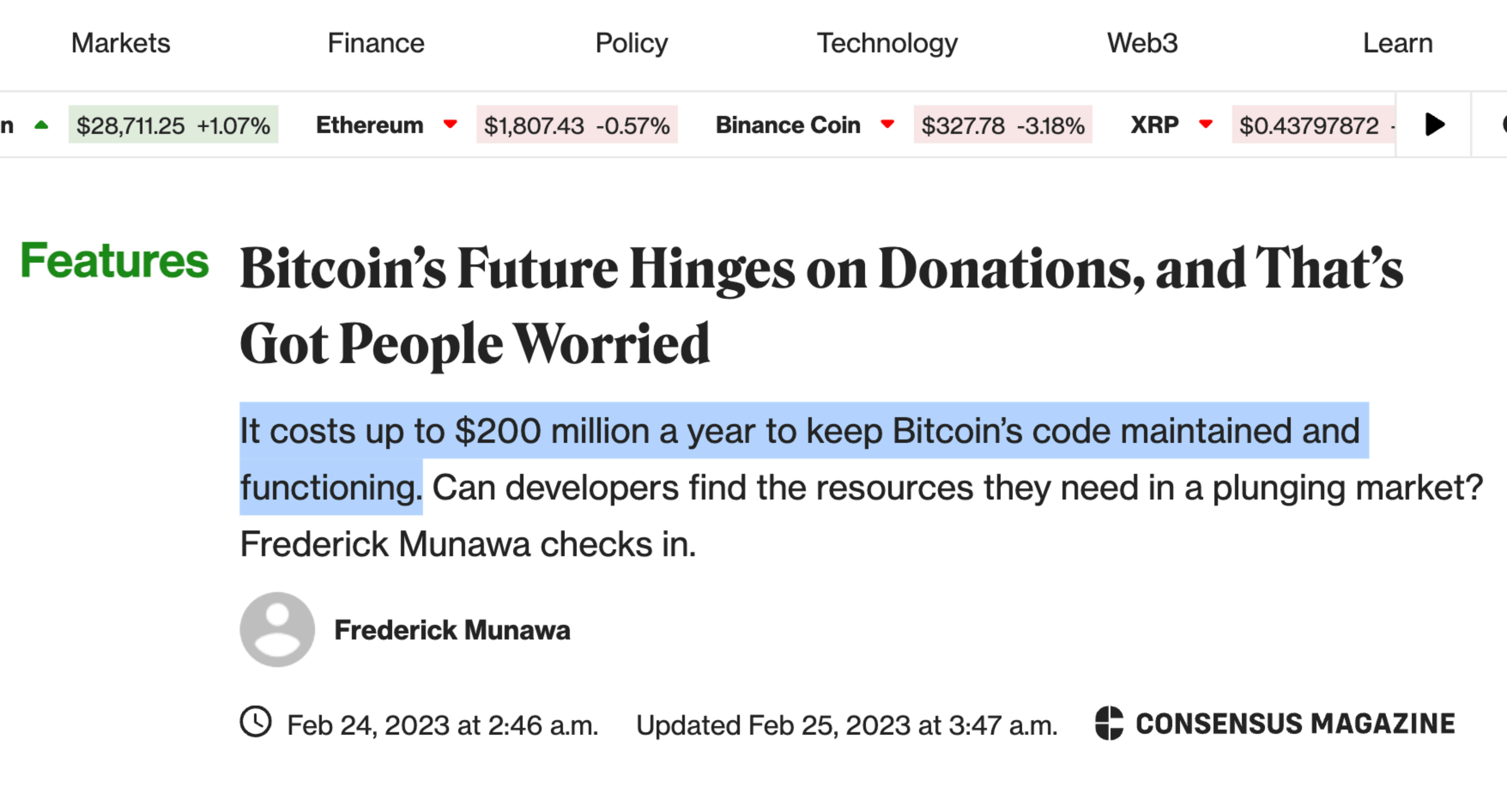
It'd be really nice to see more Bitcoin businesses step up. This could get bad.

