Abstract VEDA

VEDA is a technology for converting any securities and data into digital format.

VEDA is the bridge between the traditional and digital economies. It works in any existing legal field.

The using of technology will allow organizing cross-border investment attraction without the participation of financial intermediaries: banks, funds and depositories. There will be created a system where investment will be available to an unprofessional investor. It becomes possible to circumvent any restrictions on the borrowing of digital capital, remaining within the framework of market principles. The technology solves two main problems - the fear of users of events in the system of international regulation of networks and the Internet (GDPR, decentralization and cryptography), the second problem is the storage of information about valuable data: family, finances and medicine.

VEDA technology eliminates the main problem of privacy in the blockchain. entering data into web platforms without leaving traces or storing data on their servers.

The client works with the technology through a user application. There are also public nodes that display the status of the network.

The technology application industry should be divided into b2b / b2c / b2g interaction. Business is able to get an optimized resource for storing confidential information and sharing it with partners / government. Clients receive a line of products for working with confidential information with protection against unauthorized access. The state can get the same as a business, but in the form of a boxed solution that will translate into digital / process / store / exchange data and documents.

Introduction

VEDA technology is a combination of a distributed registry, encrypted communication channels, token files and smart contracts.

VEDA technology is based on the VEDA network - many VEDA clients working on the nodes that make up the VEDA blockchain. VEDA clients are user applications, as well as nodes, designed to form and store a distributed registry.

The main advantage of the system in working with information is the absence of a single point of failure and information collection centers, which become targets of hacker attacks.

VEDA nodes provide the following functionality:

1. account management and authentication;
2. database maintenance;
3. asynchronous data exchange between user applications;
4. synchronous exchange between nodes and
5. interaction with VEDA Virtual Machine.

The result is a technology that is user-friendly and high-speed transactional blockchain infrastructure.

The value of information has grown up in the modern world. Distributed registries resolved the issue of confidence. But the problem of privacy and data security on the network hasn’t been resolved.

System requirements

The global introduction of VEDA technology requires a flexible platform to meet the following requirements:

- Ensuring data confidentiality
- Access to data should only be the recipient and sender.
- Integration into corporate systems.
- Functional for the secure issuance and accounting of digital assets.
- Companies that introduce new technology in their business processes require flexibility to expand and improve functionality.

VEDA Technology Bases

FILE TOKEN

The VEDA network uses a VDN token with a unique structure. It is a digital entity which is stored directly by the user on his device. But in VEDA, this is a file which is stored in a cryptocontainer of its own implementation.

Each token has its own unique identifier - VedaID - VID.

The technical implementation of the VEDA token allows you to trace the history of each file token. To distinguish a specific token from all others, it can be assigned a certain “color”, which will denote the unique properties of the token, linking it to various kinds of assets. The basic infrastructure of the VEDA network will be used for operations with such tokens.

Benefits of VEDA Color Tokens:

- *Versatility. Colored tokens can be used to work with any type of data, assets and contracts.*
- *Availability. The use of “colored” tokens allows you to get rid of the participation of a third party in conducting transactions with assets.*
- *Convenience. “Colored” tokens allows you to transfer data or ownership of any asset quickly and safely through the use of a smart contract mechanism.*

The color-token in the VEDA platform is proof of ownership of any assets. Anyone who has a “color” token, which contains proof of ownership, is the legal owner of the asset. “Colored” tokens are stored in the user's wallet, and the blockchain keeps the history of operations with this token - information to which wallet it [token] belongs to.

Another purpose of the VEDA token is that it can put in itself confidential information of any type (document, image) that will be encrypted. Only the recipient account token will be able to decrypt it after the transaction is confirmed by voting nodes.

