V.E.L.A.S.

Virtual Expanding Learning Autonomous System (VLX)

AI-Operated Delegated Proof-of-Stake Blockchain

Not intended for fundraising or marketing purposes.

Written by Alexander Alexandrovych Alexandrov

Official website: velas.com

Special thanks to Alex Lightman, Jason Butcher, Mondas, Pr0d1gy, Konstantin, Timur, Orhun, Sina, Trevor, Albert, Jag, Farkhad & Ruslan.

Abstract

Velas is a self-learning and optimizing blockchain platform for secure, interoperable, extremely scalable transactions and smart contracts. Velas uses AI-Operated DPOS (AIDPOS) consensus to secure the blockchain for high volume transactions processing without sacrificing decentralization, stability and security. Through the use of intelligent AI-operated DPOS consensus algorithm, corruptible human dependencies are removed resulting in a fault tolerance system preventing most major issues like the 51% attack and nothing-at-stake problem.

Introduction

Velas blockchain platform uses neural networks optimized by artificial intuition to enhance it's consensus algorithm. The purpose of Velas is to address and fix existing issues and challenges faced by most existing blockchains.

Neural networks are used to calculate the rewards for nodes' operators and time of block formation. Matrix calculation servers (neural network weights) are located on nodes of the network members for receiving rewards. Similar to Bitcoin miners, nodes need to have large computational resources to calculate the matrix. For example, installing a powerful dedicated video card (GPU).

To train a neural network, genetic algorithm is used during the pre-training period and using the method of back propagation of error to find the minimum of the objective function.

Algorithm of network training

Genetic algorithm:

1. Creation of a specimen: a matrix with random weights (genes)
2. Competition: getting the minimum of the objective function
3. Selection: ranking specimens by error. The smallest error is a victory.
4. Reproduction: exchange of the matrix' elements, or genes, 50/50 from the first two most successful.
5. The cycle is repeated until a 70% probability is obtained

Each node forms its own dataset (training sample) from the blockchain data.
On this data, a layered training of the genetic algorithm occurs. Every next layer is learning the previous one: training on the principle of an autoencoder.

After the pre-training, the backpropagation method is applied to tighten the weights to a minimum.

The network must be trained before the next block cycle.

Choosing the best network.

- The nodes form a test dataset prior to the formation of a new block cycle.
- After the formation of the test sample, the matrices are checked.
- The matrix with the smallest error wins and gets loaded into the next era.
- The node of the winning matrix receives Velas (VLX) from the network

A market for the sale and purchase of the most effective specimens will evolve which will open up the development of the blockchain network maintenance algorithm with rewards for other systems based on Velas.

In the era of ultra-fast computing systems based on quantum technology, it's possible to achieve close to an ideal state of a neural network in a short period of time having trained it in a commercially available quantum computers, while only adding new features over time.

By solving blockchain network maintenance tasks, each AIDPOS node makes an intellectual contribution by further developing a neural network. This all happens without requiring any understanding programming languages by operators of selected nodes.

Network Versions Roll Out Schedule

Stage 1 (Pre-Alpha):
Creation of the blockchain system structure, coins, transactions on 4 nodes. Nodes will be run by network organizers during pre-alpha stage. Smart contract in wallet will allow all CPS coin (CoinpaymentsCoin) users to swap 1:1 to VLX (Velas). Creation of tokens or custom digital assets.

Stage 2 (Alpha):
Creation of a stable system, Velas deployment on 10 nodes and testing AI from network organizers 4 servers. Introduction of Multi-Wallet Containers public and private send functions.

Stage 3 (Beta):
Addition of AI offsprings populating test nodes in order to compete with server side AI. Further expansion for all the leading cryptocurrencies for Multi-Wallet container system for sending, receiving and smart contracts functions.

Stage 4 (Release Candidate):
Integration of AI into existing advisor nodes, amount of which is now set by the AI logic.

Stage 5 (Release):
Launching the full functionality of the system. Users can now download neural network kits and optimize them for their projects using visual tools and earn for their contribution. Coin nodes will be spun up using trained neuron network pretrained for most environments and will allow easy setup and maintenance of coin nodes.

Velas Platform Overview

Terms and Definitions

- **VelasCycle** – a limited time period and the number of blocks. Each **VelasCycle** is made of one **SimpleBlock** + one **CycleBock**;
- **CycleBlock** – a block that contains a list of nodes allowed to create blocks in the current **VelasCycle**;
- **SimpleBlock** – Includes the list of transactions – not to be confused with a **CycleBlock**;
- **NodeID** – Each node contains both a secret and public key. The public key is the Node identifier and secret key used to generate a **BlockSign**;
- **BlockSign** – The block creator's signature identifying the block number;
- **TxQuery** – A query or "special transaction". It must be broadcasted to the network to show its intention of producing blocks within the next **VelasCycle**. This transaction will include the **NodeID**. The amount of 100,000 Velas coins is required to generate a **TxQuery**.

At the end of a **VelasCycle** each node must define the next algorithm by the following:

1. Collect all **TxQueries** from the previous **VelasCycle**;
2. Lexicographically sort the list of the **TxQueries** by **NodeIDs**;
3. Produce a list of potential nodes for next **VelasCycle**;
4. Collect all **BlockSigns** from the previous **VelasCycle**;
5. Produce a Merkle Tree from **BlockSigns** that will result in a **VelasSeed**. **VelasSeeds** are used to synchronize the random function between all nodes in the **Velas** network. The algorithm is deterministic;
6. Using the time periods of a **VelasCycle** and block time the number of blocks for next **VelasCycle** is calculated. e.g. **VelasCycle** – 20 hours, block time – 1 sec; 20hrs * 60 min / 1 sec = 72000 blocks per **VelasCycle**;
7. The **VelasSeed** is used to randomize the function;
8. The final step includes calling the random function as many times as blocks in next **VelasCycle** to synchronize all nodes;

The selection criterion for nodes to be chosen to create new blocks would be initially solely based on the amount of **Velas** staked. Therefore, the higher amount of coins staked the higher the chances a node will be selected to create a block and receive Velas (VLX).