7:30 PM · Jan 27, 2023 · **204.9K** Views

92 Retweets **34** Quotes **506** Likes **36** Bookmarks

Dunning–Kruger Effect





Features

Bitcoin’s Future Hinges on Donations, and That’s Got People Worried

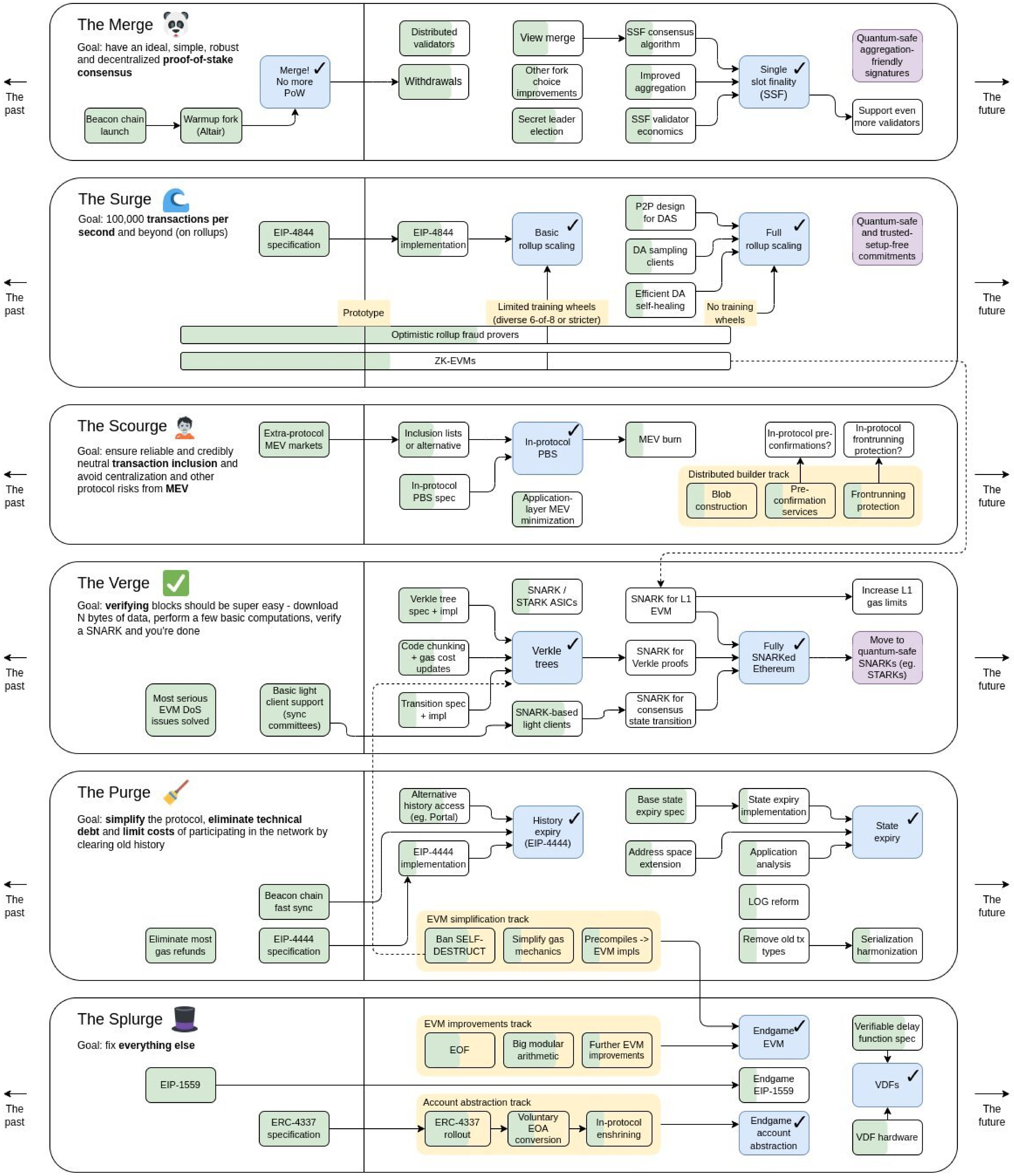
It costs up to $200 million a year to keep Bitcoin’s code maintained and functioning. Can developers find the resources they need in a plunging market? Frederick Munawa checks in.

Frederick Munawa

Feb 24, 2023 at 2:46 a.m. Updated Feb 25, 2023 at 3:47 a.m. CONSENSUS MAGAZINE

- the technical complexity of Bitcoin fitting in a human brain is far, far more important than any of the block data fitting within any storage and transfer limits

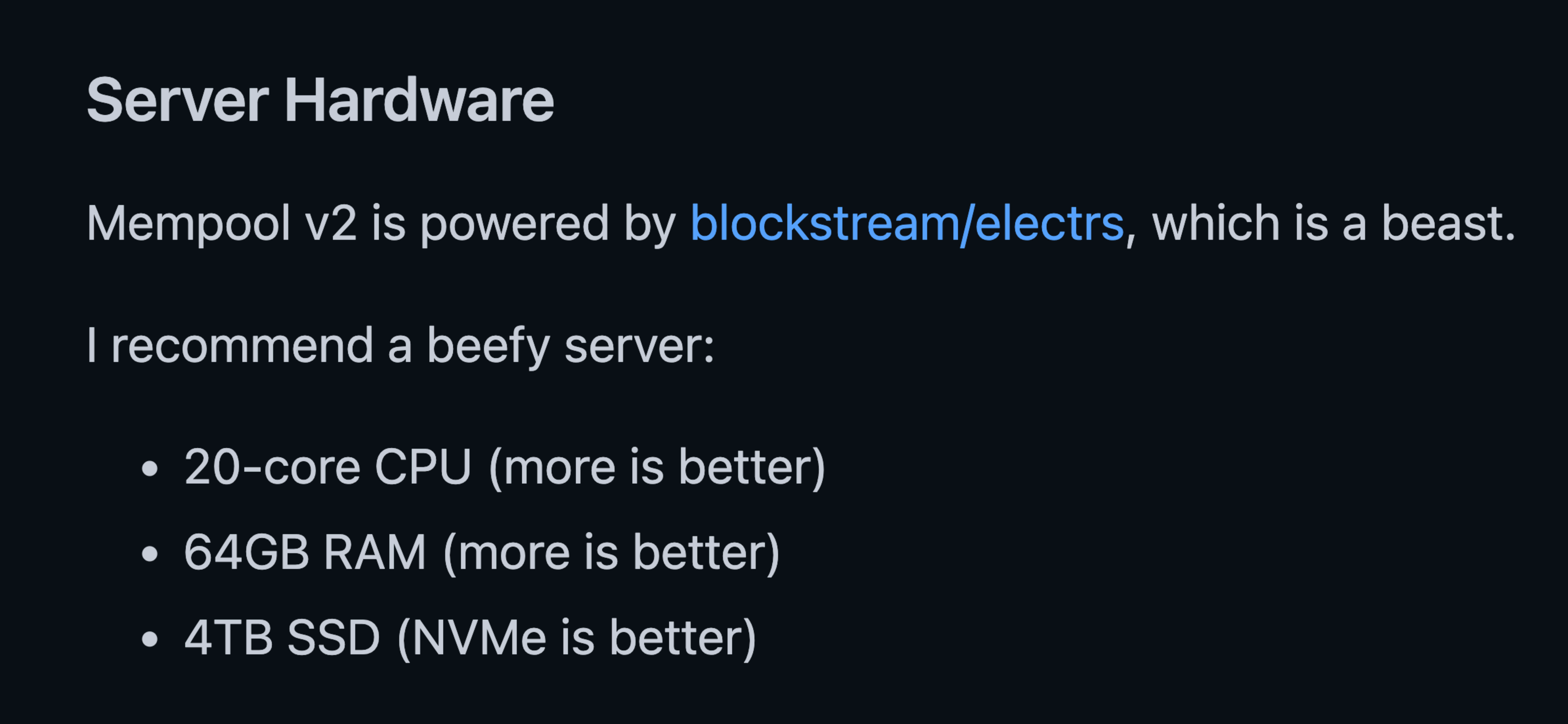
- Bitcoin Whitepaper (2008) — 9 pages, 8 references — notably it did not define Bitcoin but came as a companion to the source code which provided the actual definitions and 1.5 years of R&D by Satoshi
    - +160 BIPs as of March 2023
- Ethereum Whitepaper (2014) — 36 pages, 21 references
    - +600 EIPs as of March 2023


