# STELLITE

## A MODERN, SECURE AND DECENTRALIZED CRYPTOCURRENCY.

April 2018

## SUMMARY

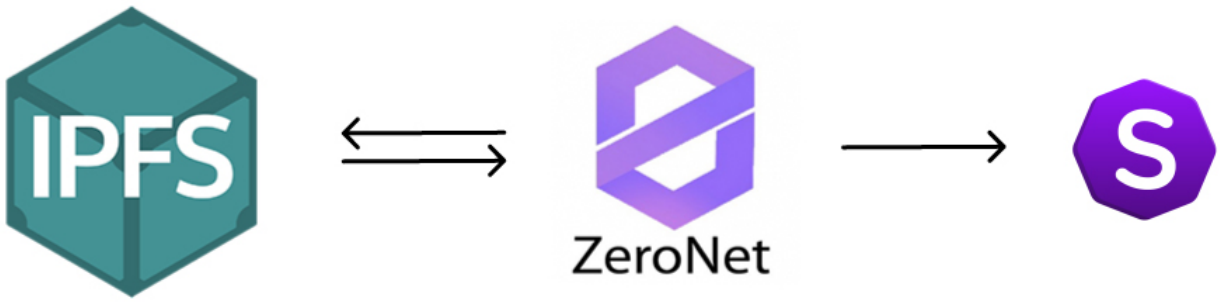
Stellite solves issues that have been puzzling cryptocurrency developers for years on topics such as an efficient and decentralized method of distributing peer list without hard coding them, using the billions of small devices to combine their computing power and form a part of a huge Proof-of-work network which helps people use their own small devices to be not only a supporting factor to the network but also be provisioned a small reward in doing so. Stellite is unique in linking both the IPFS and ZeroNet technologies into a cryptocurrency and scaling it globally for both mobile and desktop usage. Stellite will be the first cryptocurrency worldwide to provide scalable global mobile mining. In the next coming paragraphs, we will be explaining the implementation methodologies and the technologies used for making Stellite.

## Node list distribution

Most cryptocurrencies (including bitcoin, monero, etc.) hard-code trusted nodes and peers to help users set up their own nodes. This has been proven to work, but calls for a noticeable security risk. Hacking a hard-coded node will cause all future nodes from that point on to create hiccups for new users and making their daemons and clients useless. Stellite's proposed solution is to publish the node list on a decentralized network called IPFS and using ZeroNet technologies to serve as the front end for adding new nodes to the IPFS ledger or list of nodes. So in simple terms IPFS acts like a permanent storage of the node list and ZeroNet will be the dynamic front end of this permanent storage.

ZeroNet uses Bitcoin cryptography and BitTorrent technology to build a decentralized censorship-resistant network. Users can publish static or dynamic websites into ZeroNet and visitors can choose to also serve the website. Websites will remain online even if it is being served by only one peer. When a site is updated by its owner, all nodes serving that site (previous visitors) will receive only the incremental updates done to the site content. ZeroNet comes with a built-in SQL database. This makes content-heavy site development easy. The DB is also synced with hosting nodes with incremental updates.[1]

The Inter Planetary File System (IPFS) protocol works like a single BitTorrent swarm, people exchanging objects as part of a single tree repository. They're based on a content-addressed block storage model. IPFS combines a distributed hashtable, an incentivized block exchange, and a self-certifying namespace. IPFS has no single point of failure, and nodes do not need to trust each other. These make IPFS the best candidate to host a single peer list file that contains nodes which can contribute to being the backbone for the entire Stellite network.[2]




Traditionally the addition of a new file onto IPFS and the retrieval has 3 steps, creation of the file, addition of the file and the actual retrieval. This happens automatically at a certain predefined interval but for the sake of demonstration we are going to be doing it manually.

```
VA:~ hayzam$ cat node_list.dat
111.111.111.111:20188
112.112.112.112:20188
113.113.113.113:20188
114.114.114.114:20188

VA:~ hayzam$
```

Viewing of the node_list.dat file I created to be added to IPFS

```
VA:~ hayzam$ ipfs add node_list.dat
added QmPHA4hQzrdZ5cQwq2qXMatzJMFZQzUFsmhN2RPxLKJvKF node_list.dat
VA:~ hayzam$
```

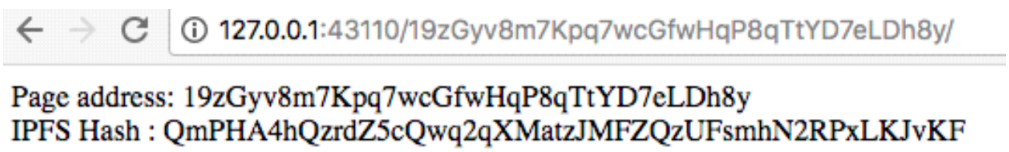
Adding the file to IPFS and retrieving the hash for the file

The hash is unique for each file that is uploaded onto IPFS, so the list can be distributed across a multitude of IPFS peers without it being tampered with.


