Virtacoin: A Peer-to-Peer Electronic Cash System
VirtaCoin:Crypto-Currency with proof-of-work and Scrypt algorithm

Virtacoin Core version 2.9.1
2014

Abstract

A peer-to-peer crypto-currency design derived from Satoshi Nakamoto's Bitcoin. It uses proof-of-work to provide most of the network security. The development of modern cryptocurrencies began in 2008 with the publication of an article by Satoshi Nakamoto and Bitcoin release, although it should be noted, some work in this direction was made earlier. In Bitcoin resistance against rewriting (also known as double-spend) is supported by the mechanism of Proof of Work based on the double-sha256, which requires from attacker to have more computing power than other "honest" miners altogether. Some solutions marked by Bitcoin development team were criticized by experts and as a result alternative currencies were created. NameCoin suggested to use blockchain as a distributed database. For the first time Namecoin implemented merged mining method, which allows to protect the namecoin chain with Bitcoin network. Other researchers have proposed changes to the algorithm PoW, this as it was thought to be more resistant against centralization, for example, Litecoin (scrypt). Like Bitcoin and Litecoin, we also propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work.

Introduction

Since the creation of Bitcoin (Nakamoto 2008), proof-of-work has been the predominant design of peer-to-peer crypto currency. The concept of proof-of-work has been the backbone of minting and security model of Nakamoto's design. While keeping this in mind, we have realized that, the concept of scrypt can facilitate to add more security to Bitcoin's proof-of-work system. We have since formalized a design where proof-of-work is used to build the security model of a peer-to-peer crypto currency and part of its minting process. **Scrypt** is a password-based key derivation function created by Colin Percival, originally for the Tarsnap online backup service. The algorithm was specifically designed to make it costly to perform large-scale custom hardware attacks by requiring large amounts of memory. A simplified version of scrypt is used as a proof-of-work scheme by a number of cryptocurrencies first implemented by an anonymous programmer called ArtForz in Tenebrix followed by Fairbrix and Litecoin. We have also implemented the same in our development process.

Until it became a cryptocurrency, Virtacoin was an online payment system called Virtapay. It had a centralized administrator and a big dream. In January, 2013, however, a project called "The Satoshi Project" changed the game. With the Satoshi Project, the owners of Virtapay started toying with the idea of turning the online payment system into a cryptocurrency. So on 1st July, 2014 'Virtacoin' was born. All virtapay balances were

converted to Virtacoin, shredding off 86% of all Virtapay balances it ended up as a premined genesis of 8.4 billion (8400000000). The ownership of the coin was then decentralised. Everybody who had Virtacoins became joint owners of it.

Proof-of-Work

We need a way for the payee to know that the previous owners did not sign any earlier transactions. For our purposes, we have followed the Nakamoto's proposal as close as possible. The earliest transaction is the one that counts, so we don't care about later attempts to double spend. The only way to confirm the absence of a transaction is to be aware of all transactions. In the mint based model, the mint was aware of all transactions and decided which arrived first. To accomplish this without a trusted party, transactions must be publicly announced like Bitcoin, and we need a system for participants to agree on a single history of the order in which they were received. The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received.
The solution we propose begins with a timestamp server. To implement a distributed timestamp server on a peer-to-peer basis, we have used a proof-of-work system similar to Bitcoin, balances and issuance are done through scrypt, a sequential memory-hard function. Legitimate users only need to perform the function once per operation (e.g., authentication), and so the time required is negligible. However, a brute-force attack would likely need to perform the operation billions of times, at which point the time requirements become significant and, ideally, prohibitive.

Block Generation under Proof-of-Work and Scrypt

By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them. The steady addition of a constant of amount of new coins is analogous to gold miners expending resources to add gold to circulation. In our case, it is CPU time and electricity that is expended. With a premined amount of 8.4 billion, a total of 21 billion coins (20,999,999,999.9769) will be mined and there is currently a market supply of over 11.5 billion coins. VirtaCoin has a block time of 60 seconds and with each block mined there is a reward of 6414 coins (Genesis: 8000 coins) which is being reduced at a rate of 0.5% per week since February 28, 2014.

