EchoRand

Данный документ описывает основной механизм работы алгоритма консенсуса **EchoRand**, лежащий в основе блокчейн-сети Echo.

За основу для алгоритма **EchoRand** взята теоретическая работа Algorand v9, которая описывает приход к консенсусу в децентрализованной сети на основе решения задачи о византийских генералах. В работе Algorand v9 излагается несколько возможных вариантов алгоритма. За основу для **EchoRand** взят вариант под названием **Algorand′2** с некоторыми изменениями.

А именно:

- Изменен способ определения участия конкретного исполнителя в конкретном шаге
- Отказ от модели однократных, производных ключей для подписи сетевых сообщений
- Изменен способ генерации разделяемого случайного состояния на третьем шаге алгоритма BBA

Основные компоненты

Основой EchoRand являются следующие компоненты:

- **испольлнитель** - аккаунт сети, выбранный на шаге раунда для выполнения определенного действия, связанного с консенсусом
- **набор создателей блока** - список исполнителей, определенных прооколом для текущего блока, которые получают возможность предложить вариант блока на текущем раунде
- **набор верификаторов** - список исполнителей, определенных протоколом для конкретного шага, которым доверяется выполнить действия верификации, определенные шагом, на котором они были выбраны
- **проверяемая случайная функция (VRF)** - псевдослучайная функция, которая предоставляет публично проверяемые доказательства правильности своих выводов
- **сид раунда** - псевдослучайное значение, изменяемое на каждом блоке. Служит основой для генерации набора верификаторов и создателей блоков

- **Graded concencus** - один из этапов консенсуса, на котором каждый из верификаторов должен объявить свое предварительное решение относительно текущего блока
- **Бинарное Византийское Соглашение (BBA)** - решение проблемы византийского соглашения, в основе которого лежит передача двоичных данных между участниками и сверка результатов с общей картиной
- **узел сети** - запущенный экземляр приложения echo, находящийся в сети остальных узлов, имеющий последнее актуальное состояние (синхронизация до последнего блока)

Условные обозначения

Обозначение	Описание
$r >= 1$	текущий раунд алгоритма, фактически равно количеству блоков в базе плюс 1
$s >= 1$	текущий номер шага алгоритма в раунде
Q_r	сид раунда **r**
$VRF(r, s)$	упорядоченное множество исполнителей, которые участвуют в шаге **s** раунда **r**
$VRFN(r, s)$	упорядоченное множество индексов исполнителей из $VRF(r, s)$, которые зарегистрированы на текущем узле и участвуют в шаге **s** раунда **r**

VRF

Понятие проверяемой случайной функции (VRF) было введено Микали, Рабином и Вадханом. Это псевдослучайная функция, которая предоставляет публично проверяемые доказательства правильности своих выводов. При заданном входном значении `x` владелец секретного ключа `SK` может вычислить значение функции $y = F_{SK}(x)$ и доказательство $P_{SK}(x)$. Используя доказательство и открытый ключ $PK = g^{SK}$, каждый может проверить, что значение $y = F_{SK}(x)$ действительно вычислено правильно, но эту информацию нельзя использовать для поиска секретного ключа.

Применение VRF в EchoRand заключается в следующем - имея псевдорандомное значение для каждого раунда Q_r и функцию VRF каждый из улов сети может определить список исполнителей $VRF(r, s)$ в шаге **s** раунда **r** и на основании его совершить необходимые действия, если авторизованный аккаунт на узле является частью $VRF(r, s)$, а также верифицировать действия других участников на наличие у них права дейсвия на этом шаге.

Определения активных исполнителей

Проверяемая случайная функция на каждом раунде **r** и шаге **s** строится итеративно, следующим образом:

$$1.VRF_0(r, s) = SHA256(Q_{r-1}, r, s)$$
$$2.VRF_n(r, s) = SHA256(VRF_{n-1}(r, s))$$

Результатом работы данной функции является массив случайных значений:

$$VRF(r, s) = VRF_0(r, s), VRF_1(r, s), ...$$

Конкретный исполнитель вычисляется из хеша $VRF_i(r, s)$ таким образом, чтобы вероятность выбора исполнителя активным была пропорциональна его балансу в системе на момент блока `r - 2`.