The Merge
Goal: have an ideal, simple, robust and decentralized proof-of-stake consensus
Beacon chain launch
Warmup fork (Altair)
Merge! ✓ No more PoW
Distributed validators
Withdrawals
View merge
Other fork choice improvements
Secret leader election
SSF consensus algorithm
Improved aggregation
SSF validator economics
Single ✓ slot finality (SSF)
Quantum-safe aggregation-friendly signatures
Support even more validators
The Surge
Goal: 100,000 transactions per second and beyond (on rollups)
EIP-4844 specification
EIP-4844 implementation
Basic ✓ rollup scaling
P2P design for DAS
DA sampling clients
Efficient DA self-healing
Full ✓ rollup scaling
Quantum-safe and trusted-setup-free commitments
Prototype
Limited training wheels (diverse 6-of-8 or stricter)
No training wheels
Optimistic rollup fraud provers
ZK-EVMs
The Scourge
Goal: ensure reliable and credibly neutral transaction inclusion and avoid centralization and other protocol risks from MEV
Extra-protocol MEV markets
Inclusion lists or alternative
In-protocol PBS spec
In-protocol ✓ PBS
Application-layer MEV minimization
MEV burn
In-protocol pre-confirmations?
In-protocol frontrunning protection?
Distributed builder track
Blob construction
Pre-confirmation services
Frontrunning protection
The Verge
Goal: verifying blocks should be super easy - download N bytes of data, perform a few basic computations, verify a SNARK and you're done
Most serious EVM DoS issues solved
Basic light client support (sync committees)
Verkle tree spec + impl
Code chunking + gas cost updates
Transition spec + impl
SNARK / STARK ASICs
Verkle ✓ trees
SNARK-based light clients
SNARK for L1 EVM
SNARK for Verkle proofs
SNARK for consensus state transition
Fully ✓ SNARKed Ethereum
Increase L1 gas limits
Move to quantum-safe SNARKs (eg. STARKs)
The Purge
Goal: simplify the protocol, eliminate technical debt and limit costs of participating in the network by clearing old history
Eliminate most gas refunds
Beacon chain fast sync
EIP-4444 specification
Alternative history access (eg. Portal)
EIP-4444 implementation
History ✓ expiry (EIP-4444)
EVM simplification track
Ban SELF-DESTRUCT
Simplify gas mechanics
Precompiles -> EVM impls
Base state expiry spec
Address space extension
State expiry implementation
Application analysis
LOG reform
Remove old tx types
State ✓ expiry
Serialization harmonization
The Splurge
Goal: fix everything else
EIP-1559
ERC-4337 specification
EVM improvements track
EOF
Big modular arithmetic
Further EVM improvements
Account abstraction track
ERC-4337 rollout
Voluntary EOA conversion
In-protocol enshrining
Endgame ✓ EVM
Endgame EIP-1559
Endgame ✓ account abstraction
Verifiable delay function spec
VDFs ✓
VDF hardware
The past
The future

- drivetrain moving parts Tesla 20 vs gasoline car 200
    - total parts Tesla 10k, gasoline 30k
    - space shuttle 2.5 million moving parts, Space X rocket 100k total parts (order of magnitude estimate by ChatGPT-3)

- security vs openness and incentives: two cases
    - OpenBSD is arguable the most secure general-purpose and generally useful OS with dozens of default features that are only partially present as advanced options in Linux and other systems (the gap has been closing)
        - still, every professional security audit always uncovers new serious vulnerabilities
    - closer to Bitcoin, Satoshi Labs, the company behind Trezor, looked at using hardware secure elements in their wallets
        - after signing watertight NDAs with a vendor, they looked at a highly secure chip and found glaring vulnerabilities — https://blog.trezor.io/introducing-tropic-square-why-transparency-matters-a895dab12dd3

- secure platforms are great, all of them have privately known vulnerabilities, Bitcoin Core and Bitcoin Protocol are no exceptions
  - why are they not widely exploited? why have I not been pwned yet?
    - you're not a worthy target yet
    - you're expected to be a better target later
    - you've been pwned, but haven't realized it yet
    - DeFi scams are 100x easier and safer



Server Hardware

Mempool v2 is powered by blockstream/electrs, which is a beast.