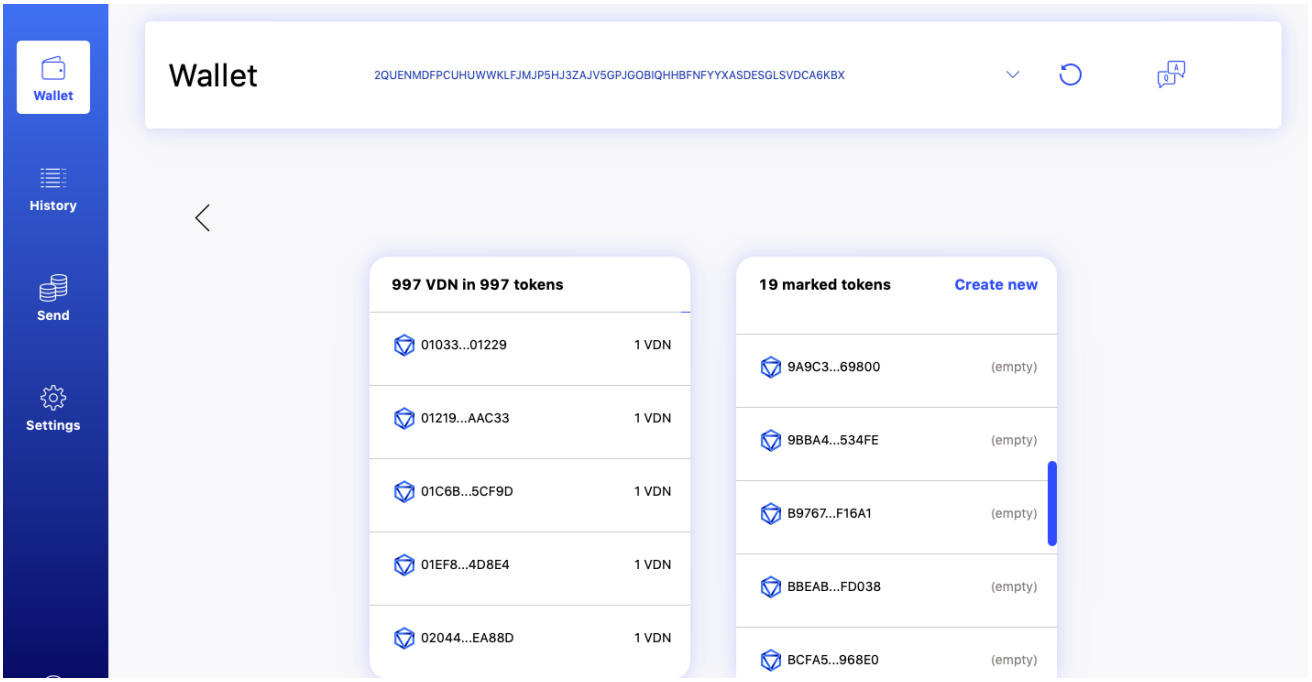


Figure 1. Types of tokens

CRYPTOCONTAINER

All tokens secure in a reliable storage (cryptocontainer) in encrypted form on user's devices. Cryptocontainer also stores SSL certificate and an account private key.

Even if an attacker can steal user tokens, he still will not be able to dispose of them, since the distributed registry stores information about the tokens belonging to a specific user. Each VDN contains information about the address of the account, which it belongs, and the transaction ID during, which it was received. To carry out this attack, an attacker will need to fake part of the transactions in a distributed registry. That's technically impossible.

The user will be able to return the stolen tokens to his wallet through the special functions of the wallet by sending a request to the VEDA network.

In all existing blockchain platforms, the user is completely unprotected from viruses and trojans that can send transactions to transfer funds to a "strange" wallet on behalf of the user. VEDA protects users from such attacks by providing the ability to store cryptocontainers with tokens on portable storages. Malware will not be able to send a transaction to transfer tokens to the attacker's wallet, as they simply aren't on the wallet.

Also, VEDA users will be able to transfer tokens between their wallets, which are installed on different devices. Multiple wallets can be registered on one device.

In addition, the tokens in the files are signed by the electronic signature of the owner's wallet. Token transfer transactions themselves are also signed by wallets before being sent to the network.

Each VDN token has a creator signature, which is the token issuing server. It is impossible to fake a token in the VEDA system, because for this you need to know or pick up the keys of the creator.

SSL CERTIFICATES

The SSL certificate mechanism is used to organize encrypted transmission channels, as well as for signing transactions, tokens and contracts in the VEDA network.

