



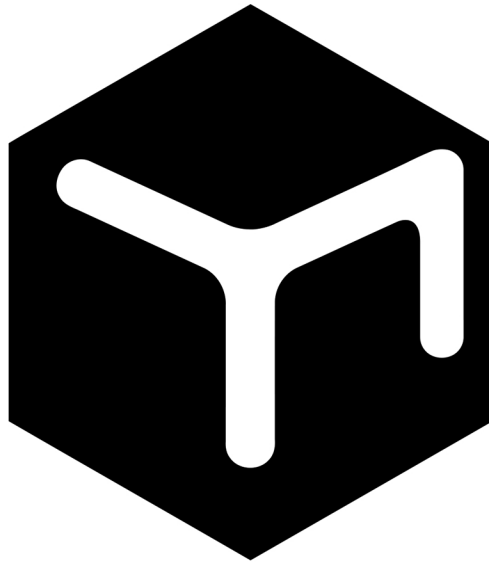
Relictum Pro

Blockchain 5.0

Decentralized Ledger Technology

Yellowpaper

version 2.05.21.ru-2



Relictum Pro

Blockchain 5.0

Global platform covering all the aspects
of human life in a distributed registry

With the use of HYPERNET technology based
on peer-to-peer peering networks

简介

Relictum Pro 是崭新一代 **Blockchain 5.0** 区块链，具备确保四代区块链运行必要的基本与足够的条件。

1. 超级网一是虚拟频道交际 4 维网络。
2. 链（智能合同）多维组织确保瞬时存取。
3. 全球化事项形式化平台，形式化个人与国家经济与日常生活中的各种事项。
4. 信息（区块）到每个节点的速率从 100 000 TPS/s 开始。
5. 动态链
6. 节点作用分级结构。
7. 网络再生
8. 意义非单一性开除
9. 规模可伸缩性特征是，总存量包的二进制文件完全相同。起始的时候，节点自动化决定它的地位（作用），或者所有人同样可以随时手动调换其作用。
10. 分布式存储器。不可避免的发生创造一个分布式存储器的必要，因为数字化文件的单单哈希不能确保本身内容的储存，随后在私密或分享的模式下提供文件。
11. 按时针差同步。
12. 采用理想的，基于残留辐射背景数据的获得（量子化）的随机函数发生器。
13. 应该用外部资源验证二进制文件的完整性，无论是固定模式下或存储器加载的文件。
14. 三级连锁完整性检查。由获得散列、全部散列、间断散列、检查求和组成。

目录

引言	6
功能	7
1. 总存量	7
2. 区块链锁	8
3. 分布式存储器	9
网络组织原理。网络协议	11
1. 沙皇	11
2. Z level- 将军	11
3. P level - 普通节点	11
4. L level - 轻节点	12
5. S level - 私密节点	13
6. O level - 睡眠节点	13
7. 云节点	13
节点种类功能表格	14
链锁结构	15
适用领域	16
法律	16
保险	16
金融	16
医学	16
物流	16
票市场难题	16
碳氢化合物	17
代币与其分类	17
支付手段	17
实用代币	17
数字化投资资产	17
代币化资产	17
代币经济	17
系统结构（结构设计师）	17
网络运行的主要阶段	18
加密算法	19
记录模式	19



区块网络目的	19
沉淀槽（功能）	19
公式与实现	20
挖矿例子	20
伪随机数发生器与行列散列获得公式	21
1. 发生行列的开始数值	21
2. 行列下一个成分的发生，GenP[byte]，发生结果	21
3. 获得散列行的第-N 中间成分	21
完全开除区块链锁的碰撞	22
分节智能合约	22
节点完整性检查	23
1. 整个链锁区块的完整性检查模块	23
2. 所有自己交易链锁区块的完整性检查模块	24
3. 当地（原本）可执行二进制文件的完整性检查模块	24
4. 插件开启	25
行列散列哈希的获得	25
阶段 1	25
阶段 2	25
阶段 3	26
阶段 4	26
阶段 5	26
挂牌	27

