Essentia blockchain

Nomenclature

- **Validator** - a participant in the PoS-chain consensus system.
- **Active Validator Set** - those validators who are currently participating in a shard.
- **Jury** - a randomly sampled subset of the active validator set.
- **Proposer** - the validator that creates a block.
- **Witness-** a validator that is part of a jury that needs to sign off on a block.
- **PoW Chain** - the main PoW chain.
- **PoS Chain** - the central PoS chain that is the base of the sharding system.
- **Shard Chain** - one of the chains on which transactions take place and account data is stored.
- **Crosslink** - a set of signatures from a jury attesting to a block in a shard chain, which can be included into the PoS chain. Crosslinks are the main means by which the beacon chain “learns about” the updated state of shard chains.
- **Slot** - a period of 8 seconds, during which one proposer creates a block
- **Decade** - a period of 8 seconds, during which a Node has the ability to create a block and some voters have the ability to validate.
- **Century** - a period of 64 slots.
- **Chain pair** - consists of a valid PoS-chain and its corresponding PoW-chain.
- **Random oracle -** a kind of hash function, such that we know nothing about the output we could get for a given input message, until we actually try it.

Preamble

There are many different blockchain architectures and each has been made with a particular focus in mind. Sharding in Essentia creates a whole set of “Essentia’s” that run side by side and can transfer data between them. This allows for significantly increased throughput and security. Blockchains based on a top of third-party "PoW base", however do not enjoy these benefits. Essentia embodies the technology and tools that allow user, developers, and organizations alike to safely use cross-channel transactions between Essentia and other

blockchain projects using smart contracts and decentralized applications. The Essentia PoS mechanism combines PoS consensus and Byzantine fault tolerant consensus theory.

The modular structure of Essentia allows you to modify and expand the functionality of the current blockchain. This feature will allow Essentia to always stay up-to-date with the latest technological developments. Essentia uses a decentralized Kademlia DHT network architecture and a Zero-Knowledge authorization protocol, which avoids common blocking attacks (51% attack and cartels), as well as DDoS attacks on DNS-Seek nodes and MITM key exchange during authorization.

Essentia's performance is powered by a deep modification of the algorithms using hardware and system which is the processing of one data element for one instruction (SISD), and the processing of several data elements for one instruction (SIMD). Also, data structures are optimized and data is split into shards.

The Design of Essentia

Essentia is a modular decentralized system of interactions and data management [1], which is based on a hybrid PoW/PoS consensus protocol.

The use of a hybrid system provides greater safety and stability. Large mining pools have zero chance of gaining too much influence over the network (i.e. a 51% attack) as the extracted block is checked and signed by randomly selected PoS voters. This approach will detect and prevent unwanted behavior by bad actors.

A conspiracy between PoW and PoS miners to overtake the network will also be impossible, since voting cards are selected only after the block is created. A conspiracy between PoS miners themselves is also highly improbable, since the choice of PoS for voting occurs randomly using a crypto-stable generator. The likelihood that the system would randomly select several tickets belonging to one group, from the large pool of Essentia PoS miners, is negligibly small. Even if this were to happen, each transaction needs to be checked **at least twice**.

Essentia consists of three main components:

1. **Essences**: contain grouped entities and associated data and services. They can consist of individuals, companies, groups or organizations. Interaction between them is carried out with the help of subIDs, permissions and smart contract self-amendment;
2. **Synergies**: allow interaction between various platforms, resources and modules of the system. This component is governed by PoS-consensus.

3. **PoW Miners**: create new blocks. Can be grouped into mining pools through the Supernode mining pool service.

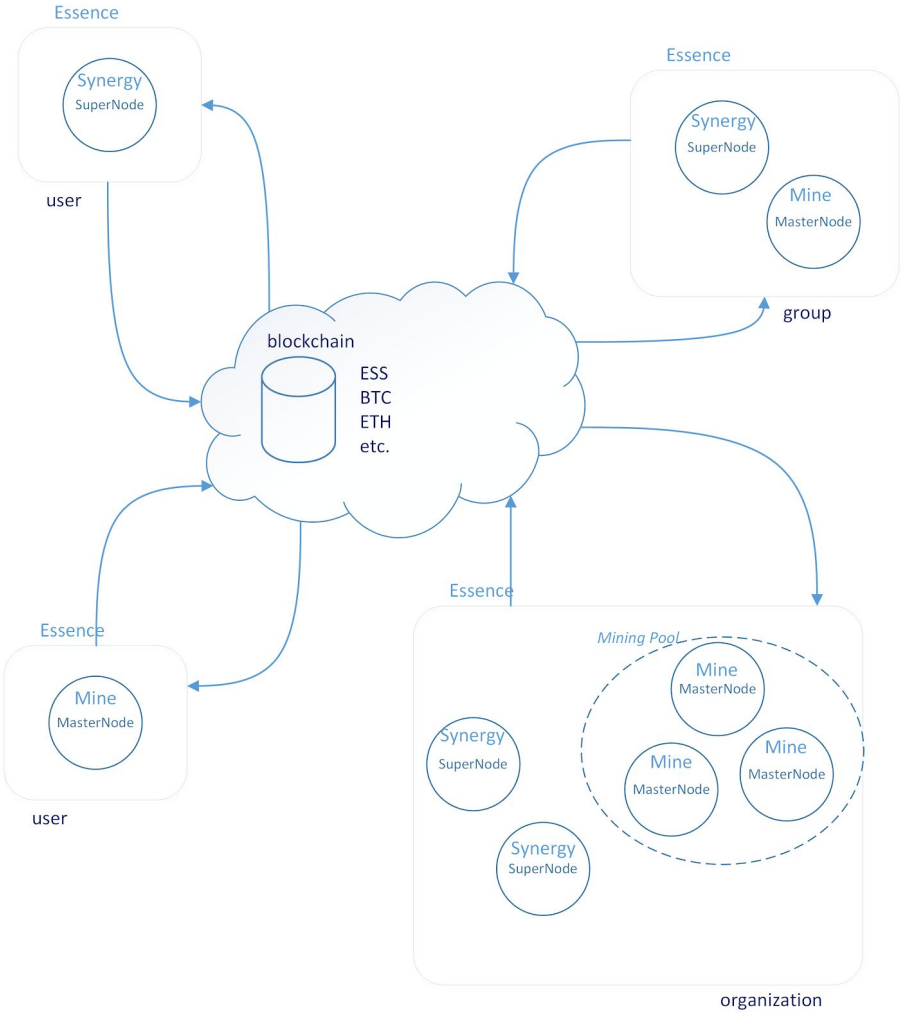

Fig. 1. A schematic representation of the general structure of Essentia

Entities interact with each other both within the framework of the blockchain and within the shards. Entity interaction takes place at the decentralized network level via API.

Entity group is a separate subnet within the Essentia core network. Figure 2 shows an outline for grouping Essences. The group does not interact with the main chain, but rather within the shard. Management of an entity group is carried out using a smart contract deployed in the main chain. The group is connected to the main chain block through the root entity which contains a list of nodes that forms the group.

Creation of a block and voting for a block occurs in the same way, except that when selecting the root entity, the next step is to select one node for voting among the available nodes in the group.

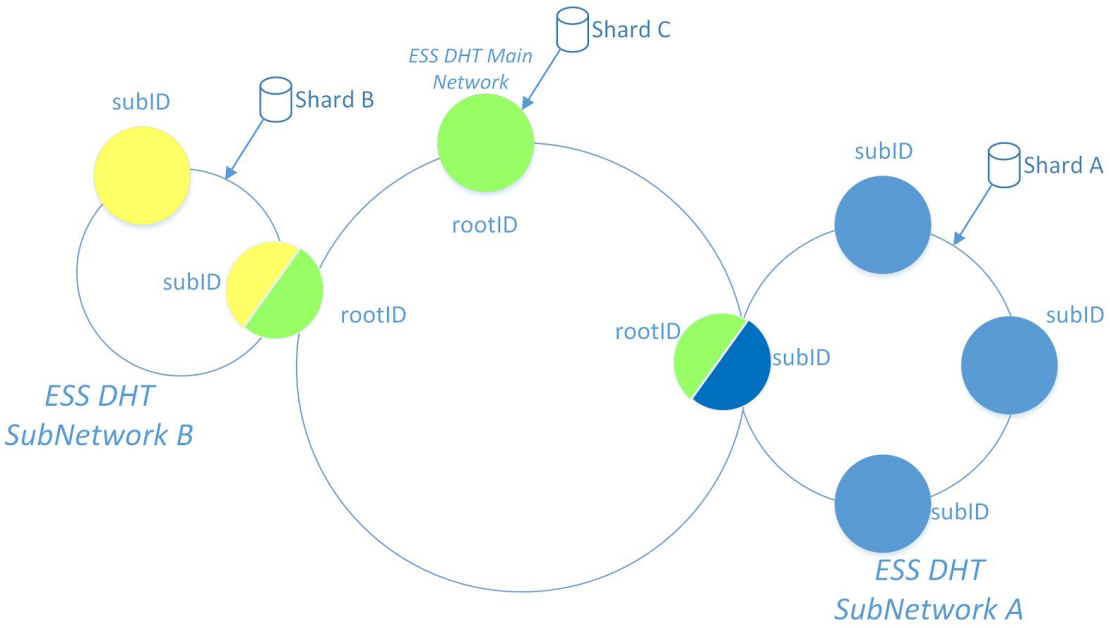

Figure 2. Grouping entities

Main components of the Synergy component :

1. ESS-ID: the Identity core of the Essentia Framework. It works like an HD wallet and can be created or managed via software or dedicated hardware devices. It has optional multi-signature features and capabilities.
2. ESS-CORE: the software or hardware-implemented core-module that allows the ESS-ID to perform its fundamental and basic operations.
3. ESS-Modules: implements interfaces for interaction with applications layer.
4. ESS-Base: implements functions to create/manage/restore/use/ the Root-ID by dialoguing with the CORE.
5. ESS-Home: connects ESS-Base and the ESS-Core functions to all of the other modules and components along with top-level user interaction.
6. DApps: decentralized applications.
7. Services: ESS services such as Payment, Config etc... which are implemented for providing additional functionality.

