

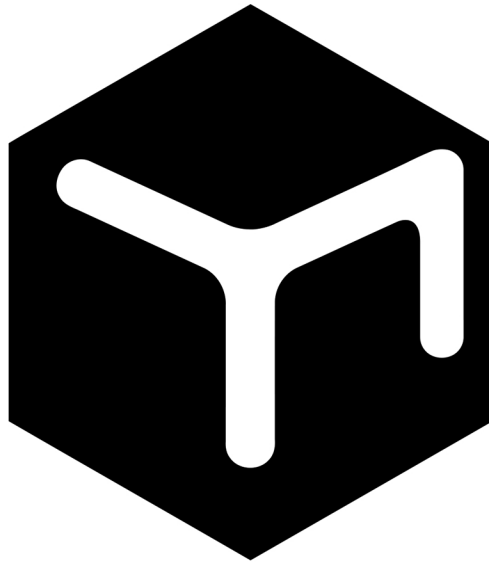
Relictum Pro

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

Аннотация

Relictum Pro представляет собой блокчейн новейшего поколения Blockchain 5.0, имеющий

необходимые фундаментальные и достаточные условия обеспечения работы блокчейна четвертого поколения:

1. Гипернет - 4-х мерная сеть коммутации виртуальных каналов.
2. Многомерная организация цепочек (смарт-контрактов) с мгновенным доступом.
3. Глобальная платформа формализации всех событий в экономической и повседневной деятельности человека и государств.
4. Скорость доставки сообщения (блока) к каждой Ноде от 100 000 TPS/с.
5. Динамические цепи.
6. Иерархия ролей нод.
7. Регенерация сети.
8. Исключение неоднозначностей.
9. Масштабируемость. Отличается тем, что бинарные файлы портфелей абсолютно идентичны. При инициализации Нода автоматически определяет свой статус (роль) или владелец может поменять роль вручную в любой момент.
10. Распределенное хранилище. Неизбежно возникает необходимость создания распределенного хранилища, так как простое хеширование событий подписи цифровых документов не обеспечивает хранение самого контента и последующего предоставления документа как в приватном, так и в расшаренном виде.
11. Синхронизация по времени хода часов.
12. Использование идеального генератора случайных чисел на основе получения данных (отсчетов – приведенной энергии каждого пикселя) реликтового фона радиоизлучения (квантизация).
13. Проверка целостности бинарного файла, как в стационарном режиме, так и загруженного в памяти, должна производиться внешними ресурсами.
14. Трехуровневая проверка целостности цепей. Заключается в получении хеша, полного хеша, хеша через промежутки, контрольной суммы.

Содержание

Введение	6
Функционал	7
1. Портфель	7
2. Цепочка блоков	8
3. Распределенное хранилище	9
Принцип организации сети. Сетевой протокол	11
1. Царь	11
2. Z level- Генералы	11
3. P level – обычная нода	11
4. L level – легкие ноды	12
5. S level - приват нода	13
6. O level - слип нода	13
7. Облачная нода	13
Таблица функций типов нод	14
Архитектура цепей	15
Область применения	16
Право	16
Страхование	16
Финансы	16
Медицина	16
Логистика	16
Проблемы билетного рынка	16
Углеводороды	17
Токен и его классификация	17
Средство платежа	17
Утилитарный токен	17
Цифровой инвестиционный актив	17
Токенизированный актив	17
Токеномика	17
Системная архитектура (архитекторы)	17
Основные фазы работы сети	18
Алгоритм шифрования	19
Режим записи	19
Цели блокчейн сети	19

Отстойник (функции)	19
Формулы и реализация	20
Пример майнинга	20
Формула псевдослучайного генератора и получения хеш строки	21
1. Генерируется стартовое значение ряда	21
2. Генерация следующего элемента ряда, GenP[byte], результат генерации	21
3. Получение N-го промежуточного элемента хеш строки	21
Полное исключение коллизий цепочки блоков	22
Коленчатый смарт-контракт	22
Проверка целостности нод	23
1. Модуль проверки целостности блоков всей цепи	23
2. Модуль проверки целостности блоков цепочки всех собственных транзакций	24
3. Модуль проверки целостного локального (оригинального) исполняемого бинарного файла	24
4. Запуск плагина	25
Получение хеш строки	25
1 этап	25
2 этап	25
3 этап	26
4 этап	26
5 этап	26
Листинг	27

Введение

Предоставляем платформу [IaaSB](#) - инфраструктуру в виде сервиса, где В - портфель.