I recommend a beefy server:

- 20-core CPU (more is better)
- 64GB RAM (more is better)
- 4TB SSD (NVMe is better)

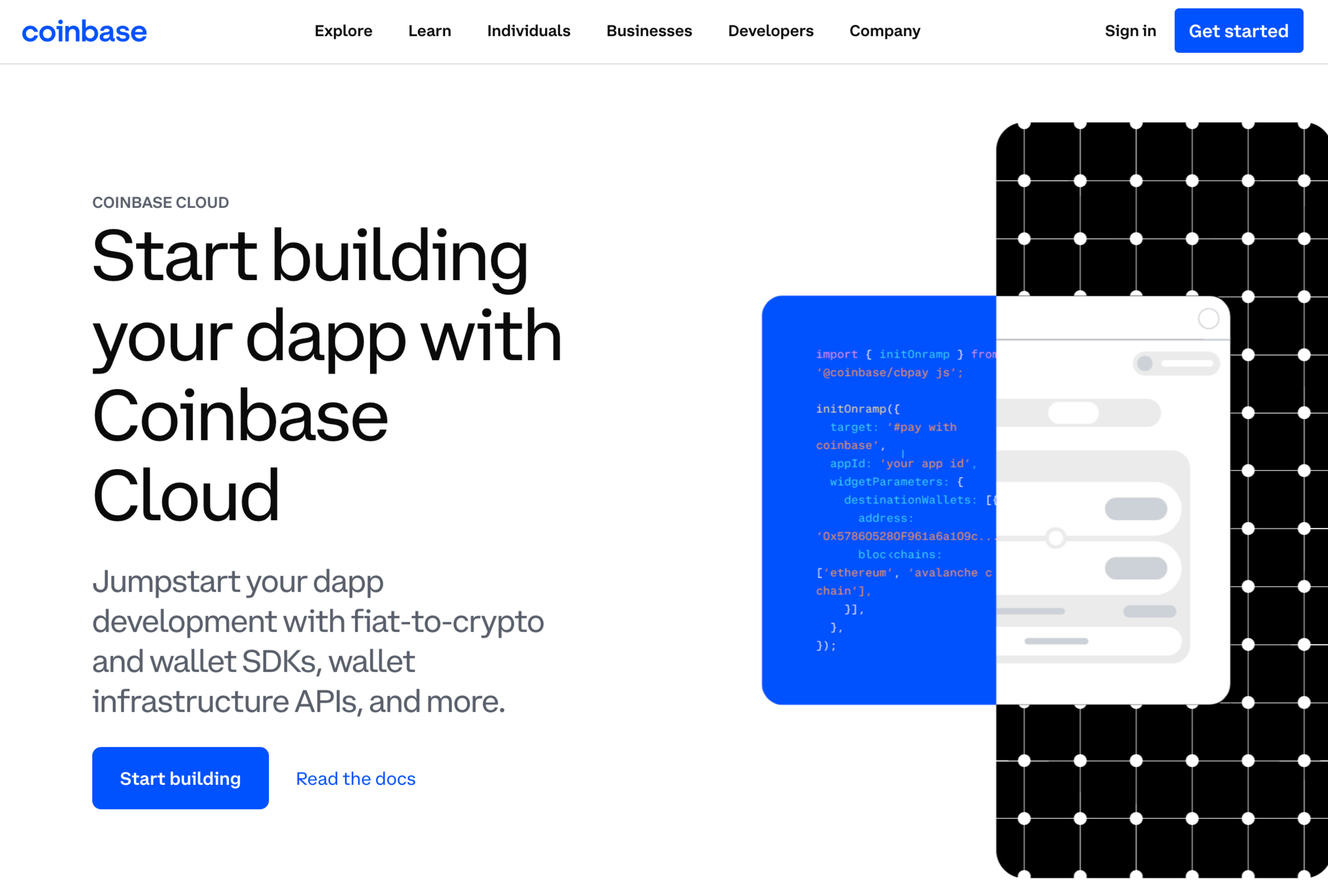
- Bitcoin averaged 3tx/sec in 2022, total data size 465GB as of March 2023, 815M total txs and capped to grow linearly
    - for any DBA, this a trivial database problem that should easily run on 20–year-old server hardware with zero CPU load
    - if a real DB is used for Bitcoin, then you can replace dozens of Electrum server implementations written in Python, Rust, C++ with a few materialized views and indexes, most taking minutes to calculate — and perhaps one page of highly readable SQL code

- so why not? — not for lack of trying
  - lots of attempts at full nodes, a cluster around 2014, and another around 2017, most achieved functionality but later gave up — complexities kept growing
  - some engineers got focused on technical problems in isolation and fell into forks like BCH or other chains
  - other engineers got enticed by the quick riches

An open source Bitcoin node built to power large scale web applications.

Toshi

Coinbase has released Toshi, a free API toolkit for bitcoin app developers that runs on a full bitcoin node backed by a SQL database.



COINBASE CLOUD

Start building your dapp with Coinbase Cloud

Jumpstart your dapp development with fiat-to-crypto and wallet SDKs, wallet infrastructure APIs, and more.