Viewing the stored file using an IPFS gateway

The gateway is a tricky thing because there are 2 ways of approaching this, we could use a single gateway and retrieve the file using the unique hash from the centralized gateway or we could bundle an IPFS daemon and gateway along with the Stellite daemon and then retrieve the file using the unique hash locally. We are implementing Stellite to run its own IPFS daemon and fetch the list from the unique hash without needing a centralized gateway.

Another question that arises is how the hash will be delivered to the Stellite daemon. This is where ZeroNet comes in. We have hard coded a ZeroNet address to retrieve the latest peer list hash from. ZeroNet is a python application which runs along with the daemon every time updating it self with the latest list with the latest hash that has been provisioned in the ZeroNet decentralized site. The below given image is an example of a ZeroNet address giving us an IPFS hash for file retrieval.



The ZeroNet site that is hardcoded into the Stellite daemon is used to get the hash which in turn is used to download the node list.

## IPFS and ZeroNet based blockchain import utility

Syncing the blockchain is a tedious and time consuming process especially if the chain in question is a few years old, even if a user has a considerable amount of network bandwidth the downloading, verifying an addition of the blockchain to the local database from other peers will take time. To solve this issue we propose a raw copy of the blockchain to be stored on IPFS and the hash on IPFS to be stored on Zeronet so it can be easily be changed in weekly or even daily intervals. How this compares to using the traditional websites is that IPFS content is tamper proof and the raw blockchain can be downloaded with utmost confidence.

The utility will be  cross platform with an intuitive frontend and as more people support the storage of the content the download will get faster.

## Complete node anonymity

Nodes are not private, they are open to the public internet, other nodes or malicious actors can take advantage of this specific property to track transactions that are being relayed by them and then break the untraceability of the coin. To avoid this we will be providing the users with an easily configurable tor and or Kovri (an open source implementation of I2P protocol) instance with which a user of the particular node can mask their clear net IP address.

## True mining for all devices

Stellite plans to bring mining to all sorts of devices, from smartphones to smart TV's, we do this by running our custom miner in such a way that it doesn't damage any of the internal aspects of the machine it's being run on. A person can create an address and mine to that address from an endless number of devices. This can help countries such as India where there is an abundance of hash power in the form of smart devices as the ones mentioned above and all of them currently are being under utilized. In our calculations the Stellite app takes less CPU toll than apps such as the Facebook app.

The math involved this is simple, a mid tier smartphone provides around 23H/s at full throttle, when at 25% usage provides around 6H/s. A user can use her specific wallet address 'X' to mine on as many devices as possible. So if Alice has 200 devices she has 6 * 200 = 1200 Hashes or 1.2KH/s. Which is equivalent to 2 GTX 1080 graphics card running at full potential. This is possible due to a variable difficulty algorithm we have worked on, if a device such as a smartphone can only provide 10H/s the improved pool server immediately gives out block templates exactly corresponding to that hashrate improving not just speed but efficiency too. The new pool works on an event loop and also has adequate proxy servers for running millions of workers. This will not just strengthen the network but also redistributes wealth globally.

The main problem with many of the past implementation were that software which are in use for most mobile miners don't take care of the device's internal aspect but with our custom miner with it's own protocol for power and temperature control, we aim to solve that problem exactly.

## AMAYC Protocol

AMAYC protocol uses deep learning to find devices optimal settings, allowing to mine without damaging hardware while easing power consumption. Benchmarking is done on the client side in the first few mining CPU cycles which takes around 8.6 seconds on a mid tier smartphone.

It makes use of a pre trained neural network to give an efficiency rating used by the miner to calibrate later on. The dataset training the network is a culmination of data from hundreds of devices, providing us with a clear cut idea of how good it can perform without damaging itself.

Below given is a sample set of data which is used to train the neural network.

| Hashes per second | Temperature | Rating |
|---|---|---|
| 15 | 38 | 1 |
| 16 | 38.15 | 1 |
| 10 | 45 | 0 |
| 9 | 35 | 1 |

## Mobile Miner

The Mobile miner is an application able to solve low diff blocks given by any pool server that supports it. The limit of the miner is set to 40 degree celcius, even though a device can get warmer, which seems to be a safe limit for most new generation smartphones. When the CPU reaches that limit, the miner turns itself off until the phone gets back to normal temperatures and then begins once the device is back to 0 potential range. This way we can ensure the device never gets so warm that the lithium ion battery suffers ANY damage.

