RASMART

WHITE PAPER

Ver 1.0

Contents

1. Characteristics of the system

Rasmart is a platform based on the blockchain technology. The specialization of Rasmart is enhancing the applicability of financial services based on a distributed ledger, cryptocurrencies and self-executing smart contracts.

Today, blockchain is an extremely rapidly developing technology, which formed the range of requirements which Rasmart, being a blockchain-based platform, fulfills offering a set of its improvements and extensions.

1. Speed of transactions. The quality characteristic showing the number of all transactions carried out in the system per second

2. Flexibility of use. Rasmart provides a flexible and adaptive API for integration with third-party services

3. Protection. Thanks to the block time of 0.2 seconds and the building of a consensus, various attacks and vulnerabilities may be neglected

4. Own smart contracts. The smart contracts of Rasmart system are fast thanks to the use of a time-tested programming language (Lua) whose advantages are the fast execution and the small interpreter not requiring a lot of computational resources; in sum, the large transaction capacity and fast execution + the regularly updated SDK with a range of capabilities

2. Introduction

Bitcoin, as the first blockchain platform, provided for the development of a new decentralized way of interaction between participants, in which asset exchange is carried out without any supervision of regulatory authorities. Therefore, Bitcoin was soon in demand among users. In the Bitcoin network, only one operation type is carried out—exchange between users without intermediaries.

Ethereum is a platform allowing to create almost any decentralized online blockchain-based services which work on the basis of smart contracts.

However, the first blockchain-based platforms did not achieve much recognition of users and had only several enthusiastic fans. Besides, the low speed of only several operations per second did not allow them to attract a wider audience. All the blockchain platforms appearing later followed the classic scheme first applied to Bitcoin.

Today, it's only the financial industry that attempts to fight the decentralized (direct) interaction between participants. Yet, from the technical and the organizational points of view, it's easier to create a decentralized system of financial services than a classic bank. All the needed experience and knowledge are already gained.

So, in order to create a maximally comfortable decentralized system of financial services based on a distributed ledger, you need:

1. A high speed of data processing (hundreds of thousands of transactions per second), provided that the cost of an operation must be minimum;

2. Creation of a united, maximally user-friendly system able to unite all its participants and elements in order to provide decentralized financial services: settlement centers for fiat money, user personalization, a credit bureau, terminals for cashing out cryptocurrencies, etc.
Rasmart platform is the way to accomplish all this.

Rasmart is a united decentralized technological platform able to unite all the participants in financial services, carry out high-speed and secure transactions based on the distributed ledger concept. The federative system and the self-executing smart contracts allow to create unique solutions, establish interaction between various financial products. Rasmart allows to fulfill the enormous potential of the application of blockchain projects in the financial and other industries where a distributed ledger was not possible to apply due to technical (a low speed) and financial restrictions in the form of a high cost of transactions.

2.1 Problems and solutions

Network lags are one of the main restrictions many blockchain projects face. The average time one Bitcoin transaction takes is 10 minutes or more. At the same time, bank transactions are carried out almost instantaneously, taking not longer than a second.

Technical problems

We see such problems as:

Capacity – the time required to process a certain amount of data

The **speed** of transactions

Latency – the real time of the system's response compared to the expected

Amount of saved data – the amount of data saved in the network
(Ethereum is constantly gaining tens of gigabytes)

High cost – the key parameter for carrying out all operations and, in particular, those of the two main groups

1. Operations of the Internet of Things, allowing to unite various objects in a global network;
2. Micropayments with a low cost of transactions, such as a usual purchase of a coffee, paying for food in a supermarket, all kinds of microloans.

All these problems and restrictions prevent expansion of the blockchain technology to many industries, including the financial one.

Blockchain-based technologies allow to implement random databases with the following characteristics:

Availability. A distributed database is stored at all the network nodes simultaneously. That is, in case of no connection with the rest of the network, all the information can be accessed at just one node.

Uniformity. Blockchain makes faking impossible and allows to quickly detect the fact and place of any faking attempt.

Confidentiality. Thanks to the asymmetric data encryption, only the recipient of information can get access to it. An electronic signature guarantees that the information really belongs to the sender.

Reliability. Storing the database at each node allows to recover all the information upon a fault of any node. The network can fully recover even on the basis of just one undamaged node.

Scalability. It allows to quickly and easily increase the size of the network by adding nodes. To connect to the network, it is sufficient to register and connect to any node existing at the moment.

Integration. Thanks to the API, there is the ability to integrate any software with the distributed database, and such software will be able to receive and save any data in it.

Thanks to the characteristics listed above, the ways of application of the database go far beyond the financial industry. It is a universal way of protected information exchange, electronic document flow, accounting automation and various data collection. Its application will be in demand in any industry which requires a great number of participants and top security of information storage for process automation.