The release of Virtacoin-Core wallet version 2.9.0 (Based on the Bitcoin wallet) and later drops the default fee required to relay transactions across the network and for miners to consider the transaction in their blocks to 0.01mVTA per kilobyte. Note that getting a transaction relayed across the network does NOT guarantee that the transaction will be accepted by a miner; by default, miners fill their blocks with 50 kilobytes of high-priority transactions, and then with 700 kilobytes of the highest-fee-per-kilobyte transactions. The minimum relay/mining fee-per-kilobyte may be changed with the minrelaytxfee option. Note that previous releases incorrectly used the mintxfee setting to determine which low-priority transactions should be considered for inclusion in blocks. The wallet code still uses a default fee for low-priority transactions of 0.1mVTA per kilobyte. During periods of heavy transaction volume, even this fee may not be enough to get transactions confirmed quickly; the mintxfee option may be used to override the default.

Difficulty is a measure of how difficult it is to find a new block. It is a human-friendly way of expressing the target. The target is a 256-bit number (extremely large) that all VTA clients share. The scrypt hash of a block's header must be lower than or equal to the current target for the block to be accepted by the network. The lower the target, the more difficult it is to generate a block. It is important to realize that block generation is not a long, set problem (like doing a million hashes), but more like a lottery. Each hash basically gives a random number between 0 and the maximum value of a 256-bit number (which is huge). If the hash is below the target, then you win. If not, you increment the nNonce (completely changing the hash) and try again. For reasons of stability and low latency in transactions, the network tries to produce one block every 60 seconds. Every Virtacoin client compares the actual time it took to generate each block with the 60 seconds target and modifies the target by the percentage difference. In other words, the difficulty is re-targeted at every block, compared to 2016 blocks by BTC and LTC. If nNonce is 0xffff0000 or above, the block is rebuilt and nNonce starts over at zero. The formula for difficulty calculation is as follows: **difficulty = difficulty_1_target / current_target (target is a 256 bit number)**

Code Snippet:

```
// Crypto++ SHA256
        // Hash pdata using pmidstate as the starting state into
        // pre-formatted buffer phash1, then hash phash1 into phash
        nNonce++;
        SHA256Transform(phash1, pdata, pmidstate);
        SHA256Transform(phash, phash1, pSHA256InitState);

        // Return the nonce if the hash has at least some zero bits,
        // caller will check if it has enough to reach the target
        if (((unsigned short*)phash)[14] == 0)
            return nNonce;

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

    // Check for stop or if block needs to be rebuilt
            boost::this_thread::interruption_point();
            if (vNodes.empty() && Params().NetworkID() != CChainParams::REGTEST)
                break;
            if (nBlockNonce >= 0xffff0000)
                break;
            if (mempool.GetTransactionsUpdated() != nTransactionsUpdatedLast &&
GetTime() - nStart > 60)
```

```
    break;
if (pindexPrev != chainActive.Tip())
    break;
```

Block Chain Protocol

The protocol for determining which competing block chain wins as main chain has been done by the usage of proof-of-work in Bitcoin's main chain protocol, whereas the total work of the block chain is used to determine main chain.

It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction to the block it's timestamped in. He can't check the transaction for himself, but by linking it to a place in the chain, he can see that a network node has accepted it, and blocks added after it further confirm the network has accepted it. A transaction is usually considered confirmed after six verfications. As of Dec 2015 the blockchain has a size of about 1 GB.

We Ignore big transactions, to avoid a send-big-orphans memory exhaustion attack. If a peer has a legitimate large transaction with a missing parent then we assume it will rebroadcast it later, after the parent transaction(s) have been mined or received. Orphan size is limited at 10,000 orphans, each of which is at most 5,000 bytes big is at most 500 megabytes of orphans.

Checkpoint: Protection of Double Spending

The verification is reliable as long as honest nodes control the network, but is more vulnerable if the network is overpowered by an attacker. While network nodes can verify transactions for themselves, the simplified method can be fooled by an attacker's fabricated transactions for as long as the attacker can continue to overpower the network. One strategy to protect against this is to accept alerts from network nodes when they detect an invalid block, prompting the user's software to download the full block and alerted transactions to confirm the inconsistency. Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

The incentive helps encourage nodes to stay honest. If a greedy attacker is able to assemble more processing power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to play by the rules, such rules that favor him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth.

