RELICTUM PRO: DECENTRALIZED LEDGER TECHNOLOGY
BLOCKCHAIN 5.0 VERSION 17.11.21.EN-3

2021, RELICTUM PRO LTD.
GLOBAL PLATFORM COVERING ALL THE ASPECTS
OF HUMAN LIFE IN A DISTRIBUTED REGISTRY

Abstract. Relictum Pro is a blockchain of the latest generation, Blockchain 5.0, which has all the necessary fundamental and sufficient conditions to ensure the work of the fourth generation of blockchain:

1. Hypernet - 4-dimensional network for switching virtual channels.
2. Multi-dimensional organization of chains (smart contracts) with instant access.
3. A global platform for the formalization of all events in the economic and daily activities of humans and states.
4. Speed of message delivery (block) to each node from 100,000 TPS/s.
5. Dynamic circuits.
6. Hierarchy of nodes roles.
7. Regeneration of the network.
8. Elimination of ambiguities.
9. Scalability. The difference is that binary files of portfolios are identical. When initialized, the node automatically determines its status (role), or the owner can change the role manually at any time.
10. Distributed storage. The necessity to create a distributed storage is inevitable since simple hashing of events of signing digital documents does not ensure the storage of the content itself and the subsequent provision of the document both in private and shared form.
11. Synchronization of the clock rate.
12. Using the ideal random number generator based on the acquisition of data (samples - reduced energy of each pixel) of the relictum background of radio emission (quantization).
13. Checking the integrity of a binary file, both in stationary mode and loaded in memory, must be performed by external resources.
14. Three-level circuit integrity check. It consists of receiving a hash, a full hash, a hash at intervals, a checksum.

1. Introduction

We provide the IaaSB platform with the infrastructure in the form of a service, where B is a portfolio.

Value - all events a person generates are determined by their value and resistance to unauthorized access to this information.

Relictum Pro - the first global blockchain service is providing the best use of modern equipment laying the groundwork for the future.

Major characteristics include:

1. Independence from transport. I.e., it is not necessary to use the Internet, you can use WiFi, Bluetooth, telephone networks, fiber-optic communications, you can even transmit information via a USB flash drive or optical, quantum promising networks.

2. Independence from the type of Client (thin client, fat client, data centers, push-button mobile devices) with any operating system.

3. TPS is close to the thickness of Masternode channels.

4. Grid blockchain has 100% distribution. For a finite period, within one day of network operation, authority is delegated up to 100,000 times to any of the nodes, which makes decisions for the entire network. The delegation of authority protocol is associated with a random number generator derived from the Relic of radio emission, as it best describes the Gaussian normal distribution curve.

5. Organization of the cranked smart contract.

6. Availability of dynamic circuits for storing dynamic data, for example, for repetitive logistic cranked smart contracts.

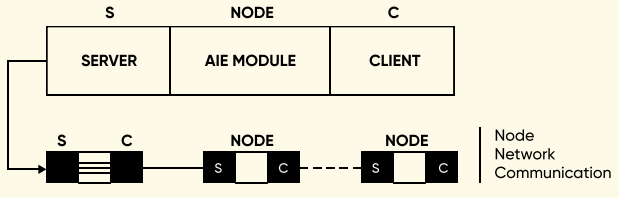
7. Use of end-to-end numbering of hashes of chains for guaranteed elimination of collisions for a long period.

8. Distributed storage. Fragmentation and differential placement of digital data on user devices containing a node.

9. Algorithm of transferring blocks over the network to each node provides for a complete cycle when all the nodes receive this data: 100% - (minus) Sleep (formula).

10. A node can get out of a slip only when it is fully initialized confirming its binding to the hardware, receiving hashes, checking hashes, checking a binary file with an external resource, and completely pumping circuits.