2.2 Role of blockchain in the project

Blockchain is a chain of connected information blocks. Each of such blocks contains the cryptographic function (or the hash) of the previous block of the chain. If we only have two blocks, e.g. the first and the last in the chain, we can be 100% sure that all the blocks between them are verified.

A database built on this basis allows to add information to it by the way of recording a new value, having the rest of the values unchanged and, thus, forming a consecutive history of changes. Deletion of information is not possible. This way, any activity in the database may be detected and have its authenticity checked. This is a perfect solution for opening ledgers.

But even an open database provides the ability to save confidential information. Asymmetric encryption by the recipient's public key allows only the holder of the private key to gain access to it. The RSA encryption algorithm can guarantee it, provided that the number of keys is sufficient. The rest of the participants in the network can only see the fact of the addition of the information, but will not be able to gain access to it.

The platform provides users with technical mechanisms for the storage, protection and synchronization of the distributed ledger. There are no restrictions of the structure and format of information storage. RSA only complements it with the information on the connection and authenticity of the data added to the ledger.

3. Definitions

1. System – a combination of decentralized network nodes which carry out processing, storage and transferring of transactions, as well as execution of smart contracts and confirmation of their conditions. It processes requests from third-party systems, provides information upon request

2. Network node – a computer which the full version of the network client has been installed on, connected to the united system, participates in conducting transactions, data storing and transferring

3. Ledger – all confirmed transactions saved at the network nodes

4. Transaction - a request for execution of a certain smart contract method, with further recording of that in the blockchain

5. Smart contract - computer protocols which check and ensure observation of the conditions of interaction. In most cases, smart contracts emulate the logic of trustful relationships. Being decentralized and independent of the central source is the key feature of a smart contract

6. Smart contract method - a program code responsible for computation of the operation of a smart contract and recording of the received data in the ledger

7. Party to a contract – a user of the system and end participant of the network

3.1 Network nodes

In order to build an independent decentralized network based on free access and node connection, we use several node types simultaneously, depending on their purpose:

1. A usual node participates in checking transaction validity but has a minimum trust coefficient, is a candidate for the role of a trusted node or a node for current processing in the next round of selection of the nodes' roles in the network

2. A trusted node participates in transaction checking, has a predominant trust coefficient

3. The main network node accumulates transactions sent from usual nodes and distributes them to trusted nodes

4. A recording network node is one of the trusted nodes chosen during a round, which forms the block, records it in the database and sends to all the participants in the network, getting back the hash and forming a new round

3.2 Last saved block

The GBL (general block ledger) is a fully synchronized state of the entire ledger of blocks at all the system's nodes.

The content of a ledger block is a unit of information stored in it, such information containing the hash code of the previous block and the set of data related to the ledger. After receiving the block from another node, it takes its place in the general ledger, according to its number. Thus, the capacity of the network can be saved significantly.

During synchronization, only the number
s of blocks undergo checking. In case there is no block at a given node, it is downloaded and saved to a hard drive.

Thanks to this approach, all the units of the system always contain only up-to-date information. We have named it the LL, i.e. the latest ledger. The LL is automatically created by the node responsible for the formation of the ledger after the consensus is found. The block is sent to all the nodes in the system, so all the nodes keep the ledger up-to-date. All the network nodes are interconnected, continuous and incessant exchange of information is conducted between them (transactions and blocks). Thus, a certain set of transactions is formed from the blocks, which will further be added to the ledger. Meanwhile, each server forms suggested sets as candidates for other servers. After checking, a decision is made on whether to add them to the ledger or not.

Eventually, the ledger data is saved multiple times on a great number of servers (separate system nodes), having all the information highly protected. The more nodes in the system, the better it is protected.

4. Consensus of the network

The Rasmart consensus is the method of group decision-making whose aim is elaboration of the final decision acceptable to each and every node of the network.

4.1 Comparison of consensus types

To compare different consensus types, we first need to describe the principles of the decentralized Rasmart ledger:

Availability of the ledger. All the nodes can record data in the ledger and read it from it at any moment

Modifiability. All the nodes in the network can make changes

Accordance. All the nodes in the system correspond to each other, which means they see one and the same version of the ledger

Division-tolerance. If one or several nodes experience a fault, this can not anyhow affect the operation of the system

4.2 The notion of the main network node and recording network node

All the nodes in the network are decentralized, and none has advantages over the others.

In order to ensure the necessary level of data storage security, increasing the speed of transaction processing and further information transferring, Rasmart platform uses a combined protocol of its own design.

The recording node signs a formed transaction block and saves it in the ledger, after which it sends the formed block to all the participants in the system. Each participant in the system records the received block in its ledger and sends the hash of the recorded block back to the recording node. The recording node forms a new list of the main node and trusted ones, and sends the formed list to all the participants in the network, thus creating a new round of Rasmart platform.

4.3 Equipment of network nodes

Our goal is to create a platform featuring the fastest transaction processing, which is why we suggest using material stimulation of maintaining network nodes in the best conditions by means of powerful servers and a high Internet connection capacity.

