Proof of Pact – 'The Twin Blockchain'

Aaron Mathis

The Proof-of-Pact is a new design of blockchain technology in that two chains work together to perform safe and secure transactions. We've taken a lot of great things from proof-of-work structure enabling our blockchain protocol to inherit a lot of its security benefits, but none of its issues. We then took the principal of proof-of-stake and wanted to put a programming language behind our chain. With this we feel we have created the first blockchain that is totally scalable, immune to forks, and secure to the highest of standards.

Soferox LLC

www.soferox.com

Version 1.0

September 20th 2017

Table of Contents

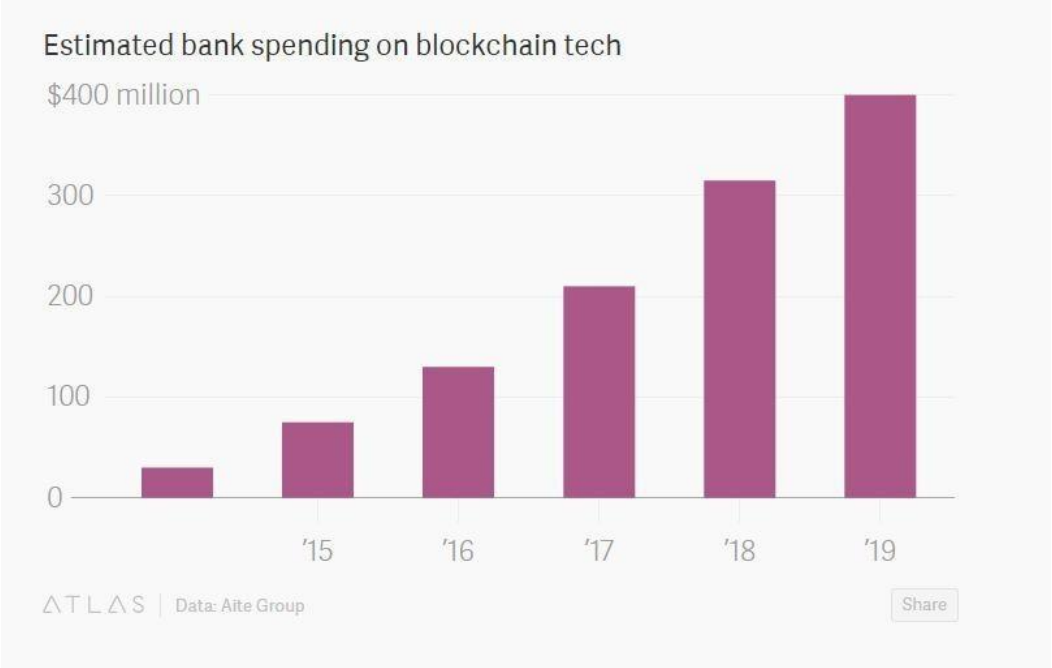
Introduction

With blockchain technology on the rise and being used more and more an inevitable discovery of some weaknesses were bound to happen. Two of the largest issues facing blockchains today are their scaling issues and their ability to maintain consensus. This is something that really rang true to us and something we felt passionate about to truly research and dive into. Another issue we felt was a problem was the high entry cost into proof-of-work. The electricity and hardware cost to actually mine it have increased over the years and making it costly to enter.

In answer to these criticisms, a variety of alternative consensus mechanisms have been proposed and developed, including proof-of-stake where users hold balances in native coins to mine, *Practical Byzantine Fault Tolerance and the Ripple Protocol Consensus Algorithm* which adapt the ideas behind classical consensus algorithms like RAFT and Paxos to function on large-scale and trustless systems, and federated nodes or trusted nodes which act as network authorities and resolve consensus conflicts.