There is no certification authority in the VEDA network, and the users themselves issue certificates using the VEDA application. Each certificate has a unique serial number (SID) throughout the network, which is created by applying a hash operation to the public key of the electronic signature of the wallet. The hash-sum of SSL certificates after creation are placed on the VEDA blockchain system.

- The structure of the entries in the certificate registry:
- SID | UID | Public-key | HASH-sert | DateTime | SIGN-old | HASH-str
  - where HASH-str – hash record amount,
  - fields DateTime | SIGN-old – not filled

- The compromise of any node doesn’t lead to hacking of accounts, since the password is not stored in any form on the server.

The blockchain is formed by G-Node nodes.

A copy of the chain is stored on the G-Node nodes and on all other nodes of the VEDA network.

The user sends a certificate to the node with an asymmetric encryption session key to confirm the identity and set the encrypted channel, during the connection to the network.

- The node, upon receipt of the certificate, first checks the signature, and then searches for the SID (certificate ID) in the SSL Certificate blockchain.
- If there was a request to register a new wallet, then in the event of a reliable signature and lack of certificate information with such SID in the blockchain, the sent certificate is placed in the certificate pool of the nearest G-Node node.
- If there was a request for authorization of an existing wallet, then in the event of a reliable signature and the presence of certificate information with such SID in the blockchain, a positive response is returned to the user and a secure communication channel is established. Otherwise, a negative response is returned to the user and the connection with him is interrupted.

Algorithm for forming a chain of SSL certificate blocks

The block of the chain of SSL Certificates consists of one certificate in order to quickly organize the search by SID of the requested certificate during user authorization in the VEDA network. Block storage is organized in KEY-VALUE Storage.

Key is a concatenation of the SID and attribute (Add or Revoke). Value - hash of the certificate, hash of the previous block, date of revocation of the certificate and signature of the owner.

After receiving the SSL Certificate (adding or revoking) from the user's wallet, the node verifies the signature of the certificate, and if it is correct, the node sends it to the G-Node node, where it is placed in the SSL Certificate pool. The G-Node node that receives the certificate sends it to all other G-Node nodes. The process of adding another block to the chain is named “round”. A G-Node node called SPEAKER is selected for each round. This node selects the next certificate from the pool, forms a block and sends it to the G-Node nodes. All nodes, that receive a block, check it and send each other CHECK YES or CHECK NOT messages. In the next step, each G-Node node counts the CHECK YES and CHECK NOT messages, and if 2/3 + CHECK YES messages are accumulated, the node sends PRE-COMMIT YES to everyone, otherwise PRE-COMMIT NOT. In the last step, each G-Node node counts PRE-COMMIT YES and PRE-COMMIT NOT messages, and if 2/3 + PRE-COMMIT YES messages are accumulated, the node adds a block to its copy of the chain, otherwise the block is discarded. Another SPEAKER is selected according to the Round-Robin algorithm to add the next certificate.