## Neural Network implementation for AMAYC protocol

Traditional benchmarking techniques do not fit well to the mobile devices use cases. They are too slow, computationally intensive and usually do not take the device temperature in account. With the neural network we will be able to find the standard in a very short span of time and also resource efficiently. For the implementation we will be using a simple classifier. We have made a custom node module for tests, the usage is very simple as shown below, the code for this module will be opensourced along with the miner.

```
var minerClassifier = require('stelliteNN');
 var network = new minerClassifier.NeuralNetwork();
/* Just some data for a very simple neural network. */
network.train([{input: { h: 15, t: 41 }, output: { bad: 1 }},
        {input: { h: 16, t: 38 }, output: { good: 1 }},
        {input: { h: 25, t: 45 }, output: { bad: 1 }},
        {input: { h: 20, t: 39 }, output: { good: 1 }}]);
var result = network.run({ h: 5, t: 35 });
//{ bad: 0.49398529529571533, good: 0.5060188174247742 }
```

From the above code it shows that the hash rate of 5H/s at 35 degree celsius is somewhat okay clocking in at 50% rating. From this we can truly understand the potential of this benchmarking approach.

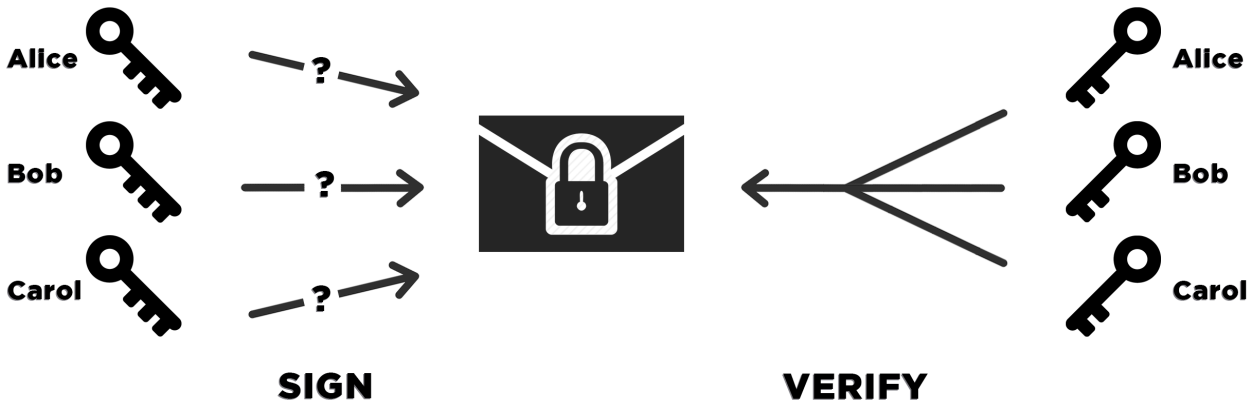
## CryptoNote Algorithm[3]

The CryptoNote algorithm is released under an open source license and has been adopted and incorporated into Stellite as it forms the basis for a solid, well tested cryptocurrency application. It is the same technology used by some of the best currencies out there like monero and bytecoin. Now we can discuss some of the merits of the currency as well as cryptonote.

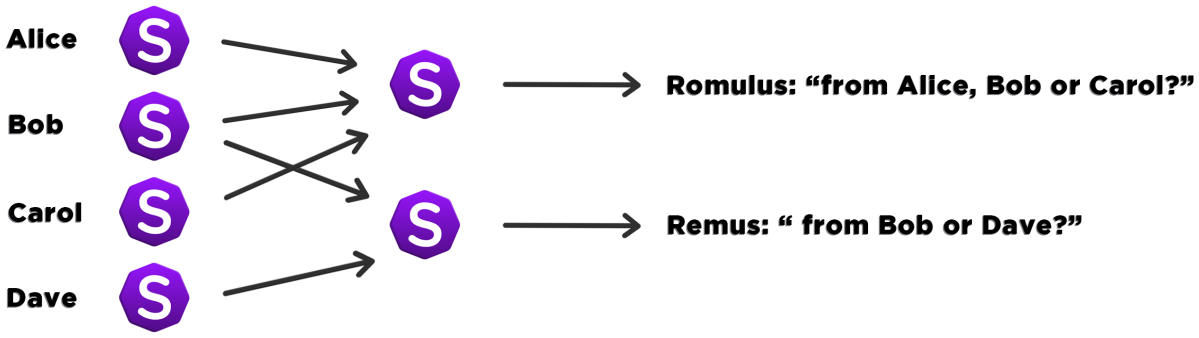
### Untraceable payments

The ordinary digital signature (e.g. (EC)DSA, Schnorr, etc...) verification process involves the public key of the signer. It is a necessary condition, because the signature actually proves that the author possesses the corresponding secret key. But it is not always a sufficient condition.


Ring signature is a more sophisticated scheme, which in fact may demand several different public keys for verification. In the case of ring signature, we have a group of individuals, each with their own secret and public key. The statement proved by ring signatures is that the signer of a given message is a member of the group. The main distinction with the ordinary digital signature schemes is that the signer needs a single secret key, but a verifier cannot establish the exact identity of the signer. Therefore, if you encounter a ring signature with the public keys of Alice, Bob and Carol, you can only claim that one of these individuals was the signer but you will not be able to pinpoint him or her.