As a material compensation, each node owner will receive a reward in the Rasmart currency, which is a part of the fees collected for the transactions of the given processed ledger. The remainder will be distributed between the trusted nodes (those that participated in solving the BFT consensus). The percentage may be changed, set out to the system of rate formation by the way of federative voting among separate network nodes, after the ICO, in not less than three years.

Thus, we stimulate the sever owners to maintain the given ledger with use of much more powerful equipment, because the higher the data processing speed is, the more a certain server owner can eventually earn.

4.4 Achieving a consensus

The main system node plays the role of the accumulator of transactions and the sender of those to trusted system nodes, trusted nodes validate transactions, choose one recording note which signs the formed transaction block and saves it in the ledger, after which it sends the formed block to all the participants in the system. The processing of the received information and the formation of the latest up-to-date ledger on the basis of it is the achievement of a consensus. And the result of formation of the latest up-to-date ledger is the solution of the consensus.

This process may be divided into several stages:

• Search of the main node

• Formation of the list of trusted nodes

• Reception of the transaction list, formation of the list of candidates for further addition to the ledger

• Processing of the candidate list (voting among nodes)

• Deletion of unconfirmed transactions or those which a negative confirmation has arrived for from the candidate list

• Formation of the list of confirmed transactions to be added to the ledger

• Addition of the transactions to the ledger with the hash code of the block containing the transactions?

• Sending of the ready block with the transactions to all the nodes participating in the network, the block will be automatically added to the ledger of all the devices upon reception

4.5 Formation and initiation of the ledger

This process may be described as follows:

• The final participant forms a transaction

• In case all the conditions are met, the user initiates the transaction by calling the needed method in the platform's special software

• The validators' core monitors all the stages of synchronization and the unchanged state of the ledger version

• At the moment of achieving a consensus, all the transactions are collected in a united block

• Each block receives its own number which consists of a mark and a node identifier transformed into a hash code. Then, the block is put in the module of consensus finding

• After the candidate list is made, the hash of the transactions and the block will be recorded in the ledger, it will always be possible to check the source's authenticity on the basis of those

• The given hash is a sort of unique block signature, in addition, it contains the detailed information on how exactly and when the given block with the transactions was created

• After a consensus is found with the help of the federative algorithm, the transactions in the block will be transferred to the validator's core for consideration, to further be recorded in the ledger

4.6 Transactions not included in the ledger

All the transactions that have not eventually got in the list of the ready ones will be marked as declined. The information on this event will be immediately shown to the initiator of the transaction. The transactions not included in the ledger are declined.

5. Transaction processing

5.1 Transactions

A transaction is the minimum unit of the system which carries information on the execution of the contract methods or direct transfers of assets without the need to create a smart contract. Further, the result will be placed to the peering network.

5.2 Finding a consensus

In our system, a federative model of consensus solution finding is applied by the way of voting among trusted nodes-validators. In addition, algorithms of consensus solution finding are applied - the algorithm of the finite-state machine arrival. The consensus itself works in cycles, and, for each cycle, transactions are extracted from the network and put into the pool. Then, all the transactions are sent to trusted nodes with the purpose of getting a response. In case the response is received, a request is sent for such a transaction, for that validator to be added to the ledger. Upon reaching the end of the chain in the form of a consensus and the full confirmation of the legality of the transfer, the transaction will further be given for validation with a mark for its further addition to the ledger.

5.3 Transaction processing

In order to achieve the system's decentralized state, each server should not only have a spacious depository for the ledger, but be a full-fledged productive processor of all transactions.

In our system, we use such a notion as "the system's core". What we mean is the data processor which carries out the assigned tasks regardless of the operability of the other participants in the system. At the input, each core gives a result, negative or positive, or an error, etc. Besides the main data set, the response code is always contained in the system's core. Such a structure may be needed for the maximally high speed of each process, provided that they should operate independently of each other.

5.4 Structure of records in the ledger

In order to achieve the necessary level of the system's productivity, we suggest using the database of the ledger without the need to form Merkle tree-like hashing from the hash code of the previous block and the result of the transaction carried out.

Merkle tree is the type of hashing applied in order to check the entirety of data with the purpose of reception of a unique identifier of the chain and recovery of the correct sequence of blocks. Any information received is divided into separate blocks which undergo individual hashing with the help of Leaf Tiger Hash. Then, Internal Tiger Hashes are calculated out of each pair of hashes. In case a hash has no pair, it is transferred further unchanged, into a new chain. This procedure will be repeated until only one hash is set out.

Merkle tree significantly slows down the operation of a ledger built on its basis, the speed of transactions becomes so low that it certainly can not be called comfortable. Besides, due to the low optimization level, the computational resources are heavily loaded, which means a lot of powerful expensive nodes will be needed to maintain a high speed, which may finally not let it break even. We believe it is absolutely irrational to use a data depository this way.