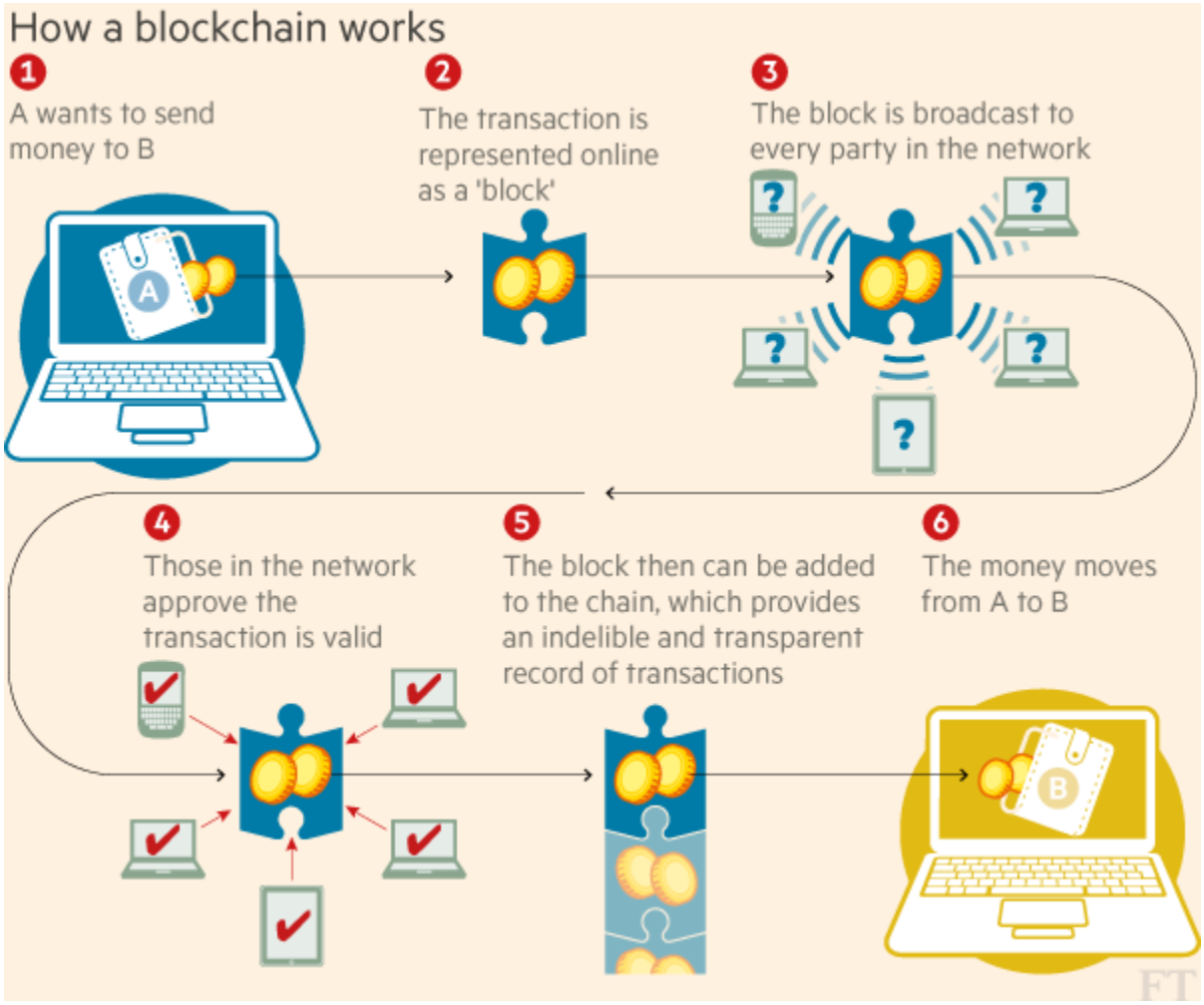
Each of these consensus algorithms trade off some of the advantages of a proof-of-work consensus mechanism: thermodynamically-sound security expectations, trustless and permission-less involvement of miners, mathematically-verifiable replaying of network history for new nodes, significant opportunity costs to attack, etc.

Our Proof-of-Pact protocol aims to take the advantages of these two ideas and address the concerns by the introduction of a second chain, or side chain, that sits on top of the transaction chain to help with some of the current issues facing today's chains. We are excited to bring this technology and we see the future of blockchain tech growing and more money being spent.



Current Technology

To better understand what we aim to do, lets first discuss Proof-of-Work, mining, and how it is as of today. Going deeper, proof of work is a requirement to define an expensive computer calculation, also called mining, that needs to be performed to create a new group of trustless transactions (the so-called block) on a distributed ledger called blockchain.



Mining serves as two purposes:

1. To verify the legitimacy of a transaction, or avoiding the so-called double-spending.
2. To create new digital currencies by rewarding miners for performing the previous task.

When you want to set a transaction this is what happens behind the scenes:

- Transactions are bundled together into what we call a block.
- Miners verify that transactions within each block are legitimate.

- To do so, miners should solve a mathematical puzzle known as proof-of-work problem.
- A reward is given to the first miner who solves each blocks problem.
- Verified transactions are stored in the public blockchain.

This “mathematical puzzle” has a key feature: asymmetry. The work, in fact, must be moderately hard on the requester side but easy to check for the network. This idea is also known as a CPU cost function, client puzzle, computational puzzle or CPU pricing function.

All the network miners compete to be the first to find a solution for the mathematical problem that concerns the candidate block, a problem that cannot be solved in other ways than through brute force so that essentially requires a huge number of attempts.

