The Mini-Blockchain Scheme

J.D. Bruce
July 2014. Rev 2.
www.cryptonite.info

Abstract

Almost all P2P crypto-currencies prevent double spending and similar such attacks with a bulky "blockchain" scheme, and the ones which do not typically use some sort of pseudo-decentralized solution to manage the transactions. Here we propose a purely P2P crypto-currency scheme where old transactions can be forgotten by the network. Since nodes only require the newest portion of the blockchain in order to sync with the network, we call this portion of the chain the "*mini-blockchain*". We argue that the loss of security this trimming process incurs can be solved with a small "*proof chain*" and the loss of coin ownership data is solved with a database which holds the balance of all non-empty addresses, dubbed the "*account tree*". The proof chain secures the mini-blockchain and the mini-blockchain secures the account tree. This paper will describe the way in which these three mechanisms can work together to form a system which provides a high level of integrity and security, yet is much slimmer than all other purely P2P currencies. It also offers other potential benefits such as faster transactions and lower fees, quicker network synchronization, support for high levels of traffic, more block space for custom messages, and potentially even increased anonymity.

Introduction

It was over five years ago now that "Satoshi Nakamoto" first released Bitcoin into the public domain and forever changed the way many people think about finance and economics [1]. Back then the difficulty was low and the blockchain was small. For the first two years things looked great, many believed the blockchain wouldn't become problematic for a long time so the issue was put to the side. We are now in July of 2014 and the blockchain is close to 20 gigabytes in size [2]. While still manageable, it is becoming a serious issue of concern.

The core developers of Bitcoin now direct much of their attention to handling the ever-increasing levels of network traffic. Today the bitcointalk.org forum observes almost daily threads concerning methods for minimizing the size of the blockchain and decreasing the synchronization time. One of the most effective measures taken to date was a switch from Berkeley DB to LevelDB [3]. BDB is much slower and the switch over to LDB resulted in a major performance boost in terms of synchronization and block verification speeds.

Another promising endeavor is the "Ultimate blockchain compression" project [4] which aims to achieve "near-optimal blockchain compression" by implementing blockchain pruning techniques including "balance-tree information" which is "maintained and verified in a separate blockchain through merged mining". Blockchain pruning is indeed a promising venture and could prove to provide a high level of compression, but the results are limited and it's not exactly clear how to merge the changes with Bitcoin.

The bitcointalk forum also observes many threads about changing the max block size limit (with many for and against it). One group of people against the change released a “Bitcoin Blocksize Problem Video” [5] in which they argue against increasing the max block size because “it will become more expensive to run a node”. Gavin Andresen, Lead Core Bitcoin Developer, replies “The block size will be raised. Your video will just make a lot of people worried about nothing”. Every day this issue seems to become more pressing.

The max block size definitely will be raised at some point, right now the transaction capacity is restricted to 7 transactions per second [6] and eventually that just wont be enough. Raising the max block size in Bitcoin will require a coordinated hardfork because all the older clients won't know how to deal with the larger blocks. So there are legitimate reasons to be concerned about increasing the max block size, but that doesn't mean it's not needed. There are several cons and pros which must be considered.

Increasing the max block size would increase transaction bandwidth and lower fees, but it would cause the blockchain to grow even faster and put more stress on nodes. Many members of the bitcointalk forum say it's not the max block size which is a problem, instead they blame gambling services such as Satoshi Dice for “spamming” the network with many transactions. Other members note that Satoshi Dice “is providing stress-testing for the network, and showing that unless the block size limit is lifted, there will be a problem” [5].

Should we be Concerned?

As mentioned a moment ago, there are several ways we can attempt to deal with the size of the blockchain. On the Bitcoin Stack Exchange a user Sean Chapman asks “Are there any studies into the size of the blockchain scaling over time?”. Meni Rosenfeld, writer of the top answer, explains that although he isn't aware of any studies that meet Chapman's request, he can outline 5 reasons why we needn't be concerned about blockchain scalability [7]:

1. Not every user needs to run a full network node.
2. Spent outputs can be pruned from the blockchain.
3. Off-chain transactions can help reduce stress on the network.
4. Transaction fees can offset the block storage costs.
5. Moore's Law is still going strong for the foreseeable future.