For example, a node holding 2,000,000 **VLX** has 2x the chances of being selected over a node holding 1,000,000 **VLX**.

Please note that the majority of **Velas** stakers would be achieved by accumulating 51%. This is the minimum needed to achieve consensus on the Velas Blockchain.

The **VelasCycle** is allowed to skip SimpleBlocks. Minimum of 51% of blocks are required to be validated in a **VelasCycle**.

Approach to Consensus

AIDPOS is able to successfully address many of the shortcomings and limitations of existing alternative options. 2,000,000,000 coins are pre-mined as starting network point to swap CoinpaymentsCoin (CPS) 1:1 to Velas (VLX).

1. When forming the blockchain, all coins are shared among the service organizers.
2. Coin emissions are performed when a new block is formed and a cycle-block ends.
3. Issued coins are credited to someone who has created a block.

Verification of transactions is executed through “staking” (holding) coins on the affiliated network. The staking of Velas (VLX) builds a trusted network of validators that will process and forge a block of transactions to the chain. In essence, it is the amount of Velas (VLX) staked that brings consensus to the current state of the blockchain.

Participants will be provided with easy to use wallet software to stake coins. Validators are compensated by collecting network fees over a period of time in exchange for staking. The larger the position staked, the more Velas (VLX) is assigned. Additional software option will be available for those participants who have a dedicated GPU in order to spin up neural network and get rewarded for their contribution and training of the AI. As a result, the PoS approach incentivizes sizable, long-term investments combined with correct validating behavior and rewards, providing a reliable and sizable stream of coins, with minimal overhead or advanced knowledge of coding languages.

Velas’s Artificial Intuition DPOS Algorithm

Artificial Intuition Delegated Proof of Stake (AIDPOS) - is used to secure Velas blockchain. AIDPOS attempts to solve the problems of alternative consensus option, including Bitcoin’s traditional Proof of Work system and the Proof of Stake system of Peercoin and NXT.

Velas AI

An Artificial Intuition is a series of algorithms used to identify relationships and patterns in a set of data. The network can adapt the inputs so that it generates the best possible result without having to redesign the output criteria.

Selected technical parameters of the Velas system include

- Transactions per second: > 30000;
- Blocks per second: 1 sec - 2 min dependant on calculation from AI algorithms);

The block time depends on network load (TPS). If the network has many transactions per second, the time blocks will be short. If the network has no transactions, the block time will be long. In case an empty block is generated, it includes nothing but a block header without body.

The algorithm powering AI will be derived from the historical data of the following:

- Numbers of VelasNodes
- Transactions per VelasCycle

Special transactions of the VelasCycle
The neural algorithm will optimize the following parameters:

Velas node network

- Block size
- Block time
- Increase TPS

The optimized parameter would be the total amount of the staker’s transaction commissions.

Velas Rewards

A reward is provided to a node/block generator for active and correct participation in the system and depends on the amount of earned points.
Algorithm changes are done dynamically, taking into account the following parameters:

- Time per VelasCycle
- Time per block
- Transactions per block

Node Rating Scoring Algorithm.

Uses importance proof algorithm. The key parameters used later for optimization include:

- The number of transactions of a node, with their quality taken into account. Fake transactions lead to point subtraction and real transactions are rewarded with additional points.
- An account balance. Additional points are assigned according to the number of staked tokens. Advisor Nodes (Block Producers) estimated would need minimum 1,000,000 VLX staked at early stages.
- Time spent online. Additional points are assigned according to the duration of the total uptime of a block generation node.
- Block generation events. Each block generation node receives points for each generated block.

If a node did not generate a block for any reason (the lack of processing power or lack of uptime - due to network issues, for example), a point is taken away from the node. When forming a block, the list of verifiers is generated as well. The verifiers are added to the list taking into account received points.

Data specified above goes through the neural network and then we receive data of the **y'=f(x)** objective function. Each **y'** goes through the softmax layer. After this, we receive the percentage of deviation from maximal quality.

A reward for block generation depends on the percent of contribution received in the neural network.

Reward Coins

We equate a reward for blocks with 100% and divide among participants under the points-percent that they received in the neural network.

Artificial Intuition (AI) would be based on a linear regression model. The model is trained by stochastic methods. The AI is based on two models. A multidimensional linear regression model is used to calculate the probability distribution density of a class and the Bayes classifier uses a posteriori maximum estimate to determine the correct decision.

Due to adaptation in a super-complex environment with a huge flow of incoming data, AI based on genetic algorithms is used since it will be able to handle many times more efficiently than the standard neural network with the back-propagation method of error.

Genetic algorithms are used to solve optimization problems using the evolution method, i.e. by selecting from a variety of solutions the most appropriate. They differ from traditional optimization methods in the following properties:

1. They process the coded form of the parameters of the problem rather than their values.
2. Searching for a solution is based on a certain population.
3. The objective function is used, rather than its derivative.
4. Algorithms are stochastic.