When a miner finally finds the right solution, he/she announces it to the whole network at the same time, receiving a cryptocurrency prize (the reward) provided by the protocol.

From a technical point of view, mining process is an operation of inverse hashing: it determines a number (nonce), so the cryptographic hash algorithm of block data results in less than a given threshold.

This threshold, called difficulty, is what determines the competitive nature of mining: more computing power is added to the network, the higher this parameter increases, increasing also the average number of calculations needed to create a new block. This method also increases the cost of the block creation, pushing miners to improve the efficiency of their mining systems to maintain a positive economic balance. This parameter update should occur approximately every 14 days, and a new block is generated every 10 minutes.

Proof of work is not only used by the bitcoin blockchain but also by Ethereum and many other blockchains.

Some functions of the proof of work system are different because created specifically for each blockchain, but now I don’t want to confuse your ideas with too technical data.

To reference a specific block, its header is hashed twice with the SHA-256 function; the resulting integer value belongs to the interval [0, 2^256 − 1]. To account for different possible implementations, we will use a generic hashing function hash (·) with a variable number of arguments and range [0, M]. For example, arguments of the function can be treated as binary strings and merged together to form a single argument that can be passed to the SHA-256 hashing function.

The block reference is used in the proof of work protocol; in order for a block to be considered valid, its reference must not exceed a certain threshold:

$$hash(B) \leqslant M/D,$$

where D ∈ [1, M] is the target difficulty. There is no known way to find B satisfying (1) other than iterating through all possible variables in the block header repeatedly. The higher the value of D, the more iterations are needed to find a valid block; the expected number of operations is exactly D. The time period T(r) for a miner with hardware capable of performing r operations per second to find a valid block is distributed exponentially with the rate r/D (see Appendix A):

$$P\{T(r) \leqslant t\} = 1 - exp(-rt/D).$$

Consider n Bitcoin miners with hash rates r1, r2, …, rn. The period of time to find a block T is equal to the minimum value of random variables T(ri) assuming that the miner publishes a found block and it reaches other miners immediately2 . According to the properties of the exponential distribution, T is also distributed exponentially:

$$P\{T\ def = min(T1, \ldots, Tn) \leqslant t\} = 1 - exp(-t\ D \sum n\ i=1\ ri\ ) ;$$

$$P\{T = Ti\} = \sum ri\ n\ j=1\ rj\ .$$

The last equation shows that the mining is fair: a miner with a share of mining power p has the same probability p to solve a block before other miners. It can be shown that proof of work as used in Bitcoin satisfies Conditions 1–3.

The important thing you need to understand is that now Ethereum developers want to turn the tables, using a new consensus system called proof of stake. Proof of stake is a different way to validate transactions based and achieve the distributed consensus.

It is still an algorithm, and the purpose is the same of the proof of work, but the process to reach the goal is quite different.

Proof of stake first idea was suggested on the bitcointalk forum back in 2011, but the first digital currency to use this method was Peercoin in 2012, together with ShadowCash, Nxt, BlackCoin, NuShares/NuBits, Qora and Nav Coin.

Unlike the Proof-of-Work, where the algorithm rewards miners who solve mathematical problems with the goal of validating transactions and creating new blocks, with the proof of stake, the creator of a new block is chosen in a deterministic way, depending on its wealth, also defined as stake. No block reward

Also, all the digital currencies are previously created in the beginning, and their number never changes. This means that in the PoS system there is no block reward, so, the miners take the transaction fees. This is why, in fact, in this PoS system miners are called forgers, instead.

In proof of stake algorithms, inequality (1) is modified to depend on the user's ownership of the particular PoS protocol cryptocurrency and not on block properties. Consider a user with address A and balance bal(A). A commonly used proof of stake algorithm uses a condition as

$$hash(hash(Bprev), A, t) \leqslant bal(A)M/D, (2)$$

where

• *Bprev denotes the block the user is building on,*

• *t is the current UTC timestamp.*

For various reasons, some cryptocurrencies use modified versions of (2) which we discuss in the corresponding sections.

The Goal

Proof-of-Pact aims to take qualities from both of these creating a friendly environment for miners to verify transactions against our chains, but to allow them to come in at a much more energy efficient approach and not require expensive hardware all while benefiting from the security advantages Proof-of-Work has inherently. We will add a few things to our transactional data, but this is only to account for the rule chain.

From Proof-of-Stake we want to more take the idea of a programming language behind our blockchain. We feel this is what chains need to really enter the world and become a force to be reckoned with.

PoP Structure

PoP will be structured similar to how current blocks are with some new editions.