Ценность - все происходящие события, которые генерит человек, определяется его ценностью и устойчивостью к несанкционированному доступу к этой информации.

Relictum Pro — первый глобальный сервис блокчейна, предоставляющий возможность наиболее оптимально использовать современное оборудование с большим заделом на будущее.

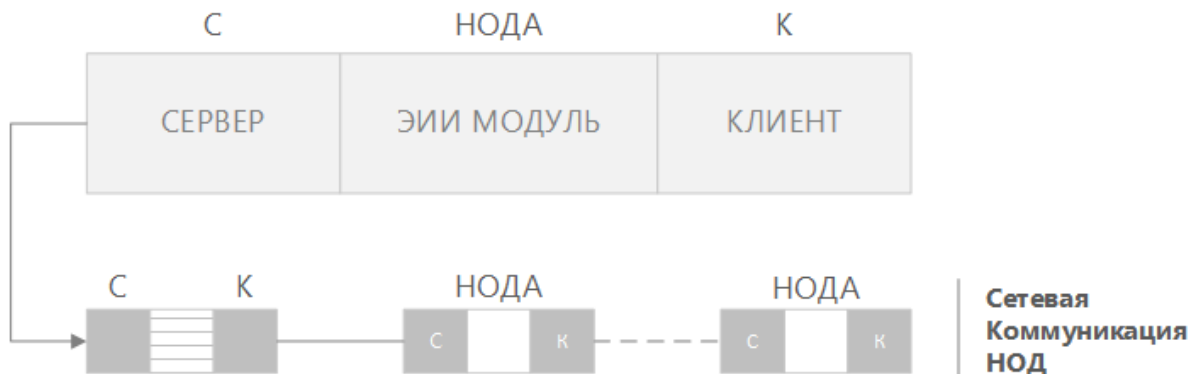
К основным особенностям относится:

1. Независимость от транспорта. Т.е. не обязательно использовать сеть Интернет, можно использовать WiFi, Блютуз, телефонные сети, оптоволоконные коммуникации, вплоть до передачи информации через флешку или оптические, квантовые перспективные сети.
2. Независимость от типа Клиента (тонкий клиент, толстый клиент, дата-центры, кнопочные мобильные устройства), где присутствует какая-либо операционная система.
3. TPS приближена к толщине каналов Мастернод.
4. Сетка блокчейн обладает 100% распределенностью. За конечный промежуток времени, в течение одних суток работы сети, полномочия делегируются до 100 000 раз какой-либо из нод, которая принимает решения за всю сеть. Протокол делегирования полномочий связан с генератором случайных чисел, получаемых из реликтового фона радиоизлучения, как наиболее оптимально описывающего кривую нормального распределения Гаусса.
5. Организация коленчатого смарт-контракта.
6. Наличие динамических цепей для хранения динамических данных, например, для повторяющихся логистических коленных смарт-контрактов.
7. Использование сквозной нумерации хешей цепей для гарантированного устранения коллизий на длительный период.
8. Распределенное хранилище. Фрагментация и дифференцированное размещение цифровых данных на устройствах пользователей, содержащих ноду.

9. Алгоритм передачи блоков по сети к каждой ноде предусматривает завершённый цикл, когда все ноды получили эти данные: 100% - (минус) Sleep (формула).
10. Нода может выйти из слипа только в том случае, когда прошла полная инициализация с подтверждением привязки к железу, получении хешей, проверки хешей, проверки бинарного файла внешним ресурсом и полной прокачки цепей.

Функционал

Нода (портфель)



ЭИИ - элемент искусственного интеллекта.

Нода представляет собой три сущности:

1. Портфель.
2. Цепочки блоков.
3. Распределенное хранилище.