5.5 Structure of the Rasmart ledger

We suggest using a ledger of transactions in the Rasmart system, instead of Merkle tree.
Each record in the Rasmart ledger consists of the hash code of the block of transactions which will be approved for addition to the list of candidates for inclusion in the ledger. In addition, the node identifier will be included in such a record, along with the formation time.
In a ledger record, there is the direction of a transaction as well as its initial and end accounts, the number of units for withdrawal, the type of depositing and the number of units which will finally be deposited. Such an approach gives the ability to significantly speed up transaction processing, increase the level of complexity of illegal changes to the ledger, eliminate the ability of retrospectively changing any records in the ledger. That is, the information processing is carried out in a maximally fast and precise way.

5.6 Size of the block

The cycle of searching for the main and the trusted nodes is a unit of time, and the length of the cycle is calculated on the basis of the complexity of the network itself. Per a unit of time, there are N transactions in it, formed and transferred for further processing from the moment the previous processing cycle ended to the beginning of the following one. The N transactions selected from the network will be put into a block. The size of it depends on the overall number of transactions in it.

5.7 Search for parties to the deal

The Rasmart peering system may be represented as a graph whose nodes are user accounts and the edges are the transactions carried out in the system, which connect the two nodes of the account. And, because all the edges have an end and a beginning, a directed graph can be built on the basis of those.

If we take the following conditions as an identical equation:

- Each transaction carried out has both a sender and a recipient
- Any of the nodes of the account will always be connected to the other with the help of a directed edge (i.e. a transaction)
- Each node has a limited number of directed edges

Based on the abovementioned, a conclusion can be made that the graph contains the sought route for meeting all the necessary conditions of conducting a transaction with the purpose of building the end chain.

A simple path is a route in a directed graph that does not have repeating nodes. And, because the graph itself is yet unknown, according to graph theory, we'll have to built the route in an unknown directed graph. It's known that in this class of graphs the length of the traversing will be equal to $\Theta(nm)$ where n is the number of nodes and m is the number of edges. For every graph, there is a traversing of the length $O(nm)$, graphs with the minimally possible traversing length of $\Omega(nm)$ also exist.

Traversing of an unknown graph is when its topology is initially unknown to us, and we can learn it only when moving along the graph. At each of the nodes it is visible which arrows go from it, yet we can only learn which node they come to following it from the beginning to the end. In fact, this is an equivalent of the task of labyrinth traversing by a robot which is exactly inside the labyrinth but does not have a plan of getting out. If the number of states is strictly limited, the robot is a finite-state machine. Such a robot is an exact equivalent of the so-called Turing machine where the tape was replaced by a graph, and its squares were not only connected to the nodes but also directly to the edges of the graph.

5.8 Data transferring channel

In order to ensure the sufficient network security level, data is transferred between validating nodes in the encrypted form, and each connection between nodes is ensured by a low-level connection based on the network library. If an error or fault happens during data transferring, the flow is automatically interrupted and a record of the error is first added to the logging system and then to the log file. All data is transferred through typed variables. All variable data undergoes encryption by a symmetric RS4 algorithm. And the work of the algorithm is carried out with a common secret key, the key itself is transferred during connection between nodes, in the encrypted form.

5.9 Action in the system

An action in the system is any transaction which characterizes a simplest payment or asset transfer from one account to another, or data transferring to a validator for further finding of a consensus in the system.

In order to avoid duplication of a transaction in one block, with one and the same identifier, a decision is made in the system that the only valid and correct identifier is the transaction that arrived for processing first. In the validator's system, there is already a record of that the transaction for the given account has already been carried out, so no consensus may be found. So, the problem of double spending is soluble.

After the transaction is carried out and the validator receives the information of its confirmation, the data on the change of the state of the ledger will be automatically sent to all the nodes in the list, the ledger will eventually start synchronizing. In order to always have an active ledger of transactions, you need to conduct synchronization of incoming transactions every time, in the ledger of all the nodes. And, in order to successfully accomplish this task, it is necessary to use a separate port for the synchronization (if there is such an ability). All this will allow to substantially increase the speed of the processing of incoming information in the validator's core, thanks to the effective distribution of port loading. The synchronization flow is continuous and cyclic. Besides, the priority of operating memory distribution and CPU loading is below average. Thus, up to 1000 operations may be stored in the memory, along with the states of the accounts connected. All this allow to optimize computational processes and, therefore, the speed of responding to requests.

5.10 Adding a transaction for validation

Adding a transaction to the ledger is possible only upon passing the validator, i.e. when a consensus is found and a white-list of conduction of transactions in the ledger is made. In order to strengthen the security, calls from third-party systems are blocked.