2. Functionality



EAI - Element of Artificial Intelligence Node is a three-entity:

1. Portfolio.
2. Chains of blocks.
3. Distributed storage.

2.1 Portfolio:

The binary file that contains the following items:

1. Management of nodes.
2. Explorer.
3. The generator of smart contracts.
4. Creation of IO.
5. Generation of Tokens.
6. Functional transactions for any kind of currency.
7. Private chat.
8. Exchange.
9. Self-diagnosis.
10. Update resource.
11. Protection against failures.
12. Binding to the hardware.
13. Module communication with an external resource to exit the slip mode.
14. API initialization module.
15. SDK initialization module.
16. Low-level API initialization module (socket).
17. The module of work with distributed storage.
18. Authorization module.
19. Biometric module. Biokhash.
20. Time synchronization module (clock rate).
21. Module for receiving data from an external source of the background radiation data in the form of a three-dimensional normalized digital map.
22. Status blocking module. It is used to block unauthorized access to both the binary file and part of the downloaded application file.
23. Tick module for self-diagnosis.
24. Self-destruction module. If any critical hashes do not match, the encapsulated program is launched inside the application, which removes all resources, including user authorization data.

2.2 Chains of blocks:

1. The blockchain is an n-dimensional matrix of chains of smart contracts, which are governed by the signature mechanism, in the form of hash lines, with the obligatory confirmation and placement of each transaction in the master chain.
2. The size of each block varies from 120 to 300 bytes. At the same time, each block has a consecutive numbering in the n-dimensional matrix.
3. These blocks correlation infrastructure is organized in such a way that data searching is conducted from the end to the beginning. This leads to the almost instantaneous acquisition of information about the data in the n-dimensional matrix. For example: to get lists of all cryptocurrency transactions, moving from the last reference to the first one, this list is obtained instantly, bypassing the intermediate blocks due to the continuous numbering of the blocks in the matrix. This allows analyzing any data in the blockchain locally; build charts; find ways of data intersecting; manage dynamic smart contracts; calculate the analytics of the interpolated data and approximate the extrapolation function to calculate the expectations for a sufficiently long period.
4. The presence of dynamic blocks.
5. Each node owner can create own database for interacting with dynamic blocks (DB smart contract). For example, counterparty profiles and so on can be stored in such a database. Also, dynamic smart contracts can be used as cranked smart contracts.

2.3 Distributed storage:

1. You must have a smart contract of distributed storage and confirmation of the portfolio owner to use the resources of his computer.
2. The minimum allocated amount of data is 200 MB. The maximum is unlimited.
3. The initialization of the distributed storage in the portfolio allows the owner to earn on the number of calls to the database of the distributed storage.
4. Distributed storage receives data as is (in the form of an array of bytes) and does not control the contents of the database in any way. The node takes care of the content of the database, the one that places differentiated data of the storage in the entire network.
5. Each data element can have a different length, from 1 Byte to the maximum value of the computer register (2^{64}).
6. Information about a piece of bytes is stored in a smart contract (chain) of distributed storage.
7. Duplication of pieces of a byte sequence repeatedly occurs at different nodes to eliminate data loss.
8. The owner of data placed in the distributed storage can place the file entirely in own database in own portfolio, both encrypted and shared.
9. Data placed in the repository has status from the open data type to the mosaic-mixed pieces. Data may be of Share or Private type.

10. When initializing a new document, a check for objectivity and reliability of data is made to exclude spam.

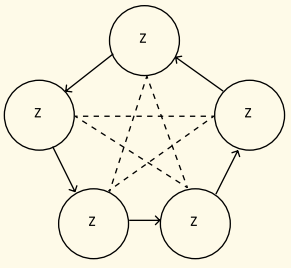
11. Data placing in a distributed repository is a paid service in order to exclude spam.

3. Principle of Network Organization. Network Protocol:

3.1 The network has five levels:

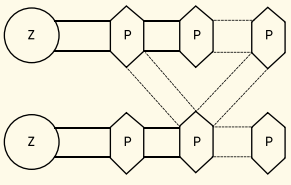
1. Tsar.
2. Z level - General.
3. P level - Ordinary node.
4. L level – Light node.
5. S level – Private node.
6. O level - Sleep node.
7. Cloud node

3.2 Z level (Generals) (Zero-Level)



1. One of them is Tsar, but it is unknown what exactly.
2. All Z work like a single computer with multiple processors.

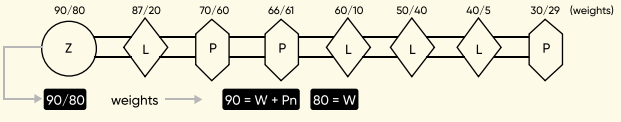
3.3 P-level (Power level) of the 1st level node



Z-P Chains

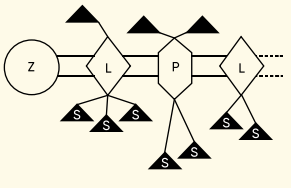
SC (switching channels) Transport.

