BitBay

SCALING THE BLOCKCHAIN

ANDOTHER IMPROVEMENTS

Sharing stakes | Atomic double deposit escrow | Anonymous Atomic Trading | Notary and Asset Issuance | Smart contracts in the output | Sidechains | Pruning | Variable Block Size | Voting for network changes | Malleability Fix | Spam protection | Stake Grinding defense | Anonymous Transactions using Bitmessage

David Zimbeck
www.bitbay.market

1.27.2018

Introduction

Blockchains get a lot of their security from strong signatures and hashes, consensus, and decentralization. One of the longest technical debates in history took place in the Bitcoin network about something as trivial as increasing the size of the blocks. This type of technical gridlock prevents popular blockchains from experimenting with improvements at the speed of the internet. This is why the altcoin industry is so important because of the great ideas being proposed and tested. Over the years of coding and watching the industry grow I have developed some concepts that I think should be used to improve blockchains specifically “Proof of Stake” chains which are energy efficient and more resistant to forks in practice. The key to scaling a chain to industrial sizes while maintaining decentralization is to face the reality of the situation. We want people to verify as much as possible, while getting as much volume as possible while having a network large enough that it is not prone to centralization. An individual can not validate every transaction and once the network scales a node can not validate every transaction. Although it can be argued that if the demand was there then the reward a node gets paid from fees will be enough to cover expensive computers and servers, it is unclear if this is the case. Besides, users still need a way to synchronize the chain with standard hardware. So this paper will explain simple improvements to blockchains that can work wonders if implemented. Some of the ideas are easy to implement while others require a good deal of engineering. If a chain were to implement all of these ideas it would be a very decentralized secure and feature rich blockchain that could scale to meet the demand.

Variable block size

The first concern is how large a block can be. I believe that the concerns about large blocks are overblown. However, the block size should be dynamic to grow or shrink based on demand. The concept is simple. The blockchain sets a very generous maximum block size (for example 128 megabytes) and a minimum block size (a few megabytes). Then within this range the block size is increased when the blocks consecutively fill to more than a percentage in capacity. So in the beginning if the block size is set for 4mb and it goes beyond 75% capacity for a thousand blocks, then the network automatically increases it to 8mb. If the network becomes less popular and traffic drops, then if the blocks are only being filled at 10% capacity, the block size is decreased back to 4mb. These numbers are not set in stone however the concept is simple. Part of the reason some people protest larger blocks has to do with centralization so this improvement will be complemented by the other sections in this paper.

Pruning the blockchain

One of the reasons it's been stated that pruning the chain is not secure, is that nodes connecting to the network should be able to work from the genesis block to the current block. It is possible with "Moore's law" that computers will have this capacity and bandwidth. Currently, this is not the case because chains such as Ethereum are growing at a gigabyte per day and this makes it impossible for consumers to run their chain which compromises their security. Therefore, it's very pragmatic to prune the blockchain. Pruning has always been voluntary and there has never really been a way for a node to connect to the network and get a pruned version of the blockchain from a certain point. Therefore the technique is fairly simple. From each pruning point, nodes start to build a database of new unspent transactions(assuming the previous database included all unspent transactions). Some spent transactions are kept because the requirement should be that a spent transaction must mature a hundred thousand blocks(for example). This database may also prune certain voting addresses and unspendable addresses that also have a lot of confirmations. The entire unspent database is hashed and then staking nodes are required to sign this database. They will then form a consensus eventually of what they believe the database to be. After perhaps anywhere from a few thousand to a hundred thousand confirmations (assuming you are using one minute blocks) then the old information is either deleted or archived and the new blockchain simply starts from the new position. Users can still choose to archive all the old spent data however it is no longer needed to run a full node and perhaps only useful for research and notary. For further protection each set of deleted information should be hashed and stored with the new genesis block header so an archival node can prove the validity of notary, transactions and other information.

