5-2022

Bitcoin, Blockchain Technology, and Cryptocurrencies

Jeffrey Dodson
University of Arkansas, Fayetteville

Bitcoin, Blockchain Technology, and Cryptocurrencies

by

Jeffrey Dodson

Advisor: Professor Steve Nolan

An Honors Thesis in partial fulfillment of the requirements for the degree Bachelor of Science in Business Administration in Information Systems.

Sam M. Walton College of Business
University of Arkansas
Fayetteville, Arkansas

May 14, 2022

Table of Contents

Introduction

The blockchain based cryptocurrency known as Bitcoin was theorized in a whitepaper published October 28, 2008, by Satoshi Nakamoto (pseudonym) (Nakamoto, 2008). The paper, titled, "Bitcoin: A Peer-to-Peer Electronic Cash System," laid out a digital currency creation/exchange structure that employs a decentralized ledger that would later run on the author's open-source application (Nakamoto, 2008). The main innovation of this technology is found within the security benefits provided by the proof-of-work consensus mechanism that requires solving a mathematic trap-door compression function to verify transactions/blocks added to the blockchain. On January 3, 2009, the genesis block, a term for the first block in any given blockchain, was created using Satoshi's Bitcoin v0.1 software that actualized the concepts in the Bitcoin whitepaper (Bitcoin Core, 2021).

Bitcoin is so well known because it was the first working implementation of decentralized cryptocurrency (Nakamoto, 2008). It also holds the top spot on the list of cryptocurrencies by market capitalization at $728,484,557,258 USD with a price of $38,279.11 USD per bitcoin (Blockchain.com, 2022). The first exchange of bitcoin for goods was 10,000 bitcoins for $41 worth of pizza establishing the initial exchange rate of 0.0041 USD per bitcoin (DeCambre, 2021). With the current exchange rate of $38,279.11 USD per bitcoin, the 10,000 bitcoins used to buy two Papa John's pizzas would be worth $382,791,100 USD today. Several relevant charts surrounding bitcoin's evolution to its current state can be found in appendix [C].

This paper's purpose is to explore the innerworkings behind Bitcoin's functionality. Bitcoin has transcended value beyond the bounds of its ledger as seen by trade volume on cryptocurrency to fiat currency exchanges and use as payment for goods and services. It is also clear that cryptocurrencies like Bitcoin have the potential to appreciate over time more than traditional assets, fiat currencies, index funds, or individual stocks. As a growing number of individuals seek to profit from acquiring cryptocurrencies and adopting blockchain technology, there is an increased risk for buying into unproductive blockchain implementations or scams if investors are not aware of certain cybersecurity fundamentals or understanding of how new coins are created. This Bitcoin centered thesis will define essential blockchain terminology, provide descriptions of cryptographic processes, and allow individuals to understand the software/hardware components that are the defining features of Bitcoin's evolving blockchain.

Using Bitcoin the New Way

In the early years of Bitcoin, its supply was in the hands of few. The owners of the currency were likely to have acquired their Bitcoin from CPU mining. There was a list of required actions a user would have to take if they wanted to acquire, request, send, or store Bitcoin. All this prerequisite knowledge and software are no longer necessary as Bitcoin is sold on several centralized exchanges. These exchanges also offer cryptocurrency wallets free to users wanting to buy the various cryptocurrencies listed on exchanges. Users now-a-days can easily buy, sell, send, and store cryptocurrency, but opt to use a third party to connect you to the blockchain making use dependent on an intermediary. These large, centralized exchanges like Coinbase have made cryptocurrency more user friendly, but at the cost of going against some of the fundamental values that Bitcoin's creator initially designed the decentralized currency for.

Summary Statistics

Listed in Table 1 below are some relevant statistics on the top 5 cryptocurrencies by market capitalization. Bitcoin has a higher market capitalization than Ethereum, the second runner up, by roughly a factor of two. While this sounds impressive, the current price is down roughly 45% from its all-time high of $68,789.63 (Blockchain.com, 2022). Overall, it is easy to see that cryptocurrencies are a rapidly growing and competitive trillion-dollar market. Another insight from Table 1 is that two of the coins are priced at exactly $1 USD. These are stable coins created to offset the price volatility of Bitcoin and other non-stable coins.

Top 5 Cryptocurrencies by Market Capitalization (May 1, 2022)

Rank	Icon	Name	Price	Market Cap	Circulating Supply	ATH
1		Bitcoin	$38,279.11	$728,484,557,258	19,027,781 BTC	$68,789.63
2		Ethereum	$2,795.67	$337,418,935,736	120,605,744 ETH	$4,891.70
3		Tether	$1.00	$83,166,955,578	83,152,877,108 USDT	$1.22
4		BNB	$386.86	$63,170,415,254	163,276,975 BNB	$690.93
5		USDC	$1.00	$49,273,953,504	49,274,562,120 USDC	$2.35

Table 1 (Blockchain.com, 2022)

Bitcoin has an interesting property where the number of coins created in the form of miner's coinbase reward halves every 210,000 blocks (Open-Source Developer Group*, 2021). This means that around the year 2140, there will be no more bitcoins added to the supply and a total of 21,000,000 bitcoins (Open-Source Developer Group*, 2021). On top of that, the difficulty to mine adjusts every 2016 blocks, or roughly every 2 weeks (Open-Source Developer Group*, 2021). As time goes on, miners will earn more from fees than coinbase rewards as seen in appendix [D].

Within Table 2 are several important measures to help understand Bitcoin. I will break down these measures. Currently there are just over 19 million bitcoins in circulation which is 90.61% of all bitcoins that can ever exist based on current protocols (Blockchain.com, 2022). For an in depth look at Bitcoin's supply schedule, see appendix [D]. These bitcoins were at one point rewarded to a Bitcoin miner in the process of adding blocks to the 734,448-block long blockchain (Open-Source Developer Group*, 2021). These blocks contain transactions and create a ledger recording who sent who bitcoins and when. Altogether, this list of transactions amounts to 403.5 gigabytes. Each block is limited at 1 megabyte of data so transactions with higher fees paid to miners will be added before those that offer a low fee to the miner (Nakamoto, 2008). Confirmed in each block on average are 994 new transactions (Blockchain.com, 2022). Unconfirmed transactions sit in a memory pool where miners compile them into blocks and attempt to solve a proof-of-work requirement before other miners. Whoever satisfies the proof-of-work mechanism first wins the coinbase reward for their computer's work in maintaining the ledgers' accuracy and integrity. A miner wins this reward and creates a block roughly every 600 seconds or 10 minutes. The unconfirmed transactions and newly added blocks are pushed across a peer-to-peer network with over 15,000 individual nodes each running the

Bitcoin Core 22.0 software (Bitnodes, 2022). For a better look into the live geo-distribution of active nodes, see appendix [B].

Web Queries from Blockchain.com	
Measure Name	Current Value
Current Bitcoin Supply	19,027,800
Number of Blocks	734,448
Avg Time Between Blocks (s)	509.0
Avg Time Between Blocks (m)	8.5
Avg Transactions per Block	994.0
Percent of Bitcoin Mined	90.61%
Bitcoin Blockchain Size (Gb)	403.5
Number of Nodes	15,184

Table 2 (Blockchain.com, 2022)

Using Bitcoin the Old Way

To understand cryptocurrency at level deeper than knowing how to buy/receive or send bitcoins, it is extremely useful to have the Bitcoin Core node/wallet software installed as a reference. However, I have provided several screenshots of the essential components of the user interface in appendix [A]. Bitcoin Core 22.0 is the most current version of the software that connects a user to the Bitcoin blockchain (Bitcoin Core, 2021). This software is free to download from the Bitcoin developer's website (Bitcoin Core, 2021). This software has several capabilities that allow a person to interact with the Bitcoin blockchain. The primary use of the software is sending and receiving blockchain data using a peer-to-peer network. The second functionality is generating a cryptocurrency wallet that enables a user to send and receive bitcoin transactions. These two functions are built on top of many sub-functions that are variable upon which version of Bitcoin Core that a user is running.

Bitcoin Core 22.0

Bitcoin software has upgraded in an iterative fashion from the version 0.1 software made public in 2009. It has an open-source codebase meaning anyone can view or edit the code running the program. The code is available on GitHub where the full list of 868 contributors and their contributions to the codebase are kept track of (Bitcoin, 2022). The node/wallet software program, known to some as the "Satoshi Client", was initially named Bitcoin, then changed to Bitcoin-Qt, and is currently called Bitcoin Core. For the full list of Bitcoin software version releases, see appendix [F]. The C language code within the program is modified as per the Bitcoin Improvement Proposal process which is often abbreviated as BIP (Bitcoin Core, 2021). The full list of software versions and BIP's for Bitcoin is in Appendix [H].

As stated before, Bitcoin Core is used to connect with the blockchain and other nodes. Table 3 below shows some important measures for how nodes connect with other nodes. All nodes must have an internet connection and an internet protocol address to start with. They connect to a hard coded domain name server to get known node IP addresses. From there, your node will attempt to open 10 connections on transmission control protocol port 8333. To see other examples of TCP Port connections, see appendix [E]. Of those connections, 2 are connections to block relays and 8 are connections to full nodes. See appendix [A] (Peers Node Window) to see these 10 connections. Block relays are nodes that only relay when a new block is added to the blockchain. This helps full nodes know if their blockchain is up to date. With the other 115 incoming connections, nodes can send each other remote process calls. These RPCs are various commands that let nodes query necessary information from other nodes to stay up to date. The full list of RPCs available to nodes is in appendix [I]. It is important to note that most nodes are still dependent on centralized internet service providers for connection.

Static Values from Most Recent Bitcoin Protocol	
Measure Name	Current Value
TCP Port	8333
Number of Peers (Block Relays)	2
Number of Peers (Outgoing Full Nodes)	8
Number of Peers (Incoming Connections)	115
Max Time for Node to Receive Full New Block	~8 Seconds

Table 3 (Open-Source Developer Group*, 2021) (Baek, 2021)

The Bitcoin Core software allows a user to set up a cryptocurrency wallet. This process is one of the most vulnerable parts of cryptocurrency. When you create a wallet, you are creating a private public key pair using the properties of an elliptic curve. The math behind this elliptic curve is too complicated to cover in this paper, but I provide a mathematic process flow to generating these key pairs in appendix [J]. The private key is a secret 64-character hexadecimal string which is the encryption key or signing key for transactions (Raj, 2022). This is like a secret passcode and if anyone steals it, then they will be able to send themselves all the user's bitcoins. A public key is a non-secret 64-character hexadecimal string and is a decryption key or verification key (Raj, 2022). A user intentionally shares this so that other nodes can verify when a transaction contains a valid signature. These key pairs can either be saved on a cold storage wallet like an ordinary USB drive or saved in a hot storage wallet where a third party like Coinbase.com stores a user's balance, transactions, and encryption keys (Raj, 2022).

Mining

Mining is the process of satisfying the proof-of-work consensus mechanism created in the Nakamoto whitepaper. When a node is said to mine, they are running the Secure Hashing Algorithm 256 (Raj, 2022). This algorithm takes advantage of the same elliptic curve properties as private public key pair creation used for cryptocurrency wallets (Raj, 2022). Table 4 and 5 show some interesting statistics about the SHA 256 algorithm. This algorithm will take in inputs and spit out a random seeming unique deterministic output that is 256 bits long as long as the input is smaller than the finite field of the elliptic curve used in the SHA 256 algorithm or 2^64 bits (Raj, 2022). Table 4 measure 1 and 2 show the number of unique outputs to the hashing function.