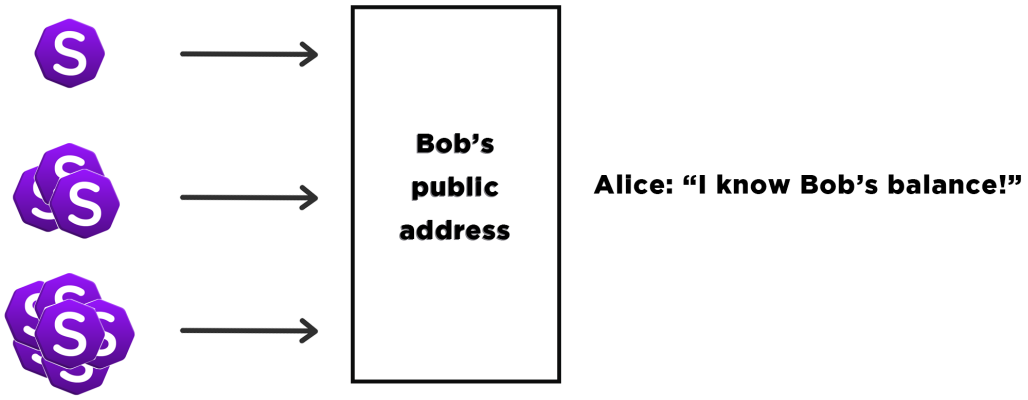
This concept can be used to make digital transactions sent to the network untraceable by using the public keys of other members in the ring signature one will apply to the transaction. This approach proves that the creator of the transaction is eligible to spend the amount specified in the transaction but his identity will be indistinguishable from the users whose public keys he used in his ring signatures.




It should be noted that foreign transactions do not restrict you from spending your own money. Your public key may appear in dozens of others’ ring signatures but only as a muddling factor (even if you already used the corresponding secret key for signing your own transaction). Moreover, if two users create ring signatures with the same set of public keys, the signatures will be different (unless they use the same private key).

## Unlinkable Transactions

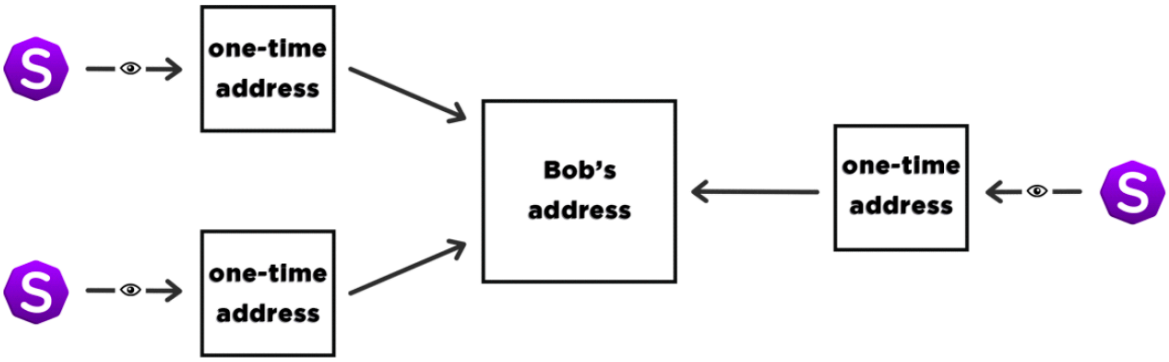
Normally, when you post your public address, anyone can check all your incoming transactions even if they are hidden behind a ring signature. To avoid linking you can create hundreds of keys and send them to your payers privately, but that deprives you of the convenience of having a single public address.



Stellite's CryptoNote solves this dilemma by an automatic creation of multiple unique one-time keys, derived from the single public key, for each p2p payment. The solution lies in a clever modification of the Diffie-Hellman exchange protocol. Originally it allows two parties to produce a common secret key derived from their public keys. In our version the sender uses the receiver's public address and his own random data to compute a one-time key for the payment.

The sender can produce only the public part of the key, whereas only the receiver can compute the private part; hence the receiver is the only one who can release the funds after the transaction is committed. He only needs to perform a single-formula check on each transactions to establish if it belongs to him. This process involves his private key, therefore no third party can perform this check and discover the link between the one-time key generated by the sender and the receiver's unique public address.

An important part of our protocol is usage of random data by the sender. It always results in a different one-time key even if the sender and the receiver both remain the same for all transactions (that is why the key is called "onetime"). Moreover, even if they are both the same person, all the one-time keys will also be absolutely unique.