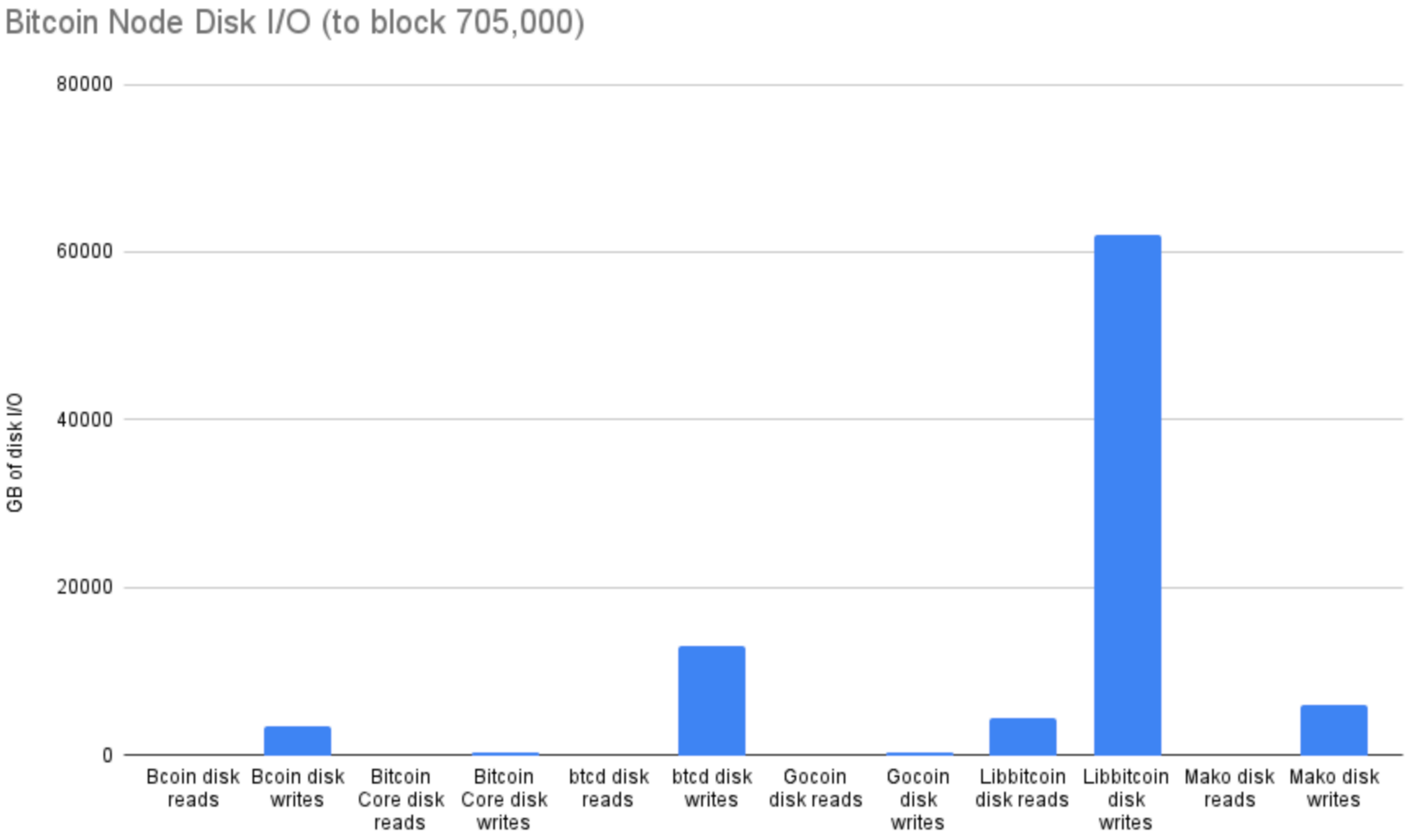
```
import { initOnramp } from
'@coinbase/cbpay js';

initOnramp({
  target: '#pay with
coinbase',
  appId: 'your app id',
  widgetParameters: {
    destinationWallets: [{
      address:
'0x578605280F961a6a109c...
      bloc<chains:
['ethereum', 'avalanche c
chain'],
    }],
  },
});
```

Full Node Software:

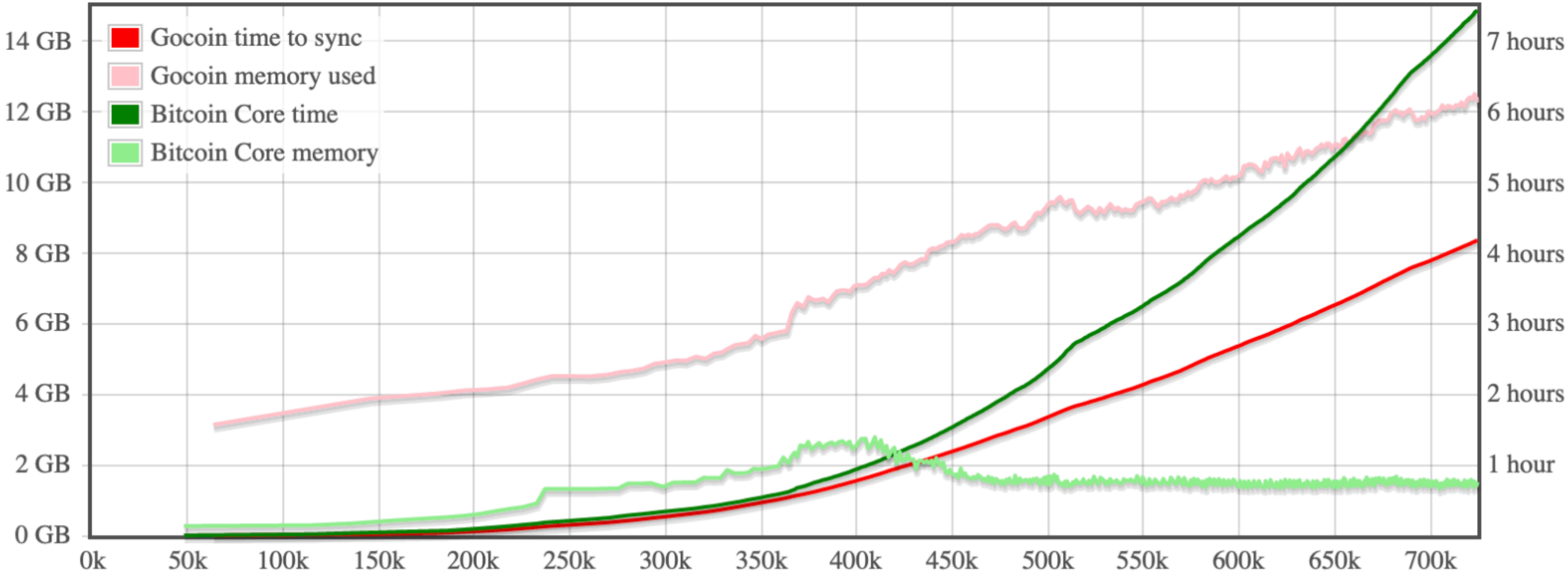
- Bitcoin Core
- Bitcoin Knots
- Bcoin
- Blockcore
- BTCD
- Gocoin
- Libbitcoin Node
- nix-bitcoin (hardened security)