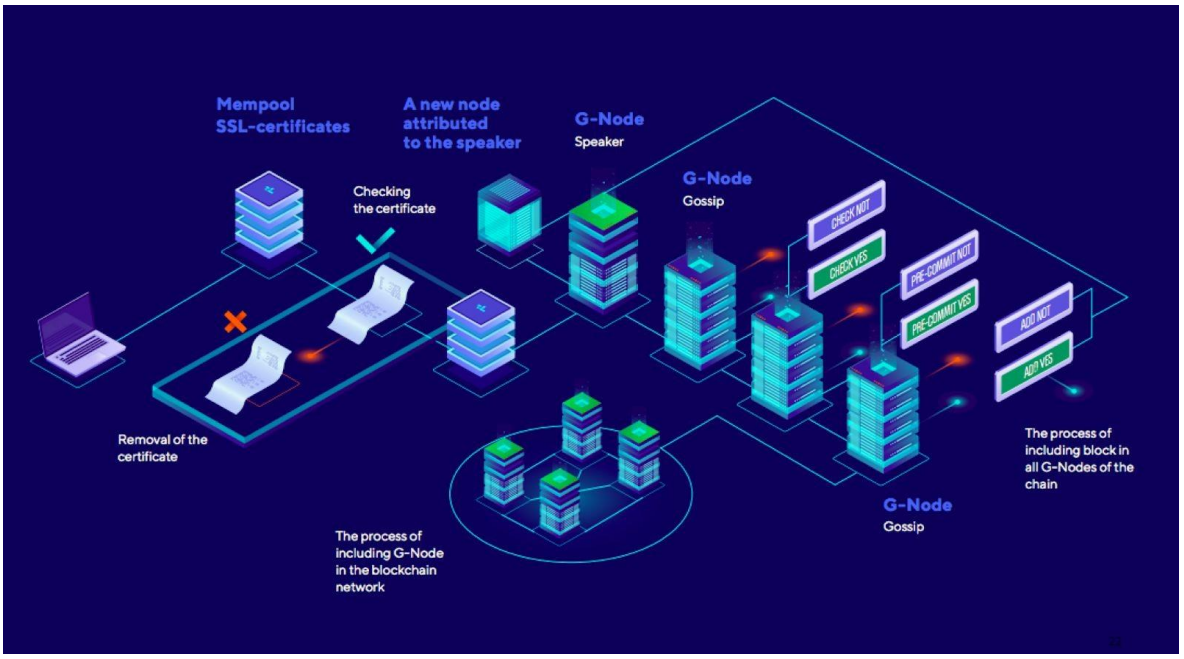


Figure 2. SSL Consensus Framework

CONFIDENTIAL DATA CHANNELS

Encryption of communication channels in the VEDA platform occurs between all objects (nodes, applications) of the network in pairs. For each pair of subscribers, its own symmetric key is generated using the Diffie-Hellman protocol. It is used to encrypt messages or generate electronic signatures. Purpose is in the distribution of keys. It allows two or more users to exchange a key without intermediaries, which can be used for symmetric encryption. According to this algorithm, a key node-wallet and node-to-node symmetric encryption key are generated.

DISTRIBUTED REGISTER

The distributed registry of the VEDA network consists of several parts:

- DAG transactions;
- chain of archive transaction blocks;
- account SSL certificate block chain.

The chain of SSL certificate blocks stores information about registered and revoked account certificates.

DAG is used to store confirmed transactions that were created and sent to the network from accounts. The vertices of the DAG are transactions signed by electronic signatures of the sender account and the host account of the transaction, which describe the movement of tokens between VEDA network accounts. The chain of blocks of archive transactions is used to reduce the height of the DAG and, accordingly, to reduce the time it takes to bypass it.

Figure 3. Blockchain consensus diagram

VEDA VIRTUAL MACHINE (VVM)

The Veda virtual machine, like the interpreter itself, is written in C and C ++, just like the platform itself.

Smart contracts are scripts in the LUA language. Since the Lua language is intended for users who are not professional programmers, the process of writing a contract does not cause much difficulty and time.

VVM is installed on the same hardware as the VEDA node. The interaction between the VVM and the node is via encrypted communication channels.

SYSTEM SECURITY

- The complex structure, the presence of the signature of the creator and owner in each token don’t allow tampering it;
- VEDA uses the GOST R 34 encryption algorithm, which hasn’t been hacked;
- Tokens are stored only in cryptocontainers on user devices or on portable hardware media separate from the wallets themselves, which significantly complicates the possibility of their theft using viruses or trojans, and user token IDs are stored in transactions recorded on the blockchain;
- Double protection of VEDA tokens - it’s not enough just to steal tokens from a user's wallet by breaking it, you need to fake transactions on the network in order to change their owner, which is almost impossible. For the organization of encrypted transmission channels, as well as for signing transactions, tokens and contracts in the VEDA network, the SSL certificate mechanism is used

ACCOUNTS

Accounts on the VEDA network divided into external and internal.

External accounts are created by users when registering a new node or application Wallet. The address of this account is the hex-sequence obtained during the conversion over the public key of the account.

Internal accounts are contract accounts created by users and uploaded to the blockchain. The address of this account is a hash from the contract code.

VEDA uses elliptic curve cryptography to ensure the confidentiality, credibility and reliability of all transactions. Each external account (Wallet account and Node account) is a pair of keys - private and public. External account associated with the changing state of the network, which is updated when transactions are confirmed by the network.