引言

我们提供 [IaaSB](#) 平台 — 一种服务形势下提供的基础设施，其中 B — 是指总存量包。

价值 — 每个人产生的所有事项，由其价值与耐于擅自访问该信息而确定。

Relictum Pro — 是提供最合理利用现代设备办法同时留下很大未来潜能的世界第一全球化区块链服务平台。

主要特征有：

1. 不依赖于传送设施。也就是说，没有一定的必要采用互联网，可以采用 WiFi、蓝牙、电话网络、光线通讯，直到通过优盘转载信息或利用光学网络、量子网络等先进技术和网络。
2. 不依赖于装有某一种操作系统的客户终端（薄客户、厚客户、数据处理中心、按钮移动机器）。
3. TPS 临近主节点频道的厚度。
4. 区块链网络具备 100%分布性。在有限的时间阶段内，网络工作一个昼夜的时候，网络会多大 10 万次把权限移交给某一个节点，为整个网络做决定。权限移交协议基于用最合理描述高斯正态分布曲线的微波背景辐射产生随机函数的发生器。
5. 分节智能合同的组织。
6. 有助于储存动态数据的动态链，比如用于重复的物流节点智能合同。
7. 采用链锁散列哈希的流水编号确保，保障能长期开除碰撞。
8. 分布式存储器。在装有节点用户机器上分别分化部署数字化数据。
9. 到每个节点网络传送区块的算法包括完整周期，保障所有节点获得这些数据。100% — （减去） Sleep （公式）。



10. 节点只有完成整个起始化周期，包括硬件绑定验证、散列哈希获取、用外部资源检查二进制文件以及全部链锁调整后才能推出 Sleep 模式。

功能

节点（总存量报）



AIE — 是指人工智能成分

节点代表三种要点：

- 1.总存量包
- 2.区块链锁
- 3.分布式存储器。

1. 总存量包：

二进制文件，包括以下成分：

1. 节点管理
2. 浏览器
3. 智能合同发生器
4. IO 创造
5. 代币发生
6. 各种货币交易功能



7. 私密聊天
8. 交易所
9. 自我诊断
10. 更新资源
11. 故障预防保护
12. 硬件绑定
13. 用于推出 SLEEP 模式的外部资源联系模块
14. API 起始化模块
15. SDK 起始化模块
16. 低级 API (socket) 起始化模块
17. 分布式储存器操作模块
18. 认可模块
19. 生物统计学模块。生物散列哈希
20. 按时间（时针差）同步模块
21. 由外部来源以标准化三维数字地图为形式获得背景辐射数据模块。
22. 状况锁定模块。适用于封闭对于二进制文件与已加载部分应用文件的非特许存取。
23. 自我诊断滴格模块
24. 自我毁灭模块。在某些敏感散列哈希不符的情况下，开启装入应用内部的程序，毁灭所有资源，其中包括用户登录数据。

2. 区块链锁：

1. **区块链锁**是智能合同链锁的 n -维矩阵，由必须经过验证并在每个交易的主链中布置的散列哈希行列为形式的签订机制调节。
2. **每个区块**的大小分别从 120 位组到 300 位组。同时，每个区块在 n -维矩阵中具有流水编号。
3. **区块数据**关联设施结构组织方式让从最后向最初实现搜索。这导致几乎瞬时获得有关 n -维矩阵中数据的信息。例如：为获得加密币所有交易的清单，从最后一个记录走向第一个记录的情况下，能几乎瞬时形成这个清单，借助矩阵中的流水编号，跳过中间区块。这允许局部完成区块中任何信息的分析；建立线图；找到数据通路的交叉路途；管理动态智能合同；为了计算出较长期的期望，计算内插数据的分析并近似外推函数。