We treat non-final transactions as non-standard to prevent a specific type of double-spend attack, as well as DoS attacks. (if the transaction can't be mined, the attacker isn't expending resources broadcasting it) Basically we don't want to propagate transactions that can't

included in the next block. Extremely large transactions with lots of inputs can cost the network almost as much to process as they cost the sender in fees, because computing signature hashes is O(ninputs*txsize). Limiting transactions to MAX_STANDARD_TX_SIZE mitigates CPU exhaustion attacks.

An attacker can submit a standard HASH... OP_EQUAL transaction, which will get accepted into blocks. The redemption script can be anything; an attacker could use a very expensive-to-check-upon-redemption script like: DUP CHECKSIG DROP ... repeated 100 times... OP_1. To avoid this kind of denial-of-service attacks checking transaction inputs, and making sure that any pay-to-script-hash transactions are evaluating IsStandard scripts are done.

Continuously rate limit free transactions is used to mitigate 'penny-flooding' -- sending thousands of free transactions just to be annoying or make others' transactions take longer to confirm. The Virtacoin system also fixes an issue where a 51% attack can change difficulty at will. It goes back the full period unless it's the first retarget after genesis. It Do not allow blocks that contain transactions which 'overwrite' older transactions, unless those are already completely spent. If such overwrites are allowed, transactions depending upon those can be duplicated to remove the ability to spend the first instance -- even after being sent to another address.

Add in sigops done by pay-to-script-hash inputs to prevent a "rogue miner" from creating an incredibly-expensive-to-validate block.

Code Snippet:

```
if (fStrictPayToScriptHash)
                    {
                        nSigOps += GetP2SHSigOpCount(tx, view);
                        if (nSigOps > MAX_BLOCK_SIGOPS)
                        return state.DoS(100, error("ConnectBlock() : too many sigops"),
                        REJECT_INVALID, "bad-blk-sigops");
                    }
```

Privacy

The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone. This is similar to the level of information released by stock exchanges, where the time and size of individual trades, the "tape", is made public, but without telling who the parties were. As an additional firewall, a new key pair is used for each transaction to keep them from being linked to a common owner. Some linking is still unavoidable with multi-input transactions, which necessarily reveal that their inputs were owned by the same owner. The risk is that if the owner of a key is revealed, linking could reveal other transactions that belonged to the same owner.

It is also possible to run VirtaCoin as a Tor hidden service, and connect to such services.

Other Observations

VirtaCoin addresses are similar to Bitcoin addresses and can be used to send and receive Bitcoins as well, though it is to be noted that bitcoin blockchain is enitrely different and needs to be in reach of the wallet software in order to use such feature. One can also send VirtaCoins to a Bitcoin address and Bitcoins to a VirtaCoin address, if they have both of the two blockchains stored in their PC. A single VirtaCoin or Bitcoin address can as well be used to separately manage balances of both cryptocurrencies. VirtaCoin is the first and only scrypt-based coin to do all this. Here is an example of a VirtaCoin address:

1PZiaTusAqZtiyfxUSUUUnzZSXzfAzTE1Y

It can be seen that the address also starts with a **"1"**, just like most Bitcoin addresses. **VirtaCoins** can be sent to **Bitcoin** addresses that begin with a **"3"** such those provided by **BitGo** and **GreenAddress** and generated by **CryptoLife's Universal Address Generator**. **"3"** Bitcoin addresses are mostly used in multi-signature wallets and can be used to send **Bitcoins** to **VirtaCoin** addresses as well. You will need to know the private key of your **VirtaCoin** or **Bitcoin** address to be able to view and manage the opposite balance of the wallet that originally created the address, meaning for a **VirtaCoin** address created with **VirtaCoin Core** you'll need the private key in order to see any bitcoins sent to that **VirtaCoin** address in a **Bitcoin** wallet. Exchanges and some online wallet providers don't normally provide you with the private keys of your **VirtaCoin** or **Bitcoin** address so its best to use addresses created from wallets that you have total control of, such as desktop or mobile wallets. Desktop wallets such as **VirtaCoin Core**, **MultiBit** and others available at **https://bitcoin.org/en/choose-your-wallet** can be used to export the private keys to a file on your computer. Mobile apps such as **Bitcoin Paper Wallet** and the online **Bitcoin Wallet Generator** at **BitAddress.org** can generate addresses with private keys.