Sharing stake rewards

Now that the block size is solved we need a way to handle bandwidth and processing time for checking transactions. This is one of the biggest bottlenecks in the industry. It is clear that we don't want centralization however we know that every node has different processing capacity. The solution to this is creating a system that allows nodes to share block rewards and fees as the system scales. First, a node who thinks they are eligible to stake performs a proof of work to determine their processing and storage capacity and this can be burned to the chain or shared privately. Then nodes voluntarily group with other nodes publishing this group in a transaction. There should be a limit for the number of confirmations before a node can change their groups and a limit to the number of groups a node can join based on their staking weight. When an input wins a stake, the hash determines what transactions each node confirms based on it's staking weight and self-published processing capability. This means the balances of the staking addresses should be known in advance. Therefore a node with a small stake in the network should not have the ability to process all of the networks transactions. Each node that participated in sharing should sign the hash of the transactions they validated and connect their address to that hash. Therefore, if they violate network rules then the next group to stake in the chain is the owner of the funds in that address which is paid as a penalty as fees. This penalty would then be the stakers total network weight. Increasing the minimum deposit size is okay as long as it keeps the entire network distributed and should probably not exceed .002% of the total supply. There may also be a maximum penalty not exceeding a percentage of the total supply. Fees are distributed based on network weight. Stakers should confirm previous blocks based on their staking weight as well. If a node has a high staking weight but low processing capacity they would not be asked to confirm so many transactions. New nodes connecting to the network would only verify transactions based on their computer hardware unless they specify otherwise. Once they get a balance they might be informed about their staking options. It might be possible to attempt to prevent double spend and increase the probability a transaction gets to a block by asking a prospective staking node to voluntarily sign the transaction. Therefore if a transaction is not published then they might pay penalties. The node might have their own requirements for transactions it can guarantee.

Using POW from other chains to prevent stake grinding

Proof of stake has been shown to be more resilient to forks. The reason for this might seem counter-intuitive because it would seem like it's easy to "buy a network". However in practice, users of POS chains have a vested interest to protect the network. POW chains on the other hand constantly get hard forked by larger mining networks when their difficulty drops. This is because the security of POW chains is not self-contained making it too easy for competitors to attack other chains. However, it's clear that POW chains do contain useful block information. It could be good to make a light version of Bitcoin to only download block hashes and use those hashes to increase the randomness of block selection to prevent stake grinding. Currently the "Proof of Stake 3" system has been reliable for years. However there is a theoretical attack where a user tries to get a slight advantage in the number of stakes he wins. If this feature is added then it should also be possible for it to be activated and deactivated by voting from stakers just in case the blockchain in question stops functioning.

Defense against spam attacks

Nodes that stake can be easily taken offline by spam attacks from other nodes. This attack is surprisingly effective and cheap to perform. To solve this, there should be some basic defenses in most wallets which allow users to filter out payments of less than a certain amount. They might choose to use different wallets for staking and spending. There should also be spam detection tools (for example many small and redundant payments) with potential penalties. There should also be tools for spending spam transactions which might be unintentional for example when a user is donating their stake profits.

Voting for network change

Technical debates can prevent a large network from scaling. Therefore, it is important to allow nodes to vote on potential forks. Although in theory they do this by choosing which transactions to accept and reject, there is issues with “replay” attacks and traffic problems if the decision to change protocol is taken casually. Therefore if more than 50% of the nodes decide they want to fork for a month or more of voting, then the chain can be stopped until the new nodes consecutively publish their confirmation that they are running the new version to the chain for a certain number of blocks. They might also include a transaction that is different from the previous protocol to indicate their support of the soft fork. Also nodes might be able to vote on other protocol changes such as whether or not to include POW info from Bitcoin, the size of deposits for staking, the size of staking groups, the amount of confirmations before a user can change their group and so forth. Stakers that do not vote on these proposals would be automatically rejecting them. Once a proposal is taken, the nodes no longer vote on the issue unless manually asked to do so. It should be mentioned here that I do not believe in Segwit or the idea that transactions should be stripped of data to be backward compatible. In fact, lightning network and patches to malleability and other changes are all completely possible without this change. Bitcoin was a very well developed system and there has been some changes which have almost compromised it’s security in recent years. However, improvements such as the lightning network are interesting scaling concepts but these concepts are outside the scope of this paper.

Fixing malleability

Malleability can be useful in some edge cases but in other cases it might not be desired by participants in a transaction. Therefore users should be able to publish a transaction with a special tag. This tag will tell the network that when this transaction output is spent, the user requests that all participants to the transaction sign it's transaction ID. As some transactions might require more than one signature, each public key included in the transaction will be asked to sign. When spending one of these transactions, it will be required that you sign the signed transaction ID instead of the transaction ID for the entire transaction. This means there will be more than one way to reference a transaction.

Anonymous Broadcasting using Bitmessage

Networks such as BitBay use Bitmessage for communicating data for their decentralized markets. Although Bitmessage is designed as an "everyone downloads everything" type of system, there are ways to scale the network because most of the data can be discarded as trivial. Because it is encrypted end to end and it is hard to locate the origin of a message due to the redundancy of passing the same message around the network, this system is ideal for protecting the privacy of nodes. Although staking nodes will not benefit from the luxury, everyday users can submit transactions through this system giving them a great deal of anonymity only second to Zero Knowledge Proof (which is the best system for anonymity). After all, the major issue with anonymity is the ISP sees the data unencrypted and the IP address is published. Having random users broadcast transactions is a cheap and effective solution to the problem. The process is fairly simple. The user chooses a Bitmessage channel they want to submit to. Then when the message arrives at the channel the participants decrypt it and check the mempool to see if it has been published. If it has not been published they do so.

