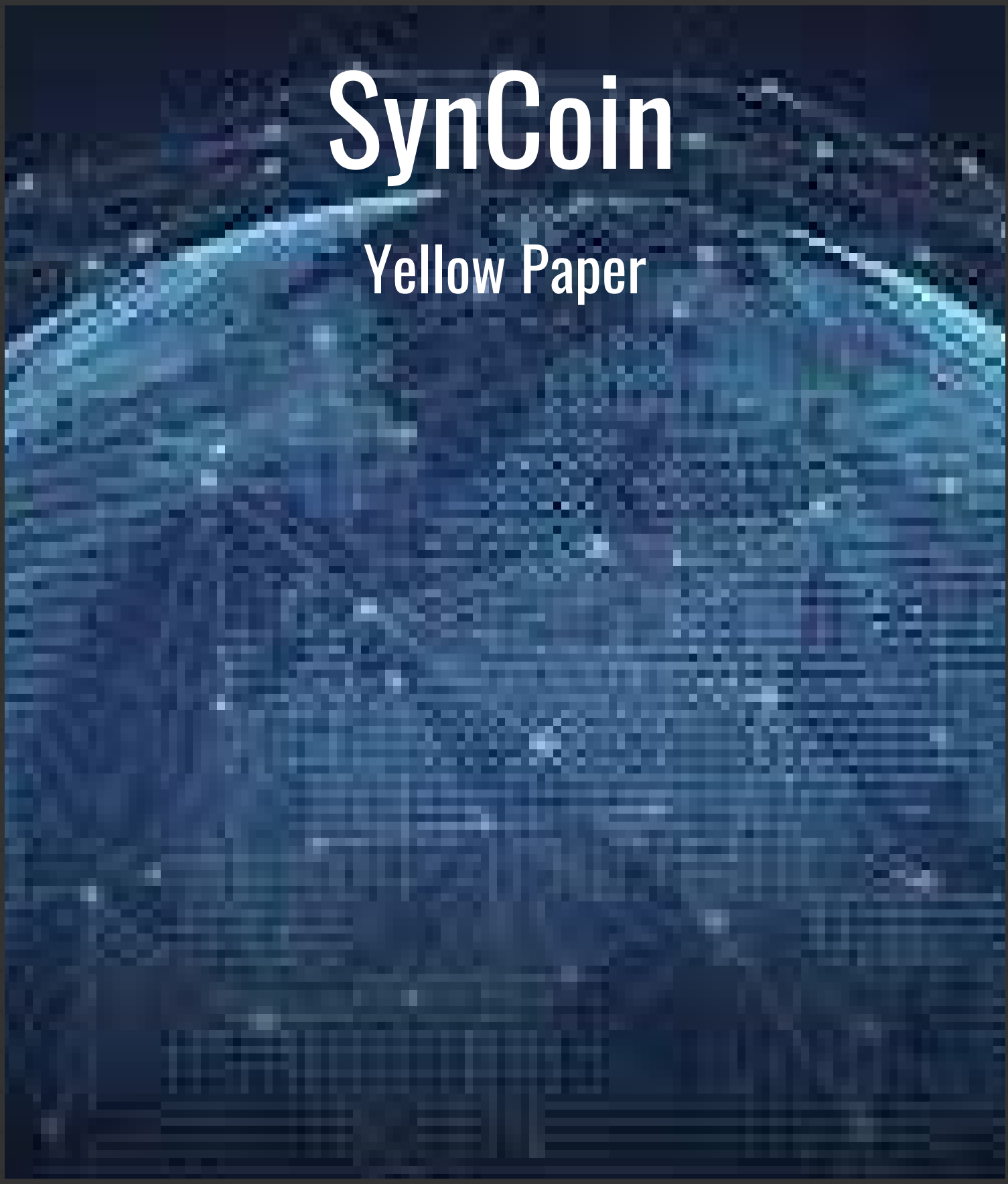


SynCoin

Yellow Paper

ShredTech, Inc

SynCoin Yellow Paper 2019

Syncoin Yellowpaper

Abstract. Syncoin is a cryptographic currency based on code from Bitcoin Core[1] Version 0.14.0[20] and Emercoin[3]. Syncoin's Proof-of-Work (PoW) uses the 'Scrypt' algorithm and also utilises Proof-of-Stake (PoS) from Peercoin. This system secures the protocol through an efficient, decentralised process called "minting".