Hashing Measures		
Measure Name	Definition	Value
2^256 Unique Combinations of Binary Output	0 or 1	115,792,089,237,316,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000
16^64 Unique Combinations of Hexidecimal Output	0-9 or a-f	115,792,089,237,316,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000
Typical Hashes per Second Range for CPU		1,000-20,000
Typical Hashes per Second Range for GPU		10,000,000-60,000,000
Typical Hashes per Second Range for ASIC		1,000,000,000,000-100,000,000,000,000
Current Network Hash Rate per Second		245,860,613,763,000,000,000
Blocks Between Difficulty Adjustments		2016 (Roughtly 2 Week Intervals)
Probability of Correct Hash (Guess)		0.000000000000000000001%

Table 4 (Open-Source Developer Group*, 2021) (Cryptopedia, 2021)

Mining blocks and getting a reward known as a coinbase, currently 6.25 bitcoins plus transaction fees included in the block mined, is done by brute-force guessing inputs into the SHA 256 algorithm (Raj, 2022). Table 4 above shows the probability of getting a correct guess per attempt is very low. Different computers can perform more guesses per second. The fastest ASIC miners perform the algorithm up to 100 trillion times per second. Table 5 below shows a series of inputs and outputs to the SHA 256 algorithm to explain what the goal of mining is. For binary conversion tables, see appendix [G]. Each input produces a seemingly random but deterministic output. Miners attempt to get an output that begins with a certain number of zeros. Currently the difficulty requires miners to get an output of 19 leading zeros. The number of leading zeros determines the difficulty of the network. All the miners in the network currently 220 million terrahashes per second (Blockchain.com, 2022). A terrahash is a trillion hashes per second. So that's 2.2e+20 hashes per second. This difficulty is updated every 2016 added blocks so that blocks are added at a rate of 1 every 10 minutes no matter how many miners are on the network (Nakamoto, 2008).

Secure Hashing Algorithm- Input to 256 Bit Output				
Input	Funtion	Output Type	Output	Length
Input	SHA 256	Hexadecimal	59a513a31d7ddca35e18069758d0e1eab4b9d0109c583419b622ec8b5cebffcb	64
Input1	SHA 256	Hexadecimal	c9a28cb6bcf4f2b6d944579278e90bc0d001fdb88a32b874891de6c119b3a946	64
Input2	SHA 256	Hexadecimal	54f194e065e9bb36218955e86a2d3abbcad506b126b86c9381c6a91d6b9d58c7	64
SecretPassword	SHA 256	Hexadecimal	2a8e9faf6b65c79233feaf2de6960888ce60987057effd87af94f81e6b76f8b8	64
0	SHA 256	Hexadecimal	5c56c2883435b38aeba0e69fb2e0e3db3b22448d3e17b903d774dd5650796f76	64
1	SHA 256	Hexadecimal	28902a23a194dee94141d1b70102accd85fc2c1ead0901ba0e41ade90d38a08e	64
2	SHA 256	Hexadecimal	729577af82250aaf9e44f70a72814cf56c16d430a878bf52fdaceeb7b4bd37f4	64
3	SHA 256	Hexadecimal	8491452381016cf80562ff489e492e00331de3553178c73c5169574000f1ed1c	64
39	SHA 256	Hexadecimal	**0**3fd5ff1048668cd3cde4f3fb5bde1ff306d26a4630f420c78df1e504e24f3c7	64
990	SHA 256	Hexadecimal	**000**1e3a4583f4c6d81251e8d9901dbe0df74d7144300d7c03cab15eca04bd4bb	64
52,117	SHA 256	Hexadecimal	**0000**642411733cd63264d3bedc046a5364ff3c77d2b37ca298ad8f1b5a9f05ba	64
1,813,152	SHA 256	Hexadecimal	**00000**c94a85b5c06c9b06ace1ba7c7f759e795715f399c9c1b1b7f5d387a319f	64
19,745,650	SHA 256	Hexadecimal	**000000**cdccf49f13f5c3f14a2c12a56ae60e900c5e65bfe1cc24f038f0668a6c	64
243,989,801	SHA 256	Hexadecimal	**0000000**ce99e2a00633ca958a16e17f30085a54f04667a5492db49bcae15d190	64
856,192,328	SHA 256	Hexadecimal	**0000000000000000**e067a478024addfecdc93628978aa52d91fabd4292982a50	64
2E99F445C007A9158207CC30CEBAD2B3D26C45FDAB2EBDF50D261335FC00D92C	SHA 256	Hexadecimal	000000000000000000095913f2dc133348dcbc4fcac513e66847fd4cee7149da	64

Table 5 (ETH.BUILD, 2022)

Miners brute-force their guess in what's known as a nonce. Appendix [K] shows a miner forming a block header with a successful hash output. The header has a version, Merkle root, hash of the previous block, nonce, bits, time, and the output hash with the correct number of leading zeros. The version is a number associated with BIP's, the Merkle root is the hash at the top of the Merkle tree for all the verified transactions in the block, the time is a timestamp value for when the algorithm was attempted, and bits/nonce are values that a miner can change to attempt to get the rest of the information in the header to input into the SHA 256 algorithm and output a hash beginning with the required number of leading zeros.

Because of how rare a correct guess is, it is rare that more than one miner gets a correct guess before getting the signal that another miner has guessed correctly before they did. But when this happens, a fork is created. Nodes receive two correct solutions to the SHA 256 algorithm. The fork that has the longest blockchain always takes priority and will resolve within the next few blocks added to the chain. Miners prove that they have done computational work by solving the SHA 256 algorithm at a specified difficulty making it impossible to corrupt the blockchain without more than 50% of the mining computing power (Raj, 2022). When a block is added, the transactions are solidified, and a new block is ready to be filled with new transactions. The difficult mining process is what's known as a consensus mechanism for the Bitcoin decentralized ledger and is the principal security behind Bitcoin's blockchain. This is what Satoshi called a proof-of-work chain (Nakamoto, 2008). See appendix [L] for a visual of a blockchain.

Conclusion

Bitcoin went from a fad to being worth more than the market cap of Facebook in just 13 short years. However, it failed to be what Satoshi Nakamoto wanted it to be. The creator of the first cryptocurrency wanted to cut out intermediaries like central banks or credit card companies. They wanted a cheap, peer-to-peer, decentralized ledger system to do daily transactions. With transaction fees peaking at $60 to send a transaction, the cryptocurrency became more of a speculative asset to buy and sell (Blockchain.com, 2022). Moreover, the fact that it is mainly traded on centralized exchanges and mining pools dominate the mining process speaks to the failure to cut out large intermediaries. However, bitcoin is a good store of value compared to come coins because it has a finite supply. It is being adopted by many financial institutions and businesses and has become ubiquitous among everyday investors. Bitcoin is in an evolutionary state. Blockchains are complicated, ever-changing, versatile, disruptive, and have the potential to change the long-term landscape of transaction validation and show that individuals can use decentralized networks and open-source applications to take the place of the services governments, businesses, and firms have historically provided and controlled.

Works Cited

Baek, S., Nam, H., Oh, Y., Tran, M., & Suk Kang, M. (2021). *On the claims of weak block synchronization in bitcoin*. Retrieved May 1, 2022, from https://eprint.iacr.org/2021/1282.pdf

Bitcoin. (2022). *Bitcoin/Bitcoin: Bitcoin Core Integration/Staging tree*. GitHub. Retrieved May 2, 2022, from https://github.com/bitcoin/bitcoin

Bitcoin Core. (2021, September 13). Retrieved May 1, 2022, from https://bitcoin.org/en/bitcoin-core/

Bitnodes. (2022). Retrieved May 1, 2022, from https://bitnodes.io/

Blockchain Explorer API Charts & Statistics. Blockchain.com. (2022). Retrieved May 1, 2022, from https://www.blockchain.com/api

Cryptocurrency address generator and validator (V1.1). (2021). Retrieved May 1, 2022, from https://www.mobilefish.com/services/cryptocurrency/cryptocurrency.html

Cryptopedia. (2021, December 3). *Crypto Mining Rigs & Bitcoin Mining Rigs explained*. Gemini. Retrieved May 2, 2022, from https://www.gemini.com/cryptopedia/crypto-mining-rig-bitcoin-mining-calculator-asic-miner#section-asic-miners-take-over-bitcoin-btc

DeCambre, M. (2021, May 22). Bitcoin Pizza Day. MarketWatch. Retrieved May 2, 2022, from https://www.marketwatch.com/story/bitcoin-pizza-day-laszlo-hanyecz-spent-3-8-billion-on-pizzas-in-the-summer-of-2010-using-the-novel-crypto-11621714395

ETH.BUILD. (2022). Retrieved May 2, 2022, from https://sandbox.eth.build/

Nakamoto, S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf

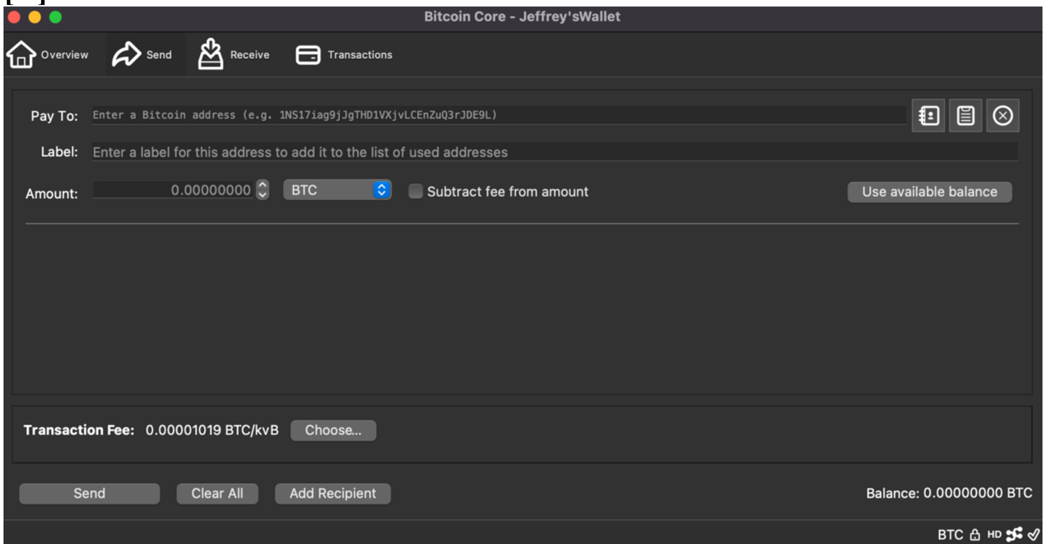
Open-Source Developer Group*. (2021, September 13). Bitcoin Core Version (22.0). Retrieved from https://bitcoin.org/en/releases/22.0/.