3.4 L-level (light node) the 2nd level



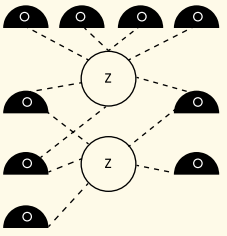
W - is a current node weight
Pn - is the result of the formula
Pn = Gp (R)
Gp - is a pseudo-random number generator
R - is a selection from the relic normalized to the range (1-100%)
Weight formula (current + dynamic):
Wd = W + Gp (Rn)
Gp → number in Rn array
n - is the network regeneration number.

3.5 S (Self node) = private node



1. Your transactions
2. Selected nodes (only their transactions)
3. Data storage permission for distributed storage

3.6 O - (Offline node) = Slip node



1. Attempt to connect with Z-node.
2. Getting the address of the connection (IP).
3. Check the integrity of the local resource.
4. Iron check.
5. Hashes synchronization.

4. Table of node type functions

0	1	2 W IP	3	4	5	6	7	8
0	Z	+	+	+	+	+	+	-
1	P	+	+	+	+	+	+	-
2	L	±	±	+	+	+	+	-
3	S	-	±	+	+	+	+	-
4	C	±	-	-	-	-	-	+
5	O	-	-	-	+	-	-	+

0. Number
1. Thumbs
2. White IP
3. Become a general
4. Become a tsar
5. Transaction Initiation
6. Participation in voting
7. Earnings
8. Rating restriction

5. Chain architecture

1.	Master chain	S
2.	Smart chain 1	S
	Smart chain 2	S
	Smart chain N	S
3.	Dynamic chain	D
	Dynamic chain 2	
	Dynamic chain 3	
4 D	Synchro chain	Synchronizes incomplete nodes
5 D	Time chain	Calculates the clock
6 D	Storage chain	Distributed storage smart contract chain
7 D	**Status chain**	Information about the status of all nodes in the network

S - standard node
D - dynamic node

6. Application area

Law:

Value, scalability, ability to withstand inflation, and incentives to use them.

Insurance:

1. Optimization of the insurance premium payment model
2. Reduction or elimination of fraudulent risks

3. Automatic payment of premiums using smart contracts

4. Improving customer service

Finances:

1. Improved identification mechanism
2. Reducing risks between counterparties
3. Full transparency of the services provision process
4. Fast cross-border transactions

Medicine:

1. Simplification of the electronic medical records maintenance
2. Effective collection and management of data from medical devices
3. Transparency of drug supply chain tracking
4. Health insurance upgrade
5. Multicranked Smart Contract (delivery)

Logistics:

1. Effective cargo tracking, reduced probability of loss
2. Reduced delivery time
3. Transparency of logistic information
4. Increase bandwidth in the supply chain

Ticket market issues:

1. Fakes
2. Touting

Hydrocarbons:

The blockchain technology gives the oil and gas industry opportunities to solve problems with cross-border transportation, a large volume of transactions, complex document flow.

7. Token and its classification

Token, as a digital commitment, can be classified as follows:

Means of Payment: It is analogous to traditional cash.

Utility token: Used as a "fuel" inside the blockchain system and ensures its operability.

Digital investment asset: A digital asset that has a legal basis in the real, non-digital world.

Tokenized asset: Digital commitment to exchange for a real product or service.

Tokenomics: HA set of economic rules and models, ensuring the functionality of the project economy, based on tokens.

Economic model: Formalized description of various economic phenomena and processes.

8. Main phases of the network operation

Regeneration mode (changes in TOPOLOGY, change in power).

BUILDING TOPOLOGY.