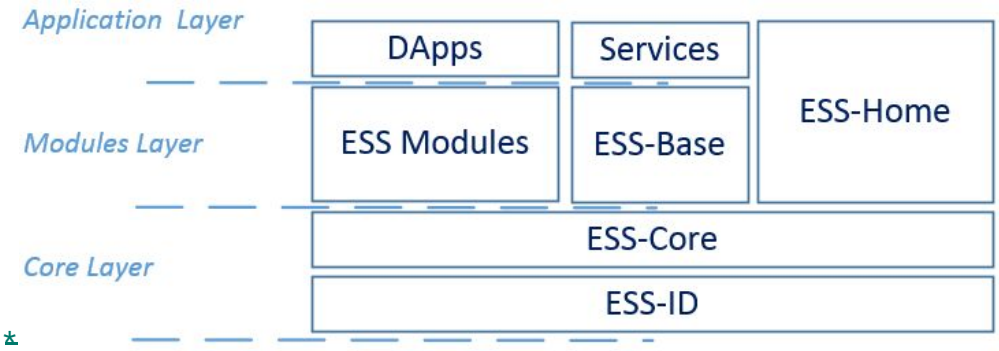

Fig.3 Synergy layered architecture

The Essentia Mathematical Model

Following Canetti's formulation [1, 2], the abstract model of the PoW / PoS hybrid protocol is defined by the formula:

$$\mathbb{P} = (\mathbb{P}^S, \mathbb{P}^W) \text{ where}$$

$\mathbb{P}^W$ u $\mathbb{P}^S$ is the result of code execution, respectively, on the PoW-Miner and PoS-Voter. The protocol $\mathbb{P}$ contains a set of functions F, inherent in the protocol.

$$\mathbb{P}^S \in \{F_s, F_{net}\},$$
$$\mathbb{P}^W \in \{F_w, F_{net}, F_{rng}\} \text{ where}$$

F_w - PoW-consensus, F_s - PoS-consensus, - F_{net} network function, F_{rng} - function of PoS-voters, CSPRNG-generator.

We consider that it $\mathbb{P}$ is controlled by the environment Z (1), which consists of n_w PoW-miners and n_s PoS-voters. Then, the execution of F in each group is initialized by the initial state s_0. This state includes all initial public information of the protocol $\mathbb{P}$, for example, the genesis block. The environment Z activates the participants ($node$) corresponding to the protocol group.

It should be noted that we are considering an idealized model of the system, where honest PoS-voters have the same shares and honest PoW-miners have the same computing resources. In reality, the shares and processing power are different. Therefore, we consider an autonomous static model that does not affect the overall concept and treats PoS-voters and PoW-miners as some arbitrary number of corresponding idealized models.

Essentia blockchain

First, the PoW-Miner creates the PoW-blocks in proportion to its computational capabilities, namely, when receiving the message, it selects the best chain-pair, and then, using the processing power, solves the PoW puzzle and sign the block. Next, the PoW-Miner randomly selects five PoS voters and transfers block to the selected PoS-voters for signature. The selected PoS-voter with the identifier from the previous PoW-block and the new PoW-block generates a PoS-block and then adds it to the shard chain. Note that for each PoS voter, the probability of being selected depends both on the CSPRNG and the availability of valid tickets.

The whole process consists of three states:

1. ***Work***: each PoW-miner Fw_i , $1 \le i \le n_w$ with the current state s_i performs the following:
    a. Receives the message m from the set of participants $N \in \{Fw_1, \ldots, Fw_{n_w}, Fs_{n_w+1}, \ldots, Fs_{n_w+n_s}\}$ from the F_{net}
    b. Execute function F_w
    c. Gets the result γ_w
    d. Executes the function F_{rng}
    e. Gets a list of ν participants (PoS voters): $\nu = \{F_{s_0}, \cdots, F_{s_5}\}$
    f. Executes the protocol $\mathbb{P}^w$ with the input parameters: the
        i. result of γ_w the function execution the F_w
        ii. result ν
        iii. message received m
    g. Updates the state s_i and the message m': $\mathbb{P}^w(s_i, N, \nu, \gamma_w, m) \rightarrow \{s_i, m'\}$
    h. Message of the message m'
2. ***Stake***: each PoS-voter Fs_i , $n_w + 1 \le i \le n_w + n_s$ with the current state s_j performs the following:
    a. Receives the message m from the set of participants $N \in \{Fw_1, \ldots, Fw_{n_w}, Fs_{n_w+1}, \ldots, Fs_{n_w+n_s}\}$ from the F_{net}
    b. Executes function F_s
    c. Gets the result γ_s
    d. Performs protocol $\mathbb{P}^s$ with input parameters: the
        i. result of γ_s the function execution the F_s
        ii. received message m
    e. Updates the status s_i and the message m': $\mathbb{P}^s(s_i, N, \gamma_s, m) \rightarrow \{s_i, m'\}$
    f. Broadcast of the message m'

Essentia Protocols

According to the Bitcoin whitepaper, PoW (proof of work) is defined as a sequence of ordered blocks that are created and maintained by PoW-miners.

PoW Protocol

A PoW protocol can be represented as:

$$B_w = \langle b, n \rangle$$

where $b \in \{0, 1\}$ a pointer to the previous block, $n \in \{0, 1\}$ is a random nonce.
A chain of PoW blocks C_w comprises a sequence of p interconnected units:

$$C_w = B_{w_1} \| B_{w_2} \| \ldots \| B_{w_p}$$

Then, the block header

$$h_w = B_{w_p} \text{header:,}$$

chain length PoW-blocks:

$$l_w = p$$

PoS Protocol

Each PoS-block is associated with a valid PoW-block - the validity of which is determined by voting. Based on B_w, a stakeholder may create B_s PoS-block, which is defined as:

$$B_s = \langle S, B_w, P_s, \sigma \rangle \text{ where}$$

S-stakeholder which generates the block P_s - the S payload σ - the $\langle S, B_w, P_s \rangle$ signature.

A chain of PoS blocks C_s comprises a sequence of r interconnected units:

$$C_s = B_{s_1} \| B_{s_2} \| \ldots \| B_{s_r}$$

Then, the block header:

$$h_s = B_{s_r} \text{Length}$$

chain PoS-blocks height:

$$l_s = r$$

Payload PoS-block:

$$P_s = \prod_{i=1}^{l_s} B_{w_i}$$

Hybrid Protocol $\mathbb{P}$

The protocol is $\mathbb{P} = (\mathbb{P}^s, \mathbb{P}^w)$ parameterized by the quality check predicate $V(\cdot)$.

1. every Fw_i PoW-miner, with $1 \le i \le n_w$ the current state s_i does the following:
   a. Selects the best local chain: Receiving through the F_{net} messages $\{N, \langle C_w, C_s \rangle\}$ $N \in \{Fw_1, \ldots, Fw_{n_w}, Fs_{n_w+1}, \ldots, Fs_{n_w+n_s}\}$ set.
   b. The calculation of the index in the previous block: $h(h_s, h_w)$
   c. Search puzzle solution: ω
   d. Create a new $B_w = \langle b, n \rangle$ $C_{w_i} = C_{w_i} \| B_w$ PoW-block:„ $st_i = st_i \cup \{\langle C_w, C_s \rangle\}$
2. each Fs_i PoS-voter, with $n_w + 1 \le i \le n_w + n_s$ the current state s_j does the following:
   a. Selects the best local chain: Receiving through the F_{net} messages $\{N, \langle C_w, C_s \rangle\}$ $N \in \{Fw_1, \ldots, Fw_{n_w}, Fs_{n_w+1}, \ldots, Fs_{n_w+n_s}\}$ set.
   b. Votes for the B_w block ,the result is v
   c. Signs, $sign((F_{s_i}, B_w, P_s), \sigma)$ for v = 1
   d. Create a new $B_s = \langle F_{s_j}, B_w, P_s, \sigma \rangle$ $C_{s_j} = C_{s_j} \| B_s$ PoS-block:„ $st_j = st_j \cup \{\langle C_w, C_s \rangle\}$
   e. Broadcast $\langle C_w, C_s \rangle$ in F_{net}

Network Architecture

The Essentia is a decentralized network combining nodes and services on the blockchain. To uniquely identify a specific node from a group of similar nodes, we use a distributed hash table (Fig. 4) that forms the the associative matrix of abstract data. Nodes are responsible for managing the copying of keys and values in such a way as to minimize system failures during the change of participants.

An ad-hoc network is an overlay computer network based on the equality of participants. Often, there are no dedicated servers in such a network and each node is both a client and a server. Unlike the client-server architecture, this organization allows you to keep the network working at any number combination of available nodes.

An overlay network is a general case of a logical network created on top of another network. The nodes of the overlay network can be connected either by a physical connection or by a logical connection for which one or more corresponding routes from the physical connections exist in the main network.

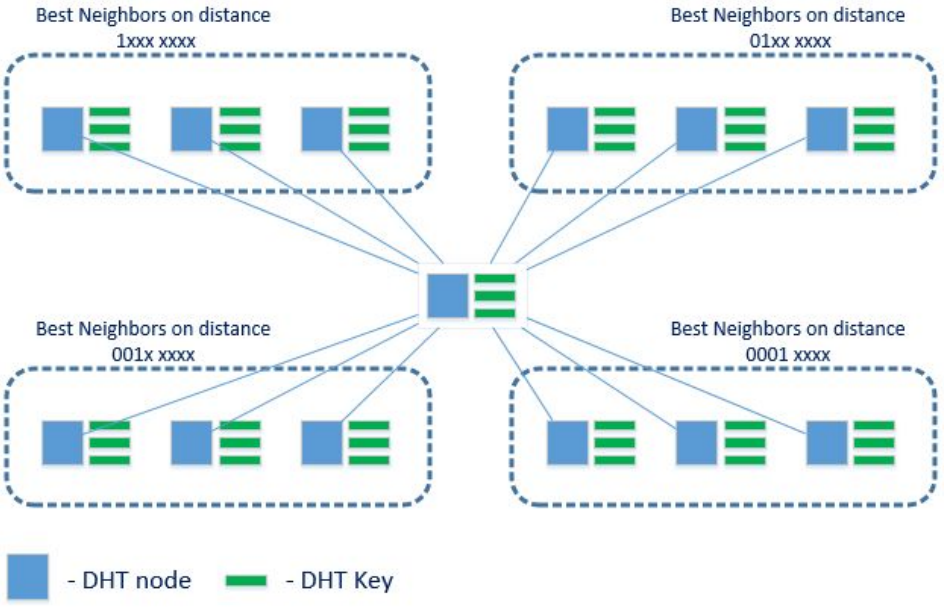

Figure 4. DHT Network Schematic

Essentia Network

The Essentia blockchain is based on the Kademlia DHT protocol (Fig.5). Essentia lacks dedicated servers and each node is both a client and operates as a server. Unlike the client-server architecture, this organization allows you to keep the network working at any number and any combination of available nodes.