An input parameter is an object that characterizes a transaction carried out. The resulting parameter—ResultValue<0 - conduction is interrupted with an error, the result value - the possible error code/0 < ResultValue - the conduction faced no errors, the result is the number of the record in the ledger.
An input parameter is an object containing a unique mark of the conducted transaction, data on the sender and the recipient, the correspondence of the value, the amount of the transferred asset, the amount of the transferred system information.

5.11 Cost of transactions

In the system, its own Rasmart currency is in circulation, which is used:

- As an internal payment means which is charged for the use of the system
- To conduct currency exchange within the system
- To exchange assets within the system
- To create and further process operations with smart contracts
- To purchase data from third party resources
- To pay for services within the system

Depending on the level of the system load, the cost of transactions can change. We suggest using the material way as a lever for managing the network load.

For the first three years of the system's operation, the cost of transactions will vary depending on the type of conducted operations and the number of transactions. In the foreseeable future, an algorithm will be designed, which, on the basis of the abovementioned factors, will form the cost of a transaction in the fully automated mode.

6. Rasmart's anatomy

Conceptually, Rasmart consists of such elements as the Node (the system's main element), along with a range of secondary modules created for end users (wallet, monitor, oracles), working in an API provided by the Node.

6.1 Structure of the network

The network consists of nodes able to exchange UDP protocol-based messages (the 4th level of the OSI model). Topologically, the network is an overlay one (over Ethernet), where a unique node identifier is used, and routing is carried out along a directed graph which is built from node identifiers. The projection of the overlay network on the real network infrastructure and the routing in it are entirely conducted by the transport level of the Node's software.

6.2 Identification and addressing of nodes in the network

Each participant in the network has a pair of asymmetric encryption keys, for both encrypting with a recipient's public key and signing with a sender's private key.
The identification of a node and its network traffic is carried out on the basis of the hash function of a public key.
For encryption and electronic signing by the Diffy-Hellman protocol, the RSA (Rivest, Shamit and Adleman) algorithm can be used, over the field of prime numbers, with the key length of 1024 bits, or ECDHE (a type of elliptic-curve algorithm with ephemeral keys).
In the second case, for comparable cryptographic precision, a key with the length of 256 bits will be sufficient.

6.3 Organization of the network

The network is a bidirected cyclic graph. Node identifiers are ordered and linked in a ring in such a way that each node has "neighbors on the left" and "neighbors on the right".
Having created a message, a node sends the pack to all its neighbors. They search for the recipient of the pack among their neighbors. If they do find it, they address the message directly to it. If they do not, those that have received the pack from the left send it again to all the neighbors on their right, and those that have received it from the right send it to those on their left.
The excess of traffic protects from UDP losses, faults of nodes on the way of a messages and attempts of pack faking.

The scheme works only when all the nodes are able to receive UDP packs at any time. The main requirement for that is a flat network without routers and hierarchically layered private networks with so-called “gray addresses”.
Nodes outside of the NAT are only able to receive UDP traffic if it comes in response to their request, only from who the request was addressed to and within the period of time which is the time to live of ARP router table records on the route.
If the nodes take such conditions into account and adapt their behavior, they can participate in the operation of the network. Analyzing the incoming and the outcoming traffic, nodes can understand what behavioral model to follow.
A criterion of such analysis is whether a node receives messages from those it has never sent messages to before such received ones. If it is not so, the node organizes its operation in such a way that it receives information in the periods of an “open window” which needs to be maintained.

6.4 DHT

The organization of a “ring” uses the equality of hash distribution, which allows to find “neighbors” in an ordered way and dynamically include new nodes between them.
The DHT algorithm used in torrent networks operates the notion of “metrics”, with the help of which it is possible to complement the neighbor selection mechanism in such a way that, in fact, they are in different data centers, on different continents. This is the protection from the Sybil attack allowing to take control over the entire round of the decision-making block.
Or, on the contrary, taking into account the speed of the node’s response, the metrics can optimize the speed of interaction, which will allow to increase the network speed by means of really close nodes.
In fact, the golden mean must be found, which would be sufficiently reliable security-wise and fast enough when it comes to transaction capacity.

6.5 PEX

The peer exchange mechanism allows to rebuild the network upon connection of new nodes. Being built in the directed graph, a node finds its place and its neighbors redistribute the subsets of their neighbors in such a way that the UDP “wavelength” remains approximately equal.

6.6 Traffic

The network operates at the UDP level. The packs created by a node have a sender, a recipient and a random payload of the command. To create a broadcast message, a nonexistent recipient’s address must be set (such as 0).

Many commands give nodes the ability to register in the network, request necessary information for ledger synchronization, create transactions, send requests for participation in the decision-making block, as well as the commands necessary for such participation.

The node creating a message signs it with a short-lived signature. The particularity of such a signature is the fact that it has a time to live (TTL) and its being more compact, not occupying a lot of space in a UDP pack. Taking the short TTL into account, it is not a security-related problem, but speeds up the checking and processing of such packs, making unauthorized integration of sending of an “alien” pack impossible.