4. 有动态模块

5. 节点的每个所有人，可以为了互动动态模块（数据库智能合同）创建自己的数据库。这种数据库可以比如储存交易对方资料等信息。此外，动态智能合同可以用作分节性智能合同。

3. 分布式存储器：

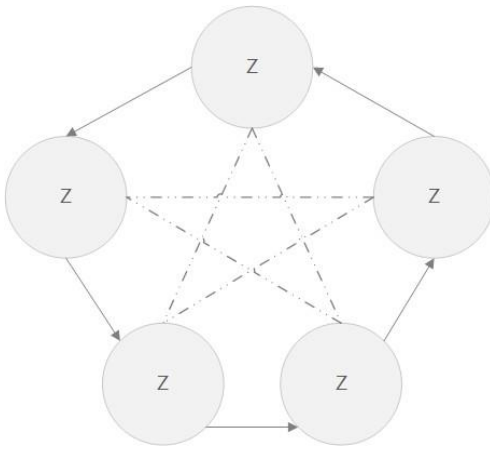
1. 必须具备分布式存储器的智能合同与总存量包所有人确认使用计算机资源。
2. 应分布的最小数据量— 200 兆字节。最大 — 无限制。
3. 分布存储器在总存量包中起始化，允许其所有人用分布式存储器存取次数挣钱。
4. 分布式存储器在已有（字节组）形势下获得数据，如何也不监控数据库内容。数据库内容由节点来管，在整个网络中放置分别的存储器数据。
5. 每个数据成分会有不同的长短，从 1 字节到电脑注册表的最大数值（264）。
6. 有关部分字节组的信息，存储在分布式存储器智能合同（链锁）。
7. 为避免数据丢失，在不同节点上多次折叠字节链片段。
8. 放入分布式存储器数据的所有人，可以到自己的，加密或分享的总存量包数据库放整个文件。
9. 放到存储器中的数据具备从开放式数据到镶嵌式转移的片段。数据能有 Share 或 Private 状态。
10. 在新文件起始化时，为开除垃圾文件，进行数据客观性与确实性检查。
11. 为了开除垃圾文件，分布式存储器放数据服务是收费的。

Prinzipien der Vernetzung. Netzwerkprotokoll:

1. 网络一共有 5 个等级: König.

1. 沙皇。
2. Z 级 — 将军。
3. P 级 — 普通节点。
4. L 级 — 轻节点。
5. S 级 — 私密节点。
6. O 级 — 睡眠节点。
7. 云节点

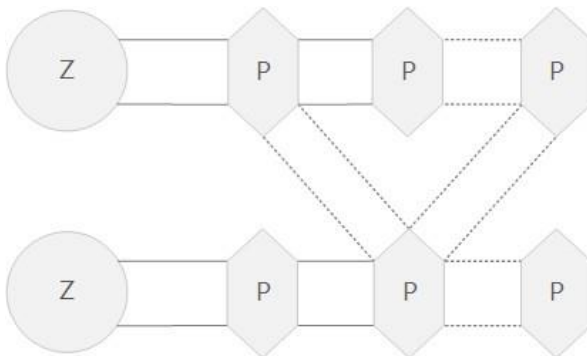
2. Z level (将军) (Zero-Level)



1) 其中的一个是沙皇，但哪一个，是不知道的。

2) 所有 Z 点工作就像有多数处理器的一个电脑。

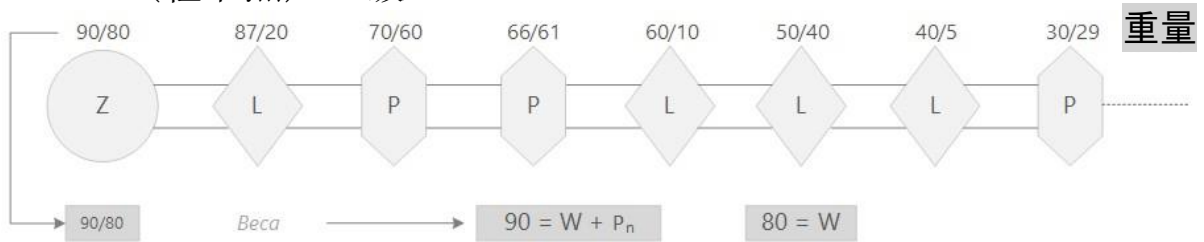
3. P-level (Power level) Knoten von Stufe 1



Chains Z-P.