The status associated with each account includes the following elements:

- current account balance
- number of outgoing transactions from this account

The address of the VEDA account is a hex-sequence of 64 bytes in upper case encoded by the Base58 algorithm. This sequence is formed by the hashing operation on the public key of the account by calling the hash-function.

All addresses of external VEDA network accounts are stored in the SSL Certificate Blockchain on network nodes. You can send a VEDA token to any valid address, if it is present in the blockchain of SSL certificates. If the transaction contains an address that isn’t registered on the network, the node will not accept this transaction and tokens will not be transferred.

CONVERT A PUBLIC KEY TO AN ADDRESS

To convert the public key to an address, the following steps are performed:

1. We execute a 256-bit public key hashing algorithm.
2. Convert the hash sum obtained in step 1 to hex-format.
3. Convert the string obtained in step 2 with the Base58 algorithm in upper case.
4. From the values obtained in step 3, we take the first 64 bytes.

Is it possible that two different public keys will have the same address? After all, if such an address contains a VDN, then an attacker can withdraw funds from such an account.

In the VEDA system, this is impossible, since such a situation can only occur if an attacker can get a key pair identical to the account that he wants to take possession of. Because the probability that the hash algorithm for different public keys can produce the same hash is zero.

CONSENSUS ALGORITHM

As mentioned above, VEDA uses the only decentralized consensus algorithm that provides high performance for transaction confirmation - Delegated Proof of Stake (DPOS).

The nodes selected for voting to confirm transactions use their own implementation of the BCA (Byzantine Consensus Algorithm), which guarantees that at any time there is only one consistent version of the blockchain. This algorithm doesn't require mining and can ensure the safe operation of the network in the presence of at least strictly more than ⅔ + 1 trusted (honest) nodes.

TRANSACTIONS

The transaction body is json array. Each transaction has extension, which can contain any digital data. Each transaction has its own unique ID (TID).

Transaction is as a data structure is validated by digital signature and is sent by VEDA Network, being put in DAG. Every transaction contains TID or TIDs of previous transactions.

Transaction associates VDN tokens with one or more accounts. Transaction isn't encrypted

Number of transactions got by node can be rather big. Network node chooses next transaction for building DAG by transaction's priority. High priority transactions come to DAG faster. Transaction priority is based on commission size – higher commission grants higher priority.

The DAG has several root nodes. Every root node is transaction of emission center having purpose to create new tokens and send them to some VEDA account (trn_create). The root nodes don't have parent nodes. They are signed by digital signature of the emission center's account. The address of emission center is specified in genesis-block of the SSL certificates blockchain. No one trn_create type transaction made by another address can be verified by VEDA Network. All transactions except trn_create type ones have links to parent transactions and theirs hash-sums. General information in the chain is data on the tokens participating in them. Every token contains TID to determine transaction. Due to the transaction, it is now in the account of its owner's account.

When new transaction is created, information about sending tokens is added to its body. At this moment, the information about parent transactions in DAG is being taken from tokens. Thus, every transaction has the hash-sum of one or more previous transactions for preventing the substitution of tokens "travel history" between accounts of the network.

JSON_TRANSACTION_BODY_NUM	“NUM”	Number of incoming transaction by specified account
JSON_TRANSACTION_BODY_TIME	“TIME”	Transaction creation time
JSON_TRANSACTION_BODY_SENDER	“SENDER”	Sender account address
JSON_TRANSACTION_BODY_RECIPIENT	“RECIPIENT”	Recipient account address
JSON_TRANSACTION_BODY_VID	“VID”	Array of token data
JSON_TRANSACTION_BODY_TYPE	“TYPE”	Transaction type
JSON_TRANSACTION_BODY_CID	“CID”	Contract account
JSON_TRANSACTION_BODY_TID_REF	“TID_REF”	TID of linked transaction
JSON_TRANSACTION_SENDER_SIGN	“SENDER_SIGN”	Sender account sign
JSON_TRANSACTION_RECIPIENT_SIGN	“RECIPIENT_SIGN”	Recipient account sign
JSON_TRANSACTION_NODE_ID	“NODE_ID”	Address of node received transaction
JSON_TRANSACTION_NODE_SIGN	“NODE_SIGN”	Sign of node received transaction
JSON_TRANSACTION_EXTENSION	“EXTENSION”	Extension