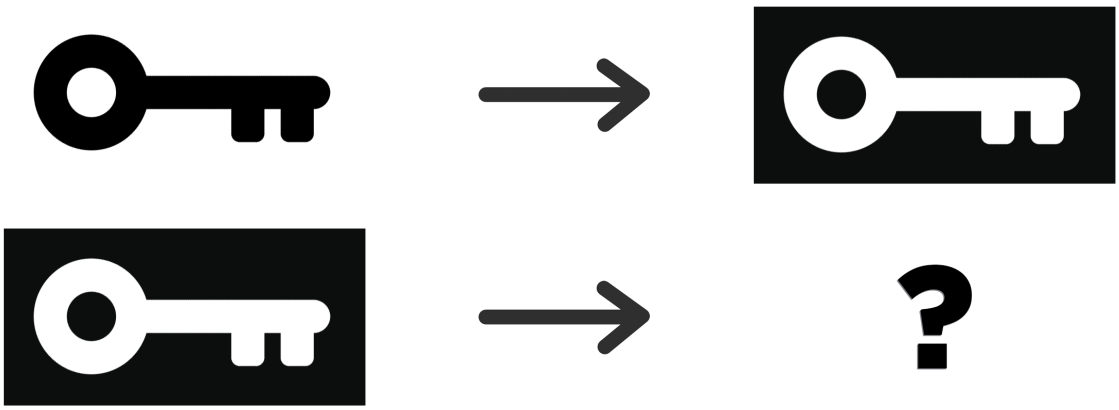



## Double-spending proof

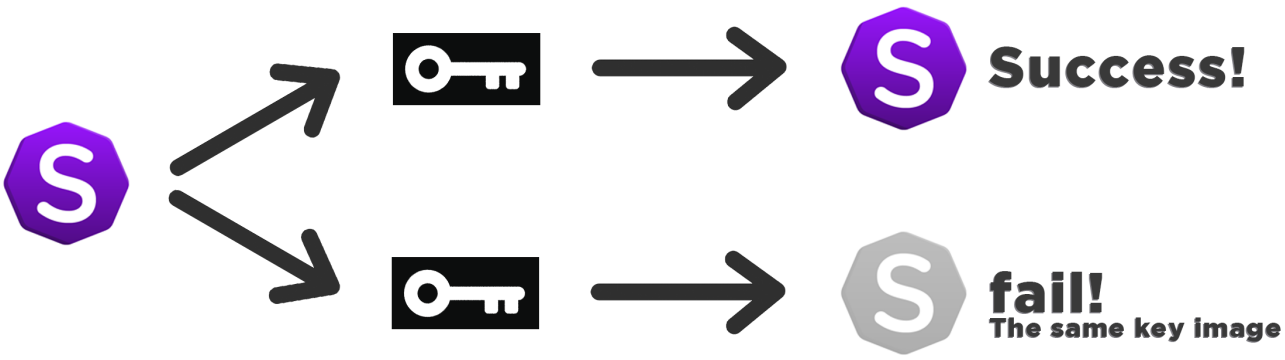
Fully anonymous signatures would allow spending the same funds many times which, of course, is incompatible with any payment system's principles. The problem can be fixed as follows.

A ring signature is actually a class of crypto-algorithms with different features. The one Stellites's CryptoNote uses is the modified version of the "Traceable ring signature". In fact we transformed traceability into linkability. This property restricts a signer's anonymity as follows: if he creates more than one ring signature using the same private key (the set of foreign public keys is irrelevant), these signatures will be linked together which indicates a double-spending attempt.

To support linkability, Stellite's CryptoNote introduced a special marker being created by a user while signing, which we called a key image. It is the value of a cryptographic one-way function of the secret key, so in math terms it is actually an image of this key. One-wayness means that given only the key image it is impossible to recover the private key. On the other hand, it is computationally impossible to find a collision (two different private keys, which have the same image). Using any formula, except for the specified one, will result in an unverifiable signature. All things considered, the key image is unavoidable, unambiguous and yet an anonymous marker of the private key.




All users keep the list of the used key images (compared with the history of all valid transactions it requires an insignificant amount of storage) and immediately reject any new ring signature with a duplicate key image. It will not identify the misbehaving user, but it does prevent any double-spending attempts, caused by malicious intentions or software errors.