Satoshi spoke about the 1st point back in late 2008 when he said as the network grows mining “would be left more and more to specialists with server farms of specialized hardware” [8]. This is perhaps a satisfactory solution but it will result in continued centralization and eventually these specialists will control a large percentage of the network power by making it harder for the smaller players to participate. The 2nd point was already mentioned; pruning offers promise but it's a complex process and the results are limited.

The main problem with proposals such as the ultimate blockchain compression scheme is that a lot of the “dust” generated by services such as Satoshi Dice still clog up the system. The way Bitcoin links transactions together makes it impossible to achieve the level of scalability we really need from a crypto-currency. The last 3 points also have some validity to them as well, but regardless of those points we are still stuck with a blockchain which never stops growing and it still doesn't provide us with a truly light weight scheme.

Odds are that Bitcoin is here to stay for a while, and even if the scheme described in this paper is verified, it wont make Bitcoin suddenly obsolete. It will offer an alternative with better functionality in some areas and poorer functionality in other areas. But at the end of the day we shouldn't play favorites, we should try to build on the work of Satoshi and create an open free market of the best crypto-currencies where competition can thrive. The scheme proposed here can provide us with truly fresh and unique advantages.

We should be looking for innovative new ways to solve these scalability issues in a concise and satisfying manner. The Bitcoin Wiki states “At very high transaction rates each block can be over half a gigabyte in size” [6]. Is it really feasible for Bitcoin to reach that level of network traffic and attempt to store it all in an ever-growing blockchain? For coins such as Bitcoin a high level of centralization may be the only solution in the end. It's clear that as we move into the future a better solution will become necessary.

Finding a Solution

What follows is a proposal for an entirely new alternative crypto-currency which is similar to Bitcoin in many aspects but also very different in other aspects. It is extremely hard to make any large changes to the Bitcoin code base, and the following scheme is not even compatible with Bitcoin. The scheme eliminates the need for a full blockchain by unlinking transactions, thus allowing all transactions to be discarded after enough time has passed, but in doing so it removes script from the protocol, and that's something Bitcoin can't do.

The proposed solution described in this paper comes by understanding the different purposes of the blockchain and then separating that functionality into individual mechanisms which are each optimized to serve their purpose. The blockchain has 3 main functions. The Bitcoin blockchain combines these functions into one single mechanism and as a result doesn't scale well. It requires you to store a lot of data which doesn't really need to be stored forever. Breaking up the functions of the blockchain is the key.

Functions of the Blockchain:

1. to coordinate how the network processes transactions
2. to encapsulate the proof-of-work which secures the network
3. to manage account balances; record the ownership of coins

Since the original version of this proposal was written in early 2013 it has been improved and modified significantly [9]. With help from other bitcointalk forum members, the mini-blockchain scheme has been fleshed out and a project wiki was created to expand upon the ideas of the original white paper. Now we have actually implemented these ideas in a new coin called Cryptonite [10]. The process of implementing it has helped immensely in fine tuning the concepts underpinning the mini-blockchain scheme.

In essence the mini-blockchain scheme works by storing the balance of all non-empty addresses in a structure we call the "account tree", so we don't actually need any of the transactions to calculate the balance of any given address. We have removed the script system and along with it the entire idea of interlocking transactions, and replaced it with a much simpler concept where transactions perform basic operations on the account tree such as "subtract coins from balance of address A and add to balance of address B".

The inputs and outputs in the transactions do not point to other transactions, they simply point to addresses in the account tree, so the transactions aren't linked together the same way they are in Bitcoin, and we can discard all transactions after a safe amount of time has elapsed (enough to make the Secret Chain Attack infeasible, discussed later). With Cryptonite nodes are able to delete all transactions a bit older than a week, but they can choose to store as much history as they want, it's unlikely the full chain will ever be lost.

The Account Tree

This proposal starts with the concept of an “account tree”. Why should we record every single transaction and save it forever if all we need to know is the balance of all non-empty addresses? The 3rd function of the blockchain is replaced with the account tree. The account tree is essentially what could be thought of as a decentralized “balance sheet”. It will contain every unique non-empty address and the balance of all those addresses, along with some other fields which make withdrawal limits possible (more on this later).

When the balance of an address changes all we need to do is update numbers in the account tree instead of adding new data to it. Of course this wont provide a truly finite amount of data to work with because new non-empty addresses will be appearing all the time, but it comes as close to finite as is probably possible. It is finite in some sense because the coins will have limited divisibility, and we can't really expect the world population or the number of Internet users to continue growing forever. In any case it's scalable and manageable.