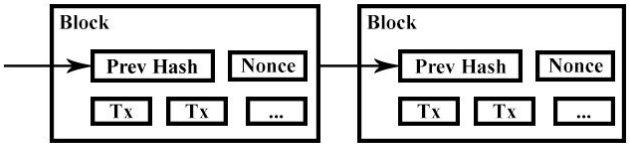
I. PROOF-OF-WORK

Syncoin's Proof-of-Work (PoW) phase exists for a period of 10,000. Because of the time taken for the initial coins to mature before staking can begin, it will be necessary to have this PoW mining period. This ensures a secure and smooth transition from PoW to Proof-of-Stake (PoS) mining.

To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a PoW system similar to Adam Back's Hashcash[8], rather than newspaper or Usenet posts.

The PoW involves scanning for a value that, when hashed for example with Scrypt, begins with a number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash.

For our timestamp network, we implement the PoW by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits. Once the CPU effort has been expended to make it satisfy the PoW, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.



The PoW also solves the problem of determining representation in majority decision making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. PoW is essentially one-CPU-one-vote. The majority decision is represented by the longest chain, which has the greatest PoW effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains.

To modify a past block, an attacker would have to redo the PoW of that block and all blocks after it, then catch up with and surpass the work of the honest nodes. The probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added. To compensate for increasing hardware speed and varying interest in running nodes over time, the PoW difficulty is determined by a moving average targeting an average number of blocks per hour. If they're generated too fast, the difficulty increases.

II. PROOF-OF-STAKE

Consensus in a decentralised digital currency, like Bitcoin[1], is achieved by requiring generated blocks to contain a proof that the node which generated the block solved a computational hard task. Unfortunately, the concept of the Proof-of-Work (PoW) based system tends to lean towards eventual self-destruction[9] due to 51% attack vulnerabilities and the high cost of energy usage.

Proof-of-Stake (PoS) aims to replace the way of achieving consensus in a distributed system; instead of solving the PoW, the node which generates a block has to provide a proof that it has access to a certain amount of coins before being accepted by the network. Generating a block involves sending coins to oneself, which proves the ownership. The required amount of coins (also called target) is specified by the network through a difficulty adjustment process similar to PoW that ensures an approximate, constant block time

As in PoW, the block generation process will be rewarded through transaction fees and a supply model specified by the underlying protocol. This process can be seen as an interest rate by common definition. The initial distribution of the currency is obtained through a period of PoW mining.

III. SECURITY ISSUES IN PROOF-OF-STAKE

Besides the clear advantage of Proof-of-Stake (PoS) over Proof-of-Work (PoW) as a method used to establish consensus on the network, there exist problems that have yet to be solved that can greatly improve network security.

A. Coin Age

In the Peercoin[2] protocol, block generation is based on coin age which is a factor that increases the weight of unspent coins linearly over time. This is the proof that has to be provided together with a new block and has to satisfy the following condition:

proofhash < coins · age · target

The proof hash corresponds to the hash of an obfuscation sum that depends on a stake modifier, the unspent output, and the current time.

With this system it is possible for an attacker to save up enough coin age to become the node with the highest weight on the network. If the attack were to be malicious the attacker could then fork the blockchain and perform a double-spend. After this is done however, a second double-spend would require the attacker to save up coin age again, as the stake resets when the block was generated.

It is worth mentioning that this situation is highly improbable and that the incentive is questionable (saving enough coin age to be the highest weight on the network would either take a lot of time or a lot of coins, and thus money, to make this happen. Next to that, performing such an attack would probably devalue the system itself so it wouldn't be profitable to do the attack in the long run).

