Whitepaper

Desire: A Privacy-Centric Crypto-Currency

Abstract. A crypto-currency based on Bitcoin, the work of Satoshi Nakamoto, with various improvements such as a two-tier incentivized network, known as the Masternode network. Included are other improvements such as PrivateSend, for increasing fungibility and InstantSend which allows instant transaction confirmation without a centralized authority.

1 Introduction

Bitcoin [1] is a cryptocurrency that has emerged as a popular medium of exchange and is the first digital currency that has attracted a substantial number of users [2]. Since its inception in 2009, Bitcoin has been rapidly growing in mainstream adoption and merchant usage [3]. A main issue with the acceptance of Bitcoin in point-of-sale (POS) situations is the time required to wait for the network to confirm the transaction made is valid, alternatively payment companies have created methods to allow vendors to take zero-confirmation transactions, but these solutions utilize a trusted counterparty to mediate the transaction outside of the protocol.

Bitcoin provides pseudonymous transactions in a public ledger, with a one-to-one relationship between sender and receiver. This provides a permanent record of all transactions that have ever taken place on the network [5]. Bitcoin is widely known in academic circles to provide a low level of privacy, although with this limitation many people still entrust their financial history to it’s blockchain.

Desire is the first privacy-centric cryptographic currency based on the work of Satoshi Nakamoto. In this paper we propose a series of improvements to Bitcoin resulting in a decentralized, strongly anonymous crypto-currency, with tamper-proof instant transactions and a secondary peer-to-peer (P2P) network incentivized to provide services to the Desire Network.

2 Masternode Network

Full nodes are servers running on a P2P network, that allow peers to use them to receive updates about the events on the network. These nodes require significant amounts of traffic and other resources that carry substantial cost. As a result, on the Bitcoin network a steady decrease in the amount of these nodes has been observed for some time [7] and as a result block propagation have been upwards of 40 seconds[14]. Many solutions have been proposed such as a new reward scheme by Microsoft Research [4] and the Bitnodes incentive program [6].

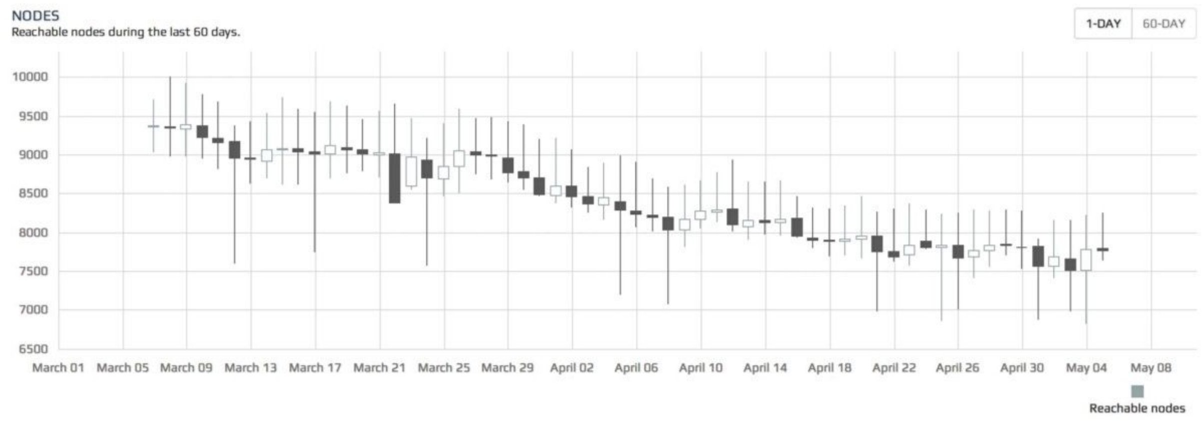


Figure 1: Bitcoin Full nodes in the spring of 2014

These nodes are very important to the health of the network. They provide clients with the ability to synchronize and quick propagation of messages throughout the network. We propose adding a secondary network, known as the Desire Masternode network. These nodes will have high availability and provide a required level of service to the network in order to take part in the Masternode Reward Program.

2.1 Masternode Reward Program - Cost and Payments

Much of the reason for the decrease of full nodes on the Bitcoin network, is the lack of incentive to run one. Over time the cost of running a full node increases as the network gets used more, creating more bandwidth and costing the operator more money. As the cost rises, operators consolidate their services to be cheaper to run or run a light client, which does not help the network at all.