Even with a population of 10 billion people where each person had 10 different non-empty addresses, we would only need to keep track of 100 billion addresses. Since we can remove empty addresses from the database and since a transaction would simply require peers to shift around numbers in this database instead of adding new data to it, the size of the account tree should always remain considerably small. By the time we reach anything close to 100 billion unique non-empty addresses our computers will be much faster.

Owners of a non-empty address in the account tree prove their ownership with their private key. Like Bitcoin, transactions are created as a signed set of data and broadcast over the network. Like Bitcoin, miners who accept the transaction then put it into their blocks and work on solving a difficult problem to get it into the mini-blockchain (more on this in the next sections). Nodes who accept the block would update their own copy of the account tree by shifting around coins or doing what ever was necessary.

The proposed database is named the “account tree” because it should have a hash tree structure. Each “account” in the tree has a corresponding hash and acts as a leaf node at the bottom of the tree. Being a hash tree, we can combine the hashes of each account to build a pyramid of hashes and calculate the “master hash” at the top. Note that an “account” isn't a collection of addresses like a “Bitcoin account”. In this case each account refers to only one address or leaf node (obviously normal “accounts” will also exist).

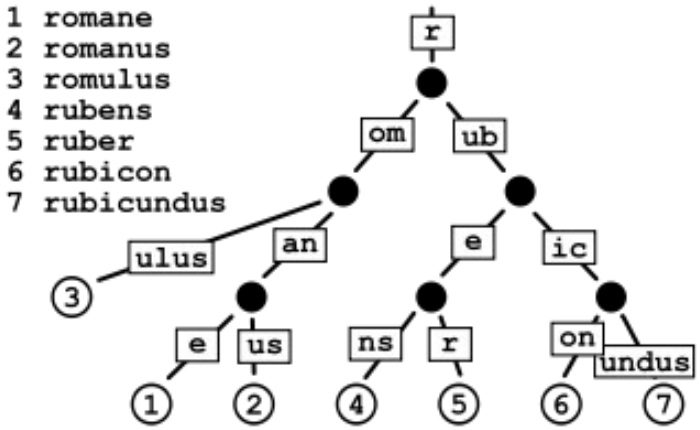

Figure 0-a

Figure 0-a shows a generic radix tree/trie structure (from Wikipedia). In reality Cryptonite uses a type of binary radix trie combined with merkle hashing. What isn't shown in the diagram is how the nodes are hashed. All the hashes together produce the master hash/root hash at the top. The master hash will change even if just one account within the tree is altered in any way. This hash tree system provides integrity to our data because the master hash is also stored in block headers so the tree is secured by the blockchain.

The advantages of using a binary radix trie structure include:

- it is well suited for looking up addresses (public key hashes)
- it is faster and more memory efficient than many other tree structures
- small parts of the tree can be verified without the whole and completeness can be proven
- inserting the same data into the trie in any order will always generate the same structure

The Mini-Blockchain

The mini-blockchain provides our 1st blockchain function. The mini-blockchain is essentially just a normal blockchain, except that we don't need to keep a copy of historic blocks. Again, it isn't truly finite, because changing the max block size could increase the average size of the mini-blockchain. Later in this paper we describe a mechanism for having a dynamically determined max block size but it's not a necessary part of the scheme.

If we are going to keep track of our database with a set of node hashes and a master hash, we can't allow every single separate transaction to alter the database on demand. We must break them up into groups of transactions which are inserted into the database in periodic intervals of time. Without having the transactions solved in groups of transactions as blocks we have no viable method of maintaining the account tree. This creates an inherent need for a blockchain, but since we can discard old blocks let's call it a “mini-blockchain”.

Bitcoin requires the full blockchain because that's the only real way to determine the full balance of all the addresses. However we have the account tree to fulfill the task of managing account balances and recording the ownership of coins. We don't need the full thing, we can throw away old blocks and save an immense amount of disk space. However we do keep a few hundred or a few thousand of the newest blocks and that makes our mini-blockchain. The mini-blockchain also provides us with a level of security.

Each block has the master hash embedded in the header and we can verify each block in the mini-blockchain starting from the beginning, making sure the transactions in each block always correspond to the master hash in the previous block. Since there is a proof-of-work process required for each block before it will be accepted into the mini-blockchain, it becomes extremely difficult for an attacker to generate a fake mini-blockchain. Although difficult, if we totally delete old blocks, it is far from impossible.