The Rule Chain:

- Block Number
- Rule Index
- Validated Boolean
- Nonce/Keyed Number
- Timestamp
- Hash
- Rule Data

The Transaction Chain:

- Block Number
- Rule Index Number
- Validated Boolean
- Nonce/Keyed Number
- Timestamp
- Hash
- Previous Hash
- Transaction Data

Definitions:

Block Number – The indentifying index number of the current block being submitted.

Rule Index Number – The indentifying number of the rule in the rule chain, or the number on a transaction that relates to the set of rules the transaction should follow.

Validated Boolean – Verifies a transaction has been verified.

Nonce/Keyed Number – This is part of a system to determine first characters of every hash.

Timestamp – This is used to record time of the submitted transaction.

Hash – This is the unique identifier, calculated in SHA-256 algorithm.

Previous Hash – The previous block's hash.

Transaction/Rule Data – This is an array of data. Most likely an array of transactions, but could be something else like rules, or other data arrays.

Merkle Trees:

Merkle trees are binary trees of hashes. Merkle trees in bitcoin use a **double** SHA-256, the SHA-256 hash of the SHA-256 hash of something.

If, when forming a row in the tree (other than the root of the tree), it would have an odd number of elements, the final double-hash is duplicated to ensure that the row has an even number of hashes.

First form the bottom row of the tree with the ordered double-SHA-256 hashes of the byte streams of the transactions in the block.

Then the row above it consists of half that number of hashes. Each entry is the double-SHA-256 of the 64-byte concatenation of the corresponding two hashes below it in the tree.

This procedure repeats recursively until we reach a row consisting of just a single double-hash. This is the **Merkle root** of the tree.

For example, imagine a block with three transactions *a*, *b* and *c*. The Merkle tree is:

```
d1 = dhash(a)
d2 = dhash(b)
d3 = dhash(c)
d4 = dhash(c)            # a, b, c are 3. that's an odd number, so we take
the c twice

d5 = dhash(d1 concat d2)
d6 = dhash(d3 concat d4)

d7 = dhash(d5 concat d6)
```

where

```
dhash(a) = sha256(sha256(a))
```

d7 is the Merkle root of the 3 transactions in this block.

Note: Hashes in Merkle Tree displayed in the Block Explorer are of little-endian notation. For some implementations and calculations, the bits need to be reversed before they are hashed, and again after the hashing operation.

PoP Protocol

Creating a block:

Blocks are created at the time transactions are submitted. When a block is submitted depends on the application/user submitting the transactions. Blocks must contain all the required fields from above. If a block contains transactions that use different rules the block is immediately rejected. Each block can only adhere to one rule set at a time.

If transaction A is defined to follow rule 1, but transaction B is defined to follow rule 2 then they must be submitted in their own blocks.

Rules are not set in stone and can be adjusted, or new rules can be added to grow with the demand of transactions. The transaction chain however is immutable and is what allows us to avoid the need for hard forks and so forth.

A block hash is created utilizing SHA-256 and the data inside the transactional data is encrypted using a Soferox algorithm we call SFX-256. This allows privatized data submission keeping transactions safe and secure.

To simplify a code example could look like this:

First create your block class to create your block.

```
class
Block
{
        constructor(blockNum, rule, previousHash, timestamp, data, hash, validated, nonce) {
            this.index = blockNum;
            this.rule = rule
            this.previousHash = previousHash.toString();
            this.timestamp = timestamp;
            this.data = data;
            this.hash = hash.toString();
            this.validated = validated;
            this.nonce = nonce;
        }
    }
```

You'll need a method to get your hash and there are plenty of libraries out there to calculated SHA-256 Hash.

For the first entry into PoP a genesis block will be manually added. From there you can begin generating new blocks similar to a way like this:

```
var
generateNextBlock
= (blockData) =>
{
                    var previousBlock = getLatestBlock();
                    var nextIndex = previousBlock.index + 1;
                    var nextTimestamp = new Date().getTime() / 9999;
                    var nextHash = calculateHash(nextIndex, previousBlock.hash,
                nextTimestamp, blockData);
```

```
        return new Block(nextIndex, previousBlock.hash, nextTimestamp, blockData,
nextHash);
};
```

Block Submission:

When a block is submitted it is first sent to miners mining the rule chain. This block is verified to meet all rules and if it does it is submitted to the transaction chain for storage. If a block does not meet the rules it is rejected as a forged or incorrectly formatted block. Once a block is deemed verified by miners it cannot be resubmitted. This is by designed to prevent forged blocks.