The checking of a short-lived signature is conducted at the network level, declined packs won’t go farther than the first redirection and will not be accounted at higher node software levels.
Large packs are divided into several small ones which are sent as large messages and collected only at the receiving node. This is an additional measure taken to protect them from capturing.

6.7 Structure of the ledger

The ledger is a base of random data, a sample of which is stored and replicated at each node. Additional structures responsible for the entirety and verity of information along with its being up-to-date are built over the data. Technically, the database is organized in the form of a “key-value” depository where the key of a random value is its hash. The local depository for the database is LevelDB, an embeddable database by Google.

6.8 Depository and Merkle tree

A binary tree is built over the records with the data, each node of which is equal to the hash of a paired union of the hashes of the previous level. Thus, all the data tree is “wrapped” into one “root” hash showing the state of all the database. All the tree is checkable and, in case an unauthorized change is made, the hash will change and cease to correspond with the same hashes at the other nodes, which would be the sign of no database entirety. This is a modification of Merkle tree.

6.9 Transaction chain and blockchain

The transaction chain is a list of records describing the changes made to the database and referring to the current state of the database (the root Merkle tree hash). The records are united in blocks signed by an electronic signature of the block’s creator, and include the hash of the previous blocks, forming a single-connection list - the so-called blockchain. This mechanism protects the data that’s already entered from alternation and authenticates those who make changes legally.

6.10 Synchronization and control over the entirety

Because the database is a key-value set, the synchronization of bases is the exchange of single records between nodes. Each time the transaction chain or the Merkle hash stop “syncing”, the place of the altered/deleted records is found and they are requested from the network. This does not only allow to update it with the latest changes but also ensures initial synchronization and recoverability of the state of the base after a long-term absence of a node in the network or an irreversible data loss in the local database. The control of entirety is conducted each time upon the start of a node, as well as in case of a mismatch of the computed state of the base’s local copy with the latest state of the ledger received from the network. Any breach launches a synchronization cycle.

7. Database entities

7.1 Wallets

A wallet is a personal account of a participant in the network. In order to send and receive funds, a participant needs a launched node and a pair of electronic signature keys. The identifier of a wallet (as well as of a node) is the hash of the public key.

7.2 Transactions

Carrying out a transfer, a participant in the network creates a transaction, signs it with their private key and sends to the network. A financial transaction is an entity with the information on the sender, the recipient, the amount transferred, the currency of the operation and random additional data. Exactly those transactions get in the chain of blocks and are part of the mechanism of database correctness control.

7.3 Random entities

Besides the information on a transfer, the ledger can contain any other kind of information of a random format. However, it cannot be saved directly. Each such entity has its owner that is a smart contract and can create, alter or delete data in the ledger.

7.4 Smart contracts

Smart contracts are stored in the database as stored procedures. It is a program code which can be referred to by its identifier. The identifier of a smart contract is the hash of its source code.

7.5 Transactions

The transactions created by participants are not yet part of the blockchain and are not included in the transaction chain. Before that, their correctness must be checked. Such checking is about two things—the sender must have more funds that they're going to send, and the transaction must be signed by the sender's private key. Any transactions that failed the checking are declined, any damaged ones are united in a block signed by the validator and included in the blockchain.

7.6 Solving the problem of "forks"

Blockchain is vulnerable to "forks". Where single-connection lists exist, trees can exist too - nothing stops several blocks from referring to one "parent" block. And this can not be "closed" by the liked to the following block, as it's done in two-connection lists, because a block cannot be altered after being recorded in the ledger.
As well as some other blockchains, we solve this problem by having only one recording block available in the network at any period of time. Which validator will become it is decided in the course of a verification round.

7.7 Decision-making block

The checking of transactions and the saving of new blocks are carried out by the decision-making block. It is a subset of nodes which carry out checking independently of each other and exchange the decision results with each other. In the condition of an untrusted environment (and a decentralized network with nodes controlled by end users is untrusted by default), such a mechanism allows to detect compromised nodes and receive correct results regardless of their existence. The basis of this method is the solution of the Byzantine generals problem, which allows to make right decisions on the basis of partly correct data.

7.8 Implementation of a consensus

A consensus is a decision on the validity of a transaction made after all the nodes exchange the information on whether the transaction sent to them is valid. Upon the decisions made on all the transactions ready for recording, the node which records the block is chosen, after which the decision-making block is formed by new nodes again, and all this is repeated. Thus, a new means of protection from "forks" is introduced - only one node records a new block in the network.

7.9 Paralleling of the decision-making block

Decision-making blocks can be more than one. If all the collected transactions are divided into subsets, all the preparatory work can be carried out simultaneously, and, from cycle to cycle, the right of decision-making will be transferred from one recording node to another. In this scheme, the recording will be almost continuous, and the transaction capacity of the network will increase manifold.