VK-Transport
(Vermittlungskanäle).

4. L-level (轻节点) 2 级



其中

W — 是指节点的当前重量

P_n — 按公示的结果

$$P_n = G_p [R]$$

其中

G_p - 是指伪随机数发生器

R - 是指对于范围 (1-100%) 定额化的残留背景辐射数据摘录

重量 (当前+动态) 公式

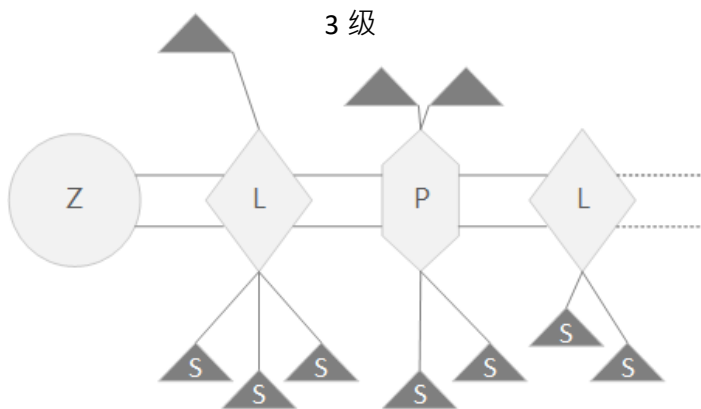
$$W_d = W + G_p [R_n]$$

$G_p \rightarrow$ 是指 R_n 矩阵中的序号

其中 n - 是指网络再生号。

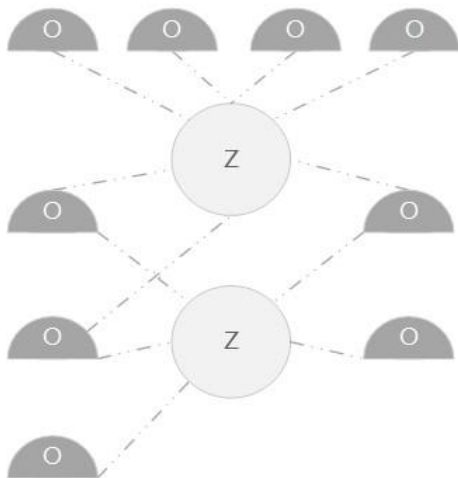


5. S (Self 节点) = 私密节点









1. 自我交易
2. 选中的节点（仅限它们的交易）
3. 分布式存储器数据的存储许可

6. O - (Offline 节点) = 睡眠(SLEEP)节点



1. Z-节点联系试图
2. 链接网址（IP）获得
3. 当地资源完整性检查
4. 硬件检查
5. 散列哈希同步

节点种类功能表格

0	1	2 W IP	3	4	5	6	7	8
0		+	+	+	+	+	+	-
1		+	+	+	+	+	+	-
2		±	±	+	+	+	+	-
3		-	±	+	+	+	+	-
4		±	-	-	-	-	-	+
5		-	-	-	+	-	-	+