Genetic algorithms for teaching neural networks are used as an alternative to the back-propagation error method. The purpose of training is to minimize the cost function. Also, the use of a genetic algorithm allows us to avoid the cost function in local minima.

It should be emphasized that the back-propagation algorithm of an error, as a rule, is performed faster than the genetic one, which scans all the many possible solutions. However, the gradient method does not always lead to the expected result, which depends on the choice of the starting point. In addition, a fundamental flaw in the back-propagation method of error is the “jamming” in local optima. That's why Genetic AI is a much more innovative and promising method for learning of neural networks.

Details on Algorithm Implementation

The Artificial Intuition algorithm for determining the rating of the node and the algorithm for determining the number of blocks in the age.

AI Algorithm for determining the number of blocks:

Each block is formed from the established number of transactions. This number depends on the intensity of use of the blockchain. If, during block formation, unconfirmed transactions remain, then it is necessary to increase the number of blocks in age by reducing the time between blocks.

When forming a new cycle of a block, it is necessary to check the number of unconfirmed transactions in previous ages, by determining the standard deviation at each epoch period using the determinant of the growth function. Next, we calculate how many transactions should be in the block, by counting transactions with calculated deviation. Then, we calculate the required number of blocks and the time of the formation of one block.

The AI Algorithm for determining node rating:

The node block generator will get the point in case of confirmation of the formed block. If a node has not formed a block for various reasons (lack of computing power or a problem with the network), the point is subtracted from the node. When forming a block cycle, a list of advisors is also formed. Advisors add to the list with the received points.

Artificial Intuition determines the rating in the voting list.
Input parameters are:

a) The number of node transactions

b) Account balance

c) Network time

d) Point for the formed block

The training sample includes errors, delays, and unconfirmed transactions when creating a block in previous cycles.

The result of the AI should be a rating in the list of candidate nodes.
These four parameters are the input data, on which we will classify the traffic.

DNA is layers of the neural network (matrix) which will be calculated on separated nodes. A few more input parameters can be added:

1. The number of unconfirmed transactions left after a block forming
2. A node participating in transaction checking (protection against the 51% and double-spent attack)
3. Truthful voting and a transaction checking speed, when a node is inside a cycle-block.

Application of AI for Velas

The purpose of utilizing AI on the Velas Platform is to reduce the cost of consensus.

As a result, the AI framework on top of Velas Platform:

1. Motivates participants of the network (nodes) to be reliably present and active in the network, maximizing relevant scores/rewards.
2. Blocks fake messages about false transactions and thus increases the quality of messages and the resistance of the network to attacks.
3. Forms timings of age, thus accelerates TPS and reduces the general computational network workload. In other words, it’s about dynamic time to form blocks coupled with allocating block formation to nodes having higher computing power during the high workload periods.
4. Correctly and optimally allocates rewards.

Protection against the 51% Attack:

We use the DPOS algorithm. The age lasts 24 hours.

When creating an age, a cycle-block in which nodes having more weight are selected is created. The nodes that were selected to a cycle-block leave their stake.

New blocks must be signed by the 80% of cycle-block nodes. Thus, to break into the system, the intruder needs to get into a cycle-block with more than 80% participation and create fake nodes that will be more than 80% of the whole system. Under these circumstances, intruders own the whole system, and it does not make sense for them to rob themselves.

Velas Node Selection

The aim of Velas node selection is to compute the matrix of the neural network.
The necessary condition is that the GPU is "on board".
The network organizer sets several nodes.
Inputs for a matrix computing are received from the blockchain.

The algorithm is public, so each network member can:

1. get inputs;
2. get finished matrix;
3. do control calculations.

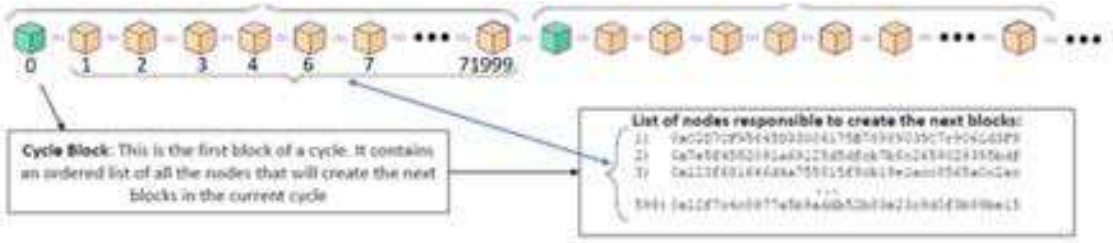
Based on the calculations of the neural network, a node forms a cycle-block with timings and sends it to all network participants.

A node stores a stake of node-participants of a cycle-block.
By calculations of the neural network, a node allocates a reward after the end of an age.

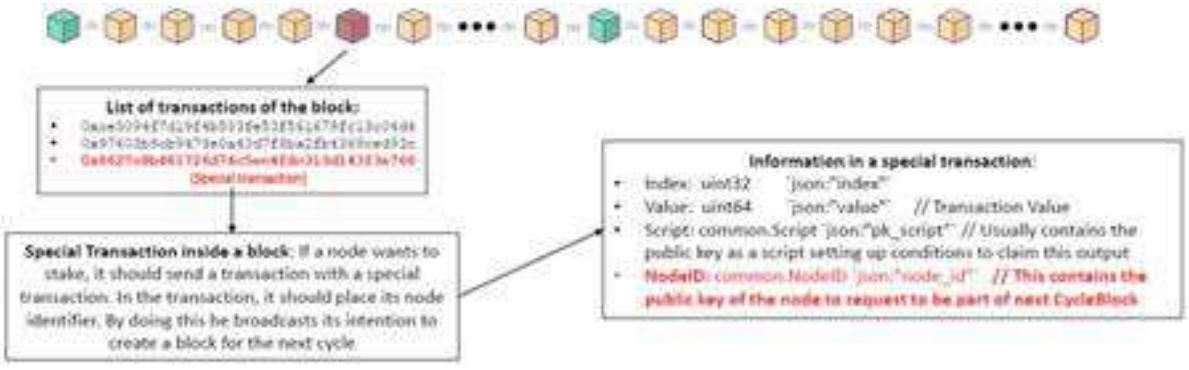
If there is an attack detected, a node dissolves a cycle-block and creates a new one.