Performance Rankings

1. **Bitcoin Core 24.0**: 7 hours, 10 minutes
2. **Gocoin 1.10.3**: 7 hours, 15 minutes
3. **Bitcoin Knots 23.0**: 7 hours, 23 minutes
4. **Mako 3d8a5180**: 23 hours, 44 minutes
5. **Bcoin 2.2.0**: 1 day, 8 hours, 8 minutes
6. **Libbitcoin Server 3.6.0**: 2 days, 8 hours, 27 minutes
7. **BTCD 0.23.3**: 2 days, 19 hours, 27 minutes
8. **Blockcore 1.1.37**: incomplete
9. **Stratis 1.3.2.4**: incomplete


Bitcoin Node Disk I/O (to block 705,000)
GB of disk I/O
80000
60000
40000
20000
0
Bcoin disk reads
Bcoin disk writes
Bitcoin Core disk reads
Bitcoin Core disk writes
btcd disk reads
btcd disk writes
Gocoin disk reads
Gocoin disk writes
Libbitcoin disk reads
Libbitcoin disk writes
Mako disk reads
Mako disk writes

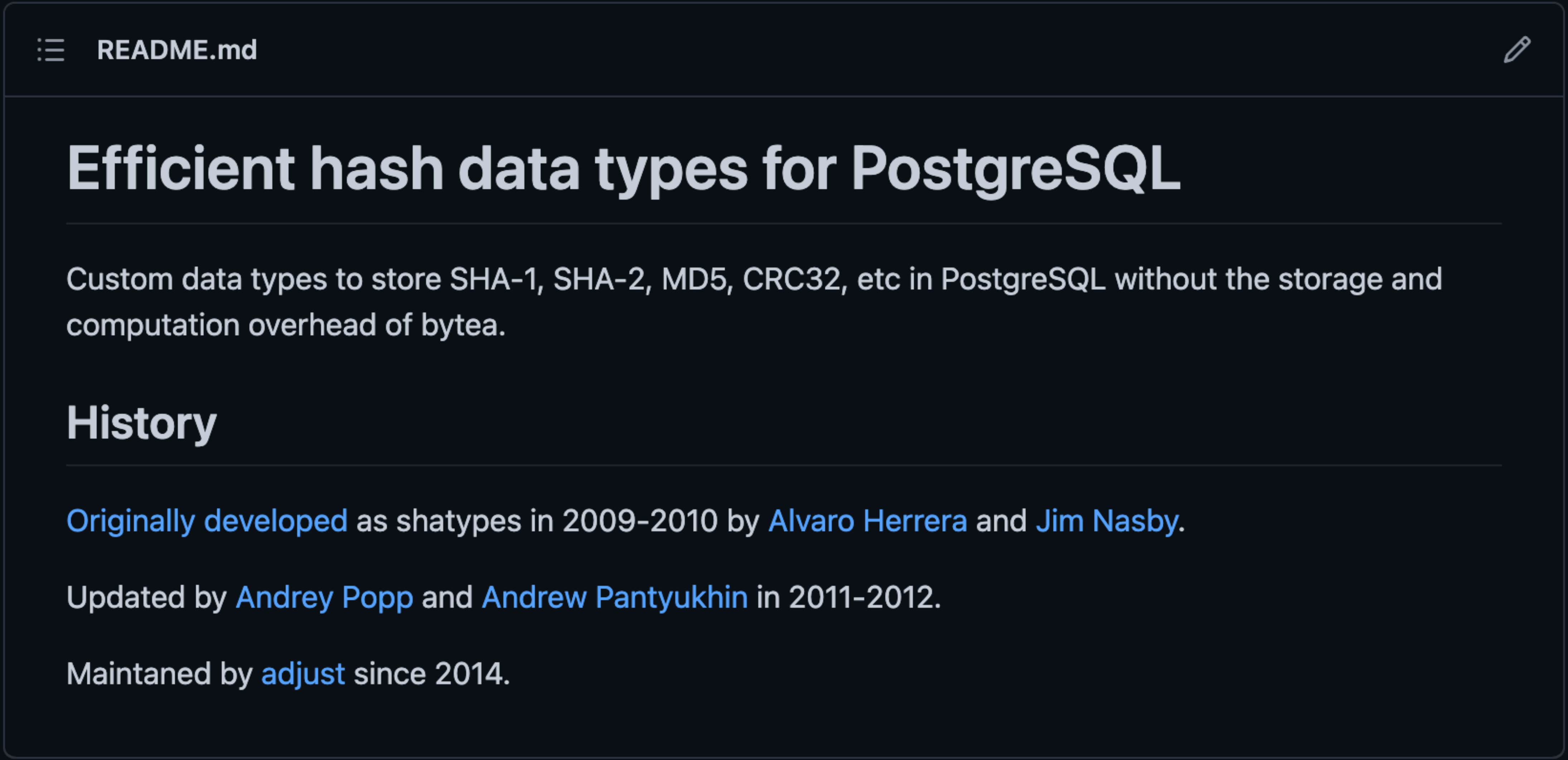
With comparision to Bitcoin Core 23.0:



Both the clients were using their default configuration.