With Bitcoin we can start at the very beginning and work our way up to the latest point because we have the full blockchain. If an attacker creates a new mini-blockchain from the oldest block available, new nodes would have trouble telling it apart from the real mini-blockchain, because before that oldest block they have no history of what happened. The attacker can spend as much time as they need building up the cumulative difficulty of their mini-blockchain because it's not an ever-growing chain they have to out-pace.

The attacker could then start broadcasting the fake chain and it might propagate enough to impose a risk of becoming the main chain. The proof chain solves this by providing a mechanism which can act as a container for storing the long term proof-of-work history so that we can calculate the total cumulative difficulty of any chain. Instead of totally deleting old blocks we must maintain the block headers so that we can always trace the history of any given chain and compare the total cumulative difficulty each chain.

The Proof Chain

The proof chain, which provides our 2nd function of the blockchain, is essentially just a chain of block headers. When nodes discard the old blocks, they will not discard the block headers, only the transactions. So basically the mini-blockchain is a blockchain pruned of all but the most recent transactions. This means all nodes can still use the chain of block headers to verify the best mini-blockchain with the highest cumulative difficulty, and thanks to the account tree they don't need old transactions to calculate address balances.

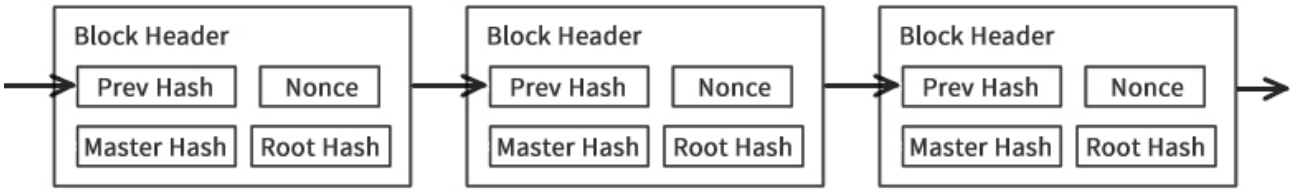

Figure 1-a

We have what is essentially a normal blockchain, so mining can work the same way as it does in Bitcoin. Nodes must hash the block headers and search for resulting hashes below the current target. It's possible to discard old transactions and store only a chain of block headers because proving the solution doesn't rely upon knowing the transactions in the block. It's secure for the same reason Bitcoin is secure, each proof in the proof chain feeds into the next proof, making it nearly impossible to generate a fake proof chain.

The master hash needs to be in the block headers as detailed in figure 1-a because it allows the nodes to verify the transactions in the block and make sure their account tree has been altered correctly by the block. The master hash stored in the block header is calculated after the transactions in the block have been applied to the account tree. We can work our way from the start of the proof chain up to where it feeds into the back-end of the mini-blockchain and we can verify that the most recent blocks we have are valid.

The proof chain proves which mini-blockchain has the most long-term computational backing. No longer does the attacker have forever to sit around generating a fake proof chain because the proof chain must feed into the mini-blockchain. So now if an attacker tries to create a totally invalid mini-blockchain they will also need a strong proof chain to go along with it. This essentially takes us back to the level of security of offered by a typical full blockchain scheme, but it's still not perfectly secure in all situations.

The main problem is that an attacker could build on the legitimate proof chain in secret using an invalid account tree, and then release the secret chain to the network when the they think it's so long that no one will have history going back that far. In that situation new nodes would not be able to detect which chain was real. We call this the "The Secret Chain Attack". If all transactions can be discarded after a week (as in Cryptonite) the attacker must maintain the majority of the hashing power for more than a week (in secret).

We believe even if this attack does happen it wont be catastrophic because:

1. the attack will only affect nodes who haven't synced with the network in more than a week, all other nodes can detect the attack and reject the fake chain.
2. a possible Secret Chain Attack underway can be detected by new nodes, although they cannot know which chain is the real one.
3. the release of "community checkpoints" will point nodes to the correct chain in the off chance this attack did happen, thereby making it a worthless attack.

Summary of Network Behavior

Network Synchronization

Network synchronization is achieved in 4 steps:

1. Acquire proof chain with the highest cumulative difficulty.
2. Acquire mini-blockchain which is associated with proof chain.
3. Build account tree by requesting slices and verifying the hashes.
4. Use recent transactions to complete sync of account tree.