Cycle Block Structure

Cycle time = **72001 blocks** (72000 **SimpleBlocks** per cycle + **Cycle Block**)



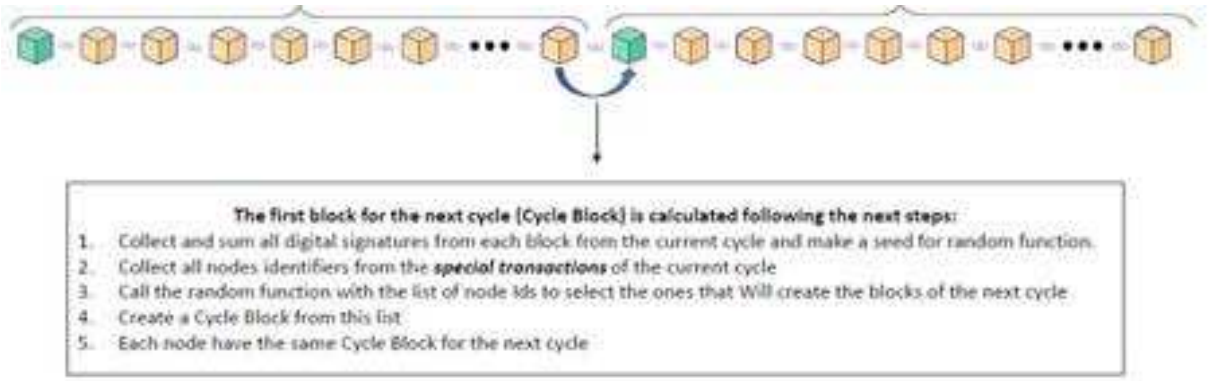
A node entering the network



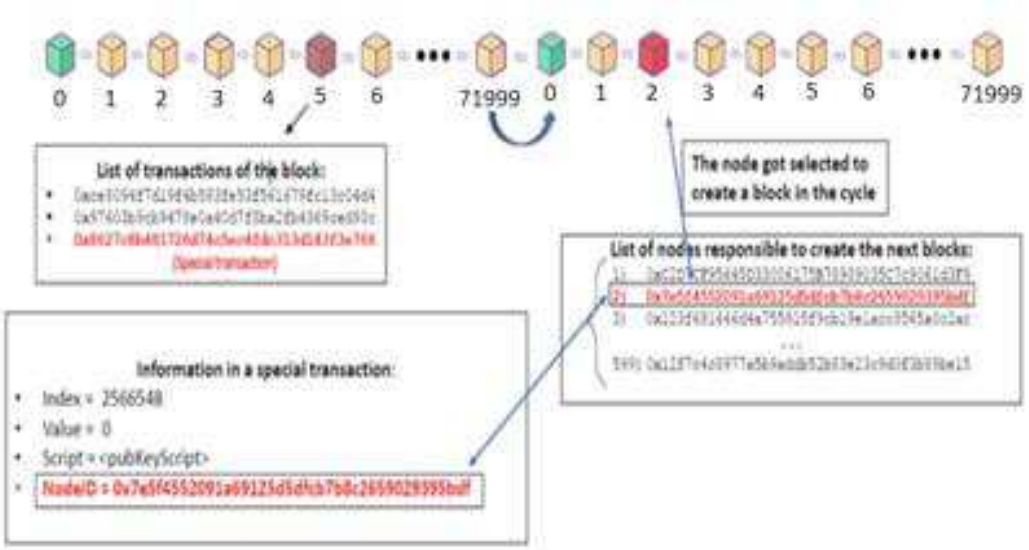
All nodes attempting to stake must send a special transaction to the network to register itself as a block creator in the following block cycle.

End of a block cycle

A node is selected to create a block



Transactions on Velas Platform



Description of the Transaction Model

The transfer of Velas coins on the Velas platform is executed by the re-issuance of the public key for the new coins to the subsequent owner while preserving the hash from the previous transaction. The verification of transactions is accomplished by validation of the chain.

Challenges, such as preventing “double-spending”, are solved with the help of a network verifying the chain’s authenticity. This is done through the introduction of the protocol “Epoch”, which provides an additional level of security, allowing blocks to be added to the blockchain within a specific timeframe. The blocks are public and available to be viewed and examined on the block explorer. Each block contains the hash of the previous block and a timestamp. The addition of each timestamp strengthens the validity of the whole chain.

Transactions per second

Example:

- Block time – 2 sec (1sec-2min);
- Transactions per block – 60,000;
- Transactions per second 60,000 / 2 = 30,000 TPS;

Transaction Process

In this section, we summarize the key details of creating and processing a Velas transaction.