0. 序号

1. Thumbs

2. White IP

3. 成为将军

4. 成为沙皇

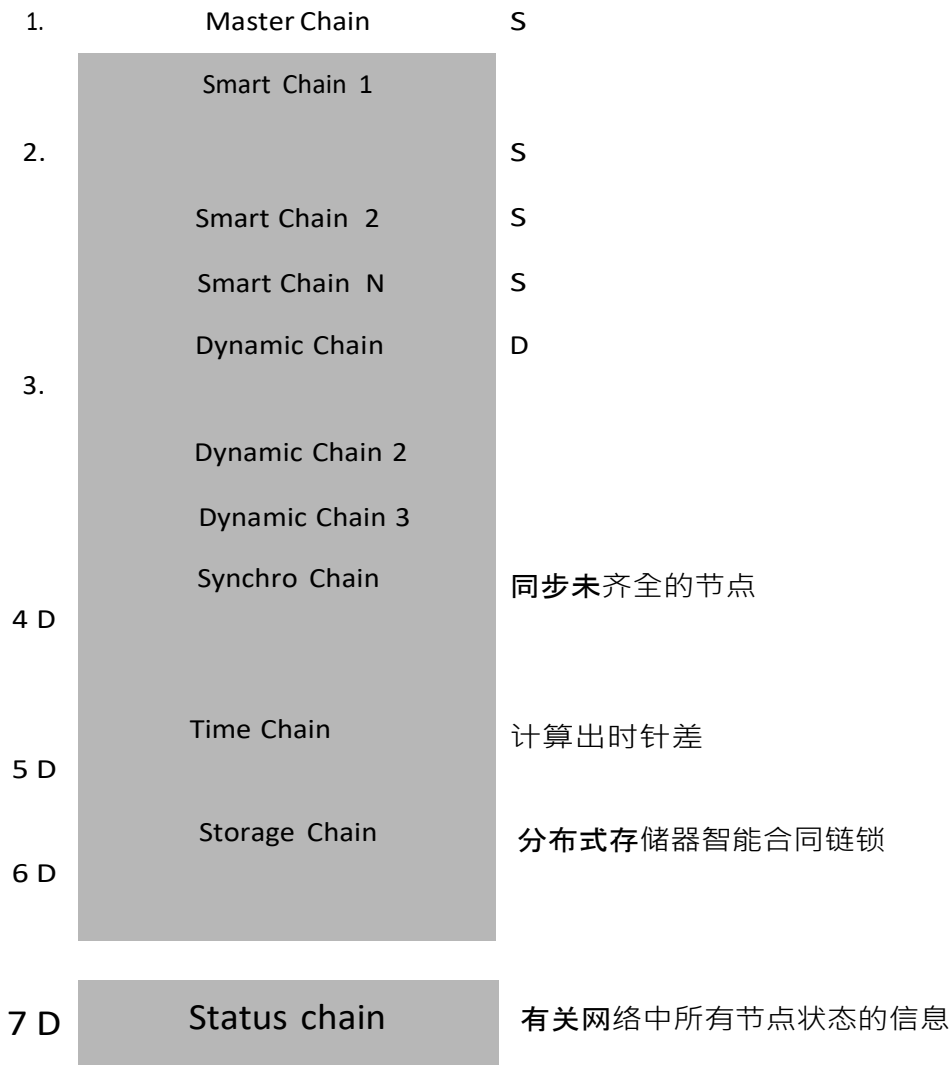
5. 交易起始化

6. 表决参与

7. 挣钱

8. 排行限制

链锁结构



S - 标准节点

D - dynamic 节点

适用领域

法律:

价值、规模可伸缩性，膨胀抵抗性的能力与使用它们的刺激因素。

保险:

保险费支出模型合理化
下降或完全开除欺诈风险
用智能合同自动化支出保险费
提高客户服务质量

金融:

改进的身份认定机制
减少交易对手之间的风险
服务提供过程的绝对透明性
快速跨境交易

医学:

简单化医学电子记录
有效收集与管理来自医用机器的数据
药物供应链跟踪透明性
医疗保险现代化
多节智能合同（送达）

物流:

货物有效跟踪，丢失机率下降
简短供应时间
物流信息透明性
供应链中的通行能力增大

票市场难题:



伪造
投机

碳氢化合物：

区块链技术给天然气石油行业提供解决跨境运输、大量交易与文件运转复杂性有关的难题的机会。

代币与其分类

代币，作为数字化债卷，可以分类如下：

支付手段：

是经典性货币资金的类似物。

实用代币：

用作区块系统的内部《燃料》，确保其工作能力。

数字化投资资产：

在真实世界，不是数字化世界中，有法律依据的数字化资产。

代币化资产：

换成实际货物或服务的数字化债卷。

代币经济：

是基于代币项目经济保障运行的一个经济规则与模型的成套。

经济模型：

各种经济现象与过程的形式化描述。

系统结构（结构设计师）

作为系统设计师采用 AGT 论（Algorithmic Game Theory，算法性游戏理论），保障在开始下一个交易之前，会无条件结束本次交易。同样采用 Proof of Relay 与 Proof of Utility — 是工作证明与效用证明，导致节点按照博弈论会无条件货币化。



网络运行的主要阶段

再生（拓扑变更、政权调换）模式

拓扑建设。

再生进行一直的或按照读取或存储征询（存取征询完后）。如果网络无动作，则按超时进行再生。

这个模式下，网络暂停处理征询（读写），网络节点（Nodes）会同时分析上一周其的工作一包括区块处理时间、存取速率（通信线速率）、交易处理数量。按照这种分析的结果，为下一个周期形成单一意义的网络拓扑（等级制度，分级）。再生模式刚开启是，在将军之间随机选择一个沙皇，随机选择过程基于 RCS 算法（Random Circle Stop，独一无二的算法）。RCS 算法是相似于手势令游戏的一种过程。在规定的时间内，节点进行顺序计算（从当前背景残留辐射数据表中领取随机数值）。规定的时间届满后，选择沙皇。将军，从按照 $Wd=W+Gp[Rn]$ 公式（第 4 页）得出上端节点中选择，其数量依赖于网络载满程度。

每一周期会更新将军节点。（不能一连两个周期作为将军）。

沙皇计算（验证，批准）每个新区块并初始化上周期将军集合的交易。

将军完成每个周期的检查后，通过网络给所有节点递交区块。

本周期末期上，沙皇计算并批准新一种网络拓扑（分级），按照本周期现行拓扑通过每个将军转交给相应路线上的每个顺序节点。

该平台上自动化处理数据过程中，不存在民主主义。

加密算法

是 SHA-1 加密算法，随后转变成 Proprietary Jump Method，用不可诊断的 Overflow 错误完成单向散列。

记录模式

收到待计入区块的任何节点，将其在网络传上去。沙皇基于接到的信息形成区块，将其计入区块链，将完成的区块转交给所有将军。

区块网络目的

- 任何节点能从任何地方连接到这个完全开放的网络。
- 网络拓扑结构有助于有效共同利用网络。
- 预防网络级创新的安全网络中立。
- 始终开放，规模始终可伸缩的。
- 自动化有效并动态路由选择。
- 代币化机制（计算机制自动化，用代币注册机制作为网络支持与扩展的代价，网络与网络连接、数据传送资产与刺激参与的节点）。
- 设计和建立下一代区块链网络。

沉淀槽（功能）

用于放置吃不下网络运行速度或因某些原因断开连接节点的沉淀槽（节点睡眠模式）。

节点从沉淀槽的返回。

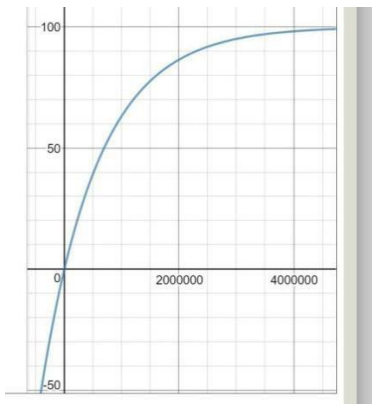
节点威信破坏（犹大搜索）。



公式与实现

在无穷上放慢 Relict 币重量 (价格) 的生长，不准许超过预先指定的最大价值 (发行量、重量) =100%。Relict 币价格生长到 100 分 (%) 公式，以时间验证 (proof of time) 挖矿智能合同为例子：