7.10 Division of the transaction graph

Representation of a set of transaction in the form of a connected graph, in which the nodes are senders and recipients and the connections between them have costs equal to the amounts of transactions, will allow to optimally divide the graph into subsets.
For this, it is necessary to have each of the subsets form a graph, in which the resulting balance of each node is non-negative. The task is the solution of the knapsack problem. The purpose of this function is the formation of the list of prepared transactions in such a way that the minimum number of transactions breaking the rule of the general non-negativity of the graph are discarded.
The transactions discarded at this stage will be passed to the next round for additional validation and “packing”.

7.11 Fees

A fee is a method of motivating the participants in the network to spend their computational resources on the maintenance of the network’s functioning. We are interested in having the nodes working with powerful equipment and having high-quality Internet channels. The nodes able to respond faster are more likely to get in the decision-making block and receive the fees for the work done.

7.12 Fee calculation method

In the table below, there is a list of reasons for forming the size of the fee (the price the sender is charged with) for the issuance of a transaction:

The dependence of the fee on the round

I.	• The number of transactions in the round • The more transactions in the round, the lower the fee	• The minimum amount of transactions in a round is 1 • The nominal value is 10000 transactions in a round • The higher limit of transactions processable in a round is set as equal to 65536
II.	• The number of trusted nodes in the round • The more trusted nodes in the round, the higher the fee	• The nominal value is 101
III.	• The physical size of the transaction • The more “disk space” the transaction occupies in the blockchain, the higher the fee	• The physical size of a transaction can range from ... to .... • There are no restrictions as of the nominal value
IV.	• The “transactional” activity of the sender • The more transactions the sender generates, the higher the fee	• The lower limit of such transactional activity is 0 • The nominal value is 5000~10000 • The higher limit of such transactional activity is set as equal to 1310720 transactions per second

8. Smart contracts

A smart contract is a program code executed upon the creation of a transaction the recipient of which is its identifier. The identifier of a smart contract is the hash of its source code.

The publication of a smart contract is the creation of a transaction containing its source code. A smart contract is added to the database and executed when a transaction with its identifier as the recipient appears in the network. Being executed, a smart contract creates, modifies or deletes some entities in the database, thus maintaining its state between calls.

As well, a smart contract can read information from the blockchain and create transactions. An example of this is a smart contract collecting bets on a sporting event and, after the information on the event appears in the network, distributes the rewards proportionally to the distribution of the overall formed “bank”. The information its decision is made on the basis of is added to the system by another smart contract. Thus, any facts in the world may become entities of a smart contract - currency rates, sporting event results, facts of notarization, etc.

Smart contracts are written in the Lua programming language and are equipped with all of its capabilities inside of an isolated virtual machine. The standard libraries of Lua are disables, and a smart contract cannot interact with the outer world.

All the necessary functions are received by a smart contract only in the framework of a specially written SDK.

When receiving a transaction the recipient of which is the identifier of a smart contract, the participants in the decision-making block execute its code, upon completion of which they exchange the hash of the state of the virtual machine including all the variables and created entities. If the results of the smart contract execution are the same at each participant in the decision-making block, the transaction is saved in the blockchain, and all the changed ledger entities are synchronized in the network.

A smart contract is not algorithmically restricted, which brings the risk of a network failure caused by a parasitic or erroneous code leading to an eternal cycle. In order to avoid that, a smart contract is given a strictly limited period for time for execution, after which it stops and the transaction is considered invalid. The fee for such a transaction is charged anyway and distributed between the participants in the decision-making block.

The entities a smart contract can alter must be initiated by itself. Any other entities can be read by it, but not recorded.

Smart contracts don’t have any input or output, they can’t fulfill network requests. All they can do is read any entities from the local copy of the ledger, as well as create, modify and delete their own entities.

8.1 Lua interpreter

The Lua interpreter is an isolated virtual registry machine, in which a program is subsequently executed. Inside the interpreter, the resources of the surrounding operating system are not available, all libraries (including the standard input-output one) must definitely be provided. This way, the virtual machine will certainly only have access to the specially written SDK and will not, for example, be able to record files or establish network connections.

8.2 Cost expression

Rasmart cryptocurrency is the cost of one contract unit, which allows to compare two absolutely different units to further achieve a consensus. The Rasmart cryptocurrency does, in fact, play the role of a connector for conduction of inter-asset transfers. Such an approach became possible thanks to the fact that every asset is liquid in relation to the Rasmart cryptocurrency.

8.3 Data sources.

For the correct and fully functional operation, maximally precise checking and provision of additional clarifying information, the Rasmart system uses third-party data suppliers. This is not conditioned by the openness of information on a party to the contract. For example, that can be the reception of a credit status of a borrower to make the decision on giving a loan.

The system may call an integration bus, which is what gives it the ability to work with third-party information systems. The given bus generates a request to a third-party system by itself, in the form of the provision of information on other participants, certainly, not free of charge. The Rasmart cryptocurrency is used as the means of payment.