First the node will use a “headers first” approach to locate the chain with the highest cumulative difficulty. Then it will acquire a collection of the most recent blocks connected to those block headers. Then it will attempt to acquire slices of the account tree until it has a complete tree. The account tree structure allows the node to prove it has received complete slices so that the node can be sure it has the all the accounts. Finally, the node can use recent transactions to update all slices of the tree to the latest master hash.

No one is required to store old account tree data but the node must build an account tree which is fully consistent with the master hash they have; they can't mix and match slices associated with different master hashes. That is why the height of each slice must be determined, so that we know what transactions need to be applied in step 4. Since all slices contain the master hash, when a node requests a particular slice it can match the slice against a particular block by comparing the hashes.

Nodes will generally attempt to acquire slices synced near the point where it becomes safe to discard old transactions. Other nodes are able to provide these old slices because because they can undo recent changes to the account tree and generate old slices upon request. New nodes will attempt to build an account tree at such an old point because they need to be able to create a large “inversion database” which holds instructions for undoing changes made to the account tree, which is useful for generating old slices and handling forks.

In simple terms, new nodes will attempt to build a complete account tree around the point where old blocks can be pruned, then it will “fast forward” using transactions from the most recent blocks and at the same time build an inversion database, so that it has inversions going back to the point where inversions are no longer needed. Of course this description of the process is very much simplified and you'd have to read the source code of Cryptonite for a more detailed technical understanding of how synchronization works.

Notice this process does not rely upon much examination of blocks, one trusts the blocks they have because the proof chain backs them. It's extremely easy to verify the proof chain and once that is complete the node only needs to make sure the blocks it gets match the proof chain. As the account tree is being built the only thing which matters is that it ends up having the master hash of the latest block. Once that is complete and the node is synchronized it can begin to update the account tree normally by accepting valid blocks.

Transactions

Bitcoin keeps track of address balances simply by reading through the blockchain to see what has happened, it's a continual ledger instead of a self contained balance sheet. Bitcoin transactions use a system which include “inputs” and “outputs” and the inputs of most new transactions usually reference the outputs of previous transactions. The mini-blockchain scheme uses the basic inputs and outputs concept but the inputs point to accounts in the account tree and the outputs also refer to accounts in the account tree.

The input accounts will fund the coins which are sent to the output accounts. This operation will cause the balance of the input accounts to decrease and the balance of the output accounts to increase. Fees are still used as normal to give priority to transactions and provide incentive to miners. Obviously a transaction should not be accepted as valid if it reduces the balance of any account below 0, or if it requests anything which conflicts with the value of any balance, or it tries anything it doesn't have permission to do.

In order to make sure the same signed transaction isn't processed by the network more than once, the block header must also contain a “lockheight” field. The transaction becomes invalid once the lockheight is outside the range of blocks which nodes are required to keep (lets call this the blocks “in view”), and same txid cannot be included twice in any of the blocks which are in view. This makes it impossible to use the same txid twice. However this solution requires that the txid is not malleable.

There are several things which need to be considered when attempting to solve transaction malleability, but most importantly we can't include the signatures when hashing a transaction because signing the same data with the same key produces different signatures each time. The sender will sign the txid, but signing it many times in no way alters the txid, only the signatures. Thus attempting to alter the contents of the transaction will always change the txid and as a result invalidate the signature(s).

Account Tree

There are a number of ways the account tree can be implemented, but the data structure must fulfill certain requirements:

1. All data should be able to be efficiently summarized by a deterministic hash (master hash / ledger fingerprint).
2. Efficient support for 4 operations: add account, modify account, remove account, lookup account.
3. After every modification one should be able to efficiently update the account tree master hash.
4. Should allow efficient verification of the correctness of a subset of the accounts without downloading the entire structure.

New accounts are inserted into the trie structure as leaf nodes when coins are sent to an address which does not already exist and they are removed from the trie when the address is emptied. When the address does exist the transaction will simply alter an existing account in the trie. The mini-blockchain acts to coordinate when the tree is updated. When a node receives a new block they will carry out the transactions listed within the block by altering their account tree accordingly.

The proposed account tree structure allows all address balances to be summarized in a "balance sheet" format and allows old transactions to be safely discarded by all nodes. However, without a way to coordinate when and how the account tree is changed we still can't ensure consistency between nodes. That is where the mini-blockchain comes in. Each time a new block is accepted into the mini-blockchain, nodes will use that block to update their copy of the account tree in a consistent and coordinated fashion.