Набор $VRFN(r, s)$ это массив индексов, различный для каждого узла сети и такой, что если $i \in VRFN(r, s)$, то из $VRF_i(r, s)$ вычисляется идентификатор пользователя являющийся исполнителем для данного раунда и шага на выбранном узле.

Иными словами, $VRFN$ является выборкой тех исполнителей из VRF, которые авторизованы на текущем узле и должны выполнятся на конкретном раунде и шаге.

На разных узлах сети, на одном и том же раунде и шаге алгоритма, множества $VRFN$ будут разные, а множество VRF будет одинаковым.

Ненерации случайного значения раунда - сида раунда

Начальный вектор $Q(0)$ выбирается случайным образом при инициализации базы.

Далее, вектор Q_r вычисляется следующим образом при создании нового блока:

Для непустого блока $B(R)$:

$$Q_r = H(signQ_{r-1}, r)$$

В этом случае для подписи используется приватный ключ исполнителя, который создает блок.

В случае, если $B(R)$ блок пуст:

$$Q_r = H(Q_{r-1}, r)$$

Генерации случайного значения на s = 7,10,13,… шаге BBA

$$BBARAND(s) = lsb(SHA256(Q_{r-1}, r))$$

Раунды консенсуса

Основные этапы консенсуса:

- Генерация блока исполнителями, выбранными на роль создателя блока
- Голосование за лучший блок
- Достижение соглашения между узналами о лучшем блоке

Генерация блока

Определенное количество исполнителей (набор создателей блока) генерируют блок.

Набор создателей блока определяется каждый блок с помощью проверяемой рандомной функции (VRF). В результате каждый узел сети получает набор $VRF(r, s)$ и подмножество $VRFN(r, s)$ - список аккаунтов, авторизованных на данном узле. Если $VRFN(r, s)$ не пустой, узел