Masternodes are full nodes, just like in the Bitcoin network, except they must provide a level of service to the network and have a bond of collateral to participate. Collateral is never forfeit and is safe while the Masternode is operating. This allows *investors to provide a service to the network, earn interest on their investment and reduce the volatility of the currency.*

To run a Masternode, the node must store 1,000DSR. When active, nodes provide services to clients on the network and in return are paid in the form of a dividend. This allows the users to pay for the services and earn a return on investment. Masternodes are all paid from the same pool of money, approximately 45% [footnote] of the total block reward is dedicated to this program.

Due to the fact that the Masternode rewards program is a fixed percentage and the Masternode network nodes are fluctuating, expected Masternode rewards will vary according to the current total count of active Masternodes. Payments for a standard day for running a Masternode can be calculated by using the following formula:

$$(n/t)^* \, r^* b^* a$$

Where:

n is the number of Masternodes an operator controls

t is the total number of Masternodes

r is the current block reward (presently averaging about 5 DSR)

b is blocks in an average day. For the Desire network this usually is 576.

a is the average Masternode payment (45% of the average block amount)

Return on investment for running a Masternode can be calculated as

$$((n/t)^* r \, ^* \, b^* a^* 365) \; / \; 1000$$

Where variables are the same as above.

The cost associated with running a Masternode creates a hard and soft limit of active nodes on the network. Currently with 3 million DSR in circulation, only 1,300 nodes could possibly be running on the network. The soft limit is imposed by the price it costs to acquire a node and the limited liquidity on exchanges due to usage of Desire as a currency and not merely an investment.

2.2 Deterministic Ordering

A special deterministic algorithm is used to create a pseudo-random ordering of the Masternodes. By using the hash from the proof-of-work for each block, security of this functionality will be provided by the mining network.

Pseudo Code, for selecting a Masternode:

```
For(mastenode in masternodes){
    current_score = masternode.CalculateScore();

    if(current_score > best_score){
         best_score = current_score;
         winning_node = masternode;
    }
}

CMasterNode::CalculateScore(){
   pow_hash = GetProofOfWorkHash(nBlockHeight); // get the hash of this block
   pow_hash_hash = Hash(pow_hash); //hash the POW hash to increase the entropy
   difference = abs(pow_hash_hash - masternode_vin);
   return difference;
}
```

The example code can be extended further to provide rankings of Masternodes also, a “second”, “third”, “fourth” Masternode in the list to be selected.

2.3 Trustless Quorums

Currently the Desire network has ~1,300 active Masternodes[8]. By requiring 1,000DSR collateral to become an active Masternode, we create a system in which no one can control the entire network of Masternodes. For example, if someone wants to control 50% of the Masternode network, they would have to buy 1,300,000 DSR from the open market. This would raise the price substantially and it would become impossible to acquire the needed DSR.

With the addition of the Masternode network and the collateral requirements, we can use this secondary network to do highly sensitive tasks in a trustless way, where no *single entity can control the outcome. By selecting N pseudo random Masternodes from the total pool to perform the same task, these nodes can act as an oracle, without having the whole network do the task.*

For an example, implementation of a trustless quorum (see InstantSend[9]), which uses quorums to approve transactions and lock the inputs or the proof-of-service implementation[10].

Another example use for trustless quorums can include utilizing the Masternode network as a decentralized oracle for financial markets, making secure decentralized contracts a possibility. As an example contract, if Apple Stock (AAPL) is over $300 on Feb 28, 2018 pay public key A, otherwise pay public key B.

2.4 Roles and Proof-Of-Service

Masternodes can provide any number of extra services to the network. As a proof-of-concept, our first implementation included PrivateSend and InstantSend. By utilizing what we call proof-of-service, we can require that these nodes are online, responding and even at the correct block height.

Bad actors could also run Masternodes, but not provide any of the quality service that is required of the rest of the network. To reduce the possibility of people using the system to their advantage nodes must ping the rest of the network to ensure they remain active. This work is done by the Masternode network by selecting 2 quorums per block. Quorum A checks the service of Quorum B each block. Quorum A are the closest nodes to the current block hash, while Quorum B are the furthest nodes from said hash.

Masternode A (1) checks Masternode B (rank 2300)

Masternode A (2) checks Masternode B (rank 2299)

Masternode A (3) checks Masternode B (rank 2298)

All work done to check the network to prove that nodes are active is done by the Masternode network itself. Approximately 1% of the network will be checked each *block. This results in the entire network being checked about six times per day. In order to keep this system trustless, we select nodes randomly via the Quorum system, then we also require a minimum of six violations in order to deactivate a node.*

In order to trick this system, an attacker will need to be selected six times in a row. Otherwise, violations will be cancelled out by the system as other nodes are selected by the quorum system.