1) Each transaction has the following parameters

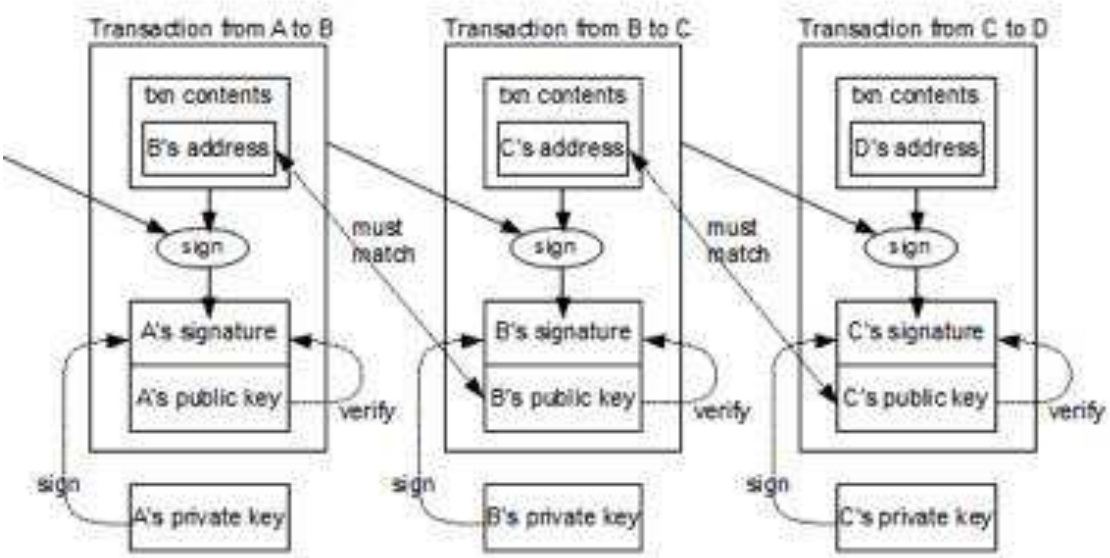
- Transaction hash
- Blockchain transaction type
- Block number or timestamp at which the transaction is unblocked
- Number of used transactional inputs
- List of used transactional inputs
- Number of created transaction outputs
- List of created transaction outputs

2) All values for transaction inputs are verified before they are processed.
Specification of the required parameters must be reviewed. For example, commissions cannot be less than or equal to zero. If the transaction in question is not confirmed, the transaction will not be processed.

3) It would be necessary to initiate and process the transaction verification procedure. This verification should ensure that the following events have occurred:

- New transaction creation has been completed
- New transaction identifier was generated for the coin
- Signature of the new coin by the owner is obtained
- Encryption of transaction data placed inside the message instructing the network nodes to process the transaction was performed
- Transmission to all nodes of the network has been successfully produced and recorded

Transaction Structure



Transaction Confirmation

All Velas transactions are considered unconfirmed until they are included in a valid block.

Recently created blocks are distributed to the network by the node that creates them. Since new blocks are added to an existing chain of blocks, each additional block adds one more validation of transaction confirmation. A transaction that is sent to the network but not included in the block remains unconfirmed. Transactions are prioritized based on the size of their related fees.

Transaction Cost

When a combination, division, or re-issuance of a coin is added to the block, all transaction fees related to the block are distributed among the nodes in the network. Members of the committee receive rewards in the form of Velas from all transactions in the blocks in the order they were elected by the network.

If the size of all transactions in the block does not exceed 1 MB, minimum Velas (vlx) will be sufficient to cover all charges related to processing. In situations when the number of unconfirmed transactions exceeds the number that can be placed in the block, the transactions with the highest commission will be selected by the node.

Transaction Hash Generation

The transaction- and block hashes are generated using the Schnorr signature algorithm. We will later discuss more our reasoning, the benefits and workings of this algorithm. Velas will always support secure multi-signature transactions.

- Version
- Locktime - 0
- Transaction input
- Hash–Transaction hash-
- Index (number of outputs when the coin was formed)

Value (coin volume in CCN)

- Signature script
- Signature
- Transaction output

Coin Structure

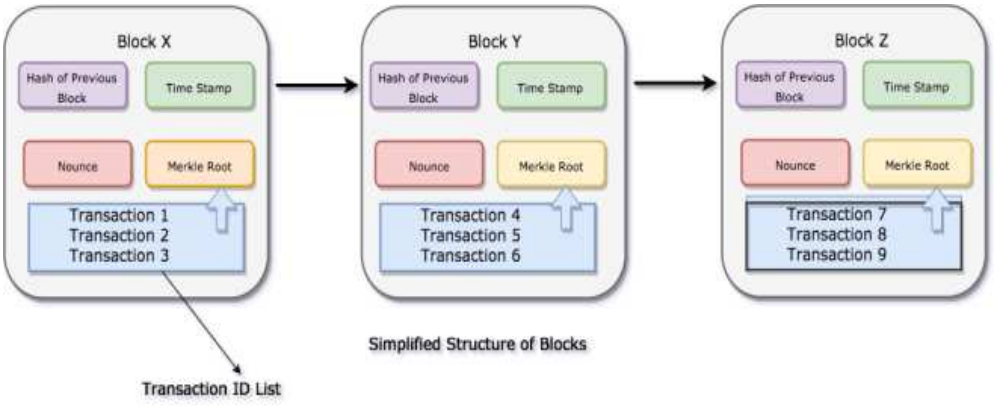
A coin is formed in the transaction process and has the following structure:

- The secret key (generated for that transaction)
- Transaction hash (exists from the moment of creation)
- Index (Transaction index)
- Units (Amount of coins)

Each block must be confirmed by the members of the committee before being admitted into the blockchain. Each member verifies all transactions placed in the block, confirming the following data: transaction amount, inputs, digital signature, and outputs. If successful, the signature is broadcasted to the rest of the network for final verification before incorporating the transactions into the blockchain. To participate, a node must inform the network of its intentions in order to receive a unique coin used to store remuneration data for work completed during a lifetime. Criteria for the selection of new committee members are as follows: activity time of the node in the network, size of the wallet, and frequency of participation in the committee.