The Zen of libbitcoin

Readability over speed.
Beauty over convenience.
Simplicity over complexity.
Architected, not hacked.
Flat, not nested.
Explicit, not implicit.
Errors should be loud.
Never is better than right now.
Now is better than never.
Be flexible and configurable.
Build houses from bricks, software from modules.



Efficient hash data types for PostgreSQL

Custom data types to store SHA-1, SHA-2, MD5, CRC32, etc in PostgreSQL without the storage and computation overhead of bytea.

History

Originally developed as shatypes in 2009-2010 by Alvaro Herrera and Jim Nasby.

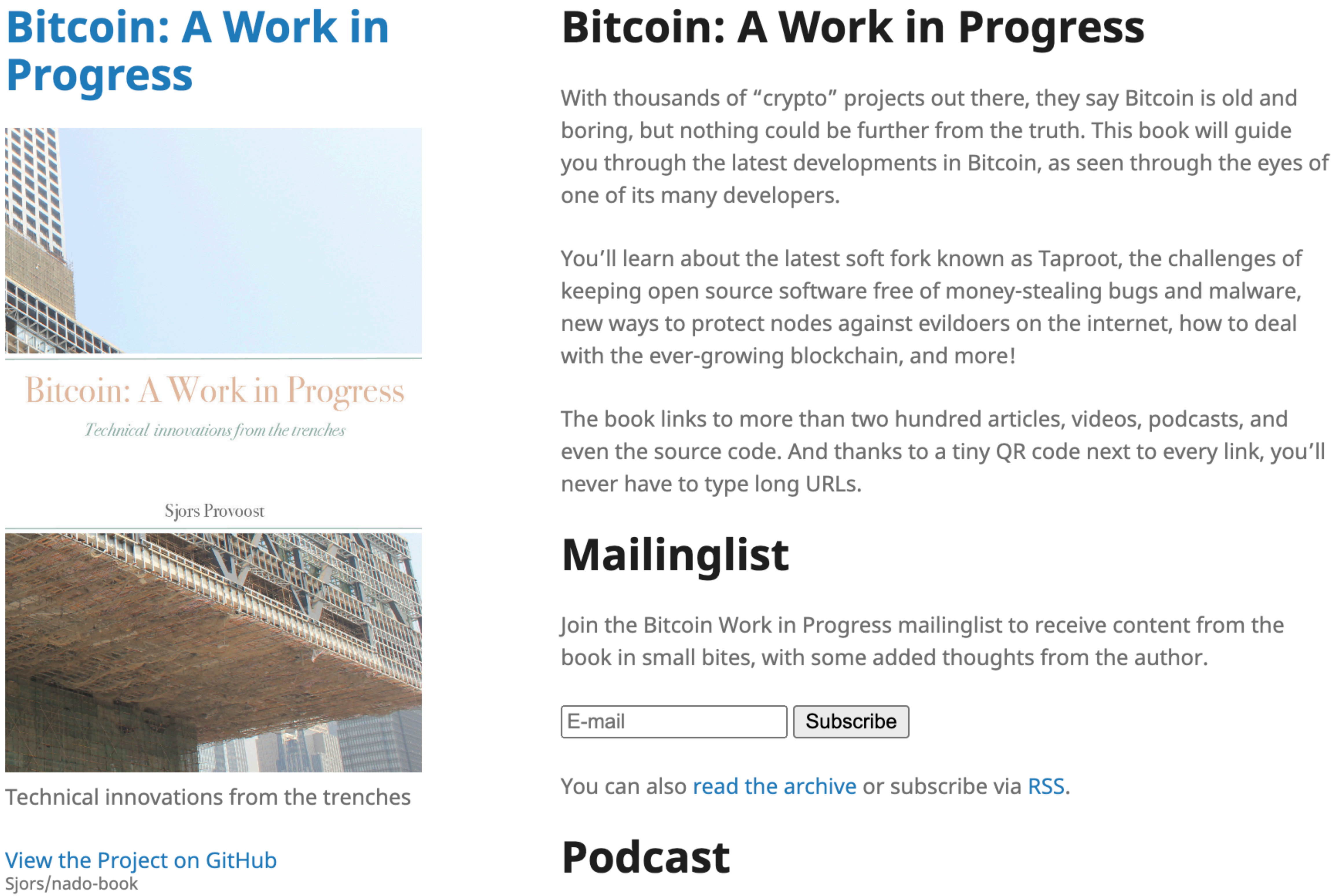
Updated by Andrey Popp and Andrew Pantyukhin in 2011-2012.

Maintaned by adjust since 2014.

- Bitcoin node software has been decoupling for a decade
  - mining / hashing / proof of work
  - wallets
    - transaction construction
    - signature coordination
      - signature construction
        - MPC (less relevant for Bitcoin)
    - balance check
    - transaction history check
    - pending transaction check
  - distributed block validation and redundant block storage —
    - the only thing bitcoin-core still does at scale, but how critical storage is?
  - distributed redundant block header storage, UTXO storage, mempool
  - features that can be core but are not
    - SPV
    - block explorers
    - Lightning
    - future: ZK

- we still lack Bitcoin specification
  - e.g. block size 1MB has never been defined outside of code, only later mention in SegWit BIP-141
- Bitcoin consensus and operation is currently not, has never been, and is currently designed to never become self-contained to any codebase
  - you have to manually track releases and decide when and how to upgrade your node software, otherwise statistically guaranteed to lose everything
  - the real consensus is very political and human

- Bitcoin Core databases
  - BDB, SQLite, LevelDB, ZeroMQ...
  - BDB replaced with SQLite in 2020 for wallet.dat, but only used as KVS
  - 220 issues just for UTXO DB and indexes (202 closed)
- addrindex
  - https://github.com/bitcoin/bitcoin/pull/2802
  - https://github.com/bitcoin/bitcoin/pull/3652
  - https://github.com/bitcoin/bitcoin/pull/5048
  - https://github.com/bitcoin/bitcoin/pull/6835



Bitcoin: A Work in Progress

Bitcoin: A Work in Progress
Technical innovations from the trenches
Sjors Provoost

Technical innovations from the trenches

View the Project on GitHub
Sjors/nado-book

Bitcoin: A Work in Progress

With thousands of “crypto” projects out there, they say Bitcoin is old and boring, but nothing could be further from the truth. This book will guide you through the latest developments in Bitcoin, as seen through the eyes of one of its many developers.