Anonymous Atomic Cross Chain Trading

Atomic trading is used mostly for decentralized exchange between different blockchains. The technique involves a shared hash that when spent on one chain, activates a transaction on the other chain. This technique requires transactions to be published asynchronously as the users involved take turns and therefore relies on time locks of at least two different lengths. One of the drawbacks of atomic trading is that there is a shared hash between the two networks that are connected. Atomic trading therefore has the weakness of a shared hash which can identify the user on both chains. One reason the hash can not be hidden is because you can not reference a “p2sh” within a “p2sh”. There is a concern about exhausting the resources of miners but it would be harmless to allow a recursion depth of at least one level. The p2sh referenced can be restricted to basic scripting techniques such as public keys, locktimes and hashes so there is no risk involved. Then when a transaction is published on both chains there will be two seemingly unrelated scripts on the second chain which actually contain two hashes instead. This allows for a script which shares the hash on both chains and a script which does not. The party which is first to act, shares both redeem scripts well in advance so it is known how they are spent at all times. Then they can share the hash they did not use with the counter-party after successfully redeeming their funds for the trade. If they refuses to send that information then the hash that is published on their blockchain can be used while spending instead. Therefore, in the worst case scenario the user doesn’t privately reveal their second secret simply breaking anonymity on the trade without compromising funds. As an added bonus, this system will allow a user to reference a p2sh address in a script. This is very useful because Bitcoin scripting allows for users to reference a “pay to public key hash” in a script but will not allow referencing a “pay to script hash” instead forcing users to know the entire script in advance. This is inconvenient for multisignature wallets such as BitBay since it requires users to send their redeem script in advance for receiving special time locked payments or it requires users to research the chain to see if the target address has spent anything previously. A limit can be placed on the number of scripts published while spending.

Example Transaction

Alice shares two secret scripts with Bob and then does the following...

Alice pays this address-> After 2 days funds funds revert to Alice, before 2 days Bob can spend with script1 (his signature and Alice's secret hash) or with script2 (his signature and Alice's second secret hash)

Bob pays this address-> After 1 day funds revert to Bob, before 1 day Alice can spend with her signature and Alices's secret hash

Alice then spends Bobs funds revealing the solution to script1. She then sends Bob the hash to script2 privately even potentially encrypted into her payment. If she doesn't do this, Bob can still spend using the secret hash she published. If she does send it the transaction is relatively anonymous because the hash is now unrelated and he doesn't need to reveal what script1 was.

Atomic Double Deposit Escrow for Mixed Currencies

It is my belief that one of the most important innovations in blockchain is double deposit escrow. This system first debuted in BitHalo/BlackHalo in 2014 and was later popularized in BitBay. Using deposits from two parties paid to a joint account, it is possible to create a two-party escrow that eliminates the incentive for deception and theft by eliminating biased middlemen and third parties. Third parties can not discover who is honest in a dispute and therefore this system takes away the incentive for a dispute by having both parties lose their deposit if they cannot come to an agreement. Deposits can be flexible and based on a reputation system. This section will expand on this system further and propose a very amazing mixed deposit system. Because of the varying popularity of many blockchains, a user might only wish to do business in their favorite currency. Users might also want to create contracts which move funds from one chain to another or make promises using a different base currency. The advantages to doing this are too much to list. Traditionally double deposit requires both parties to hold the same base currency because the deposit and withdraw must happen at the same time. However, by using clever transactions, both users can mix currencies. Here is how it works.

Bob funds this address (TX1):

Before timelock (for example 1 day)
Alice sig
Bob sig
Alice Hash
After timelock
Bob sig

Alice funds this address (TX2):

Before timelock (for example 2 days)
Alice sig
Bob sig
Alice Hash
After timelock
Alice sig

Bob first ensures that Alice signs both transactions to the 2 party escrows.
Bob then signs both transactions to the two party escrows.
Then he waits for Alice to spend his funds and publish the hash of TX1.
If she doesn't he gets his funds back.
If she funds his escrow she publishes her hash and this allows him to fund her escrow with the longer timelock.