The block header consists of the following parameters:

- Height (height of block)
- Size (size of the block)
- Version (blockchain version)
- Previous block hash
- Timestamp (current time in seconds)
- Bits (command window)
- Nonce (command window)
- Number of transactions in one block



Simplified Structure of Blocks

Velas Block Generation Node (BGN) Details

Currently nodes in the network are divided into two types:

- Block generation node (BGN);
- All remaining nodes.

When connecting, the node makes a request to DNSSeeds and receives a list (slice). The BGN makes a request to any address and receives the list (node_slice) of the BGN branch.

The node makes a request to any node from the list and tries to connect to it. If the connection is successful, the node is registered. If the node does not have free slots, it gives a list of slave nodes to which a connection request can be made, otherwise, it will connect to any possible slave nodes to register itself.

Moreover, searching through the whole “tree”, this cycle continues until the new node finds a free slot for connection.

The node synchronizes and makes the request: sync()

Furthermore, if the node is not behind the NAT, it launches a server to connect seven remaining nodes. The node sends a ping to determine the path length. The node is always connected to TCP-server. If the node loses the connection without notification, it loses the points as well.

There are two types of messages in the network:

- From a node to master and back;
- The messages not getting through slave-nodes.

Each additional node signs the hash of the previous structures and sends it until the ping reaches the BGN.

These records have the following structure:

```
{
        address_node,
        hash,
        sign,
}
```

The BGN signs the hash and sends it back. At that time, the node records hop to the BGN. When voting occurs, the node gives scores to the master based on hops as well as other parameters discussed earlier.

If it is necessary to connect a new node, the node responds with a free slot and amount of the hops to the BGN. If there is no free slot, the node responds only with “inability of connection” and gives the list of slave-nodes.

How Does the Node Control the Connection?

The basic setup is that it is completed by sending the “notify” command to a masternode, and the masternode must respond “OK”.

Every two minutes, the node must receive a new block from the masternode. If the node did not receive the block within the specified time frame, it can start to search for a new master.

If an attempt to send the block to the slave-node fails, and the node receives an error from a specific node and connection is not closed, the node closes the connection by itself.

If the masternode is unavailable, but other nodes are available (i.e. successfully respond to connection attempts), the node gives penalty scores to this masternode.

If the node is behind the NAT, it sends “notify” every 15 seconds. A proxy-client node cannot be the BGN or the masternode.

Transaction Process Overview

Transaction spreads from the slave-nodes to the masternode. The masternode verifies transactions, signs them and forwards them. The closer the node is to the BGN, the more transactions need to be verified.

If the node does not verify the transaction, the node does not send it and waits for a moment when computing power will appear.

If the transaction is incorrect, the node sends the notification about an error to its slave-nodes.

The transaction is forwarded to the masternode once per second. In the intervals among the sends, a block from transactions is formed. It is necessary to reduce the load imposed on the network as the larger the block, the less load is on the network.

The complete block (chain’s block recorded by all BGN) spreads down from the masternode to the slave-nodes.

Choice of the BGN

Regarding the length of the path to the BGN, each node votes for the master by points. The highest amount of points is 10, which is 1 hop; the lowest amount is 0, which is more than 10 hops.

Receiving evaluation from the slave-nodes, the masternode gives points to its master and adds the slave-nodes’ size.

The number of transactions, which were generated by the slave-nodes, is added to the size of a node.

Total Number of Points

The masternode verifies the correctness of data given by the slave-nodes points. If an error is detected, points of the branch will not be taken into consideration; the points of the slave-node, in which the error was detected, will be zeroed out.

Accounting of the number of transactions is necessary in order to limit the lifting of nodes (miners) not involved in the generation of transactions up the “tree”.

The node signs a vote and sends a message. Then, the points for each pair of nodes are summed up and processed.

Receiving the final message, the BGN sends it to other BGNs. After voting, the nodes having the highest score are chosen. The list of the new BGNs is formed. The BGN verifies the signs in the selection chain.

Voting

All BGNs form the list of node-candidates. If the algorithm works properly, the list will be the same for everyone.

Each BGN verifies the list and if it conforms to its list, the BGN signs it. The winning list is the one for which more than 80% of the BGNs voted.

BGN (Block Generation Node)

The node “is activated” at the beginning of an “age” and brings a BGN socket up. This node sends a greeting to other BGNs as well.

The type of relationships among the BGNs is “all-to-all”. The node creates a block by the schedule indicated in a cycle-block.

Receiving a transaction from slave-nodes, the BGN forwards it to all BGNs.

The node generating a block gathers all unconfirmed transactions and creates a block. The block is forwarded to all BGNs.

The BGNs verify the correctness of the transactions and sign the block. A block is considered as accepted if it was signed by more than 80% of the BGNs. As a result, this accepted block is spread down the “tree”.

At the beginning of each “age”, the voting for the next BGNs starts. Along with transactions, the BGN receives votes from their slave-nodes and verifies the conformity of number of transactions and the size of a branch. If there is unconformity, the whole branch loses points.