Verifying blocks just by hashes could be simply stated as this:

```
var isNewBlockValid = (newBlock, previousBlock) => {
    if (previousBlock.index + 1 !== newBlock.index) {
        console.log('invalid block index);
        return false;
    } else if (previousBlock.hash !== newBlock.previousHash) {
        console.log('invalid previous hash!);
        return false;
    } else if (calculateHashForBlock(newBlock) !== newBlock.hash) {
        console.log('invalid hash: ' + calculateHashForBlock(newBlock) + ' ' +
newBlock.hash);
        return false;
    }
    return true;
};
```

We would also want to add in a way to check the rules too against the data. This is a bit more complex as requires first to verify if a duplicate block is being submitted and if it is, it would check validation and if the block was submitted twice it would be rejected.

Communicating with other nodes

An essential part of a node is to share and sync the blockchain with other nodes. The following rules are used to keep the network in sync.

- When a node generates a new block, it broadcasts it to the network
- When a node connects to a new peer it queries for the latest block

- When a node encounters a block that has an index larger than the current known block, it either adds the block to its current chain or queries for the full blockchain.

The Mining Process:

The miners will act in two parts, first mining against the rule chain and running calculations to verify rules of a block are met and validated. Once this happens the block is passed out into the network with a validated badge and pushed to the transaction chain where it is mined and inserted into the blockchain to be dispersed to other nodes. Based on the difficulty of block and if the rules are extremely tight a higher reward is given for a tougher block. Simple transactions receive a smaller block reward as they are generally mined faster.

Mining is done all through the CPU as we want the entry point into our mining community to be small. Plans for GPU mining is something on the table and being discussed. A nice feature with our mining software is that it can be mined alongside GPU miners allowing you to leverage your rig for dual mining.

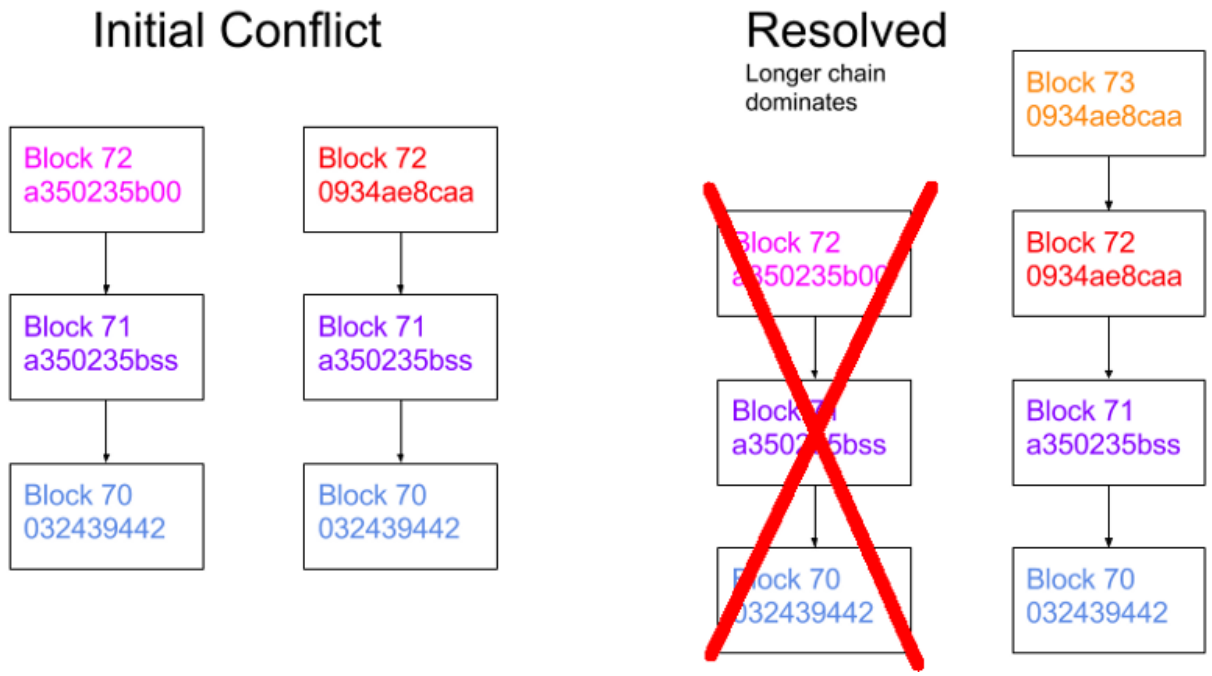
Fork and Same Block Submission:

The Soferox PoP Chain is immune to forks as any rule changes will never happen on our transaction chain. Should a same block be submitted normal hashing methods will take place to verify blocks. The security is built into a blockchain system through the distributed timestamping server and peer-to-peer network, and the result is a database that is managed autonomously in a decentralized way. This makes blockchains excellent for recording events — like medical records — transactions, identity management, and proving provenance. It is, essentially, offering the potential of mass disintermediation of trade and transaction processing.