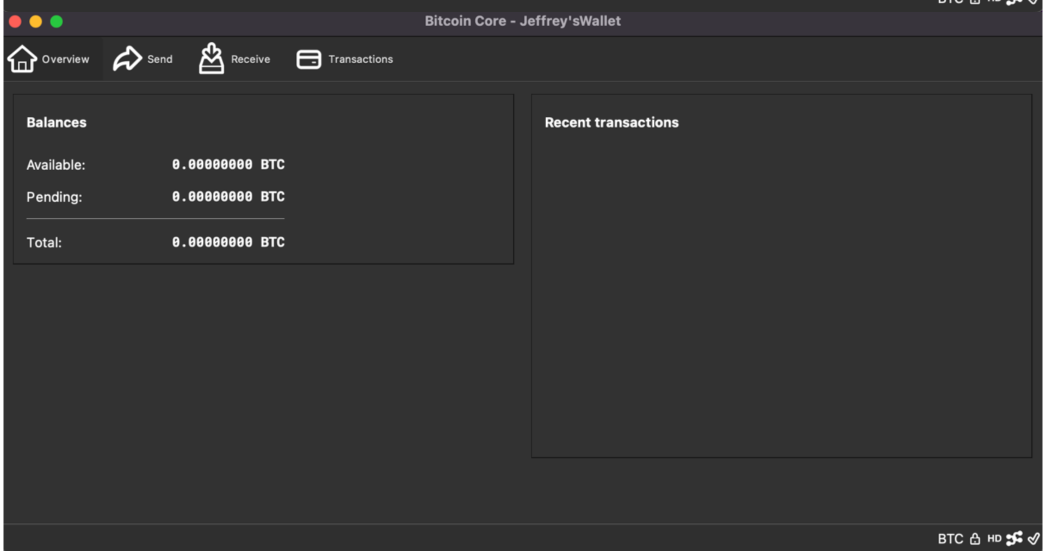
The link to the total list of 868 contributors to the codebase can be found at https://github.com/bitcoin/bitcoin/graphs/contributors.

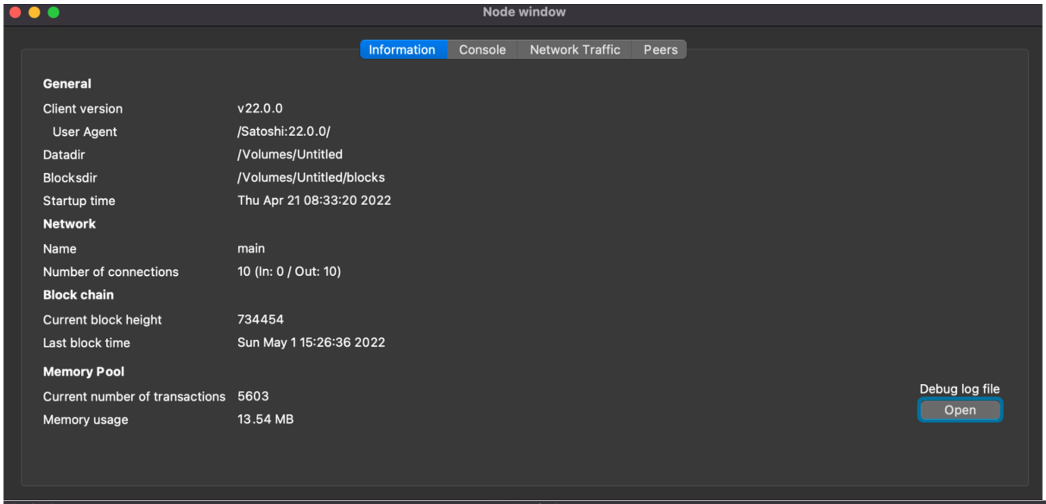
Raj, K. (2022). *Foundations of blockchain*. O'Reilly Online Learning. Retrieved May 2, 2022, from https://www.oreilly.com/library/view/foundations-of-blockchain/9781789139396/56c3bf8e-9dd2-4406-9a48-64c729163c59.xhtml

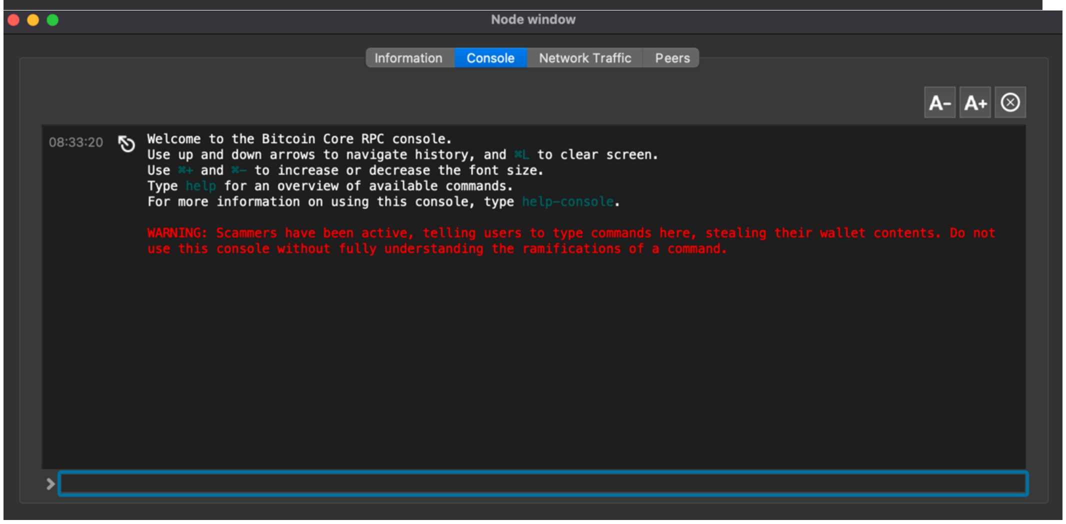
Appendix

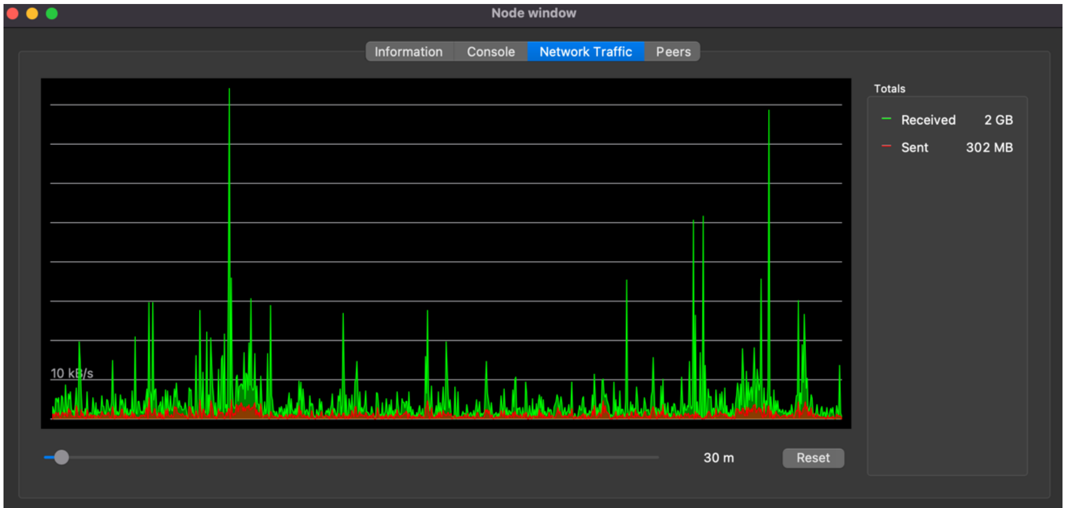
[A] Bitcoin Core 22.0 UI

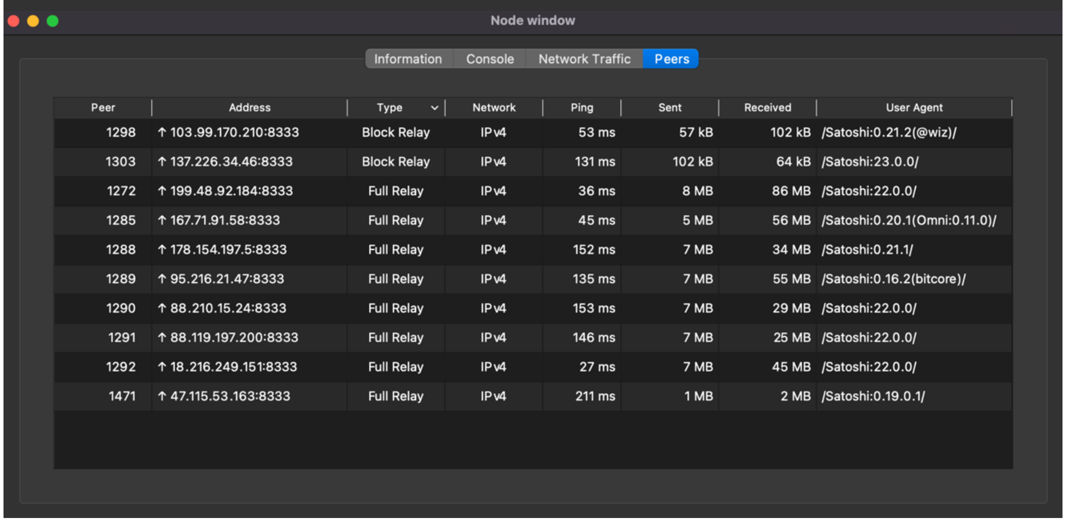





Node window
Information
Console
Network Traffic
Peers
General
Client version v22.0.0
User Agent /Satoshi:22.0.0/
Datadir /Volumes/Untitled
Blocksdir /Volumes/Untitled/blocks
Startup time Thu Apr 21 08:33:20 2022
Network
Name main
Number of connections 10 (In: 0 / Out: 10)
Block chain
Current block height 734454
Last block time Sun May 1 15:26:36 2022
Memory Pool
Current number of transactions 5603
Memory usage 13.54 MB
Debug log file
Open

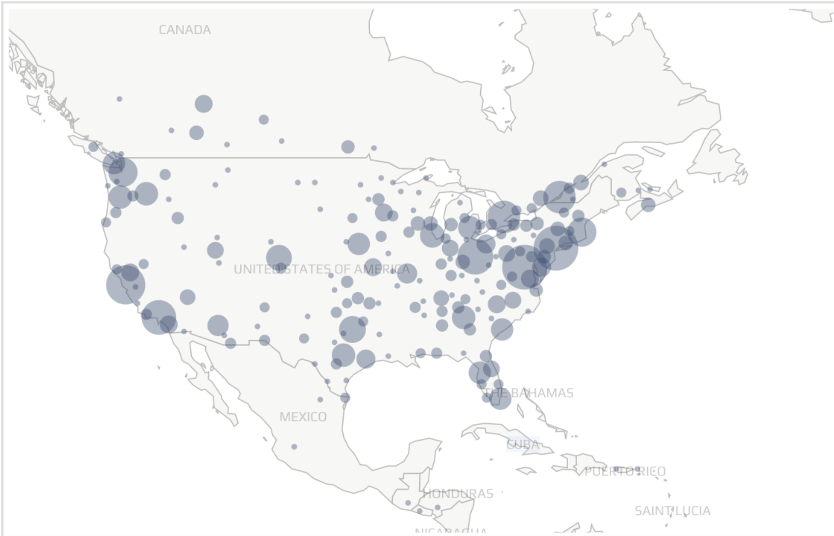

Node window
Information
Console
Network Traffic
Peers
A-
A+
08:33:20
Welcome to the Bitcoin Core RPC console.
Use up and down arrows to navigate history, and ⌘L to clear screen.
Use ⌘+ and ⌘- to increase or decrease the font size.
Type help for an overview of available commands.
For more information on using this console, type help-console.
WARNING: Scammers have been active, telling users to type commands here, stealing their wallet contents. Do not use this console without fully understanding the ramifications of a command.