Table 1. Transaction structure

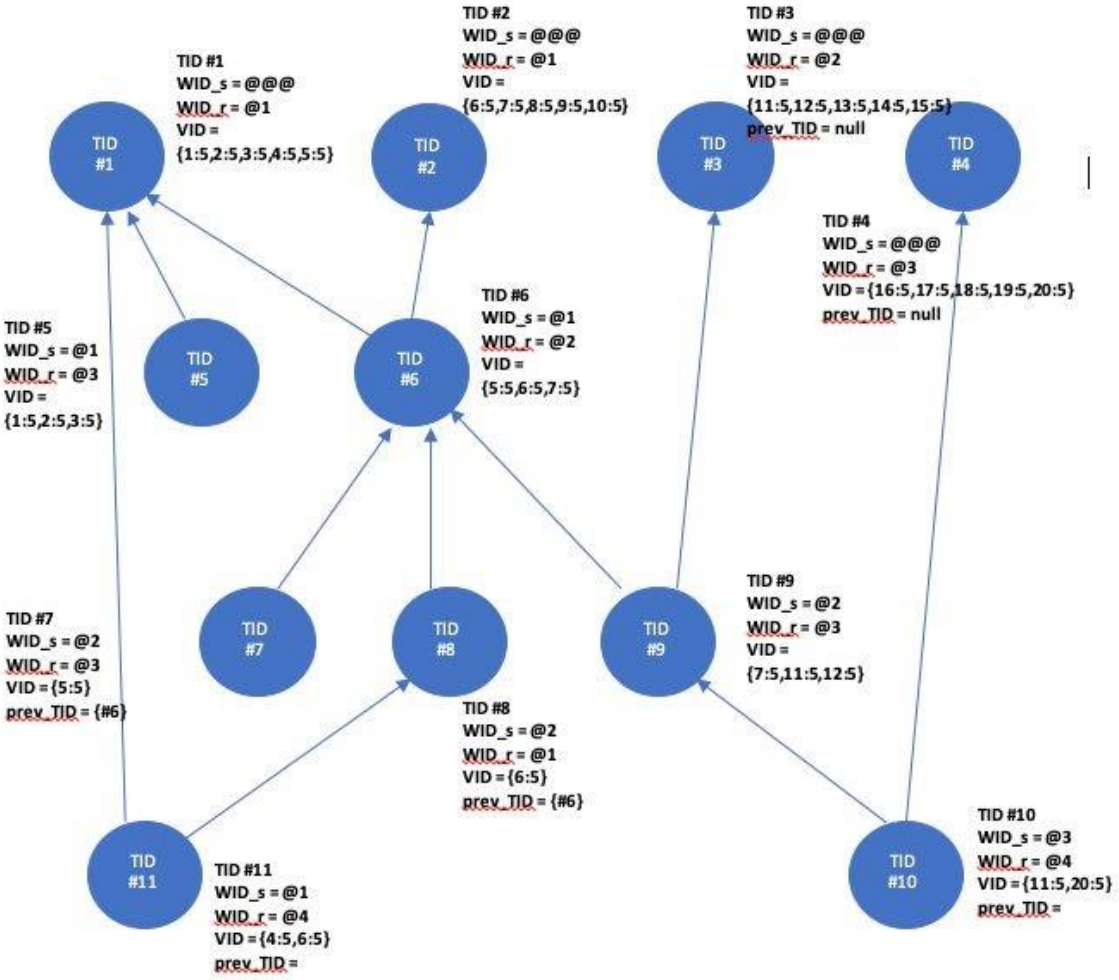
Example of transaction body in json format:


```
"BODY":
{
"NUM": 1,
"RECIPIENT": "2KMGJSRGF9X3AA1ELPKHITQ2AM9KDEXHQFMZZB8Q7GNYEN2AUZLLYJXYW5WC3HAG",
"SENDER": "22P69RENUCTB3GME92JDDZ8CEMFE4ME7NUQKQNWMZHESQVFKLV5RG7QPAKQNBGFR",
"TID": "22P69RENUCTB3GME92JDDZ8CEMFE4ME7NUQKQNWMZHESQVFKLV5RG7QPAKQNBGFR_1",
TIME": "1539700569.452466",
"TYPE": 4,
```

```
                F9C52BA095B7BD74ADBBD29AE268502C078CA7DBE0B881", "MASS": {
"VID":          "0201B8E57907E677EEF470E71D00E344964092CA5FEB90F9558DDBB19F4DB19C": 1322134912,
                "05B9DCE980A5C63F5C98569F6DCA172C94FF0E9AD4B7A45454CC001C78BDF50F":1353134912
                },
                "TID": "2KKV9JWVJDS3BQJL3KKJ9QHLJZZ77UCPODRKTERBIAEECHQJQDSPPGGTUAURMPQZ_8"
                },
], "VID_CMS": [
{
 "HASH": "2034D7AE7F986B1E2EF9C52BA095B7BD74ADBBD29AE268502C078CA7DBE0B881", "MASS": {
"0608189ABDDCBF838484B067B57CDA5CFFD96306C175AC91ECDCA8742677268F": 1316134912
},
"TID": "2KKV9JWVJDS3BQJL3KKJ9QHLJZZ77UCPODRKTERBIAEECHQJQDSPPGGTUAURMPQZ_8"
},
]
},
"NODE_ID": "2KMAZMEKHLCSNNAKUFKQWYDZDAYHH8DYCFTMUC97JSHEX7OD7QCK6OTSPYB7WPDZ", "NODE_SIGN":
"35B7D421BA03B0296A3E2DD7FD8C7EE07300E717E7803E3D4C56D58E5D47FC4964A9C36285B6B46314AD5C20BECF1D687BA
9D9762B35C923F7F855901D9D5383",
"SENDER_SIGN":
"6D754815C9427C12F36B018156F7A8E3B63FC460618AEF8029268D8F45E0EA5293B4CBFE9D4A76177887FA0B20F104BDC2AD3
4E97BF31DA65ED9B51B3C5C0639"
}
```