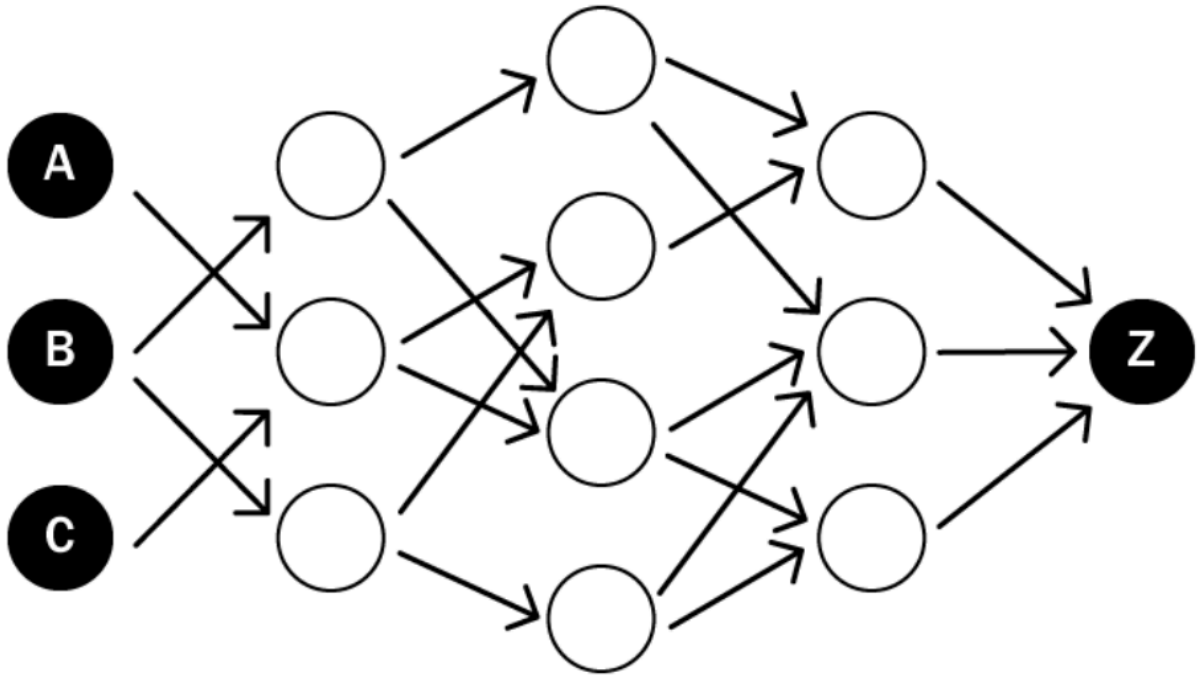

## Blockchain analysis resistance

There are many academic papers dedicated to the analysis of the Bitcoin's blockchain. Their authors trace the money flow, identify the owners of coins, determine wallet balances and so on. The ability to make such analysis is due to the fact that all the transfers between addresses are transparent: every input in a transaction refers to a unique output. Moreover, users often re-use their old addresses, receiving and sending coins from them many times, which simplifies the analyst's work. It happens unintentionally: if you have a public address (for example, for donations), you are sure to use this address in many inputs and transactions.

Stellite's CryptoNote is designed to mitigate the risks associated with key re-usage and one-input-to-one-output tracing. Every address for a payment is a unique one-time key, derived from both the sender's and the recipient's data. It can appear twice with a probability of a 256-bit hash collision. As soon as you use a ring signature in your input, it entails the uncertainty: which output has just been spent?

Trying to draw a graph with addresses in the vertices and transactions on the edges, one will get a tree: a graph without any cycles (because no key/address was used twice). Moreover, there are billions of possible graphs, since every ring signature

produces ambiguity. Thus, you can't be certain from which possible sender the transaction edge comes to the address-vertice. Depending on the size of the ring you will guess from "one out of two" to "one out of a thousand". Every next transaction increases the entropy and creates additional obstacles for an analyst.



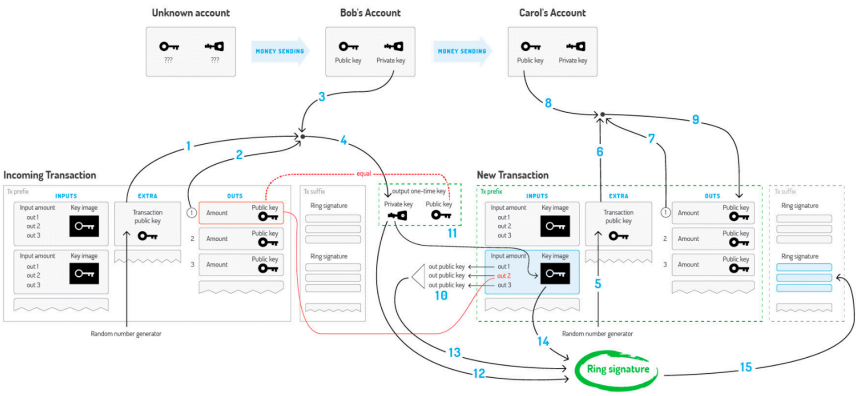

## Standard CryptoNote transactions

A standard Stellite CryptoNote transaction is generated by the following sequence covered in the white paper. Bob decides to spend an output, which was sent to the one-time public key. He needs Extra (1), TxOutNumber (2), and his Account private key (3) to recover his one-time private key (4).

When sending a transaction to Carol, Bob generates its Extra value by random (5). He uses Extra (6), TxOutNumber (7) and Carol's Account public key (8) to get her Output public key (9).

In the input Bob hides the link to his output among the foreign keys (10). To prevent double-spending he also packs the Key image, derived from his One-time private key (11).

Finally, Bob signs the transaction, using his One-time private key (12), all the public keys (13) and Key Image (14). He appends the resulting Ring Signature to the end of the transaction (15).