Regeneration occurs continuously or on request for reading or writing (at the end of the request for reading or writing). If the network is idle, then regeneration occurs on timeout.

In this mode, the network stops processing requests (READING, RECORDING) and network NODES (NODES) analyze the work of the previous cycle – the processing time of blocks, access speed (SPEED OF COMMUNICATION CHANNELS), number of processed transactions. Based on the results of this analysis, an unambiguous network topology (hierarchy, rating) is compiled for the next cycle. The choice of a Tsar occurs randomly based on the RCS algorithm (random circle stop, a UNIQUE algorithm) among the Generals at the very beginning of the regeneration mode. The RCS algorithm is a game like "rock-paper-scissors." During the designated period, the NODES produce sequential calculations (OBTAINING RANDOM DATA FROM THE CURRENT TABLE OF RELAY BACKGROUND). After the appointed time, a Tsar is selected.

The selection of GENERALS comes from the TOP OF NODES RECEIVED BY THE FORMULA Wd=W+Gp[Rn], the number of which depends on the load of the network. In each cycle, the generals are updated. (You cannot be a general for two cycles one by one).

Tsar calculates (verifies, approves) each new BLOCK and initiates transactions aggregated by the Generals in the previous cycle. After checking each block by the Generals, the blocks are transmitted down the network to all the nodes.

The new network topology (hierarchy) is calculated and approved by the Tsar at the end of the current cycle and distributed throughout the network from the NODE to the NODE, in accordance with the topology in effect on this cycle from each General in the corresponding branch.

There is no democracy on this platform, with automatic data processing.

9. Coding algorithm

It is SHA-1 algorithm with subsequent conversion to the Proprietary Jump Method, which uses the undiagnosed Overflow error for one-way hashing.

10. Record mode

Any node that has received information to write to the BLOCKCHAN transmits it UP through the network. Tsar forms a block based on the received information, writes it to the blockchain, and transfers the complete blocks to all the Generals.

11. Goals of BLOCKCHAIN NETWORK

- Any node can connect to this fully open network from anywhere.
- NETWORK TOPOLOGY facilitates effective network sharing.
- Secure network neutrality from network level innovation.
- Always open and scalable.
- Automatic effective and dynamic routing.
- Tokenization mechanism (automation of the calculation process using the token accounting mechanism, as a fee for maintaining and expanding, and optimizing the network and network connections, data transmission assets, and stimulating participating nodes.
- Design and build the next generation blockchain network.

12. Sinker (functions)

SINKER for Nodes (Sleep mode) that do not cope with the network performance or for some reason, breaks the connection.

RETURN NODE FROM SINKER.

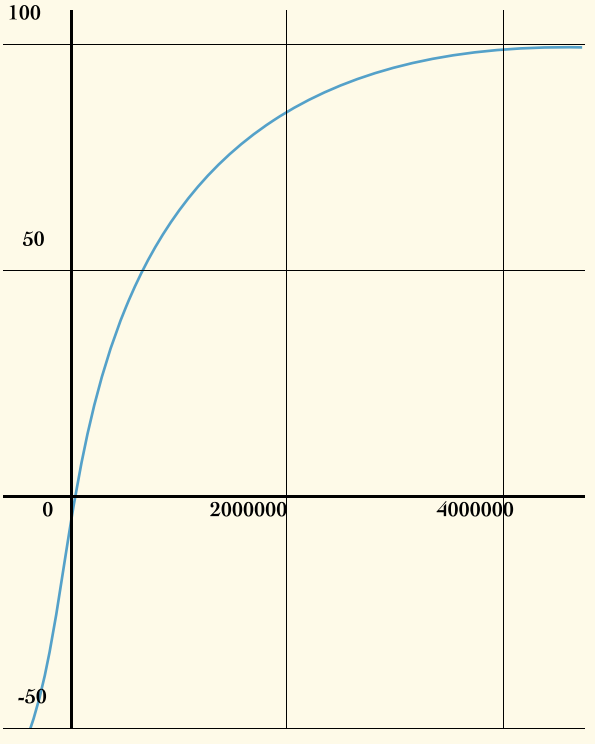
DISCREDITATION OF A NODE (SEARCH FOR JUDA).