Attacker Controlled Masternodes / Total Masternodes	***Required Picked Times In A Row***	***Probability of success*** $(n/t)^r$	***DSR Required***
1/2300	*6*	*6.75e-21*	*1,000DSR*
10/2300	*6*	*6.75e-15*	*10,000DSR*
100/2300	*6*	*6.75e-09*	*100,000DSR*
500/2300	*6*	*0.01055%*	*500,000DSR*
1000/2300	*6*	*0.6755%*	*1,000,000DSR*

Table 1. The probability of tricking the system representing one individual Masternode as failing proof-of-service

Where:

n is the total number of nodes controlled by the attacker

t is the total number of Masternodes in the network

r is the depth of the chain

The selection of Masternodes is pseudo random based on the Quorum system

2.5 Masternode Protocol

The Masternodes are propagated around the network using a series of protocol extensions including a Masternode announce message and Masternode ping message. These two messages are all that is needed to make a node active on the network, beyond these there are other messages for executing a proof-of-service request, PrivateSend and InstantSend.

Masternodes are originally formed by sending 1,000DSR to a specific address in a wallet that will "activate" the node making it capable of being propagated across the network. A secondary private key is created that is used for signing all further messages. The latter key allows the wallet to be completely locked when running in a standalone mode.

A cold mode is made possible by utilizing the secondary private key on two separate machines. The primary "hot" client signs the 1,000DSR input including the secondary signing private key in the message. Soon after the "cold" client sees a message including its secondary key and activates as a Masternode. This allows the "hot" client to be deactivated (client turned off) and leaves no possibility of an attacker gaining access to the 1,000DSR by gaining access to the Masternode after activation.

Upon starting, a Masternode sends a "Masternode Announce" message to the network, containing:

Message: (1K DSR Input, Reachable IP Address, Signature, Signature Time, 1K Desire Public Key, Secondary Public Key, Donation Public Key, Donation Percentage)

Every 15 minutes thereafter, a ping message is sent proving the node is still alive.

Message: (1K DSR Input, Signature (using secondary key), Signature Time, Stop)

After a time-to-live has expired the network will remove an inactive node from the network, causing the node to not be used by clients or paid. Nodes can also ping the network constantly, but if they do not have their ports open, they will eventually be flagged as inactive and not be paid.

2.6 Propagation of the Masternode List

New clients entering the Desire network must be made aware of the currently active Masternodes on the network to be able to utilize their services. As soon as they join the mesh network, a command is sent to their peers asking for the known list of Masternodes. A cache object is used for clients to record Masternodes and their *current status, so when clients restart they will simply load this file rather than asking for the full list of Masternodes.*

2.7 Payments via Mining and Enforcement

To ensure that each Masternode is paid it’s fair share of the block reward, the network must enforce that blocks pay the correct Masternode. If a miner is non-compliant their blocks must be rejected by the network, otherwise cheating will be incentivized.

We propose a strategy where Masternodes form quorums, select a winning Masternode and broadcast their message. After N messages have been broadcast to select the same target payee, a consensus will be formed and that block in question will be required to pay that Masternode.

When mining on the network, pool software (websites that merge the efforts of individual miners) use the RPC API interface to get information about how to make a block. To pay the Masternodes, this interface must be extended by adding a secondary payee to GetBlockTemplate. Pools then propagate their successfully mined blocks, with a split payment between themselves and a Masternode.

3 PrivateSend

We believe it is important to have a standard trust-less implementation for improving the privacy of it’s users in the reference client that provides a high degree of privacy. Other clients such as electrum, Android and iPhone will also have the same anonymity layer implemented directly and utilize the protocol extensions. This allows users a common experience anonymizing funds using a well understood system.

PrivateSend is an improved and extended version of the CoinJoin. In addition to the core concept of CoinJoin, we employ a series of improvements such as decentralization, strong anonymity by using a chaining approach, denominations and passive ahead-of-time mixing.