$$y = -100e^{(-0.00082x)} + 100$$



下面是适用于检查公式线路图验证的链接：

http://www.yotx.ru/#!1/3_h/ubWwcH@2cHBwf7Rgzhf23/aP9g/2DfT0qt7W9uHRysrW9sHkAODg5gO3u70K2Dg/2DfRINu7Fzyng83WI8bl1e7O5v7QMG

挖矿例子

$$TicS = TikL + TicD$$

$$AmC = -100e^{(-0.00082TicS)} + 100$$

$$AmD = AmC - AmL$$

$$AmD = \text{SUM}(APn)$$

$$TicD = \text{SUM}(TDn)$$

$$APn = (TDn / TicD) * AmD$$

wo:

1 Tic = 24Hour / 24 Stunden

AmL - Last Amount / Letzter Betrag

AmC - Current Amount / Aktueller Betrag

AmD - delta - SUM 周期内所有货币总额 {TickL->TikC}

APn - 周期内每个的总额

ADn - delta TicL - Last Tick / Letzter Tick

TDn - delta Tick Nod(每个节点) TSn - Sum All Ticks Nod / Summe aller Ticks Knoten

TicD - delta - SUM 周期内所有滴格总额 {TickL->TikC}

伪随机数发生器与行列散列获得公式

1. 发生行列的开始数值:

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

2. 发生下一个排列成分，GenP[byte] - 是发生的结果:

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

3. 获得散列行列 s[byte]第 N 号中间成分。s — 是指符号行列。Hash[255] - byte 矩阵。z - 矩阵成分序号 [Hesh]。i - 行列[s]矩阵成分序号。

在 $\text{length}(s) < \text{length}(\text{Hash})$ 情况下，s 增加数值 0[byte]，为了遵守条件 $\text{length}(s) = \text{length}(\text{Hash})$

周期

```
inc(z);  
inc(Hash[z], GenP(ep) + ord(s[i]));
```

在 $z > \text{length}(\text{Hash})$ 的情况下，z 被授予 1，导致散列中第 z（中间）成分的改造。

伪随机数发生器的数值用于从背景辐射重量矩阵中获得绝对随机数值。

完全开除区块链锁的碰撞

散列哈希中，首 4 个字节替代成 MasterChain 区块中的当前 Count 值。

这个说明，能为 2147483647 次交易保障碰撞开除。

给智能合同中的每个条链（给每个所有人）创造标价为 $N(\max 9223372036854775807)$ 的 1 个代币。

分节智能合同

最佳分节智能合同计算方法（代币经济）。

带有中间事项（检查站）分节智能合同的计算。

智能合同参数：

L — 长度（公里、链环、秒差距），波动（周期数，步骤等）。

T(L) — 用于通过整个长度的时间，包括检查站上的停止

L_n — 是指第 n 阶段（检查站）的长度

T_n — 第 n 阶段的时间

R_n — 积分（Rating，第 n 点的可靠性）

G_n — 是指在第 n 点变化积分伪随机数发生器的结果

W_n — 外部参数对每个阶段起到的作用

G_n — 检查站可靠性系数。

S_n — 超越 L_n 的计算速率值。用于计算先前区块其他事项的关联与未来事项的外推与环节可靠性计算，比如在检查站荷载度过高（路口问题）的情况下允许选择另一个轨道。