13. Formulas and implementation

The function slowing the growth of the token weight (value) at infinity and does not allow to exceed the predetermined maximum cost (emission, weight ...) = 100%

The formula for the token cost rise to 100 units (%), on the example of a smart mining contract, proof of time:

y=-100e^(-0.00082x)+100



The link below is for checking the formula chart:

http://www.yotx.ru/#!1/3_h/ubWwcH@2cHBwf7Rgzhf23/aP9g/2DfT0qt7W9uHRysrW9sHkAODg5gO3u70K2Dg/2DfRlNu7Fzyng83WI8bl1e7O5v7QMG

14. Mining example

TikS = TikL + TikD

AmC = -100e^(-0.00082TikS)+100

AmD = AmC - AmL

AmD = SUM(APn)

TikD = SUM(TDn)

APn = (TDn/TikD)*Amd

1 Tik = 24Hour

AmL - Last Amount

AmC - Current Amount

AmD - delta - SUM of all the coins for the period {TickL->TikC}

APn - The amount of each in the period

ADn - delta, TikL - Last Tik

TDn - delta Tik Nod (each node), TSn - Sum All Ticks Nod

TikD - delta - SUM all the ticks for the period {TikL->TikC}

14.1 Formula for a pseudo-random generator and obtaining hash strings

1.The starting value of the series is generated.

```
ep=15436757;
ep=abs(sin(ep));
ep=ep*429496729;
```

2.Generation of the next element of the series, GenP [byte]-generation result:

```
gran=255;
ep=abs(sin(ep));
ep=ln(7)/ln(ep);
ep=ep*10;
ep=ep-trunc(ep);
GenP=trunc(ep*gran);
```

3. Getting the N-th intermediate element of the hash string s[byte].

s-string of characters.

Hash[255]-array byte.

z-number of the [Hesh] array element.

The i-number of the array element of the string[s]

In the case of length(s)<lngth((Hash), the values of 0[byte] are added to s, so that the condition

```
length(s)=lngth((Hash) holds true
  Cycle
  inc(z);
  inc(Hash[z],GenP(ep)+ord(s[i]));
```

In the case of z>lngth((Hash), "1" is assigned to z, which leads to the modification of the z-th (intermediate) hash element.

The values of the pseudo-random generator are used to obtain random values from the weight matrix of the relic radio emission.

14.2 Complete Elimination of Collisions of the Chain of Blocks

The first 4 bytes in the hash are replaced by the Count value of the current number of the MasterChain block. This means that 2147483647 transactions are guaranteed to exclude collisions.

One token is created for each chain of a smart contract (for each owner) with a face value of N (max 9223372036854775807).

14.3 Cranked smart contract

The method of calculating the optimal cranked smart contract (tokenomics).

Calculation of a cranked smart contract with intermediate events (checkpoint).

Smart contract parameters: L - length (meters, links, parsecs, oscillations of waves (number of periods), steps, etc.)

T(L) - time spent on the passage of the entire length, taking into account stops in checkpoint

Ln - length of the n-th section (checkpoint)

Tn - a time of the n-th segment

Rn - rating (reliability of the n-th point)

Gn - a result of a pseudo-random rating change generator at n point

Wn - Impact of external parameters on each step

Gn - Checkpoint safety factor.

Sn - a calculated value of the rate of Ln overcoming. It is necessary for calculating correlations with other events of previous blocks and extrapolating future events and calculating the reliability of cranks, which allows you to choose another trajectory, for example, when the checkpoint is busy (the problem of intersections).

Z - Reliability of the whole way for one block

```
T(L)=sum(Tn*Rn*Gn*Wn*Gn)
Tn=Tn*Rn*Gn*Wn*Gn
Sn = Ln/Tn
S = Sn/n
Z = S/L ; при Z=1 (perfect reliability).
```

Also, you can calculate a weak crank and other parameters for analytics.

With the number of blocks> 17, we use the method of the smallest modules, and we can consider such a smart contract an element of AI (self-study + self-analytics).

For example, it can be used to control traffic lights.