You’ll learn about the latest soft fork known as Taproot, the challenges of keeping open source software free of money-stealing bugs and malware, new ways to protect nodes against evildoers on the internet, how to deal with the ever-growing blockchain, and more!

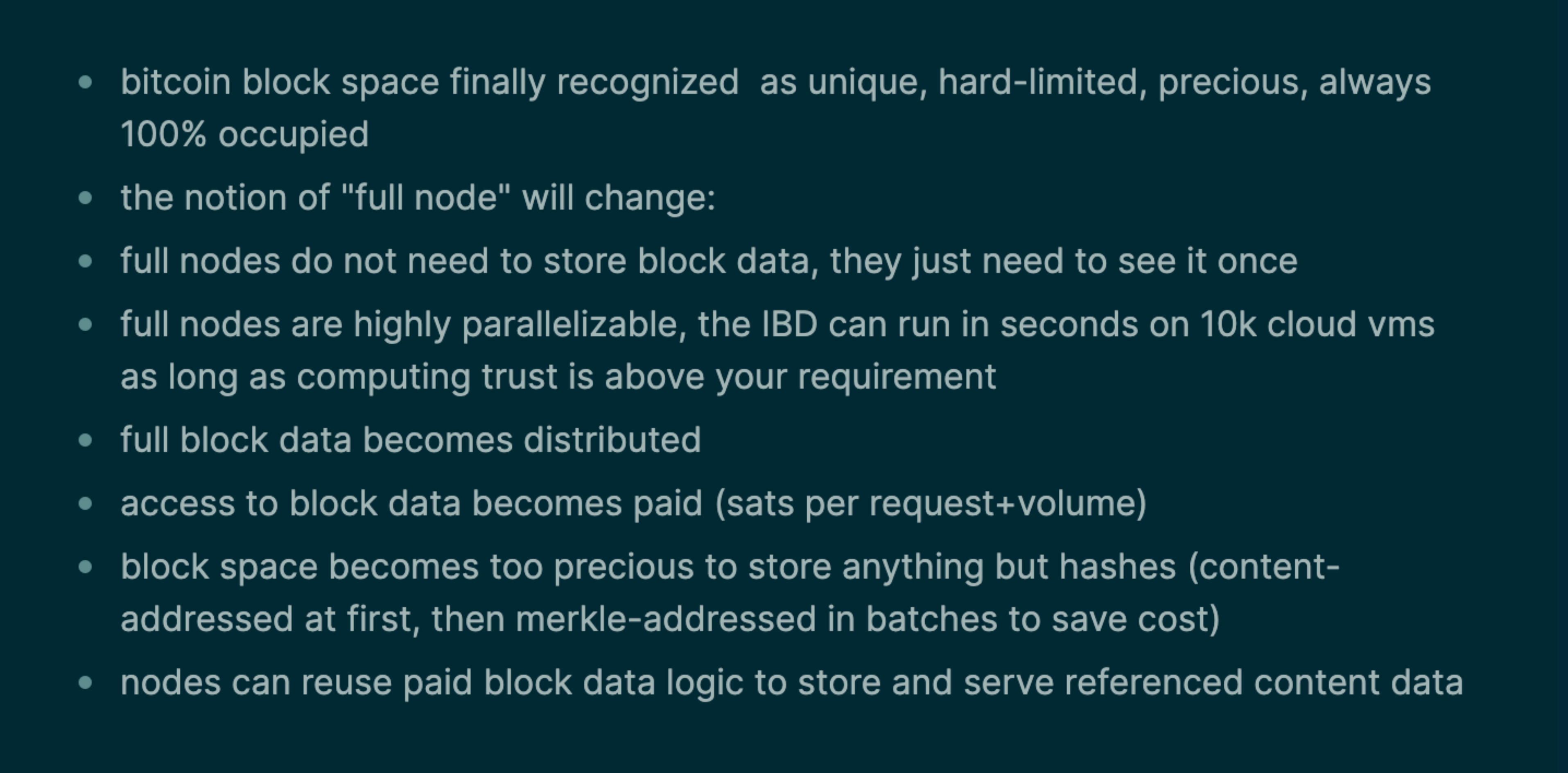
The book links to more than two hundred articles, videos, podcasts, and even the source code. And thanks to a tiny QR code next to every link, you’ll never have to type long URLs.

Mailinglist

Join the Bitcoin Work in Progress mailinglist to receive content from the book in small bites, with some added thoughts from the author.

You can also read the archive or subscribe via RSS.

Podcast



- bitcoin block space finally recognized as unique, hard-limited, precious, always 100% occupied
- the notion of "full node" will change:
- full nodes do not need to store block data, they just need to see it once
- full nodes are highly parallelizable, the IBD can run in seconds on 10k cloud vms as long as computing trust is above your requirement
- full block data becomes distributed
- access to block data becomes paid (sats per request+volume)
- block space becomes too precious to store anything but hashes (content-addressed at first, then merkle-addressed in batches to save cost)
- nodes can reuse paid block data logic to store and serve referenced content data

- ways forward
    - massive push to make the protocol fully and «centrally» documented
    - make documentation normative — leading Core vs catching up
    - experiment with industry-proven database and related technologies applied to full nodes
    - long, usually thankless effort to manage complexity