To use **VirtaCoin** address as a **Bitcoin** address in a **Bitcoin** wallet simply import the private key of your **VirtaCoin** address into the wallet. The popular **Blockchain Wallet** works very well with your **VirtaCoin** address and so does **MasterCoin's Omniwallet.** With both these wallet you can import your private key with ease and manage any bitcoins sent to your address. When you import only your private key your correct **VirtaCoin/Bitcoin** address (public key) will be revealed in your list of addresses. Once you have received or sent bitcoins you should be able to view any transaction done with your **VirtaCoin** address as well as your **BTC** balance on the **Bitcoin Network** using block explorers such as **Blockchain**, **CoinPrism** or **Blockr**.

At the moment there isn't a way of directly viewing and managing the VirtaCoins sent to a "3" bitcoin address. However if you generate a "3" Bitcoin address using CryptoLife and then import the private keys into a Blockchain Wallet it will reveal a "1" Bitcoin address. That "1" Bitcoin address is associated with the "3" Bitcoin address you generated with CrytoLife, meaning they have the same private key and "Hash 160" identifier and can now be used as a VirtaCoin address. The "Hash 160" identifier is shown in Blockchain's Block Explorer whenever you try searching it for an address. If you search for your "3" address and you click on the "Hash 160" link you'll see the same "1" address that you first saw when importing the private key. To use your Bitcoin address as a VirtaCoin address you can

import the private key of your Bitcoin address into the VirtaCoin Core wallet. Please remember to backup your wallet before importing another address into **VirtaCoin Core**.

Conclusion

Upon validation of our design in the Market, we expect proof-of-work designs to become a potentially more competitive form of peer-to-peer crypto-currency due to the more evenly de-centralised distribution resisting 51% vulnerability, also 1000 times more number of coins than bitcoin can ever generate assumes more availabilty for investment and more secure wealth management offering for investors (21 million max. vs 21 billion max.) thereby achieving lower inflation/lower transaction fees at comparable network security levels.

Acknowledgement

We would like to thank Satoshi Nakamoto and Bitcoin developers whose brilliant pioneering work opened our minds and made a project like this possible, Colin Percival for his work on Scrypt algorithm, Divine Sewornu Dzokoto, Prince and all of those 1.6 million Virtapay members/Virtacoin holders for their faith, hope and support for the outcome. The Project would be impossible without the following developers, testers, miners and other contributors:

Andrey, Ashley Holman, b6393ce9-d324-4fe1-996b-acf82dbc3d53, bitsofproof, Brandon Dahler, Calvin Tam, Christian Decker, Christian von Roques, Christopher Latham, Chuck, coblee, constantined, Cory Fields, Cozz Lovan, daniel, Daniel Larimer, David Hill, Dmitry Smirnov, Drak, Eric, Lombrozo, fanquake, fcicq, Florin, frewil, Gavin Andresen, Gregory Maxwell, gubatron, Guillermo, Céspedes Tabárez, Haakon Nilsen, HaltingState,Han Lin Yap, harry, Ian Kelling, Jeff Garzik, Johnathan Corgan,Jonas Schnelli, Josh Lehan, Josh Triplett, Julian Langschaedel, Kangmo, Lake Denman, Luke Dashjr, Mark Friedenbach, Matt Corallo, Michael Bauer, Michael Ford, Michagogo, Midnight Magic, Mike Hearn, Nils Schneider, Noel Tiernan, Olivier Langlois, patrick s, Patrick Strateman, paveljanik, Peter Todd, phantomcircuit, phelixbtc, Philip Kaufmann, Pieter Wuille, Rav3nPL, R E Broadley, regergregregerrge, Robert Backhaus, Roman Mindalev, Rune K. Svendsen, Ryan Niebur, Scott Ellis, Scott Willeke, Sergey Kazenyuk, Shawn Wilkinson, Sined, sje, Subo1978, super3, Tamas Blummer, theuni, Thomas Holenstein, Timon Rapp, Timothy Stranex, Tom Geller, Torstein Husebø, Vaclav Vobornik, vhf / victor felder, Vinnie Falco, Warren Togami, Wil Bown, Wladimir J. van der Laan.

References

Nakamoto S. (2008): Bitcoin: A peer-to-peer electronic cash system.
(http://www.bitcoin.org/bitcoin.pdf)

PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake
https://www.peercoin.net/assets/paper/peercoin-paper.pdf
https://github.com/virtacoin/
https://www.virtacoincore.com
https://en.wikipedia.org/wiki/Scrypt