Peer	Address	Type	Network	Ping	Sent	Received	User Agent
1298	↑ 103.99.170.210:8333	Block Relay	IPv4	53 ms	57 kB	102 kB	/Satoshi:0.21.2(@wiz)/
1303	↑ 137.226.34.46:8333	Block Relay	IPv4	131 ms	102 kB	64 kB	/Satoshi:23.0.0/
1272	↑ 199.48.92.184:8333	Full Relay	IPv4	36 ms	8 MB	86 MB	/Satoshi:22.0.0/
1285	↑ 167.71.91.58:8333	Full Relay	IPv4	45 ms	5 MB	56 MB	/Satoshi:0.20.1(Omni:0.11.0)/
1288	↑ 178.154.197.5:8333	Full Relay	IPv4	152 ms	7 MB	34 MB	/Satoshi:0.21.1/
1289	↑ 95.216.21.47:8333	Full Relay	IPv4	135 ms	7 MB	55 MB	/Satoshi:0.16.2(bitcore)/
1290	↑ 88.210.15.24:8333	Full Relay	IPv4	153 ms	7 MB	29 MB	/Satoshi:22.0.0/
1291	↑ 88.119.197.200:8333	Full Relay	IPv4	146 ms	7 MB	25 MB	/Satoshi:22.0.0/
1292	↑ 18.216.249.151:8333	Full Relay	IPv4	27 ms	7 MB	45 MB	/Satoshi:22.0.0/
1471	↑ 47.115.53.163:8333	Full Relay	IPv4	211 ms	1 MB	2 MB	/Satoshi:0.19.0.1/

[B] Bitnodes.io Map

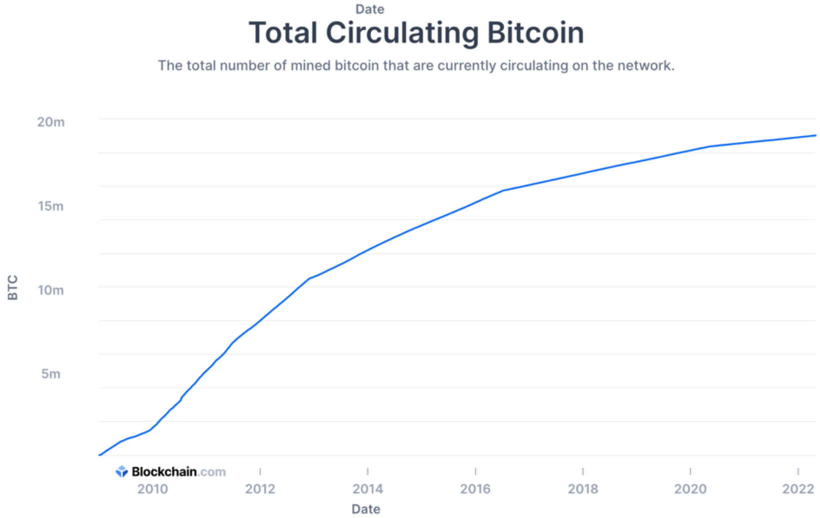


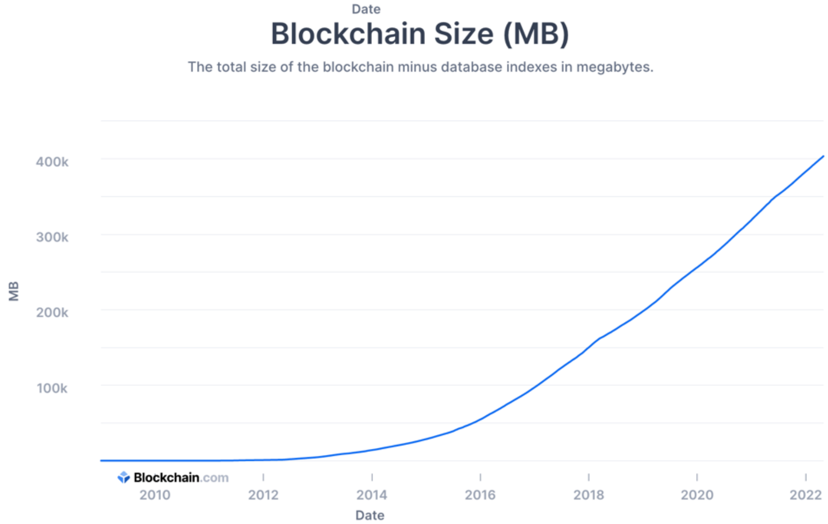
REACHABLE BITCOIN NODES

15320 nodes as of Sun May 1 16:21:24 2022 EDT

1. n/a (8214)	2. United States (1942)	3. Germany (1455)
4. France (518)	5. Netherlands (350)	6. Canada (312)
7. United Kingdom (229)	8. Finland (222)	9. Russian Federation (220)
10. Switzerland (139)	11. Singapore (121)	12. China (113)
13. Australia (104)	14. Japan (101)	15. Czech Republic (87)
16. Sweden (80)	17. Hong Kong (65)	18. Spain (62)
19. Ireland (62)	20. Brazil (55)	21. Italy (54)
22. Ukraine (53)	23. Poland (46)	24. Lithuania (46)
25. Romania (44)	26. Korea, Republic of (42)	27. Austria (39)
28. Bulgaria (36)	29. Belgium (31)	30. Norway (29)
31. India (26)	32. Hungary (23)	33. Portugal (23)
34. Argentina (23)	35. New Zealand (20)	36. Slovakia (20)
37. Taiwan (19)	38. Thailand (18)	39. South Africa (16)
40. Mexico (14)	41. Slovenia (14)	42. Denmark (14)
43. Malaysia (13)	44. Greece (12)	45. Estonia (11)
46. Moldova, Republic of (11)	47. Turkey (11)	48. Latvia (11)
49. Croatia (10)	50. Vietnam (10)	51. Iceland (9)
52. Israel (8)	53. Chile (8)	54. Luxembourg (7)
55. Kazakhstan (6)	56. Serbia (5)	57. Colombia (5)
58. Cyprus (5)	59. Iran, Islamic Republic of (5)	60. Belarus (4)
61. Panama (4)	62. Ecuador (4)	63. Malta (3)
64. Costa Rica (3)	65. United Arab Emirates (3)	66. Indonesia (3)
67. Bangladesh (2)	68. Jersey (2)	69. Gibraltar (2)
70. Montenegro (2)	71. Uruguay (2)	72. Isle of Man (2)
73. Faroe Islands (2)	74. Kyrgyzstan (2)	75. Cambodia (2)
76. Kuwait (2)	77. Seychelles (2)	78. Andorra (2)
79. Virgin Islands, U.S. (2)	80. Azerbaijan (2)	81. Qatar (2)
82. Bosnia and Herzegovina (1)	83. Belize (1)	84. Guatemala (1)
85. Honduras (1)	86. Venezuela (1)	87. Puerto Rico (1)
88. Philippines (1)	89. Zimbabwe (1)	90. Mauritius (1)
91. Tanzania, United Republic of (1)	92. El Salvador (1)	93. Dominican Republic (1)
94. Mayotte (1)	95. Lebanon (1)	96. Saint Lucia (1)
97. Armenia (1)	98. Aland Islands (1)	99. Mozambique (1)

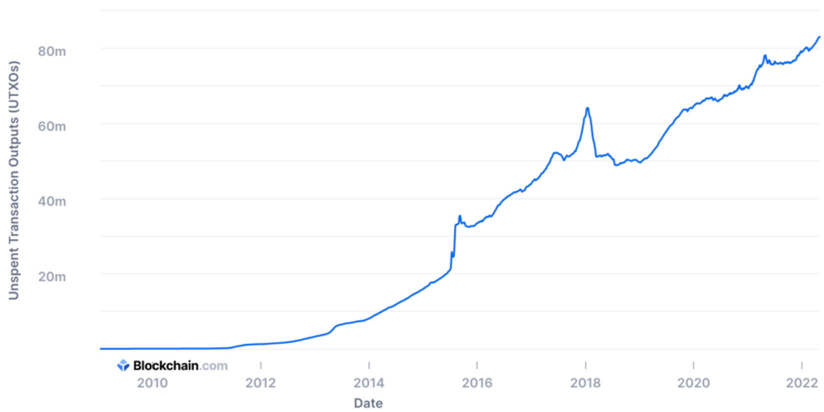
[C] Blockchain.com Graphs





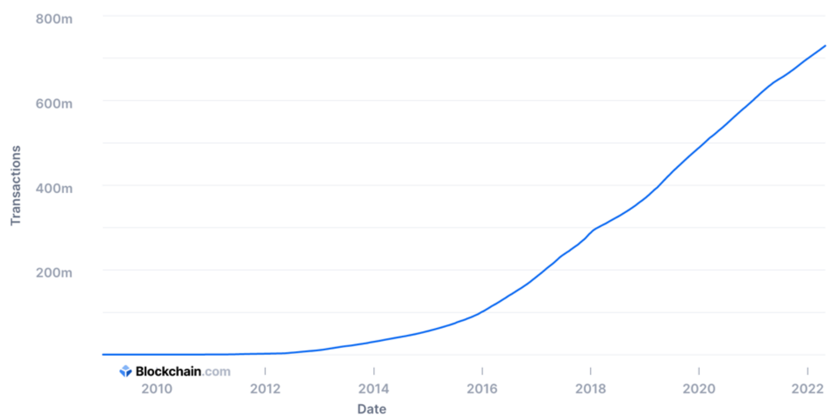
Unspent Transaction Outputs

The total number of valid unspent transaction outputs. This excludes invalid UTXOs with opcode OP_RETURN



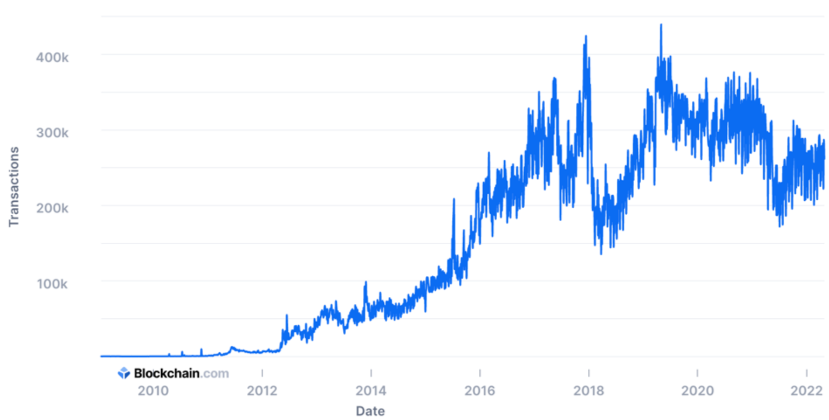
Total Number of Transactions

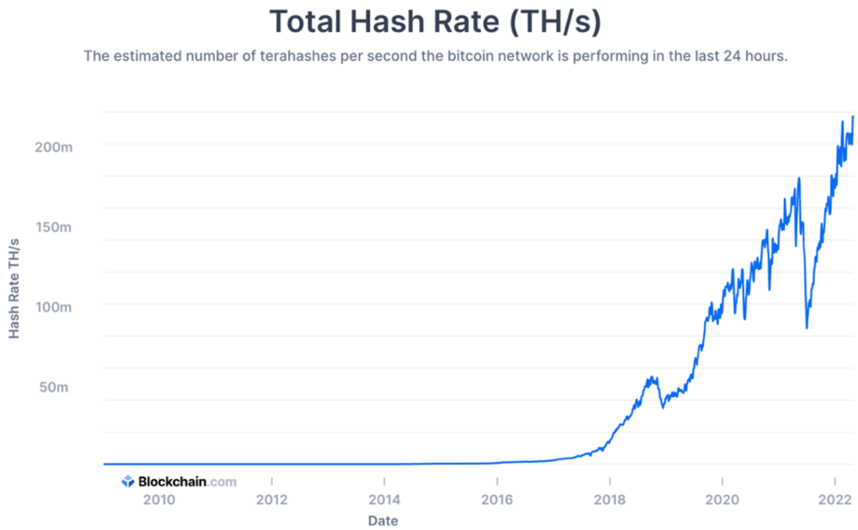
The total number of transactions on the blockchain.