In a distributed network it is impossible for every node to apply transactions at the moment it sees them. Transactions must be joined together and applied in bulk. Such a list of transactions grouped together form a block and together with headers form the blockchain. Nodes collect the transactions and apply them to the account tree to achieve a new account tree state. The master hash of the new tree state is included in the block header. Other nodes which receive such blocks can replay transactions themselves and check the hashes match.

Discussing New Network Protocols

Dynamic Max Block Size

As noted in the introduction section of this paper there's a lot of heated debate concerning the maximum block size. One potential new network protocol worth discussing is the idea of a dynamic max block size. It could be made a floating value determined perhaps by several factors. Two methods which immediately come to my mind are 1) a mining-based voting system and 2) a system which analyzes some number of previous blocks and calculates the average block size to derive a new block size limit (eg 2*average).

A voting system sounds feasible and it could allow us to manage the max block size through group consensus, but it gives groups such as mining pools a lot of power over what the max block size will be. A better solution is to simply calculate the average size of some number of recent blocks and then multiply that by some value to derive the new maximum block size, with some arbitrary lower limit. That way we don't need to store all the voting data in the blocks. This is the approach we decided to take with Cryptonite.

Even with our light weight scheme there does need to be a max block size because our network can only handle so much traffic before it virtually stops working. Even with a finite blockchain it can grow extremely quickly with large enough blocks. However in the future we may be capable of handling much larger blocks so we need a way of gradually changing it over time. As mentioned earlier, a hardfork is not very convenient. An automated readjustment system would be self-regulated and much more seamless.

Pruning the Account Tree

The bulkiest part of the mini-blockchain scheme is actually the account tree. Given enough time the account tree may become filled with many low balance accounts and it would be advantageous if we could prune this type of “dust” from the tree once the accounts were old enough. There are several ways this can potentially be achieved but none of them are remotely simple or easy to implement. The best approach seems to be one where you collect “account maintenance” fees for maintaining an account in the account tree.

The fees would be collected when a withdrawal is made from an account, and included along with the transaction fees. The fee would be calculated based on the age of the sending account. In this way low balance accounts eventually reach a balance of 0 and get pruned from the tree. Even if no withdrawals are made from the account we could have a system for allowing the pruning of accounts which would have a non-positive balance had they been paying their account maintenance fees.

The benefit of this type of system is that it places a cost on storing data in the account tree, which is economically beneficial, and it helps keep the account tree compact. The other useful feature of this system is that we can feed the maintenance fees back into the “coinbase account” (pays out block rewards) and ensure that the block reward never reaches 0, without actually inflating the money supply, just by recycling coins back through the mining system, maintaining a finite money supply while not cutting off the minting process.

Withdrawal Limits

Withdrawal limits can be set on individual accounts rather easily in this scheme due to our balance sheet approach. A withdrawal limit dictates the maximum number of coins which can be withdrawn from the account each block, which can be useful for several different reasons. Three extra fields need to be added to the account structure to make this possible: 1) time when account was last modified (this is also useful for pruning the account tree) 2) the current withdrawal limit 3) a potentially queued withdrawal limit.

The limit is self-set by account owners using a special transaction and the default withdrawal limit for new accounts is unlimited. The main purpose of withdrawal limits is to help prevent double spending and make merchants more confident in transactions with a low number of confirmations. If they know only a certain number of coins can be withdrawn from the account each block then they know even a 0-confirmation transaction is likely to go through since the attacker can't withdraw all his coins at once.

A brief outline of how withdrawal limits would work:

1. Send the network a special transaction to modify the withdrawal limit of your account. Limit is specified as number of coins per block, and is saved into queue field. Such change will take effect in eg. 100 blocks and after that time the queued value overwrites the actual withdrawal limit value.
2. Network accepts the special transaction and after 100 blocks it will reject any transaction that would cause newly specified limit to be exceeded.

A merchant can ensure he will receive funds by:

1. Checking that there is no queued withdrawal limit change on sending account.
2. Check that sending account balance is high enough so it can't be emptied too fast.
3. Ensure transaction is not low priority and has propagated enough in network.

Deciding Technical Specifications

Mini-Blockchain

If the proof chain is providing most of our security, at first glance it seems almost unnecessary to store anything beyond 1 or 2 blocks. It is of course necessary to have at least a few hundred blocks because if it's too short the Secret Chain Attack becomes much more feasible. We need at least some reasonable amount of minimum block history to be held by all nodes for several reasons. We felt that 1 week was a good number for Cryptonite but there's a lot of room for experimentation in this particular area to find the best trade offs.