Such a request is sent in the encrypted form only, to the addresses provided by the informational systems. Any response of the service, containing even the minimum amount of information needed for making a decision, is considered resultative.

8.4 API

Each node is a REST API for execution of requests in the network. Such requests may be those for transaction creation, smart contract publication, reading the transaction list and reception of values from the ledger.
An API accepts requests only locally, for any software which aims at interacting with the network and the distributed ledger it is necessary to have an own local node.
A wallet, a “table” app for transfer conduction, is an example of third-party software interacting with the network through the API node installed right next to it, at the same workstation.

9. Security and possible threats

9.1 Intermediary’s attack

Solved by the electronic signature of a transaction, which authenticates the sender.

9.2 Attack 51

As the network grows, the probability of getting in the validation round is negligibly low.

9.3 RSA key hacking

The reliability of RSA keys of the size of 2048 bits (256 bytes) has been proven mathematically. An attack by Shor’s quantum factoring algorithm is not a threat because of the absence of a quantum computer.

9.4 Sybil attack

Possible only in case of 100% surrounding by compromised nodes. One correct node is enough to detect an attack of the kind. The process is also complicated by entropy elements in terms of choosing the trusted validating nodes.

9.5 Double spending problem

Having only one block recording node at a moment protects the network from “forking” and, therefore, eliminates the possibility of spending the same funds twice.

9.6 Traffic capturing and faking

One and the same pack is copied multiple times and goes different ways. In order to block the traffic, control should be taken over the entire environment. In case of the Internet, that’s unlikely. As well, packs are equipped with a lightweight sort of electronic signature sufficient for authenticating short-lived packs and those that “spoil” as time goes.

9.7 Changes to the local depository

Any interference in the local depository will lead to recomputation of the state of the base (the Merkle tree root hashes and the latest block). If the data is incorrect, the network will withdraw from any interaction with the compromised node before it is restored.

10. Plan of implementation

10.1 Plan of the technical implementation of the project

10.2 ICO

Road-map

Stage 1
2017 Q1 - 2018 Q4

- Idea generation
- Organization of the team
- Development of the functions of the decision-making module

Stage 2
2018 Q3 - 2018 Q4

- Prototyping of the visual part of the project
- Private platform presentation, sale of tokens
- Testnet optimization
- Search for partners

Stage 3
2019 Q1

- Refinement decision module
- Optimize API usage
- Implementing a cluster of multiple entry points
- Basic functionality of smart contracts
- Rebranding the visual part
- Wallet implementation for various platforms
- Preview version of the service data collection controller
- Public tokensale

Stage 4
2019 Q1 - 2019 Q2

- Full implementation of the decision module
- Developing a split API model
- Database optimization (big data, high load)
- Implementation of the basic version of the SDK for the development of smart contracts
- New design of loading large amounts of data in explorer
- Implementation of an additional layer of data encryption
- Refinement of the functional of the collection of reliable financial data

Stage 5
2019 Q2 - 2019 Q3

- Decision Module: Parallel Processing of Pre-Transaction Pools
- The release of providing trusted data in a secure module for collecting reliable financial data
- Refinement API functionality
- Release of the release version of the smart contract SDK
- Refinement of the web wallet security system
- Wallet release version

Rasmart initial coin offering

Token details:

Token name: RAS

Token type: Utility

Token features:

Token price: $ 0.23

Hardcap: $ 50 000 000

Max tokens: 500 000 000 RAS

Minimum buying transaction: $

Accepted currencies: ETH

Public sales:

500 000 000 RAS - Initial issue of tokens

225 000 000 RAS - ICO

67 000 000 RAS - marketing and advisors

90 000 000 RAS - platform support

59 000 000 RAS - reserve fund

59 000 000 RAS - team and associates

ICO schedule

Stage name	Duration, days	Min. purchase, USD	Bonus, %*
Private sale	60	10 000	30
Pre sale	20	5 000	20
Main sale	10	500	5-10

* – for large purchases at all stages it's possible to give an extra bonus, for which tokens will be taken from Company fund/Team allocations

10.3 Rasmart cryptocurrency

After the launch of the operating version of the system, a fixed amount of tokens will be issued - 500 000 000 units of RAS. Further, one will be able to exchange them for ERC20 tokens issued during the ICO. The exchange may be carried out at a fixed rate, i.e. 1 token of the ERC20 standard = 1 unit of RAS.

After the ICO, it is planned to increase the speed of transaction processing up to 400000 per second.

The existing smart contract mechanism is based on the simple but fairly productive Lua programming language. Its security has not been doubted. The team is planning to conduct thorough auditing of the language's security for financial operations, as well as investigation of the protection methods of its virtual machine.

We are planning to significantly increase the speed of the network by developing routing mechanisms of the second level, which will help speed up finding the fastest routes for each node.