There should always be only one explicit set of blocks in the chain at a given time. In case of conflicts (e.g. two nodes both generate block number 72) we choose the chain that has the longest number of blocks. An example is demonstrated below:

```
var
replaceChain
=
(newBlocks)
=> {
                if (isValidChain(newBlocks) && newBlocks.length > blockchain.length) {
                    console.log('Received blockchain is valid. Replacing current blockchain
            with received blockchain');
                    blockchain = newBlocks;
                    broadcast(responseLatestMsg());
```

```
        } else {
            console.log('Received blockchain invalid');
        }
    };
```



Private Keys:

A blockchain allows anyone to send value anywhere in the world where the blockchain file can be accessed. But you must have a private, cryptographically created key to access only the blocks you "own."

By giving a private key which you own to someone else, you effectively transfer the value of whatever is stored in that section of the blockchain.

So, to use the bitcoin example, keys are used to access addresses, which contain units of currency that have financial value. This fills the role of recording the transfer, which is traditionally carried out by banks.

It also fills a second role, establishing trust and identity, because no one can edit a blockchain without having the corresponding keys. Edits not verified by those keys are rejected. Of course, the keys — like a physical currency — could theoretically be stolen, but a few lines of computer code can generally be kept secure at very little expense. (Unlike, say, the expense of storing a cache of gold in a proverbial Fort Knox.)

Timestamp Server

The solution we propose starts with a timestamp server or device. A timestamp server works by taking a hash of a block of items to be timestamped and then published widely to the hash. What the timestamp does is prove that the data must have existed at the time to proceed into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it. This is a core principal and goes all the back to Satoshi Nakamoto's ideas.

Incentive

The first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation. This is important since there is no central authority issuing them. In our scenario, it is CPU time and electricity that is expended. The incentive can also be funded with transaction fees, however, Soferox is going to be a fee free environment.

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

Network

The steps to run the network are as follows:

1) New transactions are broadcasted to all the nodes.
2) Each node collects new transactions into a block once approved by rule chain.
3) Each node works on finding a proof-of-work, the nonce, for its block.
4) When a node finds the nonce, it sends out the block to all nodes.
5) Nodes accept the block only if all transactions in it are valid and not pre-flagged as validated.
6) Nodes announce their acceptance of the block by working next on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Nodes tend to always consider the longest chain to be the correct one and will keep working on extending it. If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first. In that case, they work on the first one they received, but save the other branch in case it becomes longer. The tie will be broken when the next proof-of-pact is found and one branch becomes longer; the nodes that were working on the other branch will then switch to the longer one. This is something again taken from PoW as its core belief system we believe to be very solid.

Programming With PoP

Our main goal is not to just create a blockchain that exist, but to create an ecosystem that is easy to enter, cost effective, and the best option for people to use. That is why PoP will have full programming support, simple API's, and convenient ways to enter this amazing technology.

Blockchains to the world have been mysterious and just a currency. We want to change that persona and really push it out into the real world. We will release API in all popular programming language types to give people an easy entrance to this and allow for easy integration into applications.

The Twin Chain

The Rule Chain:

The rule chain is the best adaptation of the blockchain we can bring. What a rule chain can contain are the requirement for block submission, coin creation, and currency rules. Normally this exist in the first blockchain, but we decided to split them out to allow for scalability and the ability to grow as the public demands it.

The rule chain accepts new blocks, and validates them against the rules set in place. This allows for a better, more efficient way of mining blocks as bad blocks will now not ever make it to the transaction chain and resources won't be wasted trying to mine and get them into the transaction chain.

The rule chain contains most of the same information that the transaction chain does, but the difference is in the data field rules are written in to be followed.

Rules are written into the block and stored in the chain and rules must be validated by the nodes to assure they are real rules. For now rules are set in place for the launch and will evolve as developers and the community finds a need to grow them. This will also allow time for old rules to still be used and give others time to adjust. This was something important as we did not want to have the issues facing a lot of developers today whenever a change is needed. Think of it as APIs have versions and when they update you can still use the old version of the API. This gives you time to adjust your code and then use the new API.

The Transaction Chain:

This is the chain that stores the transactional information. This chain has most the same fields traditional chains, but with two added things. We have added a rule number and a validated Boolean. A thing we do different is once a transaction has been validated a Boolean is set and

this block can no longer be changed. This is to prevent someone trying to resubmit the same block, but for more, or less of the transaction. This will also help with the problem of same block submissions, or at the very least reduce it.