Atomic Double Deposit Escrow for Mixed Currencies

In this beginning phase the user who makes the larger deposit (for example Bob) should also require Alice to sign a “bomb” transaction for both escrows before he signs the funding transaction. A malleability patch is nice to have here but even without it, a user can delete their escrow keys if no such transaction is signed to take away the incentive of breaking the agreement. The bomb transaction is the disincentive for breaking any type of agreement. Also, it might be possible to update checklocktimeverify to make it so there is a time which if overstepped makes the entire script unspendable. Currently, checklocktimeverify is non-exclusive.

The Escrow Phase:
Now that deposits were linked withdraws must also be linked. The withdraw process can be relatively similar to the funding process.

Alice Escrow:	***Bob Escrow:***
Alice Sig	*Alice Sig*
Bob Sig	*Bob Sig*
Alice Hash	*Alice Hash*

The first one to sign must be Alice with a withdraw proposal for how funds are distributed for each currency. Then if Bob agrees, Alices hash unlocks both transactions when published.

This concludes the amazingly useful “Atomic Double Deposit” system. Therefore a Bob might deposit Bitcoin against Alices BitBay and pay in either currency upon the contracts conclusion. Users can mix any two currencies they want. This greatly reduces the barrier to entry in general contracting as in this case, a person would only need to hold some sort of asset as collateral. This also might create clever ways to back altcoin currencies using Bitcoin or other strong currencies. It might be useful for funding ICO projects while attempting to get a guarantee on their work and not releasing funds until certain milestones are met. It’s useful for barter, employment, cash and wires, trading, shipping and merchant contracts. The possibilities are almost endless. Our society relies on trust and unbreakable contracts such as these are a gateway to our freedom by creating a system that favors honest parties.

Further areas of research

If the blockchain in question is successful, there are some other recommendations in this paper for the future of blockchain technology. Smart contracting systems like Ethereum allow too many things at once. This creates a scaling and bloat problem far beyond what Bitcoin experienced. To solve this while at the same time offer a competitive product it is recommended to scale with demand. Therefore, it should be possible to let nodes execute code in any language as long as the code being executed matches a hash. This way, special contracts can be audited in advance for security and then various sandboxes can be made which can be used in sidechains or on the main chain. One of the strengths and weaknesses of Ethereum was allowing the building of "blockchains on top of the blockchain". This is an issue as it prevents Ethereum from scaling and also harms the economy due to fraudulent proposals and naive investing. On the other hand, it makes issuing a token easy and therefore increases the demand for the main chain. It is my belief that you should limit the way in which chains are built on top of chains by either requiring them to be part of a sidechain or to tie up some funds on the main chain to increase it's value. Additionally, notarized objects such as dividend paying stocks and real estate can drive trillions of dollars of demand and real value to blockchains. There has been a somewhat baseless criticism of blockchain technology that it's value is purely speculative. However this is not true because blockchains can replace the stock market. They can also be a secure way of transferring houses and notarized objects. If for example, a user can guarantee a percentage of his home equity as backing for a blockchain and let users work with this liquidity for a fee, it empowers them in tremendous ways by giving them direct and easy access to a way to profit without any middle men. This in theory is a trillion dollar industry. It also lets demand be driven to even the smaller blockchains as long as a company is willing to issue stocks on it or as long as real estate is allowed to be notarized on the chain in question. Lastly, there is sidechains. Sidechains such as those used in Lisk, are a great way of taking pressure and liabilities off of the main chain. Even if the sidechain is more centralized, the users can understand these risks and proceed accordingly. Some of my ideas will be proposed in the next section.

Smart Contracts in a hashed output

Most popular blockchains are either account driven or output driven. Account driven systems update accounts with a nonce every time a transaction is spent from them. Output driven systems work similar to cash and have their own scripting system for contracts. There are advantages and disadvantages to both systems. One proposal I've had for years is to simply engineer a contracting system where changes and updates happen gradually as nodes accept them. Contracts should probably not be introduced through voting (even if many trusted Githubs are queried) but instead introduced in bulk during a scheduled fork. The systems design is to allow for smart contracts based on the pre-approved hash of some computer code in any language potentially Javascript or C++ for example. These hashed files are saved separately and then run with their own fee policy implemented inside of them. Their code should be modest for example, a simple dividend paying contract or a type of promise based on a notarized object. The user who wishes to use one of these contracts would have one of the outputs tied to an input or a series of inputs and then reference the hash of the contract. The next output would then be the data payload to the contract if applicable. This system is simple, not restrictive and doesn't burden the main chain with engineering issues. It may be possible to vote contracts out of existence if a flaw is found. I'm of the opinion that the most important contracts will be business style contracts such as stocks and dividends, real estate, notary, creative accounting, contracts for backing/stability, and even games. A contract might be allowed to run code in a shared thread in some cases to check on certain information in the background while reducing workload to the nodes that verify it.