The section “VID” consists of list of token IDs (MASS) being sent to recipient account, TIDs and hash-sums (HASH) parent transactions which brought aforesaid tokens to the sender account.

The section “VID_CMS” consists of list of token IDs (MASS) being paid as commission, TIDs and hash-sums (HASH) parent transactions which brought aforesaid tokens to the sender account.



Picture 4. DAG scheme.

Explanations:

1. TID – transaction unique ID
2. WID_s – sender account
3. WID_r – recipient account
4. VID – array of token IDs and nominals
5. prev_TID – array of parent transactions IDs

6. type – transaction type (trn_cre – emission,trn_trf – token transfer)

TRANSACTION VALIDATION

Transactions are being validated by List of transactions (LOT). In every round the speaker node forms current LOT and sends it to G-Nodes, which verify every transaction and exchange with each other its vote about every transaction. After that voting all transactions with positive voting effect are being put to DAG while others become not verified.

The authenticity of transaction is provided by sender sign, node sign and combination of parent transactions hashes. Thus, if an intruder is going to substitute a transaction in DAG, one need to capture sender account private key and to substitute all transactions which is child for this transaction.

Each transaction contains hash-sums of all its parents, each of which, for its part, contains
hash-sums of all its parents etc.

To change some information about transaction, it is necessary to change its hash-sum. Changing hash-sum breaks relationship between the transaction and its child transactions. Therefore, it is impossible to change some transaction without the cooperation with all its child transactions or stealing all its private keys etc.

There is no possibility doubling nodes to appear when the DAG is forming. Transactions suspected as double-spending will never get to DAG.

The principle of DAG forming excludes width growing. This is achieved by archiving transactions. Occasionally transaction with the maximum number of child transactions appears in the DAG. It is a transaction which spent all its tokens. If its child transactions spent all theirs tokens too, the algorithm gets this transaction from DAG and puts it into the archive which is the blockchain of the archive transactions. Thus, DAG size is lowered, and significant disk space is freeing. The archive transactions don’t take part in the process of verification of the new transaction and are used to analytics only.