## Zero confirmation transactions

One of Stellite's main goals are to get adopted by the masses, to be used in everyday life for simple transactions. This is greatly hindered by the fact that transactions takes time to be added into blocks. We have made a custom point-of-sale system that works with our mobile wallets to allow transactions to go through to a merchant in less than a few seconds.

The inner working of this function are very simple. When a transaction is broadcasted onto the network, it first goes into a pool of transactions called the txpool or the memory pool, from here it is added onto a block, even with our block time being 60 seconds, it could take a couple of minutes more depending on amount of transactions in the pool. To avoid this waiting period we are allowing merchants to accept 0 confirmation transactions. The point of sale will also come with an option where the merchant can set the amount of confirmation needed, for instance if the transaction is of high value.

Another solution we provide to the users is to use a proprietary application called StellitePay which uses a centralized database of transaction to be handled in a similar fashion of how traditional exchanges handle them, which is by using a simple pool of transactions on a single server before commiting onto the network, this is mostly for the novice users for whom setting up a regular node or a node which connects to a remote node is not feasible. Thus increasing even more the adoption capabilities.

## Adaptive Limits

A decentralized payment system must not depend on a single person's decisions, even if this person is a core developer. Hard constants and magic numbers in the code deter the system's evolution and therefore should be eliminated (or at least be cut down to the minimum). Every crucial limit (like max block size or min fee amount) should be re-calculated based on the system's previous state. Therefore, it always changes adaptively and independently, allowing the network to develop on it's own. Stellite's CryptoNote has the following parameters which adjust automatically for each new block:

- **Difficulty**

The general idea of our algorithm is to sum all the work that nodes have performed during the last 720 blocks and divide it by the time they have spent to accomplish it. The measure of the work is the corresponding difficulty value for each of the blocks. The time is calculated as follows: sort all the 720 timestamps and cut-off 20% of the outliers. The range of the rest 600 values is the time which was spent for 80% of the corresponding blocks

- **Max block size**

Let MN be the median value of the last N blocks sizes. Then the "hard-limit" for the size of accepting blocks is 2*MN. It averts blockchain bloating but still allows the limit to slowly grow with the time if necessary. Transaction size does not need to be limited explicitly. It is bounded by the size of the block.

## Improvements to the difficulty adjustment algorithm

The general idea of our original difficulty algorithm is to sum all the work that nodes have performed during the last 720 blocks and divide it by the time they have spent to accomplish it. The measure of the work is the corresponding difficulty value for each of the blocks. The time is calculated as follows : sort all the 720 timestamps and cut-off 20% of the outliers. The range of the rest 600 values is the time which was spent for 80% of the corresponding blocks.

The downside of a simple moving average algorithm is that it is way too slow adjust the difficulty and causes a long time in delay if a large hashrate miner comes to the network. Hence to curb the issue we hardforked at height 67,500 to use a new difficulty algorithm which adjusts much faster to the net hashpower of a network.The new algorithm adjusts difficulty based on the harmonic mean of the difficulties (the inverse of the average target) divided by the Linearly Weighted Moving Average (LWMA) of the solvetimes. It gives more weight to the most recent solvetimes. Difficulty will double in 15 blocks

during the introduction of a high hashrate miner, and drop back down close to normal in 10 blocks ie if the high hashrate miner leaves the network.

## Smooth emission

The upper bound for the overall amount of all digital coins is also digital:

**MSupply = 264 − 1 atomic units**

This is a natural restriction based only on the implementation limits, not on intuition like “N coins ought to be enough for everybody”. To make the emission process smoother Stellite’s CryptoNote uses the following formula for block rewards :

**BaseReward = (MSupply − A) >> 18**

Where A is amount of previously generated coins. It gives a predictable growth of the money supply without any breakpoints.

## Egalitarian Proof of Work

The proof of work mechanism is actually a voting system. Users vote for the right order of the transactions, for enabling new features in the protocol and for the honest money supply distribution. Therefore, it is important that during the voting process all participant have equal voting rights. Stellite’s CryptoNote brings the equality with an egalitarian proof-of-work pricing function, which is perfectly suitable for ordinary PCs. It utilizes built-in CPU instructions, which are very hard and too expensive to implement in special purpose devices or fast memory on-chip devices with low latency. We propose a new memory-bound algorithm for the proof-of-work pricing function. It relies on random access to a slow memory and emphasizes latency dependence. As opposed to scrypt, every new block (64 bytes in length) depends on all the previous blocks. As a result a hypothetical “memory-saver” should increase his calculation speed exponentially.

Our algorithm requires about 2 Mb per instance for the following reasons:

- It fits in the L3 cache (per core) of modern processors, which should become mainstream in a few years.
- A megabyte of internal memory is an almost unacceptable size for a modern ASIC pipeline.
- GPUs may run hundreds of concurrent instances, but they are limited in other ways: GDDR5 memory is slower than the CPU L3 cache and remarkable for its bandwidth, not random access speed.
- Significant expansion of the scratchpad would require an increase in iterations, which in turn implies an overall time increase. “Heavy” calls in a trust-less p2p network may lead to serious vulnerabilities, because nodes are obliged to check every new block’s proof-of-work. If a node spends a considerable amount of time on each hash evaluation, it can be easily DDoSed by a flood of fake objects with arbitrary work data (nonce values).