1. Портфель:

Бинарный файл, который содержит следующие элементы:

1. Управление Нодами.
2. Эксплорер.
3. Генератор смарт-контрактов.
4. Создание IO.
5. Генерация Токенов.

6. Функционал транзакций любых видов валют.
7. Приват-чат.
8. Биржа.
9. Самодиагностика.
10. Ресурс обновления.
11. Защита от сбоев.
12. Привязка к железу.
13. Модуль связи с внешним ресурсом для выхода из слип режима.
14. Модуль инициализации API.
15. Модуль инициализации SDK.
16. Модуль инициализации API низкого уровня (socket).
17. Модуль работы с распределенным хранилищем.
18. Модуль авторизации.
19. Биометрический модуль. Биохеш.
20. Модуль синхронизации по времени (ход часов).
21. Модуль получения данных от внешнего источника данных реликтового излучения в виде трехмерной нормированной цифровой карты.
22. Модуль блокировки статуса. Используется для блокирования несанкционированного доступа как к файлу бинарника, так и к части загруженного файла приложения.
23. Модуль Тиков самодиагностики.
24. Модуль самоуничтожения. В случае несовпадения каких-либо критических хешей, происходит запуск инкапсулированной программы внутри приложения, которая производит удаление всех ресурсов, в том числе авторизационных данных пользователей.

2. Цепочки блоков:

1. Цепочка блоков представляет собой n-мерную матрицу цепей смарт-контрактов, которые регулируется механизмом подписей в виде хеш-строк с обязательным подтверждением и размещением в мастер-чейн каждой транзакции.
2. Размер каждого блока варьируется от 120 до 300 байт. При этом, каждый блок имеет сквозную нумерацию в n-мерной матрице.
3. Инфраструктура корреляции данных блоков организована таким образом, что поиск данных происходит от конца к началу. Это приводит к практически мгновенному получению информации о

данных в n-мерной матрице. Например: для получения списков всех транзакций криптовалюты, двигаясь от последнего отсчета к первому, этот список получается мгновенно, минуя промежуточные блоки за счет сквозной нумерации блоков в матрице. Это позволяет проводить аналитику любых данных в блокчейне локально; строить графики; находить пути пересечения данных; управлять динамическими смарт-контрактами; рассчитывать аналитику интерполированных данных и аппроксимировать функцию экстраполяции для расчета ожиданий на достаточно длительный период.

4. Наличие динамических блоков.
5. Каждый владелец ноды может создавать свою собственную базу данных для взаимодействия с динамическими блоками (смарт-контракт БД). В такой базе данных могут храниться, например, профайлы контрагентов и прочее. Также, динамические смарт-контракты могут применяться в качестве коленчатых смарт-контрактов.

3. Распределенное хранилище:

1. Необходимо наличие смарт-контракта распределенного хранилища и подтверждения владельца портфеля на использование ресурсов его вычислительной машины.
2. Минимальный выделяемый объем данных - 200 МБ. Максимальный – неограниченно.
3. Инициализация распределенного хранилища в портфеле позволяет владельцу зарабатывать на количестве обращений к базе распределенного хранилища.
4. Распределенное хранилище получает данные как есть (в виде массива байт) и никак не контролирует содержимое базы. А о содержимом базы заботится Нода, которая помещает дифференцированные данные хранилища во всю сеть.
5. Каждый элемент данных может иметь различную длину, от 1 Байта до максимального значения регистра компьютера (2^{64}).
6. Информация о куске байт хранится в смарт-контракте (цепочке) распределенного хранилища.
7. Дублирование кусков байтовой последовательности происходит многократно на разных нодах для исключения потери данных.