Another problem with coin age are greedy honest nodes. These are nodes that have no malicious intent but they keep their coins off the network and only stake every once in a while to get their stake reward. The current system actually encourages abusive behaviour of these nodes by keeping their node offline until it accumulates enough coin age to get the reward in a short period of time and then shut down the node again.

B. Blockchain Precomputation and Long Range Attacks

At the time of writing this paper there is no known solution for secure timestamping in a largely distributed network. The current block timestamp rules give an attacker a degree of freedom in choosing the proof hash described previously, and therefore increase the probability of a successful attempt to fork from several blocks in the past.

In addition, the current stake modifier doesn't obfuscate the hash function enough to prevent the attacker from precomputing future proofs. An individual who is seeking to maliciously attack the network would therefore be able to calculate the next interval for the future PoS solutions, allowing that individual to generate a few blocks in a row and execute a malicious attack that could harm the network.

III. CHANGES IN THE PROTOCOL

In the following we will describe the changes in the Syncoin protocol that address the problems described in the previous section.

A. Taking the Coin Age Out of the Equation.

The most secure way to perform a Proof-of-Stake (PoS) system is by having as many nodes online as possible. The more nodes that are staking, the less possibility for security issues like 51% attacks, and the faster the actual network will perform transactions through these nodes.

Thus, taking out the coin age will require all nodes to be online more to get their stake reward. Saving up coin age is no longer a possibility with the new system that calculates the chance of staking as follows:

proofhash < coins · target

Note that the system above will not change the actual stake reward.

B. Changing the Stake Modifier

In order to mitigate the possibility of the pre-computation attack, the stake modifier will be changed at every modifier interval – to better obfuscate any calculations that would be made to pinpoint the time for the next PoS.

C. Block Timestamp Rules

Appropriate changes have been made to the block timestamps to work more efficiently with PoS. Expected block time is 64 seconds to match the granularity. Note that it is assumed that nodes have an

external source of time, and if the internal time of a node deviates too much from the general consensus then there is a high probability that blocks generated by this node will get orphaned.

Bitcoin	
Past Limit	Median time of last 11 blocks
Future Limit	+2 hours
Granularity	1 second
Expected Block Time	10 minutes

Syncoin	
Past Limit	Time of last block
Future Limit	+15 seconds
Granularity	16 seconds
Expected Block Time	64 seconds

D. Static Proof-of-Stake (PoS) Reward

To avoid greedy honest nodes, coin age has been removed from the equation and a static PoS reward[13] has been implemented at 20 SYN.

The total reward depends on the number of UTXOs (Unspent Transaction Outputs). In general, the more UTXOs one has, the more reward one can get, since the relative weight consumption is small. On the other hand, there is an upper limit which depends on the total balance and the convergence to that limit is fast. Because of this one doesn't have to create many unnecessary UTXOs and overload the staking device.

A static PoS reward is beneficial for those supporting the blockchain through continual running of the client and staking, yet detrimental for greedy honest nodes as they cannot exploit coin age by retaining high numbers of coins and then receive large amounts of stake rewards for minutes of staking and then going back offline.

The static PoS reward will be time dependent. The more time you stake, the higher the total reward. The expected reward will be proportional to the total balance and the time of staking.

IV. DECENTRALISED DNS

Syncoin provides a service: storing **name->value** pairs in its blockchain. The inital idea has been borrowed from the NameCoin[18] cryptocurrency. However, where NameCoin[18] is mostly focused on supporting decentralized domain zone ***.bit** in their extension, Syncoin provides a universal service to store and maintain **name->value** pairs without imposing narrow specialization. Of course, Syncoin supports distributed alternative DNS service too. Moreover, each Syncoin wallet contains a built-in simple DNS server, supporting the standard RFC1034[19] UDP DNS protocol.

V. WATCH-ONLY ADDRESSES

This allows an address or pubkey script to be stored in the wallet without the corresponding private key, allowing the wallet to watch for outputs but not spend them.

The "importaddress" RPC command adds an address or pubkey script to the wallet without the associated private key, allowing you to watch for transactions affecting that address or pubkey script without being able to spend any of its outputs.