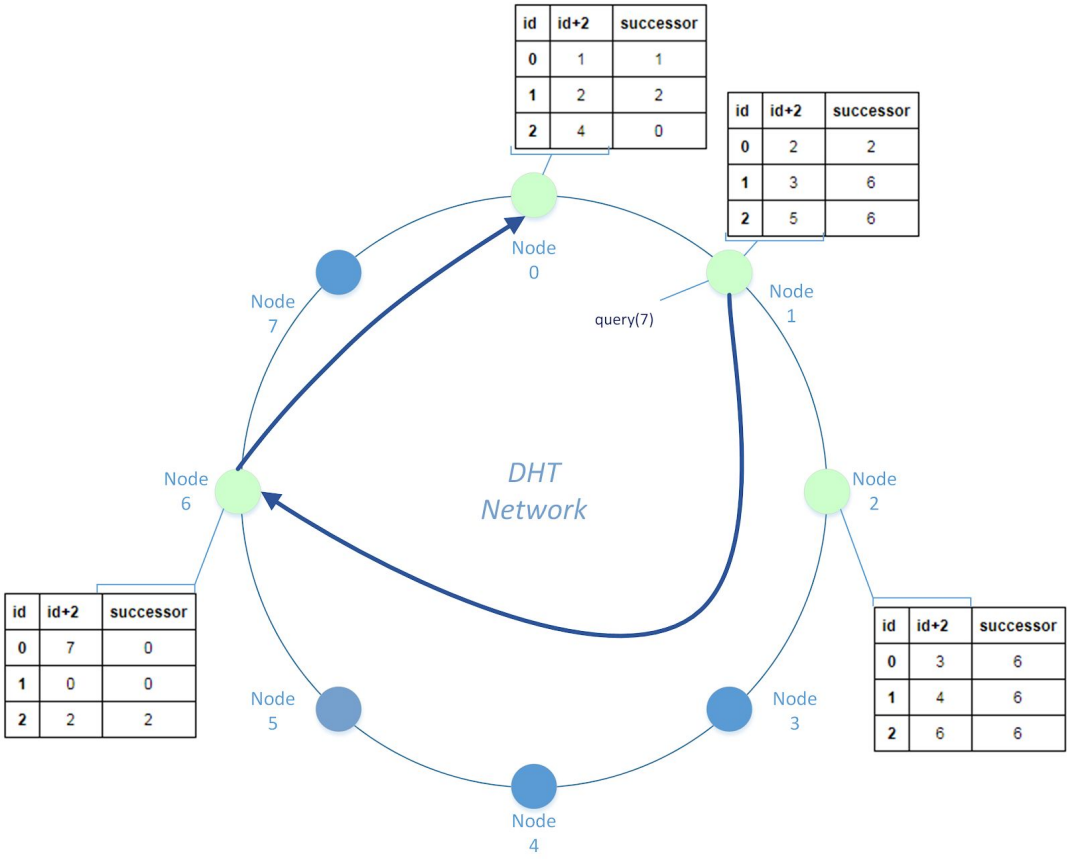

Fig.5 DHT network scheme. Node request use case.

An overlay network is a general case of a logical network created on top of another network. The nodes of the overlay network can be connected either by a physical connection or by a logical connection for which one or more corresponding routes from the physical connections exist in the main network.

Using the Kademlia protocol extends the DHT capabilities with the following properties:

- Minimizing the number of messages notifying about the appearance of a new node
- Information about nodes is distributed automatically together with the key search.
- The nodes are well-informed about other nodes. This allows you to route requests through paths with low latency.
- Using parallel and asynchronous queries that prevent timeout delays from failed nodes.

- Resistance to DDoS attacks.

The DHT hash table (Fig. 6) maps 256-bit keys to some binary values of arbitrary length. This is very similar to the formation of host addresses, for example, the IP address of a node corresponding to a given abstract address can be located on such a key. In fact, the Key (Fig. 7) is the metadata that indicates the "owner" of the record in the hash table (that is, the public key of some node), the type of stored value, and the rules by which this record can later be changed. For example, a rule can only allow the owner to change the value or prevent the value from being changed to a smaller side (to guard against replay attacks).

Figure 6. DHT Routing table

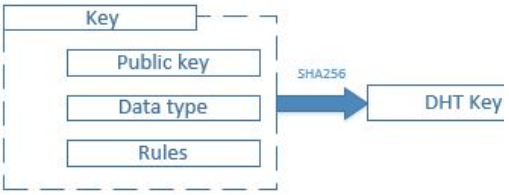

Figure 7. Hash table DHT key

Kademlia Routing table

	ID	IP	UDP Port
Bucket 1	100	109.54.34.12	3339
	101	109.54.34.32	3460
	110	109.54.34.56	3345
	111	109.54.34.23	3454
Bucket 2	ID	IP	UDP Port
	001	109.54.34.45	3391
	000	109.54.34.59	3392
Bucket 3	ID	IP	UDP Port
	001	109.54.34.45	3391
	000	109.54.34.59	3392

Figure 8. Kademlia routing table

In addition to 256-bit keys, the concept of DHT addresses is introduced. The difference with normal host addresses is that the DHT address is bound to the IP address. With DHT, node IP addresses are stored in the DHT using the hash as the key. Each node respond to put or get requests thru to the DHT interface which allows to find and store IP addresses in the DHT by hash. Than, with IP, node can establish p2p connectivity with other nodes, even if they are hide IPs .

As in the Kademlia DHT hash tables, the value corresponding to a certain key must be stored on N nodes that have the shortest distance to this key.

In order for the DHT node to communicate with other similar nodes, it must contain the DHT-DHT-routing table and the IP addresses of the nodes with which it previously interacted, grouped by the distance to them. Such groups are 256 (they correspond to the highest bit set in the distance value - that is, nodes at a distance from 0 to 255 will fall into one group, from 256 to 65535 into the next, etc.). Within each group, a limited number of nodes with the best ping to them are stored. The algorithm for building a routing table is based on calculating the distances between nodes (by applying the metric to their identifiers). The basis of the protocol and the construction of routing tables is the mechanism for determining the distance between nodes through the XOR metric:

$$d = key \ \mathrm{XOR} \ address$$

where d - the distance between nodes, key - DHT key hash table, $address$ - DHT address

Each node supports several operations:

- saving the value for the key,
- searching for nodes
- searching for values.

The nodes search implies the issuance of the nodes nearest to it from the routing table by the given key; the search for values is the same, except for the situation where the node knows the value for the key (then it simply returns it). Accordingly, if the node wants to find the key value in DHT, it sends requests to a small number of nodes closest to that key from its routing table. If, among their answers, there is no desired value, but there are other addresses of the nodes, then the request is repeated to them.

Thus, the Kademlia DHT in Essentia performs the following functions:

1. ***determining the addresses of nodes*** that implement certain services;
2. ***detection of nodes*** by their abstract addresses. To do this, the address is used as the key whose value you want to find. As a result of the request, either the node finds itself (if the desired address was its semi-permanent DHT address) or the value is the IP address and port for the connection - or the other address that should be used as the mediation tunnel.
3. ***storage of information*** about account holders in the blockchain.

Base Architecture

Essentia Framework

A user, a person or a machine, can generate their own root ESSID and associated subIDs. These identifiers enable you to abstractly bind and register data, preserve your anonymity on a blockchain or on a centralized storage.

The framework consists of modules that extend the functionality of ESS-ID and ESS-Home, allowing them to interact with each other and with resources, both centralized and traditional.
Each module is competent in its functions and has its own properties inside the Essentia Framework. The set of modules define the functionality of Essentia .

The main components of the entities are:

1. ESS-Modules: Expansion module management interface:
    - ESS-Auth: user authentication, nodes
    - ESS-DHT: implementation of connection to Essentia
    - ESS-Config: node configuration
    - ESS-Data: interface for shards
    - ESS-CC: cross-chain (atomic swap) transactions interface
    - ESS-FS: distributed file systems interface
    - ESS-DApp: decentralized applications interface
    - ESS-Wallet: wallets interface
2. ESS-Services: service management interface
    - Mining service: PoW Mining Service
    - Pool Mining service: service for organizing a pool for PoW mining
    - File service: implementation of connection to distributed file systems
    - Payment service: implementation of connection to payment systems
    - Dapp service: implementing a connection to Dapp

ESS-Modules Component

The module management interface (Figure 10) contains the functionality that allows connecting external modules as a plug-ins.
The set of Extension Modules depends on the type of entity (Masternode or Supernode) and is loaded during node start. Each module of the set is a separate thread, managed through the Management Block.

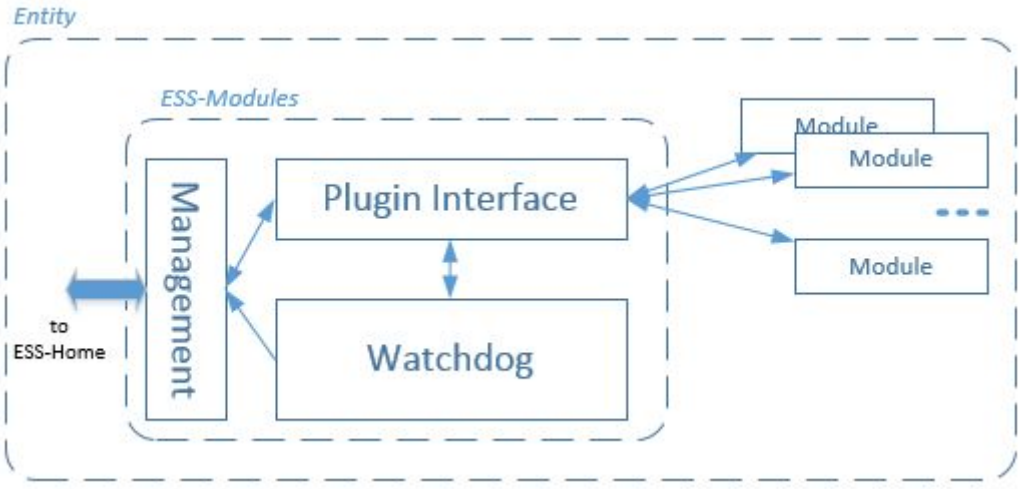

Figure 10. ESS-Modules component scheme

The Plugin Interface contains the implementation of the expansion module interfaces. The expansion modules are managed through the Management block, where the following fields are used as input parameters:

1. Module ID: unique extension module ID
2. Module Function: name of the called function of the extension module
3. Module Parameters: list of parameters and their meanings

The management block validates the request. If the validation is successful, the control unit uses PluginInterface. PluginInterface does the following:

1. Module ID sets the correct plugin
2. Module Function executes the selected function with the parameters
3. Sends the result to the Management Block

Interaction between expansion modules. Plugin Interface and Management block are implemented in asynchronous mode.

The watchdog unit is designed to monitor the status of the module and in the event that the module for some reason ceases to respond, issues a command to the Management unit to reload the plug-in.

ESS-Services Component

ESS services are separate processes running under the Entity's control (Fig. 11). The set of Extension Modules depends on the type of Entity. The start and stop of services is controlled by the Entity, through the Management block, which calls the Service Interface.