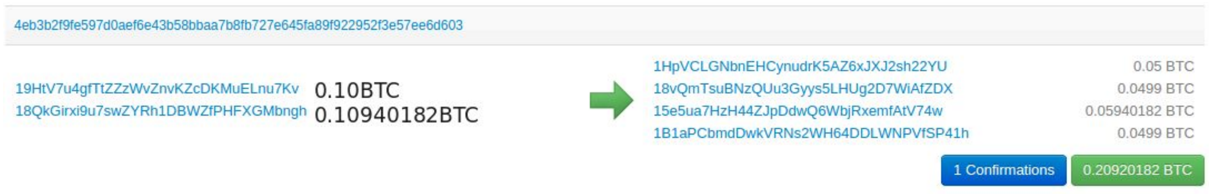
The greatest challenge when improving privacy and fungibility of a crypto-currency is doing it in a way that does not obscure the entire blockchain. In Bitcoin based crypto currencies, one can tell which outputs are unspent and which are not, commonly called UTXO, which stands for unspent transaction output. This results in a public ledger that allows any user to act as guarantor of the integrity of transactions. The Bitcoin protocol is designed to function without the participation of trusted counterparties, in their absence, it is critical that auditing capabilities remain readily accessible to the users through the public blockchain. Our goal is to improve privacy *and fungibility without losing these key elements that we believe make a successful currency.*

By having a decentralized mixing service within the currency we gain the ability to keep the currency itself perfectly fungible. Fungibility is an attribute of money, that dictates that all units of a currency should remain equal. When you receive money within a currency, it should not come with any history from the previous users of the currency or the users should have an easy way to disassociate themselves from that history, thus keeping all coins equal. At the same time, any user should be able to act as an auditor to guarantee the financial integrity of the public ledger without compromising others privacy.

To improve the fungibility and keep the integrity of the public blockchain, we propose using an ahead-of-time decentralized trustless mixing strategy. To be effective at keeping the currency fungible, this service is directly built into the currency, easy to use and safe for the average user.

3.1 Tracing CoinJoin By Amounts

A common strategy in existing Bitcoin implementations of CoinJoin is simply merging transactions together. This exposes the users to various methods of following the the users coins through these joined transaction.



```
4eb3b2f9fe597d0aef6e43b58bbaa7b8fb727e645fa89f922952f3e57ee6d603

                                                   1HpVCLGNbnEHCynudrK5AZ6xJXJ2sh22YU        0.05 BTC
19HtV7u4gfTtZZzWvZnvKZcDKMuELnu7Kv  0.10BTC        18vQmTsuBNzQUu3Gyys5LHUg2D7WiAfZDX      0.0499 BTC
18QkGirxi9u7swZYRh1DBWZfPHFXGMbngh  0.10940182BTC  15e5ua7HzH44ZJpDdwQ6WbjRxemfAtV74w  0.05940182 BTC
                                                   1B1aPCbmdDwkVRNs2WH64DDLWNPVfSP41h      0.0499 BTC

                                                   1 Confirmations  0.20920182 BTC
```

Figure 2: An example CoinJoin transaction with 2 users [11][12]

In this transaction, 0.05BTC was sent through the mixer. To identify the source of the money, one simply has to add up the values on the right until they match one of the values on the left.

Breaking apart the transaction:

- *0.05 + 0.0499 + 0.0001(fee) = 0.10BTC.*
- *0.0499 + 0.05940182 + 0.0001(fee) = 0.10940182BTC.*

This gets exponentially more difficult as more users are added to the mixer. However, these sessions can be retroactively de-anonymized at any point in the future.

3.2 Through linking and Forward Linking

In other proposed implementations of CoinJoin, it is possible that a user anonymizes money then eventually sends change from that transaction to an exchange or other entity that knows the users identity. This breaks the anonymity and allows the entity to walk backwards through that users transactions. We call this type of attack "Forward Linking":