The other important factor we need to consider is the time between blocks. Having it set too short can have problems, such as a higher number of orphaned blocks due to blocks being solved at the same time, but having it set too long makes waiting for just 1 confirmation totally unpractical. Based on a brief examination of alt-coins the optimal block time seems to be in the general vicinity of 1 or 2 minutes. Cryptonite happens to have a block time of 1 minute because it's quite fast but not so fast it causes too many orphaned blocks.

Coin Supply and Distribution

Bitcoin makes use of 2.1 quadrillion units where each coin is made up of 100 million units, resulting in a total of 21 million coins which are divisible by 8 decimal places. However, internally Bitcoin uses 64bit integers, and it could handle a much larger number of coins, so 21 million may sound like a rather arbitrary number, but in fact it is not. Many applications make extensive use of double precisions floats but doubles can only store integers up to 2^53, and that is ultimately why Bitcoin only has 2^50.9 units.

However it is possible to use extended precision floats to exploit the full range of the 64bit integer. This gives the coin supply a much higher level of granularity and is based on a natural upper limit. A coin supply with 64 bits of granularity is made up of approximately 184.4 billions coins, each of which are divisible by 8 decimal places. We have used extended precision floats in Cryptonite so that we're no longer held back by the limits of the double precision float, enabling an extremely large coin supply.

There are many ways coin distribution can be done, with Cryptonite it will take 10 years for half of the coin supply the be mined, with a repeating half life of 10 years, but the block reward is adjusted every block so that it changes gradually over time. Sudden large changes in the block reward are not healthy for the network, and it's not clear why Bitcoin was designed to have such long intervals of time between changes in the block reward. Cryptonite also updates the difficulty every block for similar reasons.

Conclusion

In this paper we have described a variant of the Bitcoin network protocol which is designed to eliminate the need for a full blockchain and significantly reduce the need for long term data storage. This is achieved by separating the functions of the blockchain into individual mechanisms optimized to perform certain tasks. Although not extensively tested, the result offers a purely P2P crypto-currency with many benefits such as increased block space, making room for extra functionality such as custom transaction messages, etc.

The nature of the account tree and mini-blockchain may also offer an increased level of user privacy since old transactions might not be locatable, but it's unlikely we wont have nodes dedicated to storing the full chain. We do achieve a much greater level of scalability at the expense of some security trade offs, but it's nothing that can't be dealt with. The end result does have a high level of security, but at the same time it's super compact and scalable. In many respects it is superior to Bitcoin but not in all areas (eg scripting capabilities).

What will the future of crypto-currency look like? With the advent of scalable crypto-currency technology it's looking brighter. Crypto-currency can change the way the world works, but only if it scales well enough to handle the needs of the world. Ideas alone aren't enough though, we must implement those ideas. Satoshi is very respectable in that sense because he took the initiative to create a vastly complex system built on an array of new and exotic concepts which hadn't ever been tested before, and he changed the world forever.

References

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

[2] Blockchain.info. 2013. Blockchain Size Data.
https://blockchain.info/charts/blocks-size

[3] Andresen, G. 2013. Bitcoin-Qt / bitcoind version 0.8.0 released.
https://bitcointalk.org/index.php?topic=145184

[4] Reiner, A. 2012. Ultimate blockchain compression.
https://bitcointalk.org/index.php?topic=88208

[5] Todd, P. 2013. Bitcoin Blocksize Problem Video.
https://bitcointalk.org/index.php?topic=189792

[6] Bitcoin Wiki. 2013. Scalability.
https://en.bitcoin.it/wiki/Scalability

[7] Rosenfeld, M. 2012. Are there any studies into the size of the blockchain scaling over time?
http://bitcoin.stackexchange.com/questions/2798/

[8] Nakamoto, S. 2008. Re: Bitcoin P2P e-cash paper.
http://www.mail-archive.com/cryptography@metzdowd.com/msg09964.html

[9] Bruce, J. 2013. Cryptocurrency with Finite "Mini-Blockchain"
https://bitcointalk.org/index.php?topic=169311

[10] Mini-blockchain Project, 2014. Cryptonite.
http://cryptonite.info/ and wiki: http://cryptonite.info/wiki/