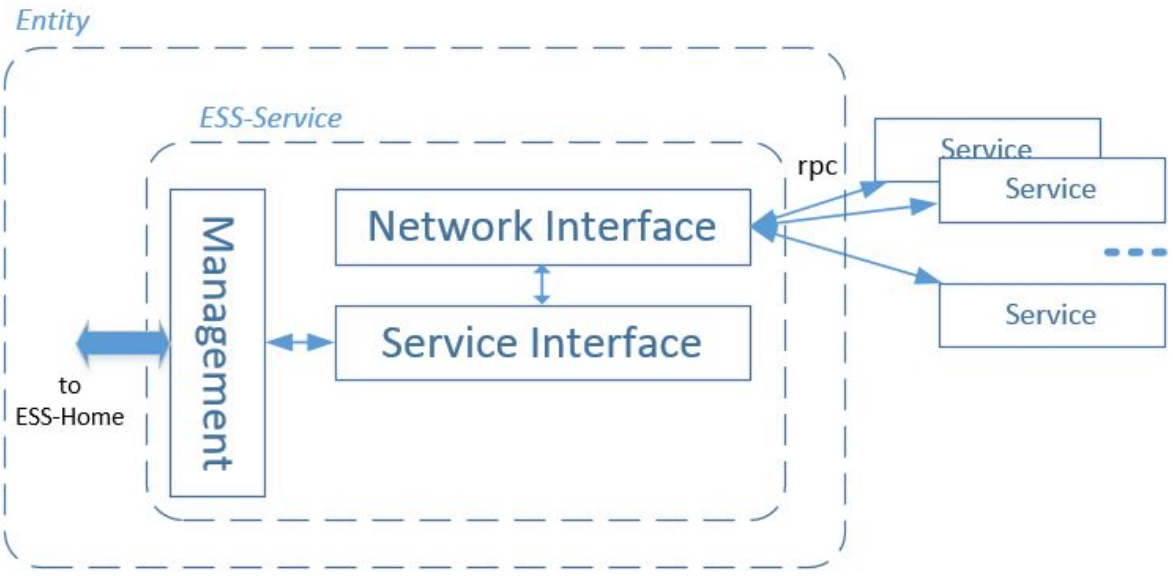

Figure 11. ESS-Service component scheme

The Interface Service contains the implementation of interaction with the services through RPC. All interaction with the services is implemented asynchronously. If the service is started and is not responsive during specified timeout, it automatically restarts.

Essentia Modules

ESS-Config

Structure: implementation of the algorithm for reconfiguring the entity from the configuration file
Purpose: reconfiguring the entities (Masternode, Supernode)

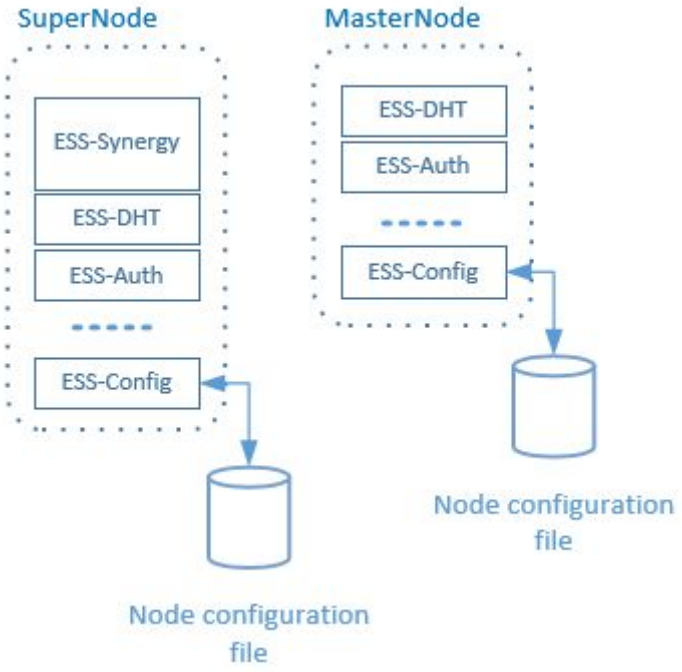

Fig.12
ESS-Config interaction scheme with configuration file

ESS-DHT

Composition: implementation of DHT and Kademlia protocols
Purpose: the organization of connection and data exchange of nodes with a block through a decentralized DHT network and contains the implementation of the Kademlia protocol. (Fig. 13)

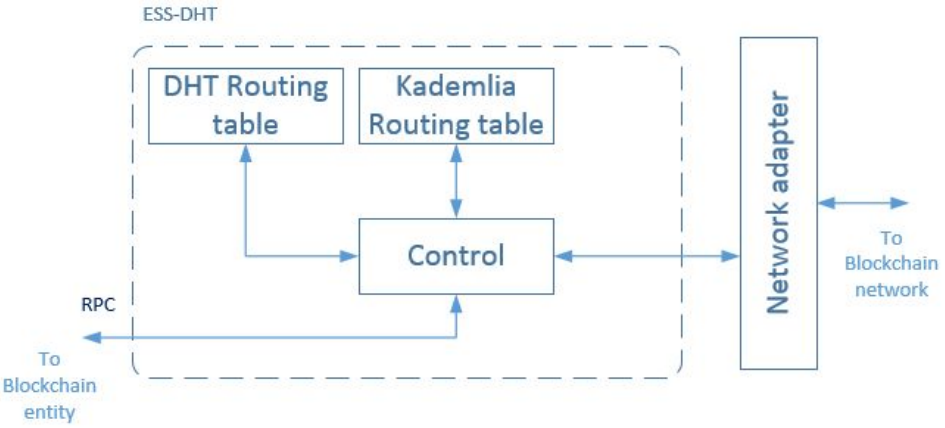

Fig.13. The scheme of the ESS-DHT module

ESS-DHT includes:

1. Control - the service control unit: contains the implementation of the Kademlia DHT

2. DHT Routing table - the routing table DHT
3. Kademlia Routing table - the Kademlia routing table

Normally, the Kademlia protocol supports four RPC commands:

- PING - return online status.
- STORE - a request to place information to the specified node.
- FIND_NODE - a query to search for a given node key, always returns nodes.
- FIND_VALUE - a query to search for a given node key, returns either ***K*** nearest nodes or the value itself.

Implementation of the Kademlia protocol in the Essentia blockchain for security reasons does not contain the STORE and FIND_VALUE commands.

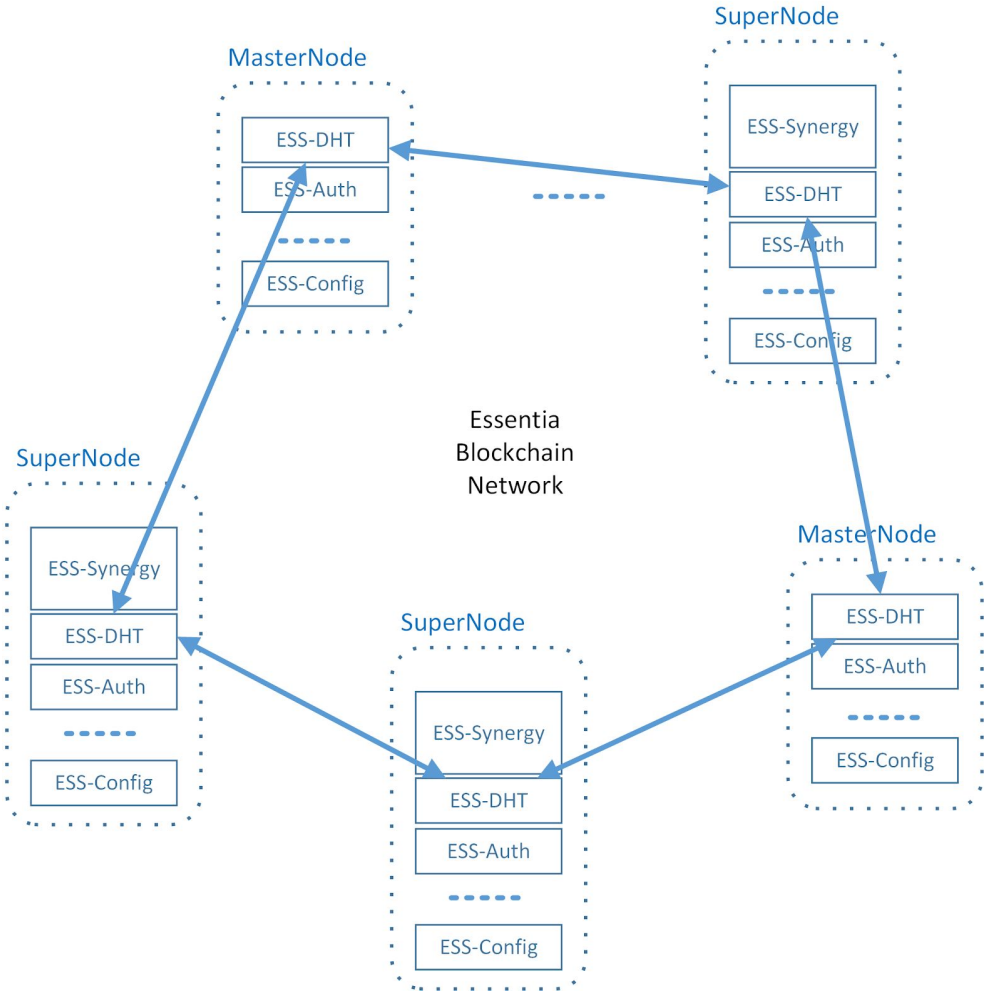

Fig.14. Scheme communication between modules ESS-DHT

ESS-Auth

Function: Cryptographically strong random number generator algorithm implementation (see Authentication). Authentication zero-knowledge (ZK-STARK) and private key generation
Purpose: user authentication (node)

ESS-Synergy

Function: implementation of interaction with the nodes of other blockchain (e.g. Bitcoin and Ethereum) and implementation of cross-chain (atomic swap) transactions.
Purpose: carrying out cross-chain transactions and interaction with other blockchains

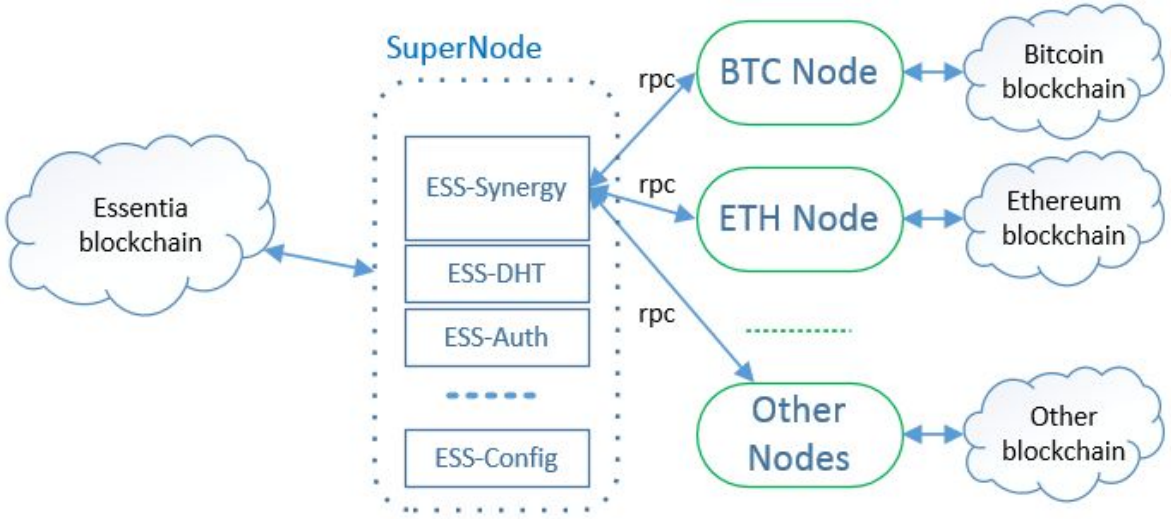

Fig. 15. Various blockchains interaction scheme of ESS-Synergy

ESS-Data

ESS-Data module has the following functions:

1. shard validation
2. synchronization of shards
3. deploy shard block to main chain

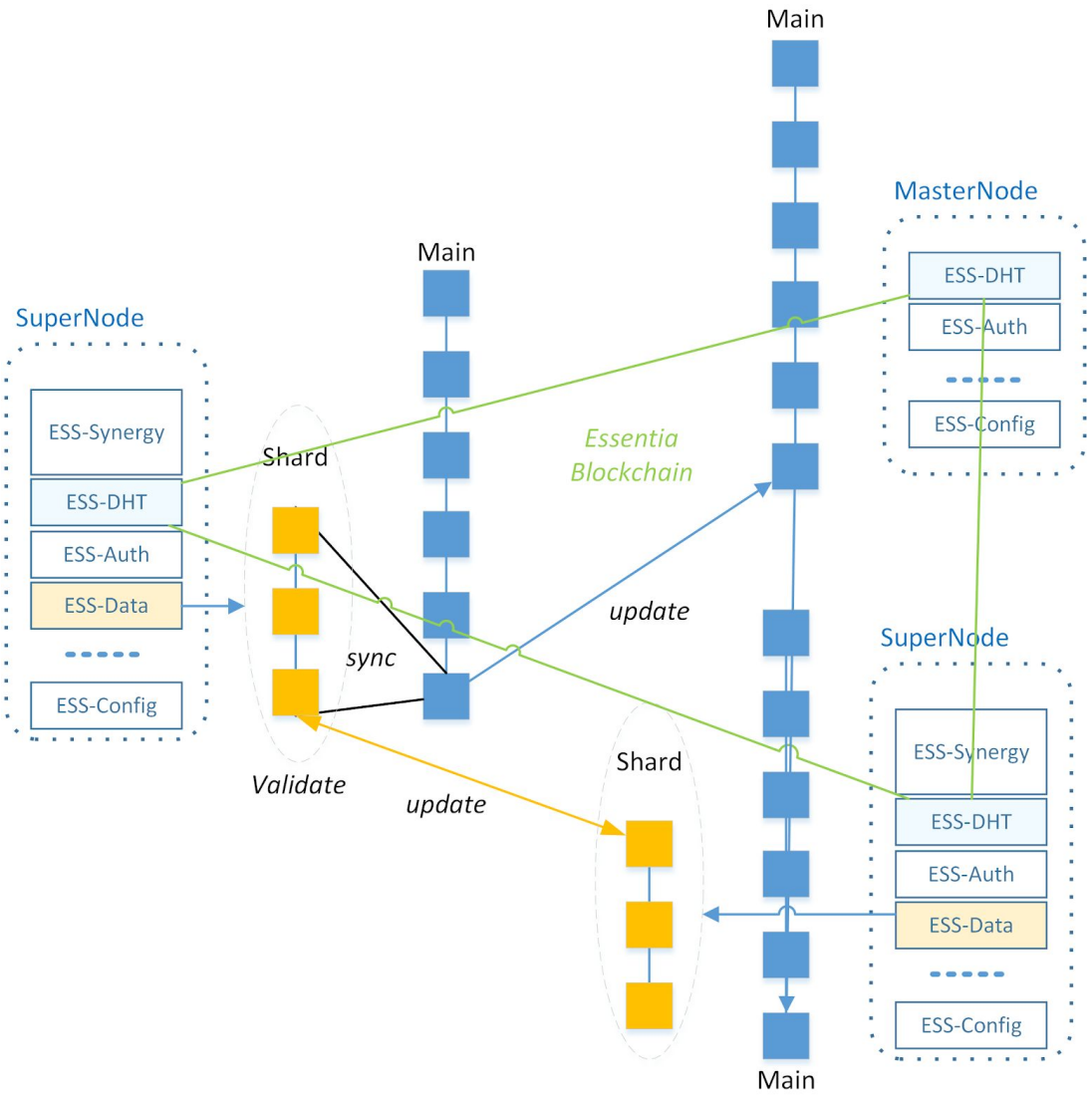

Fig. 16. Scheme of interaction between ESS-Data modules

Masternode Structure

Masternode structure (Fig.17):

1. ESS-Services:
    - PoW miner
2. ESS-Modules:
    - ESS-Auth
        i. CSPRNG
        ii. ZK-STARK
    - ESS-DHT
        i. Kademlia
    - ESS-Config

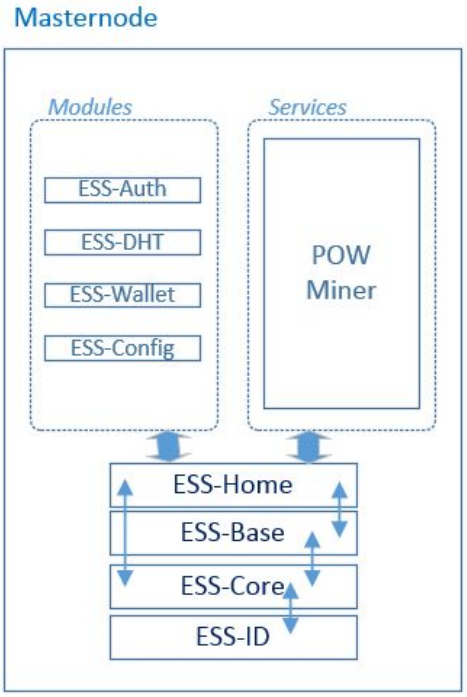

Fig.17 Masternode Scheme

Supernode Structure

Supernode contains(Fig. 18):

1. PoS Consensus
2. ESS-Modules
   - ESS-Auth
     i. CSPRNG
     ii. ZK-STARK
   - ESS-DHT
     i. DHT
     ii. Kademlia
   - ESS-Config
   - ESS-Data
     i. Sharding
   - ESS-CC(Cross-chain)
   - ESS-FS
   - ESS-DApp:
     i. Dapp VM
     ii. Dapp Validation
   - ESS-Wallet:
     i. ESS-Wallet
     ii. BTC-Wallet
     iii. ETH-Wallet
3. ESS-Services:
   - Pool Mining
     i. PoW mining pool service
   - Payment
     i. OmiseGO
   - File:
     i. IPFS
     ii. Swarm
   - Dapp:
     i. Ethereum
     ii. EOS

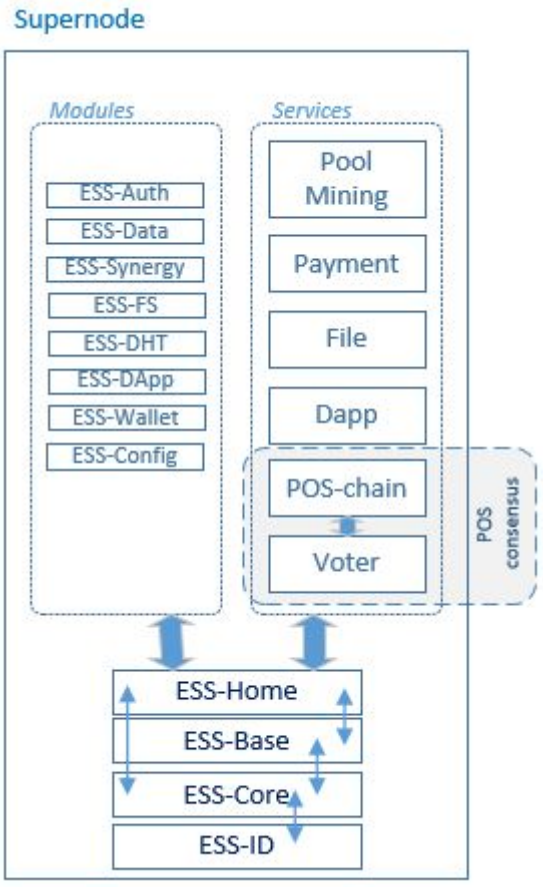

Fig.18 SuperNode Scheme

Essentia blockchain

Transaction per second (TPS)

The TPS refers to the number of transactions executed per second on the blockchain.

$$TPS = S_b / S_t / t_b,$$

were S_b - size of a block of code representing a recent chain of Essentia transactions (decred 1.25Mb, bitcoin 1Mb, essentia 8Mb); S_t-transaction size - typically it equals to 250 bytes; t_b-time needed to mining block. e.g. It takes 5 min(300 sec) for Decred and 10 min(600 sec) for Bitcoin

Network Traffic

Let's calculate network the traffic throughput for the two blockchain types:

1. DHT-based: nodes info is obtained from hash-table stored on node
2. DNS-based: dns seek server stores and returns nodes info

The number of messages Nm_{dht} which are passed between two peers per second for DHT-based networks is calculated by the following equation:

$$Nm_{dht} = 2 \cdot \log(n_p) \frac{n_p}{t_{block}},$$

where n_p - number of peers, t_{block} - block creation time in seconds.

The number of messages Nm_{dns} which are passed between two peers per second for DHT-based networks is calculated by the following equation:

$$Nm_{dns} = 2 \cdot \frac{n_p}{t_{block}},$$

With sharding, we have 7 peers (5-PoS voters, 1- transaction sender, 1- transaction receiver)

Nm_{dht} =7/ 300*2*log(7) = 0.039

Nm_{dns} =7/ 300*2 = 0.046

Size of transferred messages Sm:

Sm_{dht} = 0.039 * 250 ≈ 10 bytes

Sm_{dns} =0.046* 250 ≈ 12 bytes

Time to send one block t_0:

$t_{0_{dht}}$ = 25 seconds (using dht network)

$t_{0_{dns}}$ = 21 seconds (using dns network)

Number of blocks n_b created per second:

$$n_b = t_{block} / t_0$$

$n_{b_{dht}}$ =300/25=12
$n_{b_{dns}}$ =300/21 ≈14

8MB block

TPS =8000000/250/25= **1280** (using dht network)

TPS comparison table:

Cryptocurrency	**TPS**
Bitcoin	7
BitcoinGold	7
Ark	8
Ethereum Classic	14
Decred	17
BitcoinCash	25
Lisk	25
VeChain	25
Zcash	27
Dash	28
Dogecoin	33
Ethereum	50
Litecoin	56
Qtum	70
DigiByte	560
Monero	1,000
NEO	1,000

Nav Coin	1,120
Essentia	**1,280**
IOTA	1,500
Ripple	1,500
Stellar Lumens	2000
NEM	4,000
NANO	7000
TRON	10,000
Komodo	20,000
Stratis	20,000
BitShares	100,000
Steem	100,000

Atomic swap

By definition, the owner of a crypto currency works with their coins without the participation of a trusted third party [10,11]. It is obvious that users want to exchange these currencies on the same principle. Existing centralized exchanges can not meet these requirements. This is evidenced by numerous restrictions on the part of the exchanges themselves and frequent cases of data breaches. However, users are expected to trust the exchange and trust the owners of the exchange; that they will not withdraw money and simply disappear with their funds; trust the engineers who designed and developed the exchange as well as other professionals who provide reliable protection against malicious third parties.