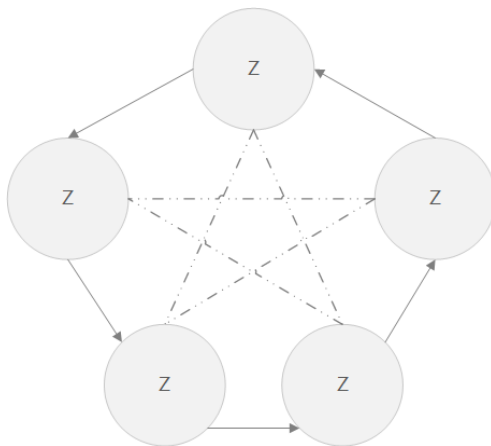
8. Владелец данных, помещаемых в распределенное хранилище, может помещать файл целиком в свою собственную БД в своём портфеле, как криптованную, так и расшаренную.
9. Данные, помещаемые в хранилище, имеют статусы от открытого вида данных до мозаично-перемешанных кусков. Данные могут иметь статус Share или Private.
10. При инициализации нового документа производится проверка на объективность и достоверность данных для исключения спама.
11. Помещение данных в распределенное хранилище является платным для исключения спама.

Принципы организации сети. Сетевой протокол:

1. Сеть имеет 5 уровней:

1. Царь.
2. Z уровень - генералы.
3. P уровень - обычная нода.
4. L уровень - легкие ноды.
5. S уровень - приват нода.
6. O уровень - слип нода.
7. Облачная нода

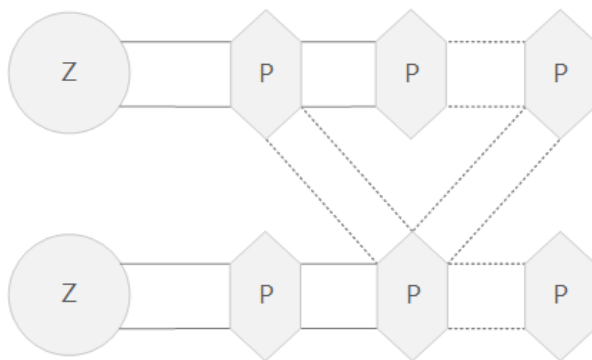
2. Z level (Генералы) (Zero-Level)



1) Кто-то из них Царь, но неизвестно.

2) Все Z работают как один компьютер с множеством процессоров.

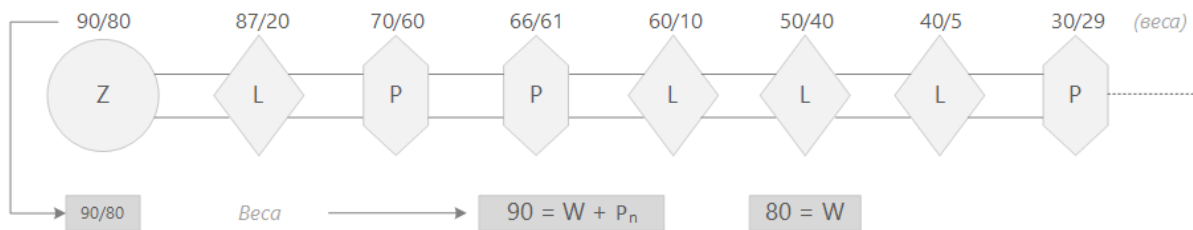
3. P-level (Power level) ноды 1-го уровня



Цепи Z-P.

Транспорт СС
(коммутационных
каналов).

4. L-level (легкая нода) 2 уровень



где

W - текущий вес ноды

P_n - результат по формуле

$P_n = G_p [R]$

где

G_p - генератор псевдослучайных чисел

R - выборка из реликта нормированного к диапазону (1-100%)

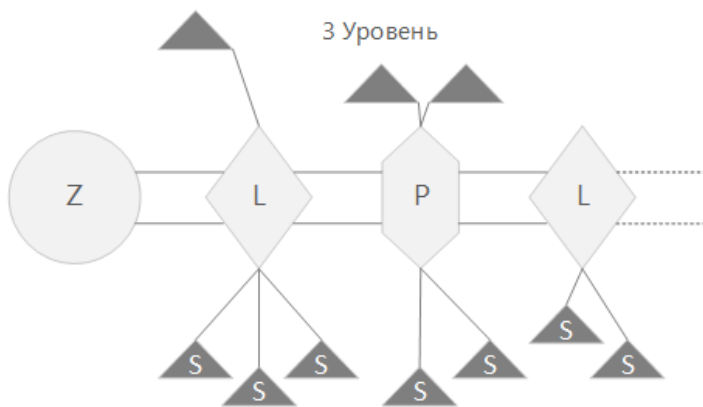
Формула веса (текущего + динамического)