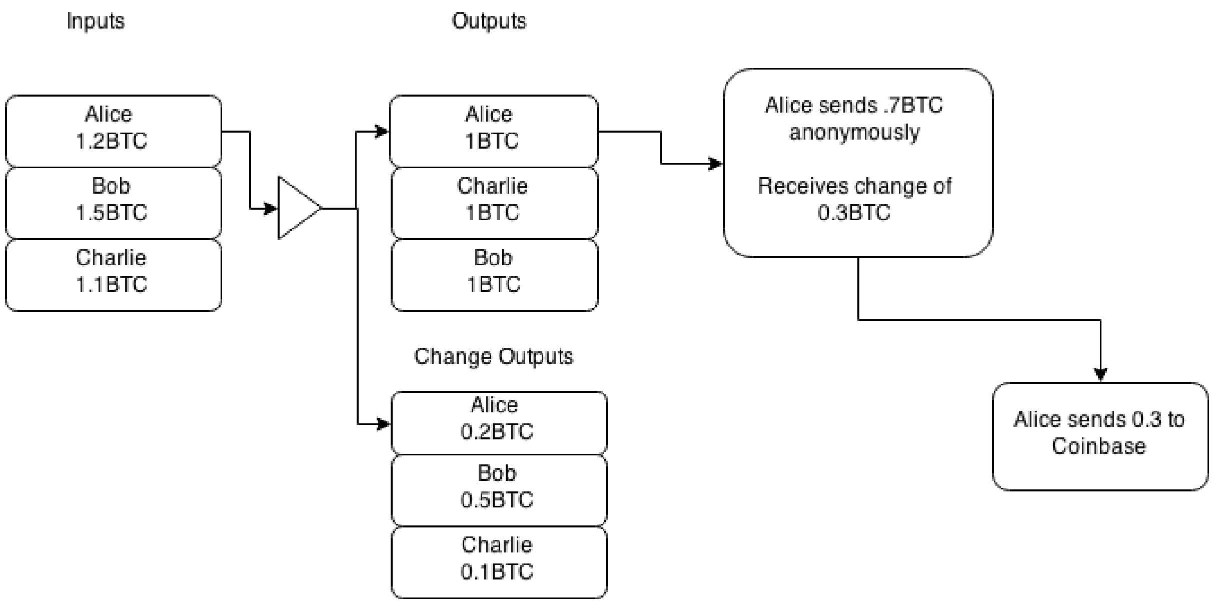


Figure 3: Forward Change Linking

In this example, Alice anonymizes 1.2BTC, which goes to two outputs, 1BTC and 0.2BTC. She then spends 0.7BTC from the 1BTC output, receiving change of 0.3BTC. That 0.3BTC then goes to an identifiable source, confirming Alice also spent the 0.7BTC in the prior transaction.

To identify the sender of the anonymous transaction, start at the "exchange" transaction and go backwards in the blockchain till you get to the "Alice sends 0.7BTC anonymously". As the exchange, you know it was your user who just recently bought something anonymously, thus breaking the anonymity completely. We call this type of attack "Through Change Linking".

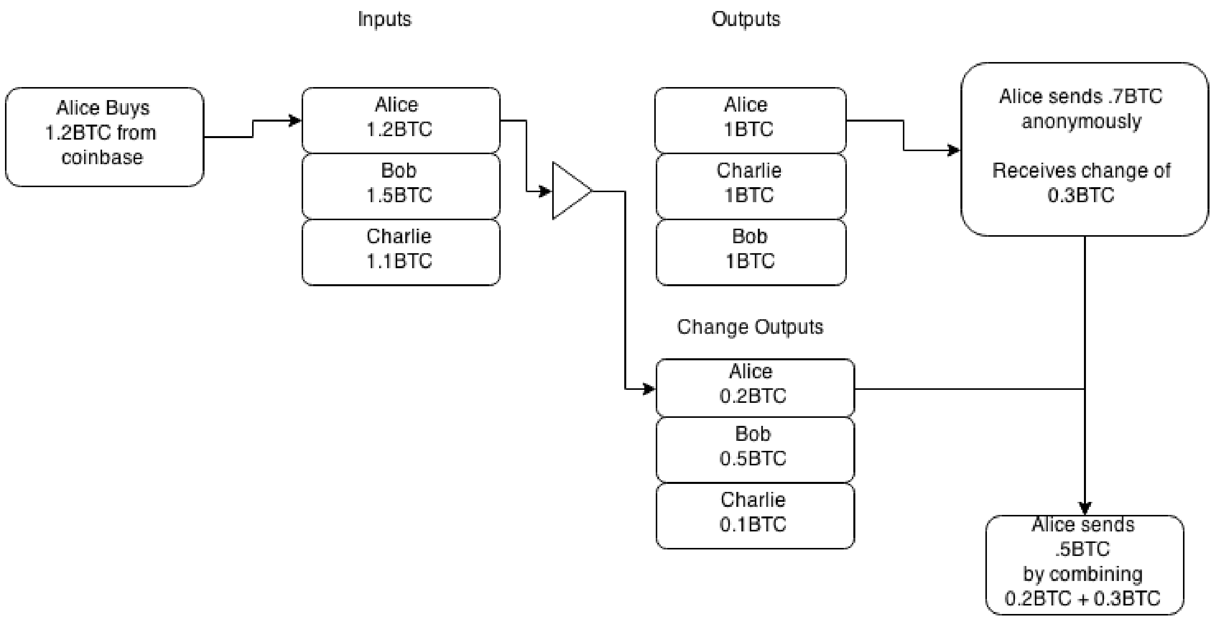

Inputs
Outputs
Alice Buys 1.2BTC from coinbase
Alice 1.2BTC
Bob 1.5BTC
Charlie 1.1BTC
Alice 1BTC
Charlie 1BTC
Bob 1BTC
Alice sends .7BTC anonymously
Receives change of 0.3BTC
Change Outputs
Alice 0.2BTC
Bob 0.5BTC
Charlie 0.1BTC
Alice sends .5BTC by combining 0.2BTC + 0.3BTC

Figure 4: Through Change Linking

In the second example, Alice buys 1.2 BTC from coinbase, then anonymizes this amount into a 1BTC output. She then spends the 1BTC, receives change in the amount of 0.3BTC and then combines that with her 0.2BTC earlier change.