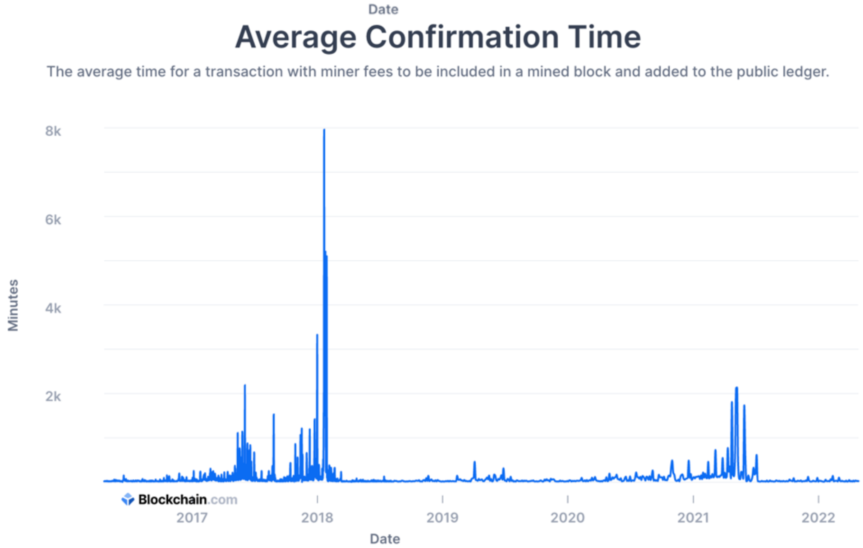


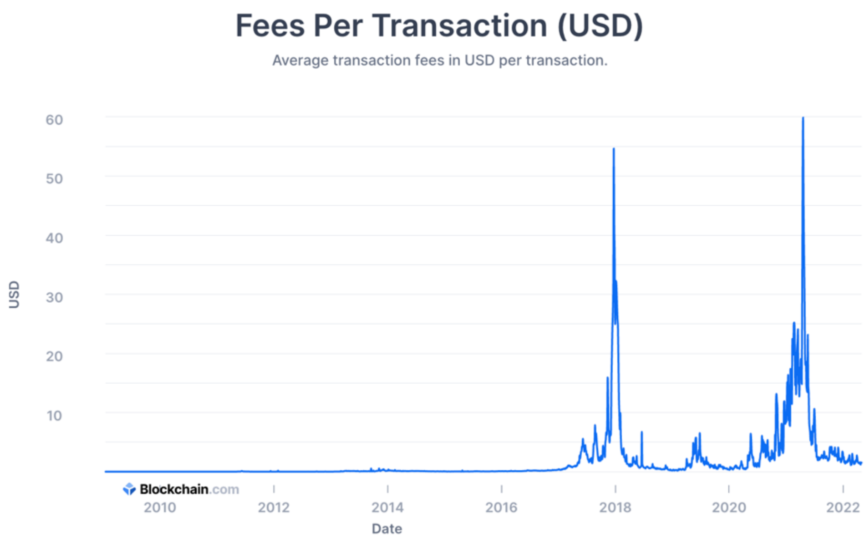
Confirmed Transactions Per Day

The total number of confirmed transactions per day.




Total Hash Rate (TH/s)
The estimated number of terahashes per second the bitcoin network is performing in the last 24 hours.
200m
150m
100m
50m
Hash Rate TH/s
Blockchain.com
2010
2012
2014
2016
2018
2020
2022
Date


Average Confirmation Time
The average time for a transaction with miner fees to be included in a mined block and added to the public ledger.
8k
6k
4k
2k
Minutes
Blockchain.com
2017
2018
2019
2020
2021
2022
Date


Fees Per Transaction (USD)
Average transaction fees in USD per transaction.
USD
60
50
40
30
20
10
Blockchain.com
2010
2012
2014
2016
2018
2020
2022
Date


Miners Revenue (USD)
Total value in USD of coinbase block rewards and transaction fees paid to miners.
USD
80m
60m
40m
20m
Blockchain.com
2010
2012
2014
2016
2018
2020
2022
Date

[D] Bitcoin Supply Schedule

Bitcoin Supply Schedule						
	Date	Block Height	Reward (Bitcoin)	Total Circulating Supply (Bitcoin)	Percent Mined	Total Unmined Supply (Bitcoin
Past	1/3/2009	0	**50**	**0.0000**	0.00000000%	**20999999.9769**
	11/28/2012	210,000	**25**	**10500000.0000**	50.00000006%	**10499999.9769**
	7/9/2016	420,000	**12.5**	**15750000.0000**	75.00000008%	**5249999.9769**
Current	5/12/2020	630,000	**6.25**	**18375000.0000**	87.50000010%	**2624999.9769**
Future	5/9/2024	840,000	**3.125**	**19687500.0000**	93.75000010%	**1312499.9769**
	5/7/2028	1,050,000	**1.5625**	**20343750.0000**	96.87500011%	**656249.9769**
	5/4/2032	1,260,000	**0.78125**	**20671875.0000**	98.43750011%	**328124.9769**
	5/1/2036	1,470,000	**0.390625**	**20835937.5000**	99.21875011%	**164062.4769**
	4/29/2040	1,680,000	**0.1953125**	**20917968.7500**	99.60937511%	**82031.2269**
	4/26/2044	1,890,000	**0.09765625**	**20958984.3750**	99.80468761%	**41015.6019**
	4/23/2048	2,100,000	**0.04882812**	**20979492.1875**	99.90234386%	**20507.7894**
	4/21/2052	2,310,000	**0.02441406**	**20989746.0927**	99.95117198%	**10253.8842**
	4/18/2056	2,520,000	**0.01220703**	**20994873.0453**	99.97558604%	**5126.9316**
	4/15/2060	2,730,000	**0.00610351**	**20997436.5216**	99.98779307%	**2563.4553**
	4/13/2064	2,940,000	**0.00305175**	**20998718.2587**	99.99389658%	**1281.7182**
	4/10/2068	3,150,000	**0.00152587**	**20999359.1262**	99.99694833%	**640.8507**
	4/7/2072	3,360,000	**0.00076293**	**20999679.5589**	99.99847420%	**320.4180**
	4/5/2076	3,570,000	**0.00038146**	**20999839.7742**	99.99923713%	**160.2027**
	4/2/2080	3,780,000	**0.00019073**	**20999919.8808**	99.99961859%	**80.0961**
	3/30/2084	3,990,000	**0.00009536**	**20999959.9341**	99.99980932%	**40.0428**
	3/28/2088	4,200,000	**0.00004768**	**20999979.9597**	99.99990468%	**20.0172**
	3/25/2092	4,410,000	**0.00002384**	**20999989.9725**	99.99995236%	**10.0044**
	3/22/2096	4,620,000	**0.00001192**	**20999994.9789**	99.99997620%	**4.9980**
	3/21/2100	4,830,000	**0.00000596**	**20999997.4821**	99.99998812%	**2.4948**
	3/18/2104	5,040,000	**0.00000298**	**20999998.7337**	99.99999408%	**1.2432**
	3/15/2108	5,250,000	**0.00000149**	**20999999.3595**	99.99999706%	**0.6174**
	3/13/2112	5,460,000	**0.00000074**	**20999999.6724**	99.99999855%	**0.3045**
	3/10/2116	5,670,000	**0.00000037**	**20999999.8278**	99.99999929%	**0.1491**
	3/7/2120	5,880,000	**0.00000018**	**20999999.9055**	99.99999966%	**0.0714**
	3/5/2124	6,090,000	**0.00000009**	**20999999.9433**	99.99999984%	**0.0336**
	3/2/2128	6,300,000	**0.00000004**	**20999999.9622**	99.99999993%	**0.0147**
	2/28/2132	6,510,000	**0.00000002**	**20999999.9706**	99.99999997%	**0.0063**
	2/26/2136	6,720,000	**0.00000001**	**20999999.9748**	99.99999999%	**0.0021**
	2/23/2140	6,930,000	**0**	**20999999.9769**	100.00000000%	**0.0000**

[E] Common Ports

Common Ports/Services (https://ipwithease.com/common-tcp-ip-well-known-port-numbers/)			
PORT NUMBER	TRANSPORT PROTOCOL	SERVICE NAME	RFC
20, 21	TCP	File Transfer Protocol (FTP)	RFC 959
22	TCP and UDP	Secure Shell (SSH)	RFC 4250-4256
23	TCP	Telnet	RFC 854
25	TCP	Simple Mail Transfer Protocol (SMTP)	RFC 5321
53	TCP and UDP	Domain Name Server (DNS)	RFC 1034-1035
67, 68	UDP	Dynamic Host Configuration Protocol (DHCP)	RFC 2131
69	UDP	Trivial File Transfer Protocol (TFTP)	RFC 1350
80	TCP	HyperText Transfer Protocol (HTTP)	RFC 2616
110	TCP	Post Office Protocol (POP3)	RFC 1939
119	TCP	Network News Transport Protocol (NNTP)	RFC 8977
123	UDP	Network Time Protocol (NTP)	RFC 5905
135-139	TCP and UDP	NetBIOS	RFC 1001-1002
143	TCP and UDP	Internet Message Access Protocol (IMAP4)	RFC 3501
161, 162	TCP and UDP	Simple Network Management Protocol (SNMP)	RFC 1901-1908, 3411-3418
179	TCP	Border Gateway Protocol (BGP)	RFC 4271
389	TCP and UDP	Lightweight Directory Access Protocol	RFC 4510
443	TCP and UDP	HTTP with Secure Sockets Layer (SSL)	RFC 2818
500	UDP	Internet Security Association and Key Management Protocol (ISAKMP) / Internet Key Exchange (IKE)	RFC 2408 - 2409
636	TCP and UDP	Lightweight Directory Access Protocol over TLS/SSL (LDAPS	RFC 4513
989/990	TCP	FTP over TLS/SSL	RFC 4217