After the verification, the BGN forwards voting results to other BGNs. This way, the list of node-candidates is formed.

In the middle of the “age”, the list of the new BGN “age” and new cycle-block are formed. The list is signed by all BGNs (80%) and forwards down the “tree”.

The Node

A node registers in a block and sends the “register” message if this node was voted for by its slave-nodes (linked to it). The list of relevant parameters includes:

```
{
        address
        weight
        hash []addresses slave nodes
        transaction []hashes
}
```

A new node is added to a blockchain. The blockchain consists of six “ages”. If a node is not active for this period, it loses all points and registration as well.

Wallet (Coins)

The aim of the wallet is to create transactions. It executes requests via its own node. The wallet is created by the special application and can be passed to any other application. The wallet balance is the number of unspent transaction outputs.

User Facing Application

The application is utilized as the storage for wallets (coins). It must be linked to a node.

Security

Protection against the 51% and Double-Spend Attack:

We use the DPOS algorithm. The age lasts 24 hours.

When creating an age, a cycle-block in which nodes having more weight are selected is created. The nodes that were selected to a cycle-block leave their stake.

New blocks must be signed by the 80% of cycle-block nodes.

Thus, to break into the system, the intruder needs to get into a cycle-block with more than 80% participation and create fake nodes that will be more than 80% of the whole system. Under these circumstances, intruders own the whole system, and it does not make sense for them to rob themselves.

Three Types of Nodes:

- **Master nodes** (from three nodes, three or more nodes are required to guarantee network availability at any time!) – belong to a network organizer. Master nodes functions will transfer to Advisors in Stage 4 release candidate version of the network in order to decentralize AI functions.
- **Advisors** – block producing nodes. In release candidate Advisors will signal to network they have sufficient GPU power and will be selected by the network based on their reputation, uptime, computation speed and coin holdings. This if selected will allow them to run the neural network and get greater reward from the network, helping further train and decentralize the AI component of the network logic.
- **Typical nodes**

All advisors check transactions when generating a new block.

If 80% of the advisors verify a block, it goes to the network.

If 20% of typical nodes defines a block as a fake, masternodes check this block.

The rate is one of the criteria for selecting a node to a cycle-block.

If the 80% of the advisors voted for a fake block and then 20% of nodes did not accept this block the block is not added to the blockchain, and, after the verification of a block by a masternode and confirmation that the block is fake, the nodes, which reported an attack, receive a reward, which was formed by the rate of advisors.

This is our solution to “nothing-at-stake” problem.

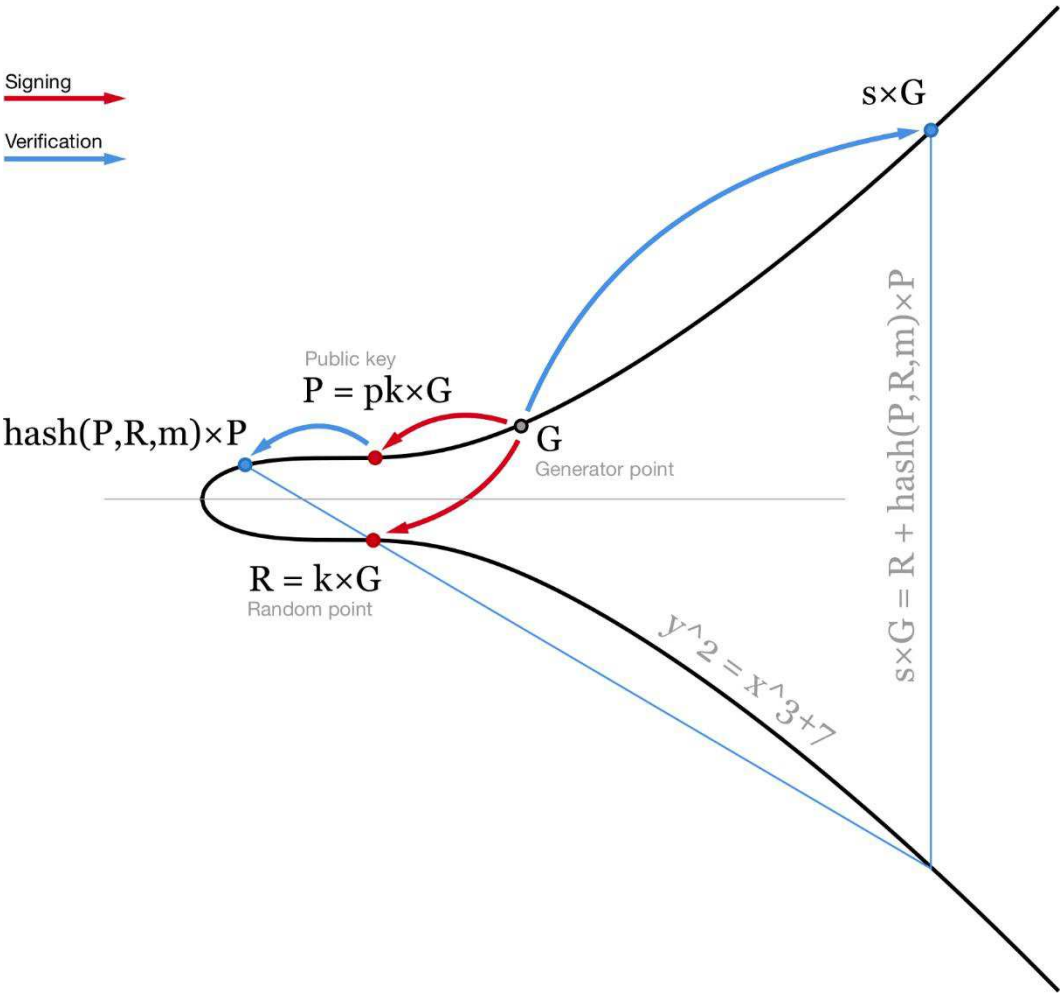
Hash Algorithm

We employ Schnorr signatures instead of ECDSA. Velas implementation will use Secp256k1 curve due to it’s predictable nature and hence better performance. The biggest benefit of using Schnorr signatures for Velas blocks is the sizable performance gain.