Parameter #1 - the address or pubkey script to watch

Name	**Type**	**Presence**	**Description**
Address or Script	String (base58 or hex)	Required (exactly 1)	Either a P2PKH or P2SH address encoded in bas58check, or a pubkey script encoded as hex

Parameter #2 - the account into which to place the address or pubkey script

Name	**Type**	**Presence**	**Description**
Address or Script	String	Optional (0 or 1)	An account name into which the address should be placed. Default is the default account, an empty string ("")

Parameter #3 - whether to rescan the block chain

Name	**Type**	**Presence**	**Description**
Rescan	bool	Optional (0 or 1)	Set to true (the default) to rescan the entire local block database or pubkey script in the wallet (including transaction affecting the newly-added address or pubkey script). Set to false to not rescan the block database (rescanning can be performed at anyt time by restarting Dynamic with the –rescan command-line argument). Rescanning may take several minutes. Notes: if the address or pubkey script is already in the wallet, the block database will not be rescanned even if this parameter is set.

Result - null on success

Name	**Type**	**Presence**	**Description**
result	null	Required (exactly 1)	If the address or pubkey script is added

			to the wallet (or is already part of the wallet), JSON null will be returned.

VI. UNITS

The SYN unit was chosen to represent a value of 10^-8 so as to give sub-unit precision rather than large whole numbers. Mirroring the standard Le Système International d'Unités, this allows for divisions of 1/10th (deci-syncoins, dSYN), 1/100th (centi-syncoins, cSYN), 1/1 000th (milli-syncoins, mSYN), and 1/1 000 000 (micro-syncoins, µSYN).

Syncoin has adopted the above method from Bitcoin and many other cryptocurrencies have adopted the same practice.

This table is to define the units of Syncoin value:

Unit	Abbreviation	Decimal (SYN)
mega-syncoin	MSYN	1,000,000
kilo-syncoin	kSYN	1,000
hecto-syncoin	hSYN	100
deca-syncoin	daSYN	10
syncoin	SYN	1
deci-syncoin	dSYN	0.1
centi-syncoin	cSYN	0.01
milli-syncoin	mSYN	0.001
micro-syncoin	µSYN	0.000001
satoshi	sat.	0.00000001

The decimal value 0.00000001, 1 satoshi or 1 sat, is being retained from Bitcoin as an homage to Satoshi Nakamoto.

VII. CHECKLOCKTIMEVERIFY

When CHECKLOCKTIMEVERIFY[16] is executed it compares the top item on the stack to the nLockTime field of the transaction containing the scriptSig. If that top stack item is greater than the transaction nLockTime the script fails immediately, otherwise script evaluation continues as though a NOP was executed.

The nLockTime field in a transaction prevents the transaction from being mined until either a certain block height, or block time, has been reached. By comparing the argument to CHECKLOCKTIMEVERIFY against the nLockTime field, we indirectly verify that the desired block height or block time has been reached; until that block height or block time has been reached the transaction output remains un-spendable.

The nLockTime field in transactions makes it possible to prove that a transaction output can be spent in the future: a valid signature for a transaction with the desired nLockTime can be constructed, proving that it is possible to spend the output with that signature when the nLockTime is reached. An example where this technique is used is in micro-payment channels, where the nLockTime field proves that should the receiver vanish the sender is guaranteed to get all their escrowed funds back when the nLockTime is reached. However, the nLockTime field is insufficient if you wish to prove that transaction output cannot be spent until sometime in the future, as there is no way to prove that the secret keys corresponding to the pubkeys controlling the funds have not been used to create a valid signature.

If Alice and Bob jointly operate a business, they may want to ensure that all funds are kept in 2-of-2 multisig transaction outputs that require the cooperation of both parties to spend. However, they recognise that in exceptional circumstances such as either party getting "hit by a bus" they need a backup plan to retrieve the funds. So they appoint their lawyer, Lenny, to act as a third-party.

With a standard 2-of-3 CHECKMULTISIG at any time Lenny could conspire with either Alice or Bob to steal the funds illegitimately. Equally Lenny may prefer not to have immediate access to the funds to discourage bad actors from attempting to get the secret keys from him by force.