[F] Bitcoin Core Version History

Bitcoin Software Version History	
Software Name & Versi	Release Date
Bitcoin Core 22.0	9/13/21
Bitcoin Core 0.21.1	5/1/21
Bitcoin Core 0.21.0	1/14/21
Bitcoin Core 0.20.1	8/1/20
Bitcoin Core 0.20.0	6/3/20
Bitcoin Core 0.19.1	3/9/20
Bitcoin Core 0.19.0.1	11/24/19
Bitcoin Core 0.18.1	8/9/19
Bitcoin Core 0.18.0	5/2/19
Bitcoin Core 0.17.1	12/25/18
Bitcoin Core 0.17.0.1	10/30/18
Bitcoin Core 0.17.0	10/3/18
Bitcoin Core 0.15.2	9/28/18
Bitcoin Core 0.16.3	9/18/18
Bitcoin Core 0.16.2	7/29/18
Bitcoin Core 0.16.1	6/15/18
Bitcoin Core 0.16.0	2/26/18
Bitcoin Core 0.15.1	11/11/17
Bitcoin Core 0.15.0.1	9/19/17
Bitcoin Core 0.15.0	9/14/17
Bitcoin Core 0.14.2	6/17/17
Bitcoin Core 0.14.1	4/22/17
Bitcoin Core 0.14.0	3/8/17
Bitcoin Core 0.13.2	1/3/17
Bitcoin Core 0.13.1	10/27/16
Bitcoin Core 0.13.0	8/23/16
Bitcoin Core 0.12.1	4/15/16
Bitcoin Core 0.12.0	2/23/16
Bitcoin Core 0.11.2	11/13/15
Bitcoin Core 0.11.1	10/15/15
Bitcoin Core 0.10.3	10/14/15
Bitcoin Core 0.11.0	7/12/15
Bitcoin Core 0.10.2	5/19/15
Bitcoin Core 0.10.1	4/27/15
Bitcoin Core 0.10.0	2/16/15
Bitcoin Core 0.9.3	9/27/14
Bitcoin Core 0.9.2.1	6/19/14
Bitcoin Core 0.9.2	6/16/14
Bitcoin Core 0.9.1	4/8/14
Bitcoin Core 0.9.0	3/19/14
Bitcoin-Qt 0.8.6	12/9/13
Bitcoin-Qt 0.8.5	9/13/13
Bitcoin-Qt 0.8.4	9/3/13
Bitcoin-Qt 0.8.3	6/25/13
Bitcoin-Qt 0.8.2	5/29/13
Bitcoin-Qt 0.8.1	3/18/13
Bitcoin-Qt 0.8.0	2/19/13
Bitcoin-Qt 0.7.2	12/14/12
Bitcoin-Qt 0.7.1	10/19/12
Bitcoin-Qt 0.7.0	9/17/12
Bitcoin-Qt 0.6.3	6/25/12
Bitcoin-Qt 0.6.2	5/8/12
Bitcoin-Qt 0.6.1	5/4/12
Bitcoin-Qt 0.6.0	3/30/12
Bitcoin-Qt 0.5.3.1	3/16/12
Bitcoin-Qt 0.5.3	3/14/12
Bitcoin-Qt 0.5.2	1/9/12
Bitcoin-Qt 0.5.1	12/15/11
Bitcoin-Qt 0.5.0	11/21/11
Bitcoin 0.4.0	9/23/11
Bitcoin 0.3.24	7/8/11
Bitcoin 0.3.23	6/14/11
Bitcoin 0.3.22	6/5/11
Bitcoin 0.3.21	4/27/11
Bitcoin 0.1	1/8/09

[G] Binary Conversion Tables

Binary to ASCII Conversion	
ASCII	Binary
null	00000000
start of header	00000001
start of text	00000010
end of text	00000011
end of transmission	00000100
enquire	00000101
acknowledge	00000110
bell	00000111
backspace	00001000
horizontal tab	00001001
linefeed	00001010
vertical tab	00001011
form feed	00001100
carriage return	00001101
shift out	00001110
shift in	00001111
data link escape	00010000
device control 1/Xon	00010001
device control 2	00010010
device control 3/Xof	00010011
device control 4	00010100
negative acknowled	00010101
synchronous idle	00010110
end of transmission	00010111
cancel	00011000
end of medium	00011001
end of file/ substitut	00011010
escape	00011011
file separator	00011100
group separator	00011101
record separator	00011110
unit separator	00011111
space	00100000
!	00100001
"	00100010
#	00100011
$	00100100
%	00100101
&	00100110
'	00100111
(	00101000
)	00101001
*	00101010
+	00101011
,	00101100
-	00101101
.	00101110
/	00101111
0	00110000
1	00110001
2	00110010
3	00110011
4	00110100
5	00110101
6	00110110
7	00110111
8	00111000
9	00111001
:	00111010
;	00111011
<	00111100
=	00111101
>	00111110
?	00111111
@	01000000
A	01000001
B	01000010
C	01000011
D	01000100
E	01000101
F	01000110
G	01000111
H	01001000
I	01001001
J	01001010
K	01001011
L	01001100
M	01001101
N	01001110
O	01001111
P	01010000
Q	01010001
R	01010010
S	01010011
T	01010100
U	01010101
V	01010110
W	01010111
X	01011000
Y	01011001
Z	01011010
[	01011011
\	01011100
]	01011101
^	01011110
_	01011111
`	01100000
a	01100001
b	01100010
c	01100011
d	01100100
e	01100101
f	01100110
g	01100111
h	01101000
i	01101001
j	01101010
k	01101011
l	01101100
m	01101101
n	01101110
o	01101111
p	01110000
q	01110001
r	01110010
s	01110011
t	01110100
u	01110101
v	01110110
w	01110111
x	01111000
y	01111001
z	01111010
{	01111011
|	01111100
}	01111101
~	01111110
DEL	01111111

Binary Hex Conversion	
Hexadecimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
a	1010
b	1011
c	1100
d	1101
e	1110
f	1111

Secure Hashing Algorithm- Input to 256 Bit Output				
Input	**Funtion**	**Output Type**	**Output**	**Length**
0000000000000000000095913f2dc133348dcbc4fcac513e66847fd4cee7149da	Hex to Binary	Binary	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000010 01011101 10110111 00111111 00110111 00100001 10111110 10011011 10001001 00110111 01000000 10101000 01111111 11111001 11010111 10000101 11110010 01100011 10100011 00010111 01011011 11000111 00111101	256
The Text "Binary" Represented in Binary Code- 010000100110100101101110011000010111001001111001	SHA 256	Binary	00101010 01011010 01000101 11111101 00100100 11110111 00110111 00001110 00100111 01111000 01010111 00010111 11000010 00010100 11000101 01100100 00011010 10110010 11111000 00110111 11010100 11000100 10010010 11111100 11001011 01001011 01011010 10111110 01111110 00001101 01011011 01000010	256
00101010 01011010 01000101 11111101 00100100 11110111 00110111 00001110 00100111 01111000 01010111 00010111 11000010 00010100 11000101 01100100 00011010 10110010 11111000 00110111 11010100 11000100 10010010 11111100 11001011 01001011 01011010 10111110 01111110 00001101 01011011 01000010	Binary to Hex	Hexidecimal	2a5a45fd24f7370e27785717c214c5641ab2f837d4c492fccb4b5abe7e0d5b42	64