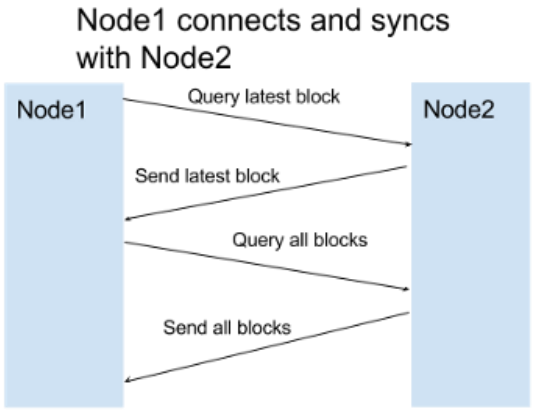
Since we have our rule chain to handle all changes and growth of the PoP network, the transaction chain is something that can remain immutable, thus avoiding the need for forks or massive changes.

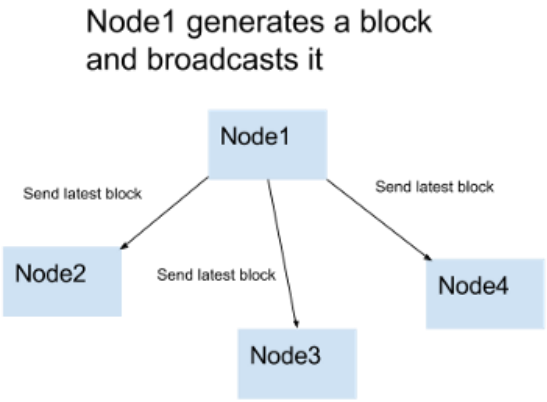
Architecture

Communicating with other nodes

An essential part of a node is to share and sync the blockchain with other nodes. The following rules are used to keep the network in sync.

- When a node generates a new block, it broadcasts it to the network
- When a node connects to a new peer it querys for the latest block
- When a node encounters a block that has an index larger than the current known block, it either adds the block the its current chain or querys for the full blockchain.





No automatic peer discovery is used. The location (=URLs) of peers must be manually added.

Controlling the node:

The user must be able to control the node in some way. This is done by setting up a HTTP server. As seen, the user is able to interact with the node in the following ways:

- List all blocks
- Create a new block with a content given by the user

- List or add peers

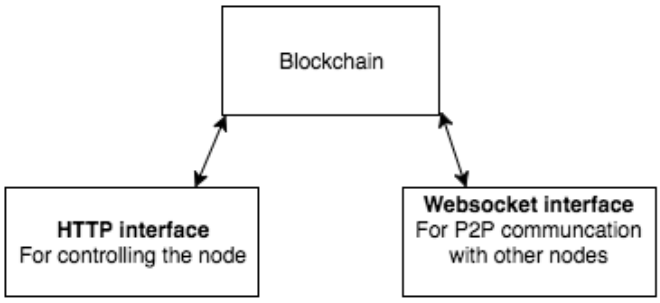
One example of this could appear like:

```
var
initHttpServer
= () => {
                    var app = express();
                    app.use(bodyParser.json());
                    app.get('/blocks', (req, res) => res.send(JSON.stringify(blockchain)));
                    app.post('/mineBlock', (req, res) => {
                        var newBlock = generateNextBlock(req.body.data);
                        addBlock(newBlock);
                        broadcast(responseLatestMsg());
                        console.log('block added: ' + JSON.stringify(newBlock));
                        res.send();
                    });
                    app.get('/peers', (req, res) => {
                        res.send(sockets.map(s => s._socket.remoteAddress + ':' +
                s._socket.remotePort));
                    });
                    app.post('/addPeer', (req, res) => {
                        connectToPeers([req.body.peer]);
                        res.send();
                    });
                    app.listen(http_port, () => console.log('Listening http on port: ' +
                http_port));
                };
```

The most straightforward way to control the node is e.g. with Curl:

#get all blocks from the node
curl http://localhost:3001/blocks

It should be noted that the node actually exposes two web servers: One for the user to control the node (HTTP server) and one for the peer-to-peer communication between the nodes.(Websocket HTTP server)



Are 'Side-Chains' Possible?

Side chains are something that have been discussed for a while and proven to be possible. Soferox is taking them a step further and moving away from possible, to reality. The basic principal here is blockchains communicating with other blockchains.

The way blockchains are today are a great example of how this technology can thrive, but we feel it can do more than thrive in a devoted community. We feel blockchain tech can dominate globally.

Applications

With the technical underpinnings out of the way, in this section we explore user-facing applications of sidechains, which effectively extend cryptocurrency to do things that it cannot today.