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

G_p → номер в массиве R_n

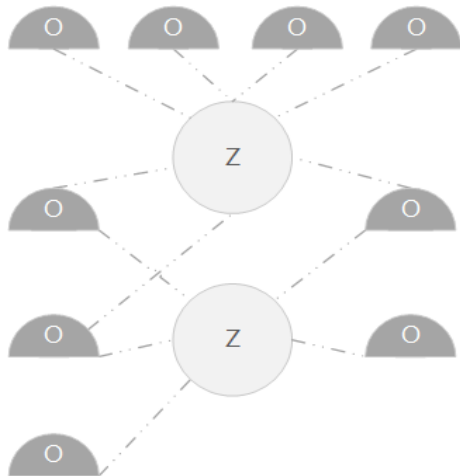
где n - номер регенерации сети.

5. S (Self нода) = приват нода









1. Свои транзакции
2. Выбранные ноды (только их транзакции)
3. Разрешение хранения данных распределенного хранилища

6. O - (Offline нода) = Слип нода



1. Попытка соединения с Z-нодой.
2. Получение адреса соединения (IP).
3. Проверка целостности локального ресурса.
4. Проверка железа.
5. Синхронизация хешей.

Таблица функций типов нод

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

- 0. Номер
- 1. Thumbs
- 2. White IP
- 3. Стать генералом
- 4. Стать царем

- 5. Инициация транзакции
- 6. Участие в голосовании
- 7. Заработок
- 8. Ограничение рейтинга

Архитектура цепей



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	синхронизирует не полные ноды
5 D	Time chain	вычисляет ход часов
6 D	Storage chain	цепочка смарт-контракта распределенного хранилища
7 D	Status chain	информация о статусах всех нод в сети

S - стандартная нода

D - dynamic нода



Область применения

Право:

Ценность, масштабируемость, способность противостоять инфляции и стимулы для их использования.

Страхование:

Оптимизация модели выплаты страховых взносов
Снижение или полное исключение мошеннических рисков
Автоматическая выплата страховых взносов с помощью смарт-контрактов
Повышение качества обслуживания клиентов

Финансы:

Усовершенствованный механизм идентификации личности
Сокращение рисков между контрагентами
Полная прозрачность процесса предоставления услуг
Быстрые трансграничные транзакции

Медицина:

Упрощение ведения электронных медицинских записей
Эффективный сбор и управление данными с медицинских устройств
Прозрачность отслеживания цепочек поставок лекарств
Модернизация медицинского страхования
Многоколенный смарт-контракт (доставка)

Логистика:

Эффективное отслеживание грузов, снижение вероятности потерь
Сокращение времени доставки
Прозрачность логистической информации
Увеличение пропускной способности в цепочке поставок

Проблемы билетного рынка:

Подделки
Спекуляции



Углеводороды:

Технология блокчейн дает нефтегазовой промышленности возможности для решения проблем с трансграничными перевозками, большим объемом транзакций, сложным документооборотом.

Токен и его классификация

Токен, как цифровое обязательство, можно классифицировать следующим образом:

Средство платежа:

Является аналогом традиционных денежных средств.

Утилитарный токен:

Используется как "топливо" внутри блокчейн-системы и обеспечивает ее работоспособность.

Цифровой инвестиционный актив:

Цифровой актив, имеющий юридическое подкрепление в реальном, не цифровом мире.

Токенизированный актив:

Цифровое обязательство обмена на реальный товар или услугу.

Токеномика:

Набор экономических правил и моделей, обеспечивающих функционирование экономики проекта, базирующегося на токенах.

Экономическая модель:

Формализованное описание различных экономических явлений и процессов.

Системная архитектура (архитекторы)



В качестве системного архитектора используется AGT (Algorithmic Game Theory), которая обеспечивает безусловное завершение сделок до начала следующей сделки. Так же используется Proof of Relay и Proof of Utility - доказательство работы и доказательство полезности, что приводит к безусловной монетизации использования Нод по закону Теории Игр.