Due to the linear nature of Schnorr signatures (‘sum’ of signatures is equivalent to the signature of the ‘sum’), we can batch validate signatures of all transactions/inputs in the block.

Even better, we can store only a single signature that aggregates all signatures for the block, with trivial computation. This results in less storage overhead, and much quicker verification. After all, the node now only has to verify the block signature against the signature of all transactions. Two relatively simple calculations instead of potentially thousands.

Using Schnorr algorithms instead of ECDSA also allows for key aggregation and improved privacy, with which it will no longer be possible to discern multi-signature transactions from regular ones.



Visualisation of the ECDSA algorithm. Elliptic curve is plotted over real numbers for illustration purposes.

Credit: https://medium.com/cryptoadvance/how-schnorr-signatures-may-improve-bitcoin-91655bcb4744

Man-in-the-Middle Protection

This system is protected against MITM attack in the following way:

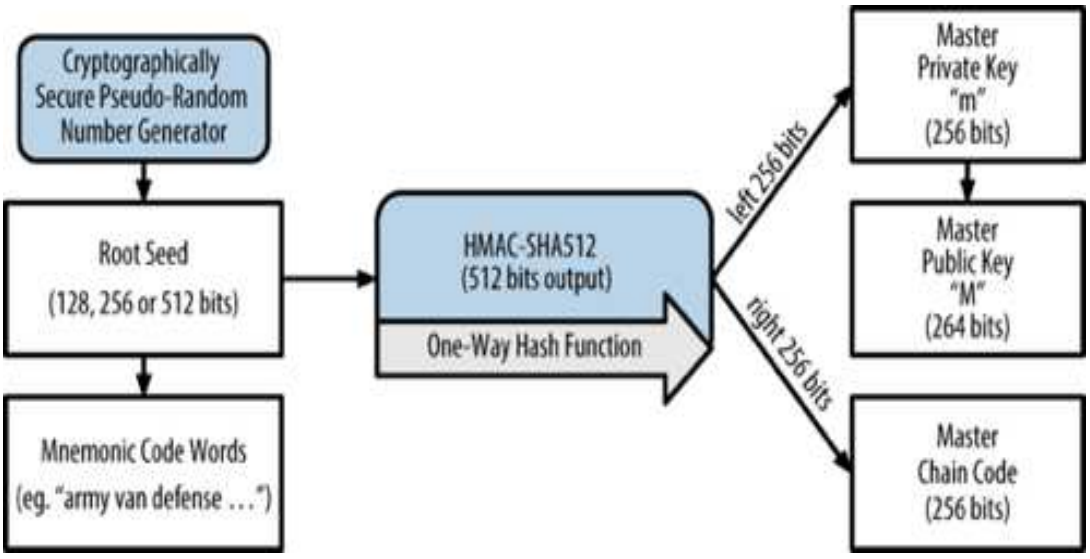
- Each node public key is stored in the blockchain.
- Also, there is an Alias (name) - public Key bunch in the blockchain.
- When sending a container from subscriber A to subscriber B, subscriber A receives the public key of subscriber B in the blockchain and encrypts them the container. It is sent to a multi-wallet of subscriber B in an encrypted form. Subscriber B decrypts the container with his private key.

Multiwallet Technology

Velas Blockchain allows creation of public and private containers for all coins supported. Velas will generate private keys for cryptocurrencies, bitcoin, ethereum and tokens, eos, xrp, monero etc. Keys will be created using users private key seed. All containers will be restored from same original seed or private key associated with Velas wallet at time of creation. These containers will allow to create an onchain offchain scalability solution for all other coins as needed and serve as one ecosystem wallet for all Velas smart contracts. Users can also benefit from storing all coins they currently hold securely in Velas wallet and create multisig backups for coins which do not natively support it.

Transferring a Container storing Private Keys.

1. From the list of nodes, the node that can work as a proxy is chosen. For instance: a flag is set, and a node has a public address.
2. Then, subscriber A sends the node address to subscriber B.
3. Both subscribers connect to a proxy server via a named channel. On the proxy server, the map, in which key is the channel name and value is a structure from two connections, is created.
4. Once the connection is done, subscriber A sends his/her public key to subscriber B.
5. Subscriber B verifies the address of subscriber A and sent a public key. If they coincide, subscriber B, owning the container that contains private keys, encrypts it with the public key of subscriber A and transfers the container to him/her.
6. Receiving the container, subscriber A decrypts it with his/her private key.
7. The connection is shut down.


Cryptographically Secure Pseudo-Random Number Generator
Root Seed (128, 256 or 512 bits)
Mnemonic Code Words (eg. "army van defense ...")
HMAC-SHA512 (512 bits output)
One-Way Hash Function
left 256 bits
right 256 bits
Master Private Key "m" (256 bits)
Master Public Key "M" (264 bits)
Master Chain Code (256 bits)

(end)