Atomic swap is a means exchange, which is either performed completely free of intermediaries or is not performed at all. This approach allows users to engage in an exchange transaction, even if users do not trust each other and the participant will not lose coins, even if the opponent wants to cheat him.

For atomic swap, Essentia supports the following properties:

1. Time delay for smart contracts
2. Implementation of various cryptographic hash functions

a. Keccak for Ethereum;
b. sha256 for Bitcoin;

3. Off-chain communication channels to discuss the terms of exchange.

Atomic swap uses time-limited locks (HTLC). In fact, a pair of special contracts (Figure 19), which guarantee that if coins of the same currency change owners, then the coins of the second currency do too. And if one of the parties of the contracts does not confirm the transfer in time, then the coins will return back to the original owners. Thus, the coins of both currencies are blocked until all conditions are met and confirmed. Either the exchange of coins is carried out atomically or it will be rejected completely.

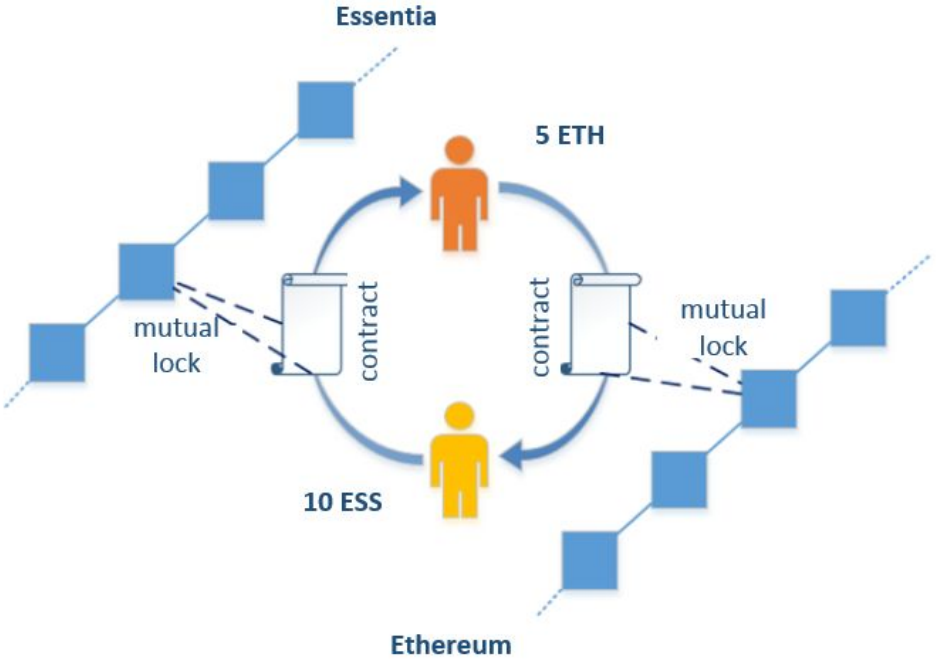

Figure 19. Scheme for exchange of crypto currency using atomic swap

To organize trustless exchange via atomic swap, 4 transactions are necessary, 2 in each blockchain, as well as off-chain communication between entities to transfer the text of the smart contract (Figure 19).

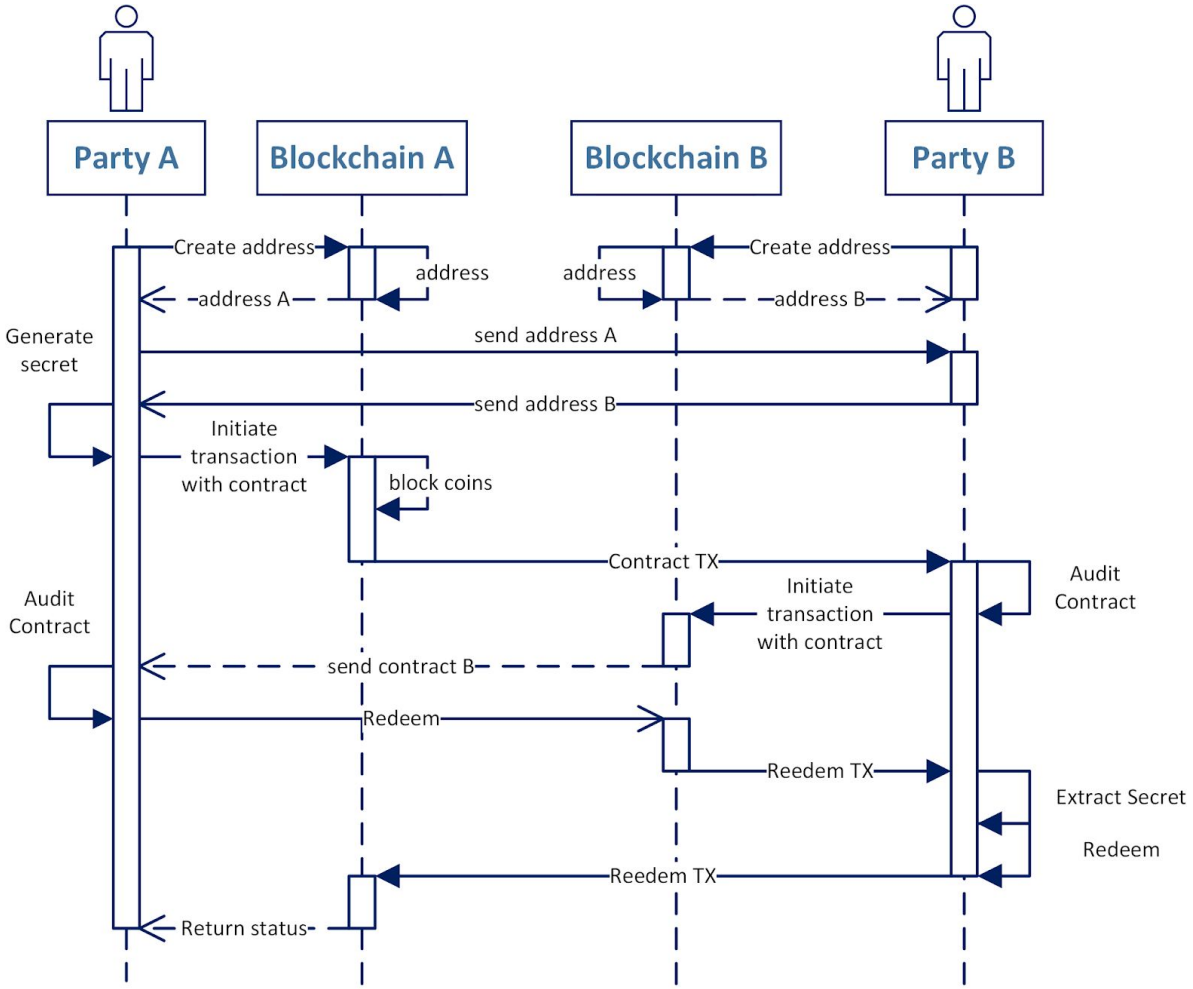

Figure 20. Atomic swap sequence diagram

The currencies that currently support atomic swap include:

1. Bitcoin (Lightning Network) and its forks: Litecoin, Monacoin, Zcoin, Viacoin, etc...
2. Ethereum (Casper) and similar projects with VM and smart contracts functionality.

Atomic Swap Model

According to [11], the Atomic Swap protocol model is an oriented graph $\mathscr{D}$ whose vertices are participants and whose arcs are the transfer of assets. Then, for any pair $(\mathscr{D}, L)$, where $\mathscr{D}=(V, A)$ is a strongly connected directed graph, a $L \subset (V, A)$, $L=\{\nu_0, \cdots, \nu_\ell\}$ is the set of feedback vertices. Accordingly, if, however, the graph $\mathscr{D}$ is not strongly related, then the participants in the exchange will deviate from the protocol.

Atomic Swap Protocol

Each time a participant notices that a contract was published on the input arc of the graph $\mathscr{D}$, they confirm that the contract is the right swap contract, otherwise they refuse the protocol. Then the parties spread secrets hashes. Contracts are distributed on the arc of the graph from the participant to the counterparty and in turn, hash keys from the counterparty to the participant. To acquire assets, each side initiates contracts on incoming arcs controlled by the counterparty.

Although contracts extend in the direction of the arcs, from party to counterparty, hash keys spread in the opposite direction from the counterparty to the party. Informally, each party is motivated to initiate contracts to enter arcs to acquire assets controlled by these contracts.

Hash σ keyhash Lok h arc (υ, ν) is

$$\sigma = \text{sign}\left(\cdots \text{sign}\left(s, u_k \right), \ldots, u_0 \right)$$ threefold:,

where the s-secret $p = \left(u_0, \cdots, u_k \right)$ is a path in the $\mathscr{D}$ graph,where $u_0 = \nu$ the u_k vertex of the graph that generated the s secret.

Each contract in the incoming arc calls the unblocking function:,

$\text{unlock}(s_i, \nu_{i,} \text{sign}\left(s_i, \nu_{i,} \right))$ where s_i is the secret generated by the top of the graph v_i . In turn, other nodes ν_i consider the hash block of the h_i contract for the outgoing arc and perform a function $\text{unlock}(s_i, p, \sigma)$ that causes the unlocking $\text{unlock}(s_i, \nu + p, \text{sign}(\sigma, \nu))$ of each incoming arc by the contract. The distribution s_i is terminated when h_i has been found either the timeout expires or when all arcs of the graph $\mathscr{D}$ are unblocked .

The deadline limiting the assets period (δ) is calculated as follows:

$$\delta = d \cdot \Delta t$$

where d- the length of the longest path between the vertices of the graph Δt- the time during which one participant publishes or modifies smart contract and the other party - detects the change.

δ limits the retention time of assets.