14.4 Integrity Check for Nodes

a) Module for checking the integrity of blocks of the whole chain.
Formula: verification of the condition preservation throughout the chain, starting from the 2nd element

h(Bn-1)=Hn for n[1..BlockCount-1]

h - hash of the block
Hn – hash in the block
Bn - block N

If the condition is not fulfilled, the circuit is considered to be damaged, and a restart event occurs for the whole circuit.

b) Module for Checking the Integrity of Blocks of Chain of All its Own Transactions.
Formula: verification of the preservation condition throughout the chain, starting from the 2nd element.

h(Bn-1)=Hn

h - hash of the block
Hn - hash in the block
Bn - block N

If the condition is not fulfilled, the circuit is considered damaged, and a restart event occurs for the whole circuit.

c) Module for Checking the Holistic Local (Original) Executable Binary File.

- A1 name verification (external)
- A2 check creation date (external)
- A3 launch location check (local)
- A4 size check (external)
- A5 check of hash file itself (external)
- A6 checking for binding to gland (local)

Formula:

h(A1,A2,A4,A5)

h - hash of the concatenation of variables A
A - c string variable of string type

Result h requires confirmation of the integrity of the unique files via a network of other nodes at the request of the smart contract

If the min (9) confirmation condition is not fulfilled, the nodes are considered to be damaged, which results in the application being completely disavowed and the network service deny.

d) When launching an executable binary file, it loads and launches a plugin that works separately from the main program. In this case, the main program waits for confirmation of authentication (waiting for the session key).

To obtain the session key, the plugin performs step 3. and sends h(A1, A2, A4, A5) to the network. The network, after checking the min (9) acknowledgment of the dynamic chain of random nodes, sends a new session key to the main program. The session key is also written to the dynamic chain of all nodes.

Thus, the status of the node changes from SLEEP to ONLINE.

15. Obtaining Hash String

Getting the hash of the string f(X(L))(hash) from the string L

Step 1. The standard function sha1 is calculated (with minor modifications); as a result, we obtain f(X(L) with a dimension of 20 bytes.

Step 2. The conversion of f (X(L) to 32-byte number using the Byte shift with summation

```
Byte shift with summation:
Type Tsb32=array[1..32+1] of byte;
for i:=1 to HashLength do sb[i]:=sb[i]+sb[i+1]+i*i;
```

f(X(L)=sb

In this case, a hardware preventive blocking of the overflow error of the processor cell is used.

```
Example:
var a,b:byte;
a:=255;
b:=1;
a:=a+b;
result: a=0;
```

Such an operation significantly saves resources; there is no need to do additional checks and calculations.

Step 3. A few jumps over the barrier 255 are taken

```
inc(b,sb[i]
inc(sb[i],b)
```

b is a byte cell

Step 4. The byte-adding function of the resulting hash f(X(L)) with the source string L is used.

```
inc(sb[k],ord(s[i]));
```

k(1..32), i(1..length(L))

Step 5. It is necessary to perform step 3 to avoid reverse action in step 4

16. Listing

(Pascal)

```
function ShaM_Bin(s:string):THash; const HashLength
= 32;
MagicByte =*****; var i,k,L:integer;b:byte;
sb:Tsb32;
SHA1Digest:TSHA1Digest; begin
SHA1(s,SHA1Digest);
k:=1;
L:=20;
for i:=1 to L do sb[i]:=SHA1Digest[i-1];
for i:=L+1 to HashLength do sb[i]:=$0;
for i:=HashLength-L to HashLength do
inc(sb[i],SHA1Digest[i-HashLength-
L]);
*****=MagicByte;
for i:=1 to HashLength do sb[i]:=sb[i]+sb[i+1]+i*i; b:=0;
for k:=1 to 4 do begin
for i:=1 to HashLength do inc(b,sb[i]);
for i:=1 to HashLength do inc(sb[i],b); end;
k:=1;
L:=length(s);
for i:=1 to L do begin
inc(sb[k],ord(s[i]));
inc(k);if k>HashLength then k:=1;
end;
for i:=1 to HashLength do ShaM_Bin[i]:=sb[i]; //
result:=SHAMDigestToHex(sb);
end;
```