From the blockstream white paper some possible uses of side chains are, but not limited to the following.

Altchain Experiments

The first application, already mentioned many times, is simply creating altchains with coins that derive their scarcity and supply from Soferox(SFX). By using a sidechain which carries SFX rather than a completely new currency, one can avoid the thorny problems of initial distribution and market vulnerability, as well as barriers to adoption for new users, who no longer need to locate a trustworthy marketplace or invest in mining hardware to obtain altcoin assets.

Technical Experimentation

Because sidechains are technically still fully-independent chains, they are able to change features of Crypto such as block structure or transaction chaining. Some examples of such features are:

• By fixing undesired transaction malleability —which can only be fixed partially — protocols which involve chains of unconfirmed transactions can be executed safely. Transaction malleability is a problem in Bitcoin which allows arbitrary users to tweak transaction data in a way that breaks any later transactions which depend on them, even though the actual content of the transaction is unchanged. An example of a protocol broken by transaction malleability is probabilistic payments

• Improved payer privacy, e.g. the ring signature scheme used by Monero, can reduce the systemic risk of the transactions of particular parties being censored, protecting the fungibility of the cryptocurrency. Improvements to this have been suggested by Maxwell and Poelstra, which would allow for even greater privacy. Today,400 ring signatures can be used with Monero coins, but not bitcoins; sidechains would avoid this exclusivity.

• Script extensions (for example, to efficiently support colored coins) have been proposed. Since such extensions are usable only by a small subset of users, but all users would need to deal with the increased complexity and risk of subtle interactions, these extensions have not been accepted into Bitcoin. Other suggested script extensions include support for new cryptographic primitives.

Since changes like these affects only the transfer of coins, rather than their creation, there is no need for them to require a separate currency. With sidechains, users can safely and temporarily experiment with them. This encourages adoption for the sidechain, and is less risky for participants relative to using an entirely separate altcoin.

Economic Experimentation

Bitcoin's reward structure assigns new coins to miners. This effectively inflates the currency but it winds down over time according to a step-wise schedule. Using this inflation to subsidize mining has been a successful complement to transaction fees to secure the network. An alternate mechanism for achieving block rewards on the sidechain is demurrage, an idea pioneered for digital currency by Freicoin (http://freico.in). In a demurring cryptocurrency, all unspent outputs lose value overtime, with the lost value being recollected by miners. This keeps the currency supply stable while still rewarding miners. It may be better aligned with user interests than inflation because loss to demurrage is enacted uniformly everywhere and instantaneously, unlike inflation; it also mitigates the possibility of long-unspent "lost" coins being reanimated at their current valuation and shocking the economy, which is a perceived risk in Bitcoin and other currencies similar to it. Demurrage creates incentives to increase monetary velocity and lower interest rates, which are considered (e.g. by Freicoin advocates and other supporters of Silvio Gesell's theory of interest) to be socially beneficial. In pegged sidechains, demurrage allows miners to be paid in existing already valued currency.

Privacy

The traditional banking model achieves a somewhat a level of privacy by restricting or limiting access to information to the parties involved and the trusted third party.

"The necessity to announce all transactions publicly precludes this method, but privacy can still be maintained by breaking the flow of information in another place: by keeping public keys anonymous. The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone. This is similar to the level of information released by stock exchanges, where the time and size of individual trades, the "tape", is made public, but without telling who the parties were."
-S.N.

Privacy is something that is a must and a single chain must carry a lot of weight on its shoulders and we feel this is not a needed amount of stress on one chain. The twin-chains are here to relieve that stress and make blockchains not only safer, but more efficient.

Conclusion

The PoP blockchain isn't just a clone, but it is a new look on chain technology and with our twin chains we hope to bring blockchains closer to the real world, and to see them being used in applications.

Soferox has proposed a system for transactions without relying on trust. Soferox started with the usual framework of coins made from digital signatures, that gives a strong control of ownership, but is incomplete and only partial without a way to prevent 'double-spending'.

To solve this, we are proposing a peer-to-peer network, similar to Bitcoin, using proof-of-pact to validate blocks over a ruling chain and then record a public history of transactions that quickly becomes impractical for an attacker to change if honest nodes control a majority of CPU power. The network is robust in its unstructured simplicity.

Nodes will work all at once with little coordination needed. Nodes can leave and rejoin the network at whenever they want, accepting the proof-of-pact chains as proof of what happened while they were away. They vote with their CPU/GPU power, broadcasting their acceptance of valid blocks by working on extending them and the rule chain focusing on rejecting invalid blocks by refusing to work on them. Any needed rules and incentives can be enforced with this consensus mechanism and by modifying our rule chain.