Finally, any atomic swap protocol using hashed time-locks should assign secrets to a set of feedback vertices. Thus, at each step of the protocol, the node ν can publish the contract on the outgoing arc only if it ν has an indegree zero in the current standby mode for the graph $\mathscr{D}$.

Sharding

In order to create a flexible scalable architecture capable of processing a multitude of transactions in a short time, it is necessary to use shading in conjunction with the PoS consensus. Sharding in the blockchain allows you to divide the block into fragments of account shards. The number of shards is dynamic, a new shard is created for each new account.

The block of the main chain contains:

1. set of shard accounts (a)
2. list of validators and shares
3. hash of last blocks and shards

The account shard block contains:

1. public key
2. smart contracts and the Dapps
3. list of nodes belonging to the same group
4. transaction list
5. cross-links to other shards

For the operation with shards, validation tasks are:

1. validating the hash of the parent block
2. validating the hash of the block in the shard
3. creating a cross-link to add the shard block to the main chain.

Each shard itself is a PoS chain. Cross-links confirm segments of chains of shards in the main chain and also are the main way by which various shards can exchange data with each other. The simplest cross-links are hashes of unrelated data blocks. Processing PoS chain blocks have to satisfy the following conditions:

1. The parent block has passed the validation
2. The block has passed the validation
3. The local node time is greater than or equal to the calculated point in time.

The block processing is postponed until all three conditions are true. Thus, Decade determines the time when the block is created. Then, when Decade over, the Supernode either creates or verifies the block.

Monitoring the State of the Chain

The result of the transaction is processed using the state tree (Fig. 16). In fact, this is proof that if the transaction T with root state S is completed, the result is the state of the root state S' and the $O(\log n)$ protocol .

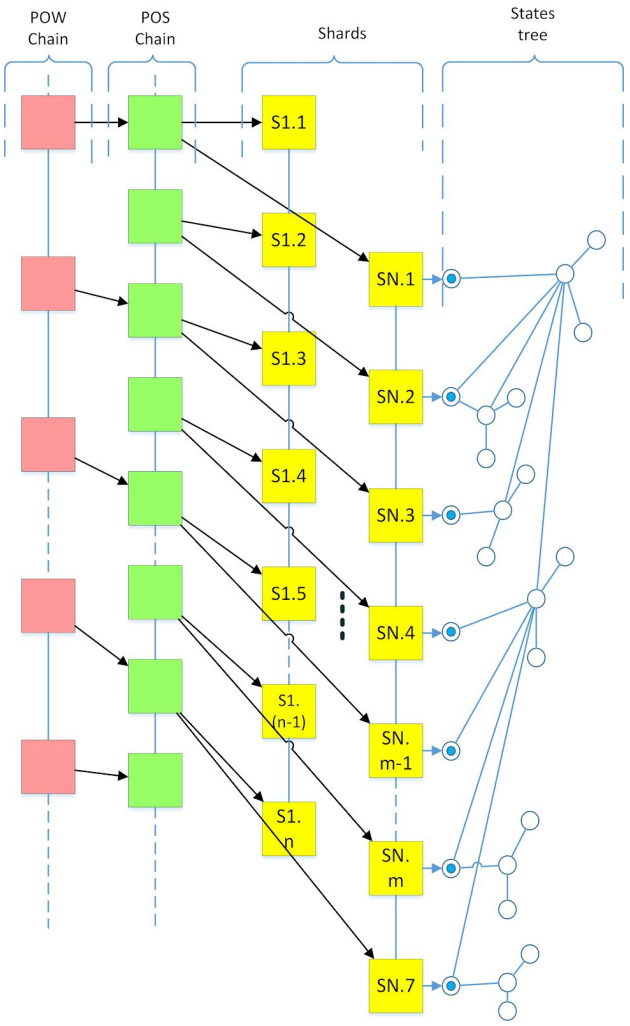

Figure 21. The Essence schema

The state transition consists of two parts:

1. Century
    a. checking the cumulative signatures
    b. keeping temporary records related to activity within the block
    c. control of state validators
    d. processing of cross-links
2. Block processing during and after the Century and cross-link processing
    a. verification of partial cross-links
    b. extension of the list of verified objects according to their state

c. verification of the verified object, it contains the first validator at the given height of the chain

The Century change is performed when the Decade boundary time passes to the next Century. Since the change itself, consists of several parts, a change in the balance of the validator can only be made after all parts have been fully executed.

How the Chain of Blocks is Formed

The Shard

The generation order of shard blocks is determined by sorting the supernode list. The shard block is created first on the supernode list and is signed by its private key. Furthermore, the block is transmitted by a randomly selected Node (more than 3). The receiving Node verifies the signature of the initiating supernode block and checks the validity of the block, performing all transactions in it and compares the result to the declared result.

If the shard block is confirmed, the receiving Node signs it and spreads its signature to the other Node in the group either directly or via "multicast".

As a shard unit will collect at least two-thirds of the Node signatures in the group, it can be designated as the next shard block.

Block of the Main Chain

The block of the main chain is generated by masternodes after the shard block is generated and confirmed. The procedure for generating the block of the main chain is similar to the generation of the shard block, except that all the supernodes, or at least two-thirds, are involved in the process.

Chain Algorithm

Step 1. A new block, containing a header that includes the hash of the previous block, is accepted by the network. If there are no tickets in the network, the block is rejected.

Step 2. The masternode checks that the new block included a valid list of recently published transactions starting from the root of the Hash-tree. The masternode votes for the block and sends a signed control block.

Step 3. Any new block has to have a consensus among at least **three** masternodes. The number of masternodes can vary depending on the entropy of the chain. For any given number of masternodes, there are three requirements:

- A block has passed validation when the majority of masternodes voted for it. This block is considered as valid and linked to the new shard. "Positive" masternodes ***receive a reward***, while absentees and "negatives" ***do not receive compensation***.
- The block is rejected if the majority of masternodes voted "negative". The "negative" masternodes ***receive a reward***, but the absentees and "positive" ***receive nothing.***
- Expired and expended tickets can not vote.

Step 4. The SHA3-256 hash of the previous blocks are included in the header of the new supernode block

In order for the masternode to participate in the voting process, it must freeze 1000 ESS. Each masternode from the full set votes only for the PoW-block in the mainnet, the remaining voters, selected by the masternode from the voters subset, vote for the shard block only.

The voting algorithm for the PoS block

1. A masternode initiates a network transaction. This transaction transfers the assets from its purse to the newly created address.
2. This transaction is included in the next chain block and is confirmed by the supernode. The transaction address is not readable after that.
3. Based on the supernode blocks, a 16-bit ticket is created.
4. After masternode selection, send PoS-block to the main chain and either confirms or rejects the last supernode block. There are two possible scenarios:
    - Voting to confirm or deny a newly created supernode occurs when the masternode creates its PoS-block. A transaction with quotes in this block ***provides a reward*** (as per the current interest rate). The output address containing the crypto currency used to send the transaction and the reward, is unlocked and can then be used.
    - The masternode block can not be created without the supernode block. In this case, the ticket is marked as "invalid", no ***reward is given***, the funds on the initial output address are unblocked and, thus, can be spent.

You should pay attention to the masternode election. To prevent attacks on the voting mechanism, it is necessary to avoid predictability (i.e. determining the result of the selection from the previous values). Essentia thus uses a crypto-stable pseudo-random number generator.

Hybrid consensus

The use of a hybrid system provides greater safety and stability. Large pool mining has zero chance to gain too much influence on the network (i.e. a 51% attack), as the block is checked and signed by randomly selected PoS-voters. This helps to mitigate unwanted behavior. A conspiracy between PoW and PoS miners is impossible, since PoS voters are selected only after the block is created. A conspiracy between PoS miners is also impossible, since the crypto-stable generator randomly chooses a PoS voter. The chance to randomly choose several tickets belonging to one group, within a large group of Essentia PoS node operators, is insignificantly small. Even if this were to happen, each transaction is checked at least twice by different voters.

The implementation of PoS consensus in a hybrid scheme is critical to security issues since an increase in TPS leads to the simplification of the voting algorithm and centralization as a consequence.

Security

Security Properties of the Essentia Blockchain

Let's highlight the basic security features of the blockchain:

1. chain growth property Q_{gp} [4]: linear chain growth depending on the
2. chain quality property Q_{qp} [5]
3. common prefix property Q_{cp} [6,7]

In a system with hybrid consensus, the block chains for PoS and PoW grow evenly, therefore we will consider the basic properties with respect to PoS, in particular, for parameters $r \in \mathbb{N}$ and the lower bound $\mu \in (0, 1)$.

Definition 1. Q_{gp} **for PoS**: Q_{gp} argues that the length of the new chain for any honest PoS-voter $Fs \in \{Fs_{n_w+1}, \ldots, Fs_{n_w+n_s}\}$ holds the statement $l_s - l_s^{'} \geq g$, where l_s - the length of the new chain C_s, - the $l_s^{'}$ length of the previous chain C_s, - the g growth coefficient.

Definition 2. Q_{qp} **for PoS**: Q_{qp} with parameters $\mu \in \mathbb{R}$ and $r \in \mathbb{N}$ argues that for any bonafide PoS-voter $Fs \in \{Fs_{n_w+1}, \ldots, Fs_{n_w+n_s}\}$ holding a chain it C_s is true that for a sufficiently large sequence of a r chain of C_s PoS-blocks, the number of honest PoS-blocks is not less μ.

Definition 3. Q_{cp} **for PoS**: Q_{cp} c parameter $k \in \mathbb{N}$, where is the k number of the last blocks, states that for two honest PoS-voters C_{s_i}, the C_{s_j} statement is true $C_{s_i}[1, r_i] \preccurlyeq C_{s_j}$ where $r_i = l_{s_i} - \Theta(k)$.

Properties Q_{gp} and Q_{cp} for PoW are similar to PoS except Q_{qp}, because an attacker can create a lot of malicious PoW-blocks, having control over the large processing power.

Definition 4. Q_{qp} **for PoW** : Q_{qp} with parameters $\mu' \in \mathbb{R}$ and $p \in \mathbb{N}$ states that for any conscientious PoW-miner $F_w \in \{Fw_1, \ldots, Fw_{n_w}\}$ holding a chain it C_w is true that for a sufficiently large sequence of a p chain of C_w PoW-blocks, the number of honest PoW-blocks is not less μ'.

Consensus Attacks Analysis

Denote the probability that a bonafide PoW-miner extracted block as:

$$\alpha_w = 1 - (1 - cp)^{(1 - cp_m) \cdot n_s},$$

where cp - the probability that a bonafide miner will find the solution cp_m - malware probability computing power miner, n_w - the total number of PoW-miners.