However, with CHECKLOCKTIMEVERIFY the funds can be stored in scriptPubKeys in the form:

```
IF
    <now + 3 months> CHECKLOCKTIMEVERIFY DROP
    <Lenny's pubkey> CHECKSIGVERIFY
    1
ELSE
     2
ENDIF
   <Alice's pubkey> <Bob's pubkey> 2 CHECKMULTISIG
```

At any time, the funds can be spent with the following scriptSig:

```
<Alice's signature> <Bob's signature> 0
```

After 3 months have passed Lenny and one of either Alice or Bob can spend the funds with the following scriptSig:

<Alice/Bob's signature> <Lenny's signature> 1

There exist a number of protocols where a transaction output is created that the cooperation of both parties to spend the output. To ensure the failure of one party does not result in the funds becoming lost, refund transactions are setup in advance using nLockTime. These refund transactions need to be created interactively but currently vulnerable to transaction mutability. CHECKLOCKTIMEVERIFY can be used in these protocols, replacing the interactive setup with a non-interactive setup, and additionally, making transaction mutability a non-issue.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." 2008 [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[2] S. King and S. Nadal, "Peercoin – Secure & Sustainable Cryptocoin." Aug-2012 [Online]. Available: https://peercoin.net/whitepaper

[3] "Emercoin" [Online]. Available: http://emercoin.com/.

[4] E. Duffield and D. Diaz, "Dash Whitepaper." Apr-2015 [Online]. Available: https://github.com/dashpay/dash/wiki/Whitepaper

[5] "Scrypt proof of work – Bitcoin Wiki." [Online]. Available: https://en.bitcoin.it/wiki/Scrypt_proof_of_work.

[6] E. Duffield and K. Hagan, "Darkcoin: Peer-to-Peer Crypto-Currency with Anonymous Blockchain Transactions and an Improved Proof-of-Work System." Mar-2014 [Online]. Available: https://www.dash.org/wp-content/uploads/2014/09/DarkcoinWhitepaper.pdf.

[7] "Darkwallet – The bitcoin wallet with focus on privacy." [Online]. Available: https://wiki.unsystem.net/en/index.php/Darkwallet

[8] A. Back, "Hashcash-a denial of service counter-measure." 2002 [Online]. Available: http://c65mcoididjlt3zo.onion.link/pdf/hashcash.pdf.

[9] N. T. Courtois, "On The Longest Chain Rule and Programmed Self-Destruction of Crypto Currencies," *arXiv:1405.0534 [cs]*, May 2014 [Online]. Available: http://arxiv.org/abs/1405.0534.

[10] "Units – Bitcoin Wiki." [Online]. Available: https://en.bitcoin.it/wiki/Units.

[11] "Bitcoin Average – Average United States Dollar Bitcoin Price." [Online]. Available: https://bitcoinaverage.com/en/bitcoin-price/btc-to-usd.

[12] "Simple Machines Forum – Free & open source community software." [Online]. Available: http://www.simplemachines.org/.

[13] "Masternode Budget API – Official Documentation." [Online]. Available: https://dashpay.atlassian.net/wiki/display/DOC/Masternode+Budget+API.

[14] A. Biryukov, D. Dinu, and D. Khovratovich, "Argon2." Jul-2015 [Online]. Available: https://password-hashing.net/submissions/specs/Argon-v3.pdf.

[15] "BLAKE2." [Online]. Available: http://blake2.net/.

[16] P. Todd, "bitcoin/bips: OP_CHECKLOCKTIMEVERIFY," *GitHub*. [Online]. Available: https://github.com/bitcoin/bips.

[17] G. Maxwell, "CoinJoin: Bitcoin privacy for the real world." [Online]. Available: https://bitcointalk.org/index.php?topic=279249.

[18] "Namecoin" [Online]. Available: https://namecoin.org/

[19] "RFC1034" [Online]. Available: https://www.ietf.org/rfc/rfc1034.txt

[20] “Bitcoin Core version 0.14.0 released” 8-Mar-2017 [Online]. Available: https://bitcoin.org/en/release/v0.14.0