By combining the change from the anonymous transaction (0.3BTC) and the change she received from the CoinJoin transaction, you can link the entire history before and after, completely breaking the anonymity.

3.3 Improved Privacy and Denial-of-service (DOS) resistance

PrivateSend uses the fact that a transaction can be formed by multiple parties and made out to multiple parties to merge funds together in a way where they cannot be uncoupled thereafter. Given that all PrivateSend transactions are setup for users to pay themselves, the system is highly secure against theft and users coins always remain safe. Currently to mix usi ng PrivateSend requires at least three participants.

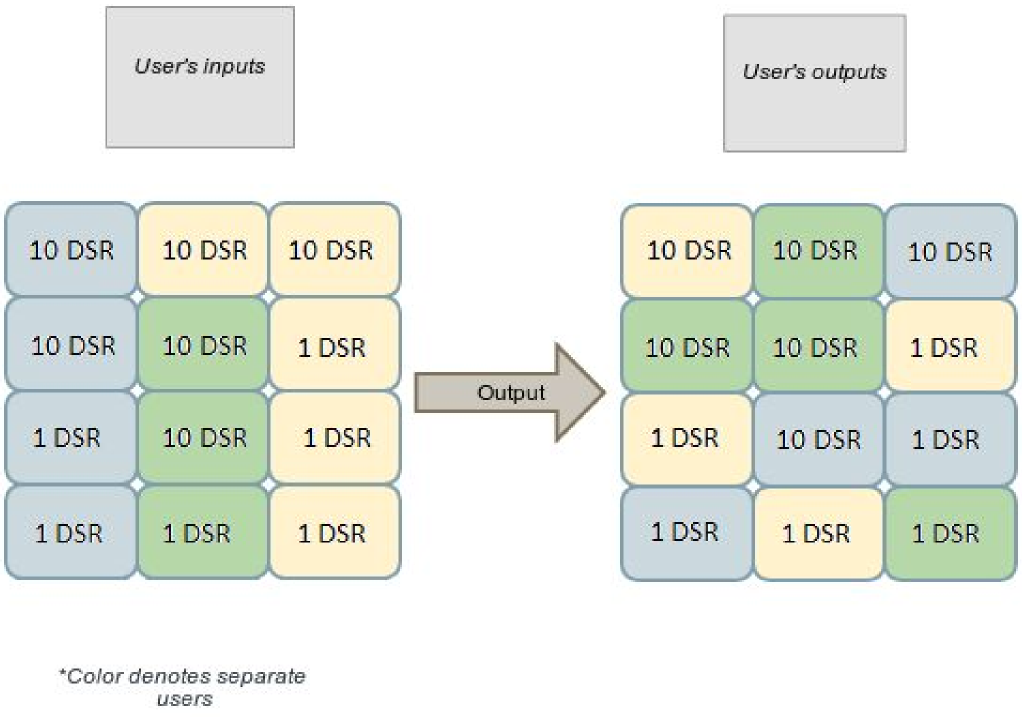


Figure 5: Three users submit denominated funds into a common transaction. Users pay themselves back in the form of new outputs, which are randomly ordered.

To improve the privacy of the system as a whole we propose using common denominations of 0.1DSR, 1DSR, 10DSR AND 100DSR. In each mixing session, all users should submit the same denominations as inputs and outputs. In addition to denominations, fees should be removed from the transactions and charged in bulk in separate, sporadic unlinkable transactions.

To address the possible DOS attacks, we propose all users submit a transaction as collateral to the pool when joining. This transaction will be made out to themselves and will pay a high fee to miners. In the case when a user submits a request to the mixing pool, they must provide collateral at the beginning of this exchange. If at any point any user fails to cooperate, by refusing to sign for example, the collateral transaction will automatically be broadcasted. This will make it expensive to do a sustained attack on the privacy network.

3.4 Passive Anonymization of funds and chaining

PrivateSend is limited to 1,000DSR per session and requires multiple sessions to thoroughly anonymize significant amounts of money. To make the user experience easy and make timing attacks very difficult, PrivateSend runs in a passive mode. At set intervals, a user's client will request to join with other clients via a Masternode.