[H] Bitcoin Improvement Proposals

Bitcoin Improvement Proposal List (Bitcoin.org)					
Number	Layer	Title	Owner(s)	Type	Status
1		BIP Purpose and Guidelines	Amir Taaki	Process	Replaced
2		BIP process, revised	Luke Dashjr	Process	Active
8		Version bits with lock-in by height	Shaolin Fry, Luke Dashjr	Informational	Draft
9		Version bits with timeout and delay	Pieter Wuille, Peter Tod	Informational	Final
10	Applications	Multi-Sig Transaction Distribution	Alan Reiner	Informational	Withdrawn
11	Applications	M-of-N Standard Transactions	Gavin Andresen	Standard	Final
12	Consensus (soft fork)	OP_EVAL	Gavin Andresen	Standard	Withdrawn
13	Applications	Address Format for pay-to-script-hash	Gavin Andresen	Standard	Final
14	Peer Services	Protocol Version and User Agent	Amir Taaki, Patrick Strat	Standard	Final
15	Applications	Aliases	Amir Taaki	Standard	Deferred
16	Consensus (soft fork)	Pay to Script Hash	Gavin Andresen	Standard	Final
17	Consensus (soft fork)	OP_CHECKHASHVERIFY (CHV)	Luke Dashjr	Standard	Withdrawn
18	Consensus (soft fork)	hashScriptCheck	Luke Dashjr	Standard	Proposed
19	Applications	M-of-N Standard Transactions (Low SigOp)	Luke Dashjr	Standard	Rejected
20	Applications	URI Scheme	Luke Dashjr	Standard	Replaced
21	Applications	URI Scheme	Nils Schneider, Matt Co	Standard	Final
22	API/RPC	getblocktemplate - Fundamentals	Luke Dashjr	Standard	Final
23	API/RPC	getblocktemplate - Pooled Mining	Luke Dashjr	Standard	Final
30	Consensus (soft fork)	Duplicate transactions	Pieter Wuille	Standard	Final
31	Peer Services	Pong message	Mike Hearn	Standard	Final
32	Applications	Hierarchical Deterministic Wallets	Pieter Wuille	Informational	Final
33	Peer Services	Stratized Nodes	Amir Taaki	Standard	Rejected
34	Consensus (soft fork)	Block v2, Height in Coinbase	Gavin Andresen	Standard	Final
35	Peer Services	mempool message	Jeff Garzik	Standard	Final
36	Peer Services	Custom Services	Stefan Thomas	Standard	Rejected
37	Peer Services	Connection Bloom filtering	Mike Hearn, Matt Corall	Standard	Final
38	Applications	Passphrase-protected private key	Mike Caldwell, Aaron Vo	Standard	Draft
39	Applications	Mnemonic code for generating deterministic keys	Marek Palatinus, Pavol F	Standard	Proposed
40	API/RPC	Stratum wire protocol	Marek Palatinus	Standard	BIP number allocated
41	API/RPC	Stratum mining protocol	Marek Palatinus	Standard	BIP number allocated
42	Consensus (soft fork)	A finite monetary supply for Bitcoin	Pieter Wuille	Standard	Final
43	Applications	Purpose Field for Deterministic Wallets	Marek Palatinus, Pavol F	Informational	Final
44	Applications	Multi-Account Hierarchy for Deterministic Wallets	Marek Palatinus, Pavol F	Standard	Proposed
45	Applications	Structure for Deterministic P2SH Multisignature Wallets	Manuel Araoz, Ryan X. C	Standard	Proposed
47	Applications	Reusable Payment Codes for Hierarchical Deterministic Wallets	Justus Ranvier	Informational	Draft
48	Applications	Multi-Script Hierarchy for Multi-Sig Wallets	Fontaine	Standard	Proposed
49	Applications	Derivation scheme for P2WPKH-nested-in-P2SH based accounts	Daniel Weigl	Informational	Final
50		March 2013 Chain Fork Post-Mortem	Gavin Andresen	Informational	Final
52	Consensus (hard fork)	Durable, Low Energy Bitcoin PoW	Michael Dubrovsky, Bog	Standard	Draft
60	Peer Services	Fixed Length "version" Message (Relay-Transactions Field)	Amir Taaki	Standard	Draft
61	Peer Services	Reject P2P message	Gavin Andresen	Standard	Final
62	Consensus (soft fork)	Dealing with malleability	Pieter Wuille	Standard	Withdrawn
63	Applications	Stealth Addresses	Peter Todd	Standard	BIP number allocated
64	Peer Services	getutxo message	Mike Hearn	Standard	Obsolete
65	Consensus (soft fork)	OP_CHECKLOCKTIMEVERIFY	Peter Todd	Standard	Final
66	Consensus (soft fork)	Strict DER signatures	Pieter Wuille	Standard	Final
67	Applications	Deterministic Pay-to-script-hash multi-signature addresses through public key sorting	Thomas Kerin, Jean-Pier	Standard	Proposed
68	Consensus (soft fork)	Relative lock-time using consensus-enforced sequence numbers	Mark Friedenbach, BtcD	Standard	Final
69	Applications	Lexicographical Indexing of Transaction Inputs and Outputs	Kristov Atlas	Informational	Proposed
70	Applications	Payment Protocol	Gavin Andresen, Mike H	Standard	Final
71	Applications	Payment Protocol MIME types	Gavin Andresen	Standard	Final
72	Applications	bitcoin: uri extensions for Payment Protocol	Gavin Andresen	Standard	Final
73	Applications	Use "Accept" header for response type negotiation with Payment Request URLs	Stephen Pair	Standard	Final
74	Applications	Allow zero value OP_RETURN in Payment Protocol	Toby Padilla	Standard	Rejected
75	Applications	Out of Band Address Exchange using Payment Protocol Encryption	Justin Newton, Matt Da	Standard	Final
78	Applications	A Simple Payjoin Proposal	Nicolas Dorier	Standard	Draft
79	Applications	Bustapay :: a practical coinjoin protocol	Ryan Havar	Informational	Replaced
80		Hierarchy for Non-Colored Voting Pool Deterministic Multisig Wallets	Justus Ranvier, Jimmy S	Informational	Deferred
81		Hierarchy for Colored Voting Pool Deterministic Multisig Wallets	Justus Ranvier, Jimmy S	Informational	Deferred
83	Applications	Dynamic Hierarchical Deterministic Key Trees	Eric Lombrozo	Standard	Rejected
84	Applications	Derivation scheme for P2WPKH based accounts	Pavol Rusnak	Informational	Draft
85	Applications	Deterministic Entropy From BIP32 Keychains	Ethan Kosakovsky	Informational	Draft
86	Applications	Key Derivation for Single Key P2TR Outputs	Andrew Chow	Standard	Draft
87	Applications	Hierarchy for Deterministic Multisig Wallets	Robert Spigler	Standard	Proposed
88	Applications	Hierarchical Deterministic Path Templates	Dmitry Petukhov	Informational	Proposed
90		Buried Deployments	Suhas Daftuar	Informational	Final
91	Consensus (soft fork)	Reduced threshold Segwit MASF	James Hilliard	Standard	Final
98	Consensus (soft fork)	Fast Merkle Trees	Mark Friedenbach, Kalle	Standard	Draft
99		Motivation and deployment of consensus rule changes ([soft/hard]forks)	Jorge Timón	Informational	Rejected
100	Consensus (hard fork)	Dynamic maximum block size by miner vote	Jeff Garzik, Tom Harding	Standard	Rejected
101	Consensus (hard fork)	Increase maximum block size	Gavin Andresen	Standard	Withdrawn
102	Consensus (hard fork)	Block size increase to 2MB	Jeff Garzik	Standard	Rejected
103	Consensus (hard fork)	Block size following technological growth	Pieter Wuille	Standard	Withdrawn
104	Consensus (hard fork)	'Block75' - Max block size like difficulty	t.khan	Standard	Rejected
105	Consensus (hard fork)	Consensus based block size retargeting algorithm	BtcDrak	Standard	Rejected
106	Consensus (hard fork)	Dynamically Controlled Bitcoin Block Size Max Cap	Upal Chakraborty	Standard	Rejected
107	Consensus (hard fork)	Dynamic limit on the block size	Washington Y. Sanchez	Standard	Rejected
109	Consensus (hard fork)	Two million byte size limit with sigop and sighash limits	Gavin Andresen	Standard	Rejected
111	Peer Services	NODE_BLOOM service bit	Matt Corallo, Peter Todd	Standard	Proposed
112	Consensus (soft fork)	CHECKSEQUENCEVERIFY	BtcDrak, Mark Friedenba	Standard	Final
113	Consensus (soft fork)	Median time-past as endpoint for lock-time calculations	Thomas Kerin, Mark Frie	Standard	Final
114	Consensus (soft fork)	Merkelized Abstract Syntax Tree	Johnson Lau	Standard	Rejected
115	Consensus (soft fork)	Generic anti-replay protection using Script	Luke Dashjr	Standard	Rejected
116	Consensus (soft fork)	MERKLEBRANCHVERIFY	Mark Friedenbach, Kalle	Standard	Draft
117	Consensus (soft fork)	Tail Call Execution Semantics	Mark Friedenbach, Kalle	Standard	Draft
118	Consensus (soft fork)	SIGHASH_ANYPREVOUT for Taproot Scripts	Christian Decker, Anthor	Standard	Draft
119	Consensus (soft fork)	CHECKTEMPLATEVERIFY	Jeremy Rubin	Standard	Draft
120	Applications	Proof of Payment	Kalle Rosenbaum	Standard	Withdrawn
121	Applications	Proof of Payment URI scheme	Kalle Rosenbaum	Standard	Withdrawn
122	Applications	URI scheme for Blockchain references / exploration	Marco Pontello	Standard	Draft
123		BIP Classification	Eric Lombrozo	Process	Active
124	Applications	Hierarchical Deterministic Script Templates	Eric Lombrozo, William	Informational	Rejected
125	Applications	Opt-in Full Replace-by-Fee Signaling	David A. Harding, Peter	Standard	Proposed
126		Best Practices for Heterogeneous Input Script Transactions	Kristov Atlas	Informational	Draft
127	Applications	Simple Proof-of-Reserves Transactions	Steven Roose	Standard	Draft
129	Applications	Bitcoin Secure Multisig Setup (BSMS)	Hugo Nguyen, Peter Gra	Standard	Proposed
130	Peer Services	sendheaders message	Suhas Daftuar	Standard	Proposed
131	Consensus (hard fork)	"Coalescing Transaction" Specification (wildcard inputs)	Chris Priest	Standard	Rejected
132		Committee-based BIP Acceptance Process	Andy Chase	Process	Withdrawn
133	Peer Services	feefilter message	Alex Morcos	Standard	Draft
134	Consensus (hard fork)	Flexible Transactions	Tom Zander	Standard	Rejected
135		Generalized version bits voting	Sancho Panza	Informational	Rejected
136	Applications	Bech32 Encoded Tx Position References	Велеслав, Jonas Schnell	Informational	Draft
137	Applications	Signatures of Messages using Private Keys	Christopher Gilliard	Standard	Final
140	Consensus (soft fork)	Normalized TXID	Christian Decker	Standard	Rejected
141	Consensus (soft fork)	Segregated Witness (Consensus layer)	Eric Lombrozo, Johnson	Standard	Final
142	Applications	Address Format for Segregated Witness	Johnson Lau	Standard	Withdrawn
143	Consensus (soft fork)	Transaction Signature Verification for Version 0 Witness Program	Johnson Lau, Pieter Wui	Standard	Final
144	Peer Services	Segregated Witness (Peer Services)	Eric Lombrozo, Pieter W	Standard	Final
145	API/RPC	getblocktemplate Updates for Segregated Witness	Luke Dashjr	Standard	Final
146	Consensus (soft fork)	Dealing with signature encoding malleability	Johnson Lau, Pieter Wui	Standard	Withdrawn
147	Consensus (soft fork)	Dealing with dummy stack element malleability	Johnson Lau	Standard	Final
148	Consensus (soft fork)	Mandatory activation of segwit deployment	Shaolin Fry	Standard	Final
149	Consensus (soft fork)	Segregated Witness (second deployment)	Shaolin Fry	Standard	Withdrawn
150	Peer Services	Peer Authentication	Jonas Schnelli	Standard	Draft
151	Peer Services	Peer-to-Peer Communication Encryption	Jonas Schnelli	Standard	Withdrawn
152	Peer Services	Compact Block Relay	Matt Corallo	Standard	Final
154	Peer Services	Rate Limiting via peer specified challenges	Karl-Johan Alm	Standard	Withdrawn
155	Peer Services	addrv2 message	Wladimir J. van der Laa	Standard	Draft
156	Peer Services	Dandelion - Privacy Enhancing Routing	Brad Denby, Andrew Mil	Standard	Rejected
157	Peer Services	Client Side Block Filtering	Olaoluwa Osuntokun, Al	Standard	Draft
158	Peer Services	Compact Block Filters for Light Clients	Olaoluwa Osuntokun, Al	Standard	Draft
159	Peer Services	NODE_NETWORK_LIMITED service bit	Jonas Schnelli	Standard	Draft
171	Applications	Currency/exchange rate information API	Luke Dashjr	Standard	Rejected
173	Applications	Base32 address format for native v0-16 witness outputs	Pieter Wuille, Greg Max	Informational	Final
174	Applications	Partially Signed Bitcoin Transaction Format	Andrew Chow	Standard	Final
175	Applications	Pay to Contract Protocol	Omar Shibli, Nicholas G	Informational	Rejected
176		Bits Denomination	Jimmy Song	Informational	Draft
178	Applications	Version Extended WIF	Karl-Johan Alm	Standard	Draft
179		Name for payment recipient identifiers	Emil Engler, MarcoFalke	Informational	Draft
180	Peer Services	Block size/weight fraud proof	Luke Dashjr	Standard	Rejected
197	Applications	Hashed Time-Locked Collateral Contract	Matthew Black, Tony Ca	Standard	Draft
199	Applications	Hashed Time-Locked Contract transactions	Sean Bowe, Daira Hopw	Standard	Draft
300	Consensus (soft fork)	Hashrate Escrows (Consensus layer)	Paul Sztorc, CryptAxe	Standard	Draft
301	Consensus (soft fork)	Blind Merged Mining (Consensus layer)	Paul Sztorc, CryptAxe	Standard	Draft
310	Applications	Stratum protocol extensions	Pavel Moravec, Jan Čape	Informational	Draft
320		nVersion bits for general purpose use	BtcDrak	Standard	Draft
322	Applications	Generic Signed Message Format	Karl-Johan Alm	Standard	Draft
325	Applications	Signet	Karl-Johan Alm, Anthony	Standard	Proposed
330	Peer Services	Transaction announcements reconciliation	Gleb Naumenko, Pieter	Standard	Draft
338	Peer Services	Disable transaction relay message	Suhas Daftuar	Standard	Draft
339	Peer Services	WTXID-based transaction relay	Suhas Daftuar	Standard	Draft
340		Schnorr Signatures for secp256k1	Pieter Wuille, Jonas Nicl	Standard	Draft
341	Consensus (soft fork)	Taproot: SegWit version 1 spending rules	Pieter Wuille, Jonas Nicl	Standard	Draft
342	Consensus (soft fork)	Validation of Taproot Scripts	Pieter Wuille, Jonas Nicl	Standard	Draft
343	Consensus (soft fork)	Mandatory activation of taproot deployment	Shinobius, Michael Folks	Standard	Proposed
350	Applications	Bech32m format for v1+ witness addresses	Pieter Wuille	Standard	Draft
370	Applications	PSBT Version 2	Andrew Chow	Standard	Draft
371	Applications	Taproot Fields for PSBT	Andrew Chow	Standard	Draft
380	Applications	Output Script Descriptors General Operation	Pieter Wuille, Andrew C	Informational	Draft
381	Applications	Non-Segwit Output Script Descriptors	Pieter Wuille, Andrew C	Informational	Draft
382	Applications	Segwit Output Script Descriptors	Pieter Wuille, Andrew C	Informational	Draft
383	Applications	Multisig Output Script Descriptors	Pieter Wuille, Andrew C	Informational	Draft
384	Applications	combo() Output Script Descriptors	Pieter Wuille, Andrew C	Informational	Draft
385	Applications	raw() and addr() Output Script Descriptors	Pieter Wuille, Andrew C	Informational	Draft
386	Applications	tr() Output Script Descriptors	Pieter Wuille, Andrew C	Informational	Draft