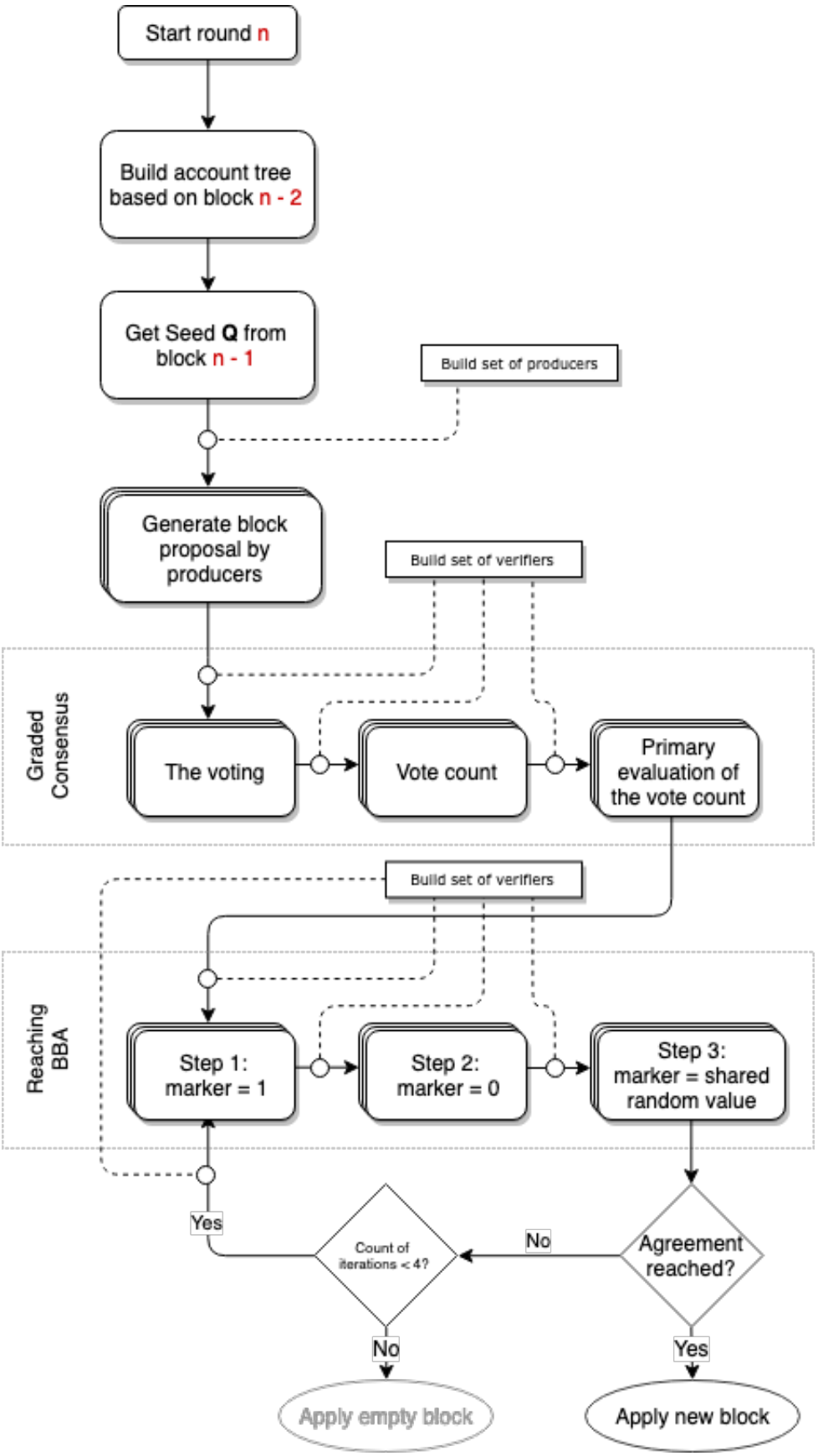

Figure 1: EchoRand steps

выпускает предложение блока на основании транзакций, находящихся в мемпуле узла.

Голосование за лучший блок (Graded concencus)

Состоит из 3-х шагов. На данном этапе цель верификаторов - проголосовать и заявить сети, кого из продюссеров они считают лучшим кандидатом для текущего блока.

Шаг 1 - голосование

Каждый из выбранных верификаторов говорит сети, какой из блоков он считает предпочтительным для секущего раунда.

Шаг 2 - подсчет голосов

Сообщения предыдущего шага получают все активные узлы сети. Выбранные верификаторы на шаге 2 подсчитывают, какой из предложенных блоков набрал больше голосов и объявляют об этом сети.

Шаг 3 - первичная оценка подсчета голосов

Имея результаты голосования на предыдущих шагах, узлы имеют представление о том, смогла ли сеть договориться о выборе продюссера для текущего раунда. Результат (договорились или нет) и о чем именно договорились каждый верификатор формирует в сообщение и отправляет в сеть.

После данного шага все узлы в сети знают результат голосования. В честной сети этого было бы достаточно для завершения раунда. Но так как мы допускаем возможность недобросовестных участников, сети необходимо сверить данные. Это задача следующего этапа.

Достижение Binary Byzantine Agreement (BBA)

На каждом шаге работы алгоритма, узлы в сети можно разбить на два множества:

- узлы, которые получили за предыдущие раунд(-ы) достаточное количество сообщений (с некоторым одинаковым значением), позволяющее им предложить это значение в качестве решения.
- узлы, которые получили два варианта решения в сообщениях и не могут отдать предпочтение какому-либо из них.

В последнем случае, неопределившихся узлы используют VRF для генерации разделяемого случайного числа из множества { 0, 1 } для принятия и отсылки своего решения. В силу того, что случайное число будет одно и то же для всех “неуверенных” узлов, все такие узлы примут одинаковое решение.

Этап состоит из циклов, в которые входит 3 шага. На каждом из шагов новый сет верификаторов отправляет свое видение результата голосования в бинарном виде. Если в результате цикла (3-шагов) `2/3 + 1` верифаеров сходятся во мнении, блок применяется. Если же нет - цикл запускается снова.

В случае, если за 4 цикла (в них участвует 12 разных сетов верификаторов) сеть не смогла прийти к единому мнению, в сети применяется пустой блок и начинается следующий раунд с самого первого шага - генерации блока.