The “travel path” of token can be obtained by going from the last DAG node with it to its parent DAG nodes and etc.

Picture 4. DAG scheme.

Transaction: 2QUENMDFPCUHUWWKLFJMJP5HJ3ZAJV5GPJGOBIQHHBFNFYYXASDESGLSVDCA6KBX_3

Recipient:	25RC8FEFWT2J1NJXH6AI81BUZF64WMKTRGPMZVDK7RSDCWSU6ASPZUPXILMNRHXZ
Sender:	2QUENMDFPCUHUWWKLFJMJP5HJ3ZAJV5GPJGOBIQHHBFNFYYXASDESGLSVDCA6KBX
Time:	Saturday, May 23, 2020, 15:44:41 UTC
Type:	Transfer
Comment:	
Global index:	17
Node ID:	XYYTJQ1IYHYCHDY42TUKAODRTXJKJKK3SADANJTTSQGYXAKJGP7CHULZ9AO1SPPC
Node sign:	7B635AF4373F9FDC3EE07D7A68E1409164E7F65404AB1148EF463155F99DEBFAB23B9F1B29247F4262327F4F1CF2094E3337BFEF4F1781813052D358A41CF3CF
Round:	17
Sender sign:	5232D8DFEFA10326C8904A0907867D6D5E7D978C89DD720C2D3F1B2CEBDBB5FFBB5231FDFA8B0680F1DC48BE1EC80D78D2D411CF96A2F6E64F5F00F43A894D3C
Transactions in round:	1

Pending tokens:

TID: 2QUENMDFPCUHUWWKLFJMJP5HJ3ZAJV5GPJGOBIQHHBFNFYYXASDESGLSVDCA6KBX_1
HASH: 6B376FF72E0B5FE1CDD3AEB7338CD229E9AD2A7214EDC58293A1264152F99847

Tokens:

5A4E35ABBE5AFFA699A7002D90433C9045396D08A8348BFC4723DC4D956DCA32 0

Comission:

TID: 29Q3J2PT9ULYWBKMKHHVVDCDJ11FRJMYMPYMBP4MQYQH96J35BJJXXST9HMYDHMK_7
HASH: BC2795BBD47A138B917A39254333870849B133BBA24B81B009337433062E862F

SMART-CONTRACTS

The smart-contracts in VEDA platform are scripts written on LUA programming language. The purposes of using smart-contracts are to simplify, verify or guarantee some real contracts. The smart-contracts are executed automatically by the network nodes, therefore they cannot be violated, declined or bypassed. It is the new level of security and cost reduction in contract processing.

Why LUA?

LUA is a dynamic language with good level of performance, quite high popularity, quite big community. It is easy to learn: there is only one textbook about LUA written by its author.

VIRTUAL MACHINE ARCHITECTURE

Virtual machine is separate process (module). It is listening to local port, so it is separated from the network. It is working by specified protocol and all the information sent between it and the Node is encrypted. There can be more than one virtual machine simultaneously. Thus, there is possibility to run smart-contracts written on different programming languages in future.

SMART CONTRACTS LIBRARY

VEDA platform provides contract templates library. It is application looks like App Store or Google Play. It has two interfaces: developer and user.

User interface helps user to find similar contract, modify it for own usage, try it on “sandbox” and use for own purposes. If there is no similar contract in library, user can take most similar template from library and modify it.

Developer can create new contracts and share them using library for free or in commercial purposes.

CONTRACT EXECUTION

The execution of the contract functions is done by the Virtual Machine placed on every VEDA Node. It is started by sending transaction to the contract address with the data determines which function must be run.

ADAPTERS

It uses the VEDA’s contracts API to write data needed to one or more of smart-contracts. Adapter provide the way to get external data and put it to the VEDA Platform. The contract used by the Adapter to receive external data, must provide public interaction methods.

CONCLUSION

VEDA software is developed using international experience, proven concepts and best practices. It provides the new level of working with data.

VEDA technology is the base for:

- the confidential workflow between companies

- securing private property in the Internet

- the platform for releasing digital assets in the existing legal field