[I] Remote Process Calls

Remote Process Calls (Bitcoin Core)
Bitcoin Core Commands
== Blockchain ==
getbestblockhash
getblock "blockhash" (verbosity)
getblockchaininfo
getblockcount
getblockfilter "blockhash" ("filtertype")
getblockhash height
getblockheader "blockhash" (verbose)
getblockstats hash_or_height (stats)
getchaintips
getchaintxstats (nblocks "blockhash")
getdifficulty
getmempoolancestors "txid" (verbose)
getmempooldescendants "txid" (verbose)
getmempoolentry "txid"
getmempoolinfo
getrawmempool (verbose mempool_sequence)
gettxout "txid" n (include_mempool)
gettxoutproof ["txid",...] ("blockhash")
gettxoutsetinfo ("hash_type" hash_or_height use_index)
preciousblock "blockhash"
pruneblockchain height
savemempool
scantxoutset "action" ([scanobjects,...])
verifychain (checklevel nblocks)
verifytxoutproof "proof"
== Control ==
getmemoryinfo ("mode")
getrpcinfo
help ("command")
logging (["include_category",...] ["exclude_category",...])
stop
uptime
== Generating ==
generateblock "output" ["rawtx/txid",...]
generatetoaddress nblocks "address" (maxtries)
generatetodescriptor num_blocks "descriptor" (maxtries)
== Mining ==
getblocktemplate ("template_request")
getmininginfo
getnetworkhashps (nblocks height)
prioritisetransaction "txid" (dummy) fee_delta
submitblock "hexdata" ("dummy")
submitheader "hexdata"
== Network ==
addnode "node" "command"
clearbanned
disconnectnode ("address" nodeid)
getaddednodeinfo ("node")
getconnectioncount
getnettotals
getnetworkinfo
getnodeaddresses (count "network")
getpeerinfo
listbanned
ping
setban "subnet" "command" (bantime absolute)
setnetworkactive state
== Rawtransactions ==
analyzepsbt "psbt"
combinepsbt ["psbt",...]
combinerawtransaction ["hexstring",...]
converttopsbt "hexstring" (permitsigdata iswitness)
createpsbt [{"txid":"hex","vout":n,"sequence":n},...] [{"address":amount,...},{"data":"hex"},...] (locktime replaceable)
createrawtransaction [{"txid":"hex","vout":n,"sequence":n},...] [{"address":amount,...},{"data":"hex"},...] (locktime replaceable)
decodepsbt "psbt"
decoderawtransaction "hexstring" (iswitness)
decodescript "hexstring"
finalizepsbt "psbt" (extract)
fundrawtransaction "hexstring" (options iswitness)
getrawtransaction "txid" (verbose "blockhash")
joinpsbts ["psbt",...]
sendrawtransaction "hexstring" (maxfeerate)
signrawtransactionwithkey "hexstring" ["privatekey",...] ([{"txid":"hex","vout":n,"scriptPubKey":"hex","redeemScript":"hex","witnessScript":"hex","amount":amount},...] "sighashtype")
testmempoolaccept ["rawtx",...] (maxfeerate)
utxoupdatepsbt "psbt" (["",{"desc":"str","range":n or [n,n]},...])
== Signer ==
enumeratesigners
== Util ==
createmultisig nrequired ["key",...] ("address_type")
deriveaddresses "descriptor" (range)
estimatesmartfee conf_target ("estimate_mode")
getdescriptorinfo "descriptor"
getindexinfo ("index_name")
signmessagewithprivkey "privkey" "message"
validateaddress "address"
verifymessage "address" "signature" "message"
== Wallet ==
abandontransaction "txid"
abortrescan
addmultisigaddress nrequired ["key",...] ("label" "address_type")
backupwallet "destination"
bumpfee "txid" (options)
createwallet "wallet_name" (disable_private_keys blank "passphrase" avoid_reuse descriptors load_on_startup external_signer)
dumpprivkey "address"
dumpwallet "filename"
encryptwallet "passphrase"
getaddressesbylabel "label"
getaddressinfo "address"
getbalance ("dummy" minconf include_watchonly avoid_reuse)
getbalances
getnewaddress ("label" "address_type")
getrawchangeaddress ("address_type")
getreceivedbyaddress "address" (minconf)
getreceivedbylabel "label" (minconf)
gettransaction "txid" (include_watchonly verbose)
getunconfirmedbalance
getwalletinfo
importaddress "address" ("label" rescan p2sh)
importdescriptors "requests"
importmulti "requests" ("options")
importprivkey "privkey" ("label" rescan)
importprunedfunds "rawtransaction" "txoutproof"
importpubkey "pubkey" ("label" rescan)
importwallet "filename"
keypoolrefill (newsize)
listaddressgroupings
listdescriptors
listlabels ("purpose")
listlockunspent
listreceivedbyaddress (minconf include_empty include_watchonly "address_filter")
listreceivedbylabel (minconf include_empty include_watchonly)
listsinceblock ("blockhash" target_confirmations include_watchonly include_removed)
listtransactions ("label" count skip include_watchonly)
listunspent (minconf maxconf ["address",...] include_unsafe query_options)
listwalletdir
listwallets
loadwallet "filename" (load_on_startup)
lockunspent unlock ([{"txid":"hex","vout":n},...])
psbtbumpfee "txid" (options)
removeprunedfunds "txid"
rescanblockchain (start_height stop_height)
send [{"address":amount,...},{"data":"hex"},...] (conf_target "estimate_mode" fee_rate options)
sendmany "" {"address":amount,...} (minconf "comment" ["address",...] replaceable conf_target "estimate_mode" fee_rate verbose)
sendtoaddress "address" amount ("comment" "comment_to" subtractfeefromamount replaceable conf_target "estimate_mode" avoid_reuse fee_rate verbose)
sethdseed (newkeypool "seed")
setlabel "address" "label"
settxfee amount
setwalletflag "flag" (value)
signmessage "address" "message"
signrawtransactionwithwallet "hexstring" ([{"txid":"hex","vout":n,"scriptPubKey":"hex","redeemScript":"hex","witnessScript":"hex","amount":amount},...] "sighashtype")
unloadwallet ("wallet_name" load_on_startup)
upgradewallet (version)
walletcreatefundedpsbt ([{"txid":"hex","vout":n,"sequence":n},...]) [{"address":amount,...},{"data":"hex"},...] (locktime options bip32derivs)
walletdisplayaddress bitcoin address to display
walletlock
walletpassphrase "passphrase" timeout
walletpassphrasechange "oldpassphrase" "newpassphrase"
walletprocesspsbt "psbt" (sign "sighashtype" bip32derivs)
== Zmq ==
getzmqnotifications

[J] Elliptic Curve Cryptography Math

Elliptic curve domain parameters over F_p associated with a Koblitz curve secp256k1 **Documented by the Standards for Efficient Cryptography Group (www.secg.org)**	
Parameter	**Value**
a	a is the constant that define the ellipc curve $y^2 = x^3 + \mathbf{a}x + b$ a = 0
b	b is the constant that define the ellipc curve $y^2 = x^3 + ax + \mathbf{b}$ b = 7
p	A finite field is a field with a finite number of elements, called its order (the size of the underlying set). The number of elements is the prime number p. F_p is called the prime field of order p, and is the field of residue classes modulo p, where the p elements are denoted 0, ..., p - 1. This means prime number p should be used for all the finite field math operations (better known as modulo operation), for example: $y^2 \bmod p = (x^3 + ax + b) \bmod p$ The output of the math operation should never be bigger than the p value. $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1 = 2^{256} - 2^{32} - 977 =$ Hexadecimal: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F Decimal: 115792089237316195423570985008687907853269984665640564039457584007908834671663
G	The base point G is a predetermined point (x_G, y_G) on the elliptic curve that everyone uses to compute other points on the curve. Often the base point G is displayed in two ways: • **Compressed form (prefix 02)** **02** 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798 If the prefix is removed, the value is the x_G coordinate. To get the y_G coordinate, calculate $y_G = (x_G{}^3 + 7)^{1/2}$ • **Uncompressed form (prefix 04)** **04** 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8 If the prefix is removed, the first half of the value is the x_G coordinate and the last half is the y_G coordinate.
x_G	Hexadecimal: 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798 Decimal: 55066263022277343669578718895168534326250603453777594175500187360389116729240
y_G	Hexadecimal: 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8 Decimal: 32670510020758816978083085130507043184471273380659243275938904335757337482424
n	The prime n which is the order of base point G. The parameter n determines which is the maximum value that can be turned into a Bitcoin private key. Any 256-bit number in the range [1, n - 1] is a valid private key. Hexadecimal: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141 Decimal: 115792089237316195423570985008687907852837564279074904382605163141518161494337 Thus any 256-bit number from 0x1 to 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364140 is a valid private key.
h	The cofactor: 01

[K] Secure Hashing Algorithm 256 Example

Raw Header Inputs (https://blockchain.info/block-index/506679)

Version	2
Prev. Block	00000000000000000A2940884E0C3BC96510CAD11912A527E9D15DF42F0E1D67
Merkle Root	2E99F445C007A9158207CC30CEBAD2B3D26C45FDAB2EBDF50D261335FC00D92C
Time	12/16/14 18:05:40
Bits	404454260
Nonce	3225483075

Block Hash	**000000000000000015A8D88216918C8DE090268A5E7F53FEEF72CD111F7F27FF**

Digest 1	**09A0D19192EF77C304FE447888F9EF5069D648465A19146FB770619714D08904**				
A	09A0D191	=	6A09E667	+	9F96EB2A
B	92EF77C3	=	BB67AE85	+	D787C93E
C	04FE4478	=	3C6EF372	+	C88F5106
D	88F9EF50	=	A54FF53A	+	E3A9FA16
E	69D64846	=	510E527F	+	18C7F5C7
F	5A19146F	=	9B05688C	+	BF13ABE3
G	B7706197	=	1F83D9AB	+	97EC87EC
H	14D08904	=	5BE0CD19	+	B8EFBBEB

Digest 2	**3EBB2D68D7007148B184E57BBA9697D76BC04141155C57F97E3B92C5FD6A46BD**				
A	3EBB2D68	=	09A0D191	+	351A5BD7
B	D7007148	=	92EF77C3	+	4410F985
C	B184E57B	=	04FE4478	+	AC86A103
D	BA9697D7	=	88F9EF50	+	319CA887
E	6BC04141	=	69D64846	+	01E9F8FB
F	155C57F9	=	5A19146F	+	BB43438A
G	7E3B92C5	=	B7706197	+	C6CB312E
H	FD6A46BD	=	14D08904	+	E899BDB9

Digest 3	**FF277F1F11CD72EFFE537F5E8A2690E08D8C911682D8A8150000000000000000**				
A	FF277F1F	=	6A09E667	+	951D98B8
B	11CD72EF	=	BB67AE85	+	5665C46A
C	FE537F5E	=	3C6EF372	+	C1E48BEC
D	8A2690E0	=	A54FF53A	+	E4D69BA6
E	8D8C9116	=	510E527F	+	3C7E3E97
F	82D8A815	=	9B05688C	+	E7D33F89
G	00000000	=	1F83D9AB	+	E07C2655
H	00000000	=	5BE0CD19	+	A41F32E7

[L] Proof-of-Work Chain