Then, the probability of mining blocks by unscrupulous PoW-miners is:

$$\beta_w = cp_m \cdot cp \cdot n_w$$

When $cp_m \cdot n_w \ll 1$ we have $\alpha_w = (1 - cp_m) \cdot n_w \cdot cp$, then

$$\frac{\alpha_w}{\beta_w} = \frac{1 - cp_m}{cp_m}.$$

Similarly, we calculate the probability of good faith α_s ()and unfair β_s ()participation

$$\alpha_s = 1 - (1 - \tilde{cp})^{(1 - \tilde{cp_m}) \cdot n_s} = (1 - \tilde{cp_m}) \cdot n_s \cdot \tilde{cp}$$ PoS-voters:,

$$\beta_s = 1 - (1 - \tilde{cp})^{n_s \cdot \tilde{cp_m}} = \tilde{cp_m} \cdot n_s \cdot \tilde{cp}$$

then perhaps the fact that conscientious PoW-miners have extracted the block and conscientious PoS-voters have signed it, is:

$$\alpha = \alpha_s \cdot \alpha_w$$ and

the likelihood that unscrupulous PoW-miners have extracted the block and unscrupulous PoS-voters signed it, is equal to:

$$\beta = \beta_s \cdot \beta_w$$

Consider the possible scenarios of the hybrid consensus:

Scenario 1: The conscientious PoW-Miner mints a new PoW-block for which the conscientious PoS-voter votes. A conscientious PoS-voter signs the corresponding PoS-block.

Scenario 2: An unfair PoW-Miner mints a new PoW-block, for which an unscrupulous PoS-voter votes. An unfair PoS-voter signs the corresponding PoS-block.
Scenario 3: A conscientious PoW-Miner mints a new PoW-block, for which an unscrupulous PoS-voter votes. An unscrupulous PoS-voter can sign or cancel the corresponding PoS-block.
Scenario 4: An unfair PoW-Miner mints a new PoW-block, for which a conscientious PoS-voter votes. The PoS-voter can sign or cancel the PoS-block.

Scenario 1 guarantees the growth of the blockchain. Unscrupulous participants can not affect its growth.

In addition, unscrupulous participants in Scenario 2 and 3 can generate PoS-blocks. If the probability of Scenario 2 and 3 is less than Scenario 1, unscrupulous participants can not generate more PoS blocks than honest players. Since unscrupulous participants can not prevent Scenario 1, they can compete with bonafide participants. Even if they win all the polls, there are still some blocks from honest players. This guarantees the property Q_{qp}. Finally, the probability that all PoW-miners will find a new PoW-block at the same time is very low.

We also assume that unscrupulous participants may delay for some time messages from bonafide participants. We assume that, in most cases, unscrupulous participants can delay any communication. Thus, unscrupulous participants can affect bonafide participants to get the best chain-pair on time. This will lead to bonafide miners working with the wrong blocks. As a result, bonafide computing power is lost. We measure the actual likelihood τ that bonafide participants can create a block. The worst case for Q_{gp} is that unscrupulous participants will cancel all honest messages on Δt in order to reduce the probability of success of bonafide participants .

In a hybrid consensus, this delay is doubled and equal $2 \cdot \Delta t$. Block delay $\delta > 0$ reduces the efficiency of mining. Because of the delays, the new PoS and PoW blocks produced by bonafide and dishonest participants will be added to the chain, and bonafide miners will not get any rewards.

Within the network delay model [12], the effective probability of bonafide miners and voters considering the delay Δt in transferring the block between the miner and the holding company is:

$$\tau = \frac{\alpha}{1 + 2 \cdot \Delta t \cdot \alpha}$$

Thus, the security properties for the block can be written in the form:

1. For Q_{gp} PoS: for the protocol in question $\mathbb{P} = (\mathbb{P}^s, \mathbb{P}^w)$ exists $\delta > 0$, such that for any bonafide voter $Fs \in \left\{Fs_{n_w+1}, \ldots, Fs_{n_w+n_s}\right\}$ in the current ν' and previous ν

votes ($V=\nu' - \nu > 0$), under which the probability, of what $l_s' - l_s \geqslant g \cdot V$ is less $1 - e^{-\Omega(V)}$, where $g = (1-\delta) \cdot \tau$

2. For Q_{qp} in the PoS: We assume $\tau = \lambda' \left(\alpha_w + \beta_w \right) \beta_s$ that. $\lambda' > 1$ Then, for the protocol in question, $\mathbb{P} = (\mathbb{P}^s, \mathbb{P}^w)$ there $\delta > 0$ is such that for any bonafide voter $Fs \in \left\{ Fs_{n_w+1}, \ldots, Fs_{n_w+n_s} \right\}$ with a chain C_s, there is a probability that, for a sufficiently large length of the chain of PoS blocks r generated for m rounds, the fair-block ratio is not less than $\mu = 1 - (1+\delta) \frac{\left(\alpha_s + \beta_s \right) \beta_w}{\tau}$ is less than $1 - e^{-\Omega(l_s)}$
3. For Q_{cp} in the PoS: We assume $\alpha = \lambda' \left(\alpha_s + \beta_s \right) \beta_w$ that. $\lambda' > 1$ Then, for the protocol under consideration, $\mathbb{P} = (\mathbb{P}^s, \mathbb{P}^w)$ there is $\delta > 0$ also a security parameter k, such that for any two bonafide voters Fs_i and Fs_j (which generate blocks on rounds m_i and m_j accordingly, where $m_i \leq m_j$ and $i, j \in \{n_s+1, \ldots, n_s+n_w\}$) and have chains C_{s_i} and the C_{s_j} probability that $C_{s_i}[1, \ell_i] \preccurlyeq C_{s_j}$, where $\ell_i = l_{s_i} - \Theta(k)$ not less than $1 - e^{-\Omega(k)}$

Authentication

In the Essentia system, the user authentication task is based on a ZK-STARK [Zero-knowledge] zero-disclosure protocol. The use of this protocol primarily solves the following problems:

1. the need to store keys on the server.
2. MITM Key capture attack.

As a rule, ZK-protocols are of a probabilistic nature. This means that the testing party can never be completely sure of the knowledge of the proving party of the secret, but it can verify this with accuracy to any preassigned probability in a finite time.

ZK-STARK has the following properties:

1. ***Completeness***: parties can always prove knowledge of the secret, if they really have it;
2. ***Correctness***: the parties can not show knowledge of the secret, if they do not own it;
3. ***Zero agreement property***: any information received by the verifying party can be obtained by themselves using a polynomial algorithm without interaction with the proving party.

The ZK-STARK protocol is scalable, transparent, universal and is currently resistant to quantization [13].

Using ZK-STARK as a confirmation of the parties' honest allows to secure users without revealing the state of their accounts and transaction amounts. Thus, only the fact of a successful transaction is open for public access.

In the ZK-STARK protocol, there is no external trusted configuration phase, instead, random oracle model is used. The use of a random oracle model is extremely important for the public to trust systems with evidence of a lack of knowledge. Otherwise a subject with huge calculation performance could have influence to settings obtaining and create false evidence. However, if the trusted phase of the installation is not used and that random oracle model is used instead, the zero-knowledge mechanism creates proven trust.

So, ZK-Stark includes implementation of the following algorithms (Fig.22):

1. Key Generation (G): The key generator G accepts the secret parameter λ and the program C and generates two public keys: the proof key pk and the verification key vk. These keys are public parameters that need to be created only once for this program C.
2. Proof P: As input values, the proof takes: a key pk, a public value χ and a secret value ω. The algorithm generates a proof $proof = P(pk, \chi, \omega)$ that the proof knows the secret value of w and that the secret value satisfies the program C.
3. Checking V: The algorithm calculates the $V(vk, \chi, proof)$ result which is *true* if the proof is correct and *false* otherwise. Thus, this function returns *true* if the verifier knows that the secret value ω satisfies $C(\chi, \omega) = true$.

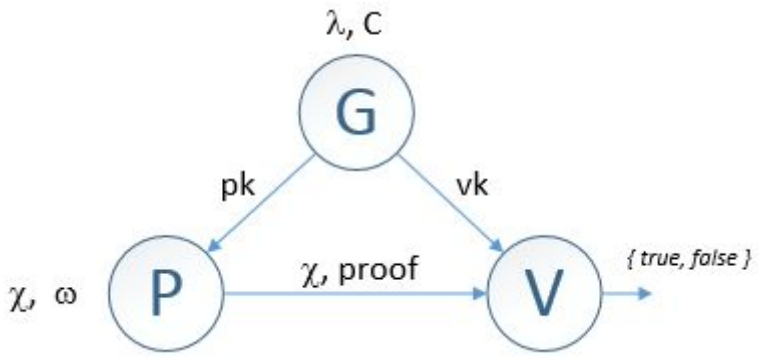

Fig.22. Authentication schematic

Using the ZK-STARK protocol gives the Essentia system a unique property: ***verified trust***, which is guaranteed to provide protection from attempts to falsify the data being processed.

Links

[1] Essensia Whitepaper https://essentia.one/whitepaper.pdf

[2] R. Canetti. Security and composition of multiparty cryptographic protocols. Journal of Cryptology 13(1):143–202, 2000.

[3] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. http://eprint.iacr.org/2000/067.

[4] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf.

[5] V. Buterin. Chain Interoperability .September 9, 2016. https://static1.squarespace.com/static/55f73743e4b051cfcc0b02cf/t/5886800ecd0f68de303349b1/1485209617040/Chain+Interoperability.pdf

[6] M. Herlihy. Atomic Cross-Chain Swaps. Computer Science Department Brown University Providence, Rhode Island 02912 . https://arxiv.org/pdf/1801.09515.pdf

[7] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. ACM Trans. on Modeling and Computer Simulation, 8(1):3–30, January 1998. http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/ARTICLES/mt.pdf

[8] A. Kiayias and G. Panagiotakos. Speed-security tradeoffs in blockchain protocols. Cryptology ePrint Archive, Report 2015/1019, 2015. http://eprint.iacr.org/2015/1019.

[9] J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In E. Oswald and M. Fischlin, editors, EUROCRYPT 2015, Part II, volume 9057 of LNCS, pages 281–310. Springer, Heidelberg, Apr. 2015.

[10] R. Pass, L. Seeman, and A. Shelat. Analysis of the blockchain protocol in asynchronous networks. In EUROCRYPT, 2017. https://eprint.iacr.org/2016/454.

[11] J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In E. Oswald and M. Fischlin, editors, EUROCRYPT 2015, Part II, volume 9057 of LNCS, pages 281–310. Springer, Heidelberg, Apr. 2015.

[12] R. Pass, L. Seeman, and A. Shelat. Analysis of the blockchain protocol in asynchronous networks. In EUROCRYPT, 2017. https://eprint.iacr.org/2016/454.

[13] E, Ben-Sasson, I. Bentov, Y. Horesh. M.Riabzev. Scalable, transparent, and post-quantum secure computational integrity, In EUROCRYPT, 2018. https://eprint.iacr.org/2018/046.pdf

[14] The ISAAC Cipher http://rosettacode.org/wiki/The_ISAAC_Cipher