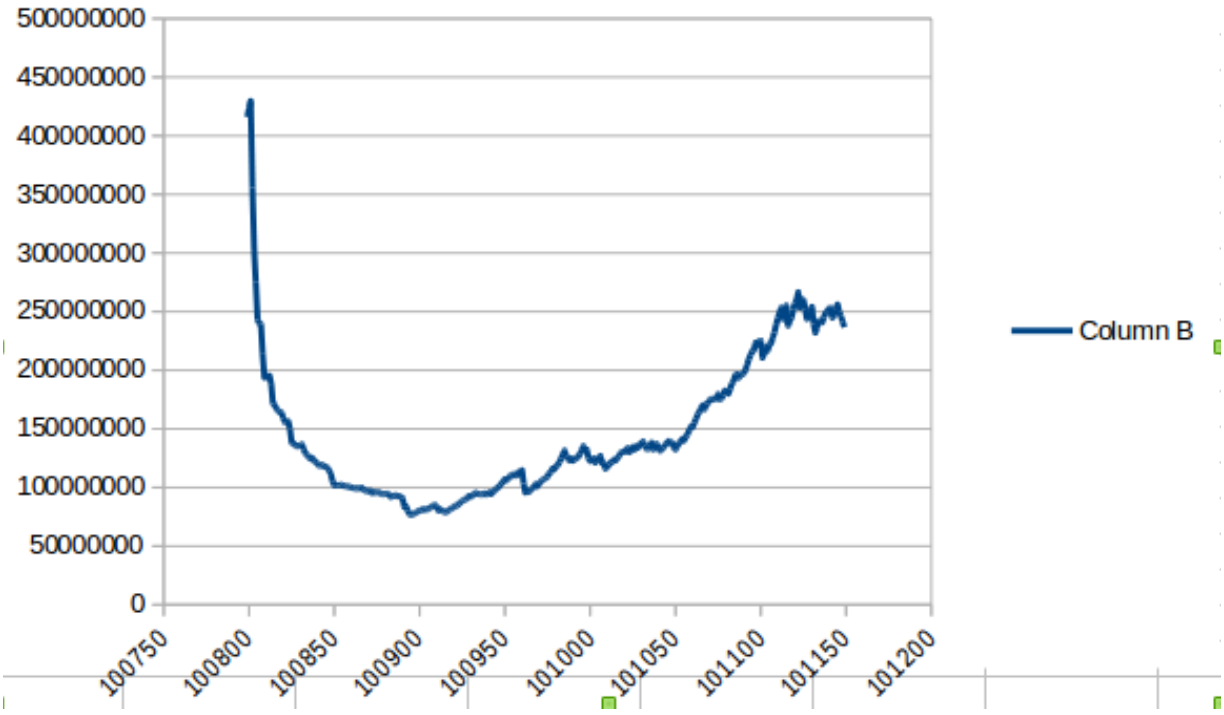
  One of the proof-of-work algorithms that is in line with our propositions is CryptoNight. It is designed to make CPU and GPU mining roughly equally efficient and restrict ASIC mining.

## A solution to restrict ASIC mining

The drawback of most cryptocurrencies like bitcoin is that the proof of work algorithm which uses the SHA-256 hash function can be mined on special Application Specific Integrated Circuits (ASICs). This is due to the huge gaps in efficiency when compared with CPU/GPU mining. This gap can make ASIC the only device that is capable of mining, this is a threat to our ideology. We wish to make mining further decentralized by helping people mine on their smartphones.

To solve the ASIC dilemma we will be changing our current PoW algorithm(CryptonightV7) slightly in regular protocol upgrades/hard forks which will render ASICs useless as they are meant to be for just one specific function and cannot adapt like regular CPUs/GPUs. Even if done in largely spaced intervals PoW changes poses a significant drop in network hash rate for a considerable amount of time, in parallel with LWMA as our difficulty adjustment algorithm the net difficulty will reflect the real network hash rate in the span of a few hours. Below given is a graph during our first PoW change with LWMA.

As you can see the difficulty dropped significantly in a short span of blocks but also regained momentum as more people joined in at a later time.



## References and Citations:


• [3] Cryptonote white paper: https://cryptonote.org/whitepaper.pdf
• [3] Cryptonote Inside: https://cryptonote.org/inside
• [3] Bitcoin white paper: https://bitcoin.org/bitcoin.pdf
• [3] Electroneum white paper: https://electroneum.com/technical-white-paper.pdf
• [2] IPFS white paper: https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf
• [1] ZeroNet Documentation: https://zeronet.readthedocs.io