Z 对于单个区块整个路线的可靠性

$$T(L)=\sum(T_n*R_n*G_n*W_n*G_n)$$



$$Tn = Tn * Rn * Gn * Wn * Gn$$

$$Sn = Ln / Tn$$

$$S = Sn / n$$

$$Z = S / L ; \text{ bei } Z=1 \text{ (理想的可靠性) 。}$$

此外，可以计算出来弱小环节等用于统计的参数。

在区块数量 > 17 的情况下，采用最小模块模式，就可以认为这种智能合同为人造智能的一个成分（自学习+自分析）。

这个可以比如用于管理红绿灯。

节点完整性检查

1. 整个网络完整性检查模块。

公式：

从第 2 个成分开始的整个链锁保持条件的检查

$$h(Bn-1) = Hn \text{ bei } n[1..BlockCount-1]$$

式中，

h — 是指区块散列哈希

Hn — 是指区块中的散列哈希

Bn — N 区块

在未符合条件的情况下，链锁被视为已被破坏的，重新启动整个链锁的事项发生。

2. 所有自己交易链锁区块的完整性检查模块。

公式:

从第 2 个成分开始的整个链锁，检查是否保持条件。

$$h(B_{n-1})=H_n$$

式中,

h — 是指区块散列哈希

H_n — 是指区块中的散列哈希

B_n — N 区块

在未符合条件的情况下，链锁被视为已被破坏的，发生重新启动整个链锁的事项。

3. 当地（原本）可执行二进制文件的完整性检查模块。

- A1 名称检查（外部）
- A2 创造日期检查（外部）
- A3 启动地检查（当地）
- A4 大小检查（外部）
- A5 本身文件哈希检查（外部）
- A6 硬件绑定检查（当地）

公式:

$$h(A1,A2,A4,A5)$$

式中,

h - 是指 A 变数并列哈希

A - string 式行性变数

h 结果需要依照无双文件完整性检查智能合同的征询，其他节点的确认。

在未符合 $\min(9)$ 次验证条件的情况下，节点被视为已被破坏的，导致应用完全撤销，拒绝网络服务。

4. 开启可执行二进制文件时，它加载和开启主程序单独运行的插件。此时，主程序等待真伪检查确认（对话键等待）。

为获得对话键，插件执行第 3 条，并到网络发送 $h(A1, A2, A4, A5)$ 。随机节点动态链 $\min(9)$ 确认检查完成后，向主程序发送新对话键。对话键同样到所有节点的动态链中记录。

这样一来，节点状态从 SLEEP 转变成 ONLINE。

— 本身插件 — 是一个无双文件，在主程序开启时由网络（按征询）生成。

— 在开启时，插件检查网络时间与开启它机器上的时间是否同步。

— 插件工作 1-2 秒。

如果在这个时间内它没有发送 $h(A1, A2, A4, A5)$ ，则所有程序会由于错误关闭。

— 同一个插件重新启动的情况下，网络会回应错误，因为该插件不再是无双的。

— 插件的无双性由所有主节点动态链中记录的插件文件哈希绝对。

行列散列哈希的获得

从 L 行获得 $f(X(L))$ (哈希) 行列散列

阶段 1 — 计算出 sha1 标准函数（带稍微的改造），
结果得出 $f(X(L))$ ，大小为 20 字节。

阶段 2 — $f(X(L))$ 转变成 32 字节的数字，采用逐字移交与求和

逐字移交与求和：

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

式中 $f(X(L))=sb$

此时采用硬件预防处理器栈溢出错误。

例如：

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

这种操作显著节约资源，没有必要进行额外检查与计算。

阶段 3

进行越过 255 栅栏的几次跳越

```
inc(b,sb[i]  
inc(sb[i],b)  
其中 b 是指字节栈
```

阶段 4

采用所获得哈希 $f(X(L))$ 逐字求和原始行列 L 的函数

```
inc(sb[k],ord(s[i]));  
wok(1..32),i(1..length(L))
```

阶段 5

为确保在第 4 阶段开除反行动，必须完成第 3 阶段。

挂牌

(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;
```