Notary Contracts and asset issuance

The issue of blockchains on top of the blockchain is one to be addressed. In this case, I propose that real estate notary and assets (such as stocks) be encouraged on the main chain and contracts should be created to trade it's value in a meaningful way. It should be pointed out that this major industry might require country-wide recognition in their respective notaries and regulatory industries to become truly valid. It is possible to code any type of notary transfer system on top of Bitcoin without changing a single thing about Bitcoin. This is because something in the real world that is notarized on a blockchain needs to be recognized as valid in the real world. Nonetheless, there should be contracts to support such an infrastructure since it will be accepted in one way or another. These types of contracts can include issuance, backing, trading, bonds, collateral, options and dividends. The first step required to acknowledge digital signatures and hashes in the real world would require a person registering their identity with their local notary such as public key and/or documentation identity hash (people wish to be anonymous but this would be the exception when they want solid title). Then they would also be able to register the documentation or signatures of a corporation, real estate, commodities and so forth. Lastly, there are countries that require strict regulation for stocks and some with no regulation at all. The users of the chain will probably issue assets at their own risk as their options expand with time. When issuing a new asset such as a stock or a coin associated with a contract, there should be one of two requirements. The first option is the user must connect X number of coins to each asset (so the asset traded is both the parent and child currency). So as an example, a company may claim 1 BitBay is 100 shares of their company and if a user holds a minimum number of shares, they receive dividends payable to the same public key in BitBay or potentially even Bitcoin (the public key can be the same across multiple networks). All subsequent transactions would ensure this tag is carried to the next recipient. It is not a good idea to allow multiple tags in the first implementation of this system. This prevents bloating the chain but allows easy asset issuance and it restricts supply of the parent currency and drives extra value to it. However in blockchains such as BitBay with a variable supply, this might not always be an option as in some cases an asset might not want to be subdivided or frozen and unfrozen dynamically. Therefore, a company might choose to issue an asset as a contract and this contract might issue it's own coins or assets. To create this asset it would be required to post a deposit to the blockchain which can only be released if the asset is dissolved. Stakers can vote to change the minimum deposit required dynamically to make room for new companies. I believe this simple system to be the best of both worlds as the system can scale with demand while protecting the parent blockchain.

Sidechains

For more advanced token issuance and platforms that require a lot of coding the only way forward is sidechains. Sidechains protect the main chain by preventing all the traffic from being on the main chain. They may be pseudo-centralized or decentralized. However the point is, failures on a side chain will never effect the security of the parent chain. The basic concept is to allow users to make a payment to a side chain thus removing coins from the main chain after a certain number of confirmations. Then the sidechain can inject the coins back into the main chain once a certain number of confirmations happen on the sidechain. The parent chain only needs to monitor this event. Stakers should be able to vote a sidechain into or out of existence and limit the number of possible chains issued and even require a deposit to issue a sidechain. This type of system is useful for many applications such as social media, businesses, zero knowledge proof, websites, games and other implementations. A good example of a working system for sidechains is Lisk. There are also various whitepapers discussing the best security practices.

Conclusion

This paper summarizes a variety of ideas ranging from scaling to smart contracts. Some things require more work to code than others so updates would come as multiple forks as each improvement is tested and proven. Current systems such as Ethereum, EOS and even Bitcoin have centralization, usability and scaling concerns. Improvements to blockchains should be gradual and practical. Contracts should be introduced at a steady pace with modest goals for the demand that businesses are requiring. Notary, proof of identity, securities and real estate should be prioritized due to the tremendous value it drives to the entire industry. Sidechains should be used to avoid putting pressure on the main chain and dividend paying securities and tokens should transfer with the parent coin in order to drive more demand and value being mutually beneficial to both the parent and child. The methods for scaling should be based on pruning, archiving, sharing the workload of confirming blocks, adding disincentives for double spend and bad blocks, benchmarking, syncing with a percentage of the network of your choice as the system scales, scaling dynamically, scaling blocks with demand, and patching current security holes such as spam, malleability, stake grinding and randomness. Voting can be used to scale blockchains, add changes as they are audited by enough coders, and protect the network from subversion and security threats by being able to remove contracts or side chains if needed. Decentralized exchange should be the norm and not the exception with atomic trading and especially anonymous atomic trading used for free trade without risk of losing funds. Double deposit escrow can be used to enforce real world agreements between multiple currencies at once regardless of the type of agreement. All of these things are important for the future of cryptocurrency and it is my hope that the ideas here inspire other developers to carry the torch from this moment forward.


BitBay