Основные фазы работы сети

Режим регенерации (изменения ТОПОЛОГИИ, смены власти).
ПОСТРОЕНИЯ ТОПОЛОГИИ.

Регенерация всегда или по запросу на чтение или запись (по окончании запроса на чтение или запись). Если сеть бездействует, то регенерация происходит по таймауту.

В этом режиме сеть прекращает обработку запросов (ЧТЕНИЯ, ЗАПИСИ) и УЗЛЫ сети (НОДЫ) производят анализ работы предыдущего цикла - время обработки блоков, скорость доступа (СКОРОСТЬ КАНАЛОВ СВЯЗИ), количество обработанных транзакций. По результатам этого анализа составляется однозначная топология сети (иерархия, рейтинг) для следующего цикла. Выбор Царя происходит случайным образом на основе алгоритма RCS (random circle stop, УНИКАЛЬНЫЙ алгоритм) из числа Генералов в самом начале режима регенерации. Алгоритм RCS представляет собой игру, подобную «камень, ножницы, бумага». В течении назначенного периода УЗЛЫ производят последовательные вычисления (ПОЛУЧЕНИЕ СЛУЙНЫХ ДАННЫХ С ТЕКУЩЕЙ ТАБЛИЦЫ РЕЛИКТОВОГО ФОНА РАДИОИЗЛУЧЕНИЯ). По истечении назначенного времени происходит выбор Царя.

Выбор ГЕНЕРАЛОВ происходит из ТОПА НОД ПОЛУЧЕННЫХ ПО ФОРМУЛЕ (стр. 4) $Wd=W+Gp[Rn]$, число которых зависит от загруженности сети. В каждом цикле Генералы обновляются. (Нельзя быть Генералом два цикла подряд).

Царь рассчитывает (верифицирует, утверждает) каждый новый БЛОК и инициирует транзакции, агрегированные Генералами в предыдущем цикле. После проверки каждого блока Генералами, блоки передаются по сети вниз всем нодам.

Новая топология сети (иерархия) рассчитывается и утверждается Царем в конце текущего цикла и распространяется по сети от УЗЛА к УЗЛУ, в

соответствии с действующей на этом цикле топологией от каждого Генерала по соответствующей ветви.

На данной платформе, при автоматической обработке данных, не существует демократии.

Алгоритм шифрования

Представляет собой алгоритм SHA-1 с последующим преобразованием в Proprietary Jump Method, который использует недиагностируемую ошибку Overflow для одностороннего хеширования.

Режим записи

Любая Нода, получившая информацию для записи в БЛОКЧЕЙН, передает ее ВВЕРХ по сети. ЦАРЬ формирует блок на основе полученной информации, записывает в блокчейн и передает готовые блоки всем Генералам.

Цели БЛОКЧЕЙН СЕТИ

- Любой узел может подключиться к этой полностью открытой сети из любого места.
- ТОПОЛОГИЯ СЕТИ способствует эффективному совместному использованию сети.
 - Безопасный сетевой нейтралитет от инноваций сетевого уровня.
 - Всегда открыта и масштабируема.
 - Автоматическая эффективная и динамическая маршрутизация.
 - Механизм токенизации (автоматизация процесса расчетов с использованием механизма учета токена, как плата за поддержание и расширение, и оптимизация сети и сетевых подключений, активов передачи данных и стимулирование участвующих узлов.
- Проектировать и строить сеть блокчейнов следующего поколения.

Отстойник (функции)



ОТСТОЙНИК для Нод (Sleep режим), которые не справляются с производительностью сети или по каким-либо причинам разрывают соединение.

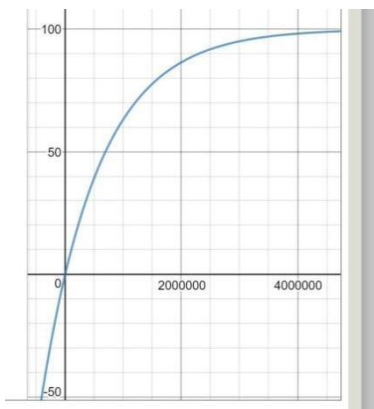
ВОЗВРАЩЕНИЕ Ноды ИЗ ОТСТОЙНИКА.
ДИСКРЕДИТАЦИЯ УЗЛА (ПОИСК ИУДЫ).