Применение блока всеми участниками сети

Все узлы сети получают все сообщения, отправленные исполнителями на всех этапах консенсуса. Соответственно, каждый из узлов на момент окончания консенсуса сам для себя определяет его окончание и понимает, какой именно блок необходимо применить и добавить в цепочку. Т.е. итоговое сообщение с результатирующей информацией никем не отправляется, так как в этом нет необходимости.

Разрешение ветвлений

Количество шагов алгоритма и зависимость от всех базы аккаунтов делает возможность ветвлений маловероятной. Однако EchoRand все равно имеет механизм разрешения ветвлений. Разрешение ветвления происходит по одному из следующий сценариев:

- Переключиться на самую длинную цепочку при наличии нескольких цепочкек.

- Если имеется более одной длинной цепочки, следовать за той, у которой последний блок не пустой. Если у всех из них в конце пустые блоки, проверять вторые и последующие блоки с конца до первого непустого блока.
- Если имеется более одной длинной цепочки с непустыми блоками в конце, скажем, цепочки длины r, следовать той, чей блок r имеет наименьшее значение хеша.

Оптимизация сетевого протокола

Для уменьшения количество сообщений с информацией о предложении блока, в протоколе имплементированы следующие оптимизации:

- в случае, если узел сети получает предложение блока, которое уже не первое для раунда и не является лучше предыдущего, узел не отправляет сообщение о нем остальным узлам.
- в случае, если на узле авторизовано несколько исполнителей для раунда генерации блока, узел сам определяет, какой из блоков лучший кандидат и отправляет в сеть только одно предложение блока.

Исключительные ситуации

Отсутствие сети

Шаги алгоритма не будут получать сообщения и все выходы из шагов будут происходить только по срабатыванию таймера. Так как конец раунда на данный момент происходит только как реакция на приходящее сообщение, то шаги BBA будут выполнятся в цикле до достижения константы μ. В результате будет сгенерирован пустой блок.

Восстановление сети

Узлы, которые в результате восстановления сети войдут в раунд в середине, будут содержать неполные данные в своих контекстах раунда. В результате они будут генерировать либо неверные оценки, либо голосовать за

пустой блок. В каждом из этих вариантов узлы будут себя вести как узел-злоумышленник. В результате информация с таких узлов будет отфильтрована алгоритмом bba. В силу неполных данных в локальных контекстах такие узлы завершат раунд:

- с неверным блоком
- с пустым блоком

Произойдет ветвление, которое будет автоматически разрешено, когда остальная сеть уйдет вперед в процессе генерации новых блоков.

Неполная база blockchain в узле

Случай, когда локальная база данных узла “догоняет” базу данных сети. При этом алгоритм не может работать в силу того, что отсутствуют значения:

- HB_{r-1} - хеш последнего созданного блока
- Q_{r-1} - случайное значение последнего раунда работы алгоритма

Требуется определить момент, когда локальная база “догонит” базу данных сети и запустить раунд алгоритма.

Отсутствие активных исполнителей на шаге

Набор исполнителей вычисляет с помощью VRF и не зависит реального наличия исполнителей в сети. Может возникнуть ситуация, когда для какого-то шага алгоритма нет активных исполнителей. В этом случае активные участники сети по истечению таймаута просто переходят на следующий шаг или применяют пустой блок, в случае, если это был последний шаг.

Делегация участия в консенсусе

Исполнителями на раундах являются аккаунты, но для участия в консенсусе необходим активный узел, так как только имея актуальное состояние сети и наличие свободных транзакций в мемпуле позволяют определять сеты исполнителей, собирать и проверять сообщения.

Учитывая, что большинство аккаунтов не имеют возможность поддерживать активный узел в сети, но могут быть выбраны для участия в раунде, в протоколе реализован механизм делегации участия в консенсусе другим аккаунтам. Это означает, что аккаунт `A` может установить для себя доверенный аккаунт `B` с заведомо запущенным узлом в сети и тем самым предоставить аккаунту `B` возможность выполускать сообщения консенсуса в тот момент, когда исполнителем был выбран аккаунт `A`.

По умолчанию доверенным аккаунтом `B` для аккаунта `A` становится тот аккаунт, который зарегистировал аккаунт `B` в сети.

> Создание аккаунта в сети Echo требует создания соответствующей транзакции, добавляенной в блок