Upon entry into the Masternode, a queue object is propagated throughout the network detailing the denominations the user is looking to anonymize, but no information that can be used to identify the user.

Each PrivateSend session can be thought of as an independent event increasing the anonymity of user's funds. However each session is limited to three clients, so an observer has a one in three chance of being able to follow a transaction. To increase the quality of anonymity provided, a chaining approach is employed, which funds are sent through multiple Masternodes, one after another.

Depth Of The Chain	***Possible Users*** $(n)^r$
2	*9*
4	*81*
8	*6561*

Table 2. How many users could possibly be involved in N mixing sessions.

As transactions are merged, Masternodes can possibly "snoop" on users funds as they pass through. This is not considered a serious limitation due to the requirement for Masternode's to hold 1,000DSR and the fact that users utilize random Masternodes that they select to host their joins. The probability of following a transaction throughout a chaining event can be calculated as follows:

Attacker Controlled Masternodes / Total Masternodes	***Depth Of The Chain***	***Probability of success*** $(n/t)^r$	***DSR Required***
10/1010	*2*	*9.80e-05*	*10,000DSR*
10/1010	*4*	*9.60e-09*	*10,000DSR*
10/1010	*8*	*9.51e-11*	*10,000DSR*
100/1100	*2*	*8.26e-03*	*100,000DSR*
100/1100	*4*	*6.83e-05*	*100,000DSR*
100/1100	*8*	*4.66e-09*	*100,000DSR*
1000/2000	*2*	*25%*	*1,000,000DSR*
1000/2000	*4*	*6.25%*	*1,000,000DSR*
1000/2000	*8*	*0.39%*	*1,000,000DSR*
2000/3000	*2*	*44.4%*	*2,000,000DSR*
2000/3000	*4*	*19.75%*	*2,000,000DSR*
2000/3000	*8*	*3.90%*	*2,000,000DSR*

Table 3. The probability of follow a PrivateSend transaction on the network given the attacker controls N Nodes.

Where:

n is the total number of nodes controlled by the attacker

t is the total number of Masternodes in the network

r is the depth of the chain

The selection of Masternodes is random.

Considering the limited supply of DSR (3 million at the time of writing, March 2018) and the low liquidity available on the market, it becomes an impossibility to attain a large enough number of Masternodes to succeed at such an attack.

Extending the system by blinding Masternodes to the transactions taking place on their node will also greatly enhance the security of the system.

3.6 Masternode Blinding via Relay System

In section 3.4 we describe the probabilities of following a single transaction through multiple sessions of PrivateSend mixing. This can further be addressed by blinding Masternodes, so they cannot see which inputs/outputs belong to which users. To do this we propose a simple relay system that users can use to protect their identity.

Instead of a user submitting the inputs and outputs directly into the pool, they will pick a random Masternode from the network and request that it relays the inputs/outputs/signatures to the target Masternode. This means that the Masternode will receive N sets of inputs/outputs and N sets of signatures. Each set belongs to one of the users, but the Masternode can't know which belongs to which.

4 Instant Transactions via InstantSend

By utilizing Masternode quorums, users are able to send and receive instant irreversible transactions. Once a quorum has been formed, the inputs of the transaction are locked to only be spendable in a specific transaction, a transaction lock takes about four seconds to be set currently on the network. If consensus is reached on a lock by the Masternode network, all conflicting transactions or conflicting blocks would be rejected thereafter, unless they matched the exact transaction ID of the lock in place.

This will allow vendors to use mobile devices in place of traditional POS systems for real world commerce and users to quickly settle face-to-face non commercial transactions as with traditional cash. This is done without a central authority. An extensive overview of this feature can be found in the InstantSend white paper[9].

5 Additional Improvements

5.1 NeoScrypt hashing algorithm

Password based key derivation function (KDF) is a deterministic algorithm used to derive a cryptographic key from an input datum known as a password. An additional input datum known as a salt may be employed in order to increase strength of the algorithm against attacks using pre-computed hashes also known as rainbow tables. The derived key length may be specified usually, and one of the most popular uses of KDFs is key stretching. It increases effective length of a user password by constructing an enhanced key to provide with a better resistance against brute force attacks. Another popular use is password storage. Keeping user passwords in unencrypted form is very undesired as it may be possible for an attacker to gain access to the password file and retrieve the passwords stored immediately. Brute force attacks may be the only possible approach against strong KDFs. This kind of attack can be parallelised usually to a great extent. High requirements on computational resources such as processor time and memory space allow to reduce parallelisation efficiency and keep these attacks expensive far beyond reasonable limits. As the name suggests, NeoScrypt is a further development of Scrypt as described in Percival. It is aimed at increased security and better performance on general purpose computer hardware while maintaining comparable costs and requirements. This document focuses on functional differences between NeoScrypt and Scrypt.