Формулы и реализация

Функция, замедляющая рост веса токена (стоимости) на бесконечности и не позволяющая превысить заранее заданную максимальную стоимость (эмиссию, вес...) = 100%

Формула роста стоимости токена до 100 единиц (%), на примере смарт-контракта майнинга proof of time:

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



Внизу ссылка проверки построения графика формулы:

http://www.yotx.ru/#!1/3_h/ubWwch@2cHBwf7Rqzhf23/aP9q/2DfT0qt7W9uHRysrW9sHkAODg5qO3u70K2Dq/2DfRINu7Fzyng83WI8bl1e7O5v7QMG

Пример майнинга

$$TicS = TikL + TicD$$

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

$$AmD = AmC - AmL$$

$$AmD = SUM(APn)$$

$$TicD = SUM(TDn)$$

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

где:

1 Tic = 24Hour

AmL - Last Amount

AmC - Current Amount

AmD - delta - SUM всех монет за период {TickL->TikC}

APn - Сумма каждого за период

ADn - delta

TicL - Last Tick

TDn - delta Tick Nod(каждой ноды)

TSn - Sum All Ticks Nod

TicD - delta - SUM всех тиков за период {TickL->TikC}

Формула псевдослучайного генератора и получения хеш строки

1. Генерируется стартовое значение ряда:

ep=15436757;

ep=abs(sin(ep));

ep=ep*429496729;

2. Генерация следующего элемента ряда, GenP[byte]-результат генерации:

gran=255;

ep=abs(sin(ep));

ep=ln(7)/ln(ep);

ep=ep*10;

ep=ep-trunc(ep);

GenP=trunc(ep*gran);

3. Получение N-го промежуточного элемента хеш строки s[byte]. s-строка символов. Hash[255]-массив byte. z-номер элемента массива[Hesh]. i-номер элемента массива строки[s]

В случае $\text{length}(s) < \text{length}(\text{Hash})$ к s добавляются значения 0[byte], чтобы соблюдалось условие $\text{length}(s) = \text{length}(\text{Hash})$

цикл

inc(z);

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



В случае $z > \text{length}(\text{Hash})$, z присваивается 1, что приводит к модификации z -го (промежуточного) элемента хеш.

Значения псевдослучайного генератора используются для получения абсолютно-случайных значений из матрицы весов реликтового фона радиоизлучения.

Полное исключение коллизий цепочки блоков

В хеше первые 4 байта заменяются на значение Count текущего номера MasterChain блока.

Это означает, что на 2147483647 транзакций гарантированно исключение коллизий.

Под каждую цепочку смарт-контракта создается 1 токен (для каждого владельца) с номиналом N ($\max 9223372036854775807$).

Коленчатый смарт-контракт

Методика расчета оптимального коленчатого смарт-контракта (токеномика).

Расчет коленчатого смарт-контракта с промежуточными событиями (чекпойнты).

Параметры смарт-контракта:

L - длина (метры, звенья, парсеки, колебания волн (количество периодов), шаги и т.п.)

$T(L)$ - время, затраченное на прохождение всей длины, с учетом остановок в чекпойнтах

L_n - длина n -го участка (чек пойнт)

T_n - время n -го участка

R_n - рейтинг (надежность n -ной точки)

G_n - результат псевдослучайного генератора изменения рейтинга в точке n

W_n - Влияние внешних параметров на каждый этап

G_n - Коэффициент надежности чекпойнта.

S_n - вычисляемая величина скорости преодоления L_n . Нужна для расчета корреляций с другими событиями предыдущих блоков и экстраполяции будущих событий и расчета надежности колен, что позволяет выбирать другую траекторию, например при загрузенности чекпойнтов (проблема перекрестков).

Z - Надежность всего пути. для одного блока

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

$$T_n = T_n * R_n * G_n * W_n * G_n$$

$$S_n = L_n / T_n$$

$$S = S_n / n$$

$$Z = S / L ; \text{ при } Z=1 \text{ (идеальная надежность).}$$

Также, можно вычислить слабое колено и другие параметры для аналитики.

При количестве блоков > 17 применяем метод наименьших модулей и можно считать такой смарт-контракт элементом ИИ (самообучение + самоаналитика).

Например, это можно применять для управления светофорами.

Проверка целостности нод

1. Модуль проверки целостности блоков всей цепи.

Формула:

проверка сохранения условия по всей цепочке, начиная со 2-го элемента

$$h(B_{n-1}) = H_n \text{ при } n[1..BlockCount-1]$$

где,

h - хеш блока

H_n - хеш в блоке

B_n - блок N

При невыполнении условия цепь считается поврежденной и наступает событие перезагрузки всей цепи.

2. Модуль проверки целостности блоков цепочки всех собственных транзакций.

Формула:

проверка сохранения условия по всей цепочке, начиная со 2-го элемента.

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

где,

h - хеш блока

H_n - хеш в блоке

B_n - блок N

При невыполнении условия цепь считается поврежденной и наступает событие перезагрузки всей цепи

3. Модуль проверки целостности локального (оригинального) исполняемого бинарного файла.

- A1 проверка имени (внешняя)
- A2 проверка даты создания (внешняя)
- A3 проверка места запуска (локальная)
- A4 проверка размера (внешняя)
- A5 проверка хеш самого файла (внешняя)
- A6 проверка привязки к железу (локальная)

Формула:

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

где,

h - хеш конкатенации переменных A

A - строковая переменная типа string

Результат h требует подтверждения через сеть других нод по запросу смарт-контракта целостности уникальных файлов



При невыполнении условия $\min(9)$ подтверждений нод считается поврежденным, что приводит к полной деауации приложения и отказу обслуживания сетью.

4. При запуске исполняемого бинарного файла он загружает и запускает плагин, который работает отдельно от основной программы. При этом основная программа ждет подтверждения проверки подлинности (ожидание сессионного ключа).

Для получения сессионного ключа плагин выполняет п.3. и отправляет $h(A1, A2, A4, A5)$ в сеть. Сеть, после проверки $\min(9)$ подтверждений динамической цепочки случайных нод отправляет в основную программу новый сессионный ключ. Сессионный ключ так же записывается в динамическую цепочку всех нод.

Таким образом, у Ноды меняется статус с SLEEP на ONLINE.

- Сам плагин - уникальный файл, генерируется сетью (по запросу) при старте основной программы.

- При запуске плагин проверяет синхронизацию времени сети и устройства, на котором он запущен.

- Плагин работает 1-2 секунды. Если за это время он не отправил $h(A1, A2, A4, A5)$, то все программы закрываются по ошибке.

- При повторном запуске этого же плагина сеть ответит ошибкой, так как плагин перестал быть уникальным.

- Уникальность плагина определяется хешем файла плагина, записанным в динамическую цепочку всех мастер нод.

Получение хеш строки

Получение хеш строки $f(X(L))$ (хеш) от строки L

1 этап - вычисляется стандартная функция sha1 (с небольшими модификациями),

в результате получаем $f(X(L))$ размерностью 20 байт.

2 этап - преобразование $f(X(L))$ в 32 байтное число с применением Побайтного сдвига с суммированием

Побайтный сдвиг с суммированием:



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

где $f(X(L))=sb$

При этом используется аппаратное превентивное блокирование ошибки переполнения ячейки процессора.

Пример:

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

Такая операция существенно экономит ресурсы, нет необходимости делать дополнительные проверки и вычисления.

3 этап

Производится несколько прыжков через барьер 255

```
inc(b,sb[i]  
inc(sb[i],b)  
где b байтовая ячейка
```

4 этап

Применяется функция побайтного сложения полученного хеша $f(X(L))$ с исходной строкой L

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

5 этап

Необходимо выполнить 3 этап для исключения обратного действия в 4 этапе.

Листинг



(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;
```