SCRYPT SPECIFICATIONS

The most popular implementation of Scrypt employed by many cryptocurrencies since 2011 is N = 1024, r = 1, p = 1 abbreviated usually to (1024, 1, 1). N is the primary parameter defining number of memory segments used and must be a power of 2. May be also described through Nfactor.

N = (1 << (Nfactor + 1))

Nfactor = lb(N) - 1

The default memory segment size for the 32-bit implementation is 128 bytes. r is the segment size multiplier. p is the computational multiplier. They may be also described through rfactor and pfactor respectively.

r = (1 << rfactor)

p = (1 << pfactor)

*A single instance of Scrypt utilises (N + 2) * r * 128 bytes of memory space, i.e. 128.25Kb for the (1024, 1, 1) configuration. Actual data mixing in memory is performed by Salsa20, a stream cipher introduced by Bernstein [2]. A reduced strength 8-round implementation has been chosen (Salsa20/8). Every run of the Scrypt core engine executes it 4 * r * N times, i.e. 4096 times for the (1024, 1, 1) configuration. Every execution of Salsa20 mixes one half of a memory segment with itself. The Scrypt core engine has no provisions for key stretching or compressing as well as salting, therefore additional cryptographic functions need to be deployed. In case of cryptocurrencies, a typical configuration operates with 80 bytes of input data (block header) which is also a salt. It is passed to PBKDF2, a password based KDF [3] capable of deriving variable length keys with salting. It works with SHA-256, a cryptographic hash function delivering digests up to 32 bytes in size through 64 internal rounds. It doesn't support keyed hashing, therefore a pseudorandom function (PRF) such as HMAC [4] is required, and the whole big endian construction may be called PBKDF2-HMAC-SHA256. It feeds r * 128 bytes of derived data to the Scrypt core and receives it back after mixing to be used as a salt for another PBKDF2-HMAC-SHA256 run which compresses 80 bytes of input data into 32 bytes of hash.*

5.2 Mining Supply

A different approach to restricting the inflation of mining is taken in Desire, using a 7% reduction of the supply per year. This is done as opposed to halving implemented by other currencies. In addition supply each block is directly tied to the amount of miners on the network; more miners result in lower mining rewards.

Production of Desire is scheduled to carry on throughout this century and onto the next, slowly grinding down until finally near the year 2150, production will cease.

6 Conclusion

This paper introduces various concepts to improve the design of bitcoin resulting in improved privacy and fungibility for the average user, less price volatility and quicker message propagation throughout the network. This is all accomplished by utilizing an incentivized two-tier model, rather than the existing single-tier model in other crypto-currencies such as Bitcoin. By utilizing this alternative network design it becomes possible to add many types of services such as decentralized mixing of coins, instant transactions and decentralized oracles using masternode quorums.

References

1. A peer-to-peer electronic cash system (2008)
2. http://eprints.qut.edu.au/69169/1/Boyen_accepted_draft.pdf
3. https://www.cryptocoinsnews.com/3-solutions-instant-bitcoin-confirmations/
4. http://research.microsoft.com/pubs/156072/bitcoin.pdf
5. http://www0.cs.ucl.ac.uk/staff/s.meiklejohn/files/imc13.pdf
6. https://getaddr.bitnodes.io/nodes/incentive/
7. https://medium.com/zapchain-magazine/why-don-t-people-run-bitcoin-nodes-anymore-d4da0b45aae5
8. https://desireninja.pl/
9. https://www.desire.org/wp-content/uploads/2014/09/InstantTX.pdf
10. https://github.com/desirepay/desire/blob/master/src/Masternode-pos.cpp
11. https://blockchain.info/tx/4eb3b2f9fe597d0aef6e43b58bbaa7b8fb727e645fa89f922952f3e57ee6d603
12. https://blockchain.info/tx/1694122b34c8543d01ad422ce600d59f8d8fde495ac9ddd894edc7139aed7617
13. http://en.wikipedia.org/wiki/NIST_hash_function_competition#Finalists
14. http://www.tik.ee.ethz.ch/file/49318d3f56c1d525aabf7fda78b23fc0/P2P2013_041.pdf
15. https://github.com/dashpay/dash/wiki/Whitepaper