УДК 004.056.55
doi:10.31799/1684-8853-2024-3-32-40
EDN: FNYMEW

Научные статьи
Articles

Распределенный протокол генерации псевдослучайных чисел на основе алгоритма проверяемой случайной функции

И. С. Величко[a], *аспирант, orcid.org/0009-0005-6662-5606*
А. В. Афанасьева[a], *старший преподаватель, orcid.org/0000-0003-3001-0990*
С. В. Беззатеев[a], *доктор техн. наук, профессор, orcid.org/0000-0002-0924-6221, bsv@vu.spb.ru*
[a]*Санкт-Петербургский государственный университет аэрокосмического приборостроения, Б. Морская ул., 67, Санкт-Петербург, 190000, РФ*

Введение: *одним из решений, применяемых для генерации случайных чисел в области безопасности смарт-контрактов, является «проверяемая случайная функция». Сегодняшние централизованные решения, основанные на этом алгоритме, не предоставляют прозрачности участникам-клиентам системы генерации, что вызывает беспокойство по поводу безопасности.* ***Цель:*** *разработать протокол работы системы на основе проверяемой псевдослучайной функции для децентрализованной блокчейн-системы с высоким уровнем защиты данных от фальсификации.* ***Результаты:*** *для решения проблемы подделки начального входного значения предложен протокол, основанный на замене классической централизованной системы на основе единичного оракула на децентрализованную. Для обеспечения безопасности функционирования разработана система формирования общего распределенного секретного ключа, а также методы генерации псевдослучайных значений на основе полученного секрета. Данный протокол реализован с использованием алгоритма обмена ключами без участия дилера. Работа протокола описана в контексте абстрактной Ethereum-подобной блокчейн-модели с применением узлов, функционирующих в рамках консенсуса «доказательство активности», в целях повышения доступности и удобства рядовых пользователей.* ***Практическая значимость:*** *разработанный протокол предоставляет эффективное решение для защиты системы генерации псевдослучайных чисел от подделки входного значения, формируемого смарт-контрактом. Благодаря введению системы разделения секретов без дилера и возможности цикличности раундов регистрации участников повышаются безопасность, гибкость и масштабируемость системы.*

Ключевые слова *— проверяемая случайная функция, блокчейн, смарт-контракты, распределенные системы, электронная подпись, схема Шнорра, разделение секрета, алгоритмы генерации случайных (псевдослучайных) чисел.*

Для цитирования: Величко И. С., Афанасьева А. В., Беззатеев С. В. Распределенный протокол генерации псевдослучайных чисел на основе алгоритма проверяемой случайной функции. *Информационно-управляющие системы*, 2024, № 3, с. 32–40. doi:10.31799/1684-8853-2024-3-32-40, EDN: FNYMEW
For citation: Velichko I. S., Afanasieva A. V., Bezzateev S. V. Distributed pseudorandom generation protocol based on verifiable random function algorithm. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2024, no. 3, pp. 32–40 (In Russian). doi:10.31799/1684-8853-2024-3-32-40, EDN: FNYMEW

Введение

Проблема безопасной генерации случайных чисел в области технологий распределенного учета является серьезным вызовом для разработчиков смарт-контрактов и вопросом доверия для пользователей. В настоящее время существует достаточное количество технологических решений [1, 2], которые в некоторой степени повышают уровень безопасности, создавая алгоритмы, усиленные открытым ключом или криптографией с задержкой по времени. Одним из представителей таких алгоритмов является «проверяемая случайная функция» (Verifiable Random Function, VRF) [3, 4]. VRF — криптографический алгоритм, который позволяет безопасно генерировать случайные числа с возможностью их верификации с использованием открытого ключа. В данной статье представлены стандартный алгоритм VRF, рассмотрены его плюсы и минусы, а также предложена усовершенствованная концепция системы генерации псевдослучайных чисел.

Классический протокол VRF

Verifiable Random Function позволяет создавать псевдослучайные числа, вовлекая стороннего участника вне блокчейна в процесс генерации [5]. Этот участник является обладателем секретного ключа, необходимого для создания псевдослучайного числа. Чтобы предотвратить манипуляции результатом со стороны участника вне цепи, в алгоритм передается некоторое псевдослучайное значение, предоставленное пользователем протокола. Это означает, что ответ (результат) любого участника вне цепи можно проверить на правдоподобие (т. е. на то, что алгоритм генерации был выполнен согласно общеизвестной схеме).

Терминология для дальнейшего описания проверяемой случайной функции:

B — генератор группы эллиптической кривой [6];

PK — открытый ключ VRF;

SK — секретный ключ VRF;

$Gamma$ — псевдослучайная точка;

$seed$ — псевдослучайное входное значение алгоритма;

q — порядок группы эллиптической кривой;

c — доказательство алгоритма;

s — подпись доказательства.

Работу протокола можно представить в виде схемы, куда входят три сущности.

— Потребитель — клиентский контракт. Расположен в сети блокчейна EVM (Ethereum Virtual Machine — виртуальная машина, позволяющая выполнять код смарт-контрактов) (например, Ethereum [7, 8]). В запросах на случайной генерации он указывает технические параметры, такие как открытый ключ внешнего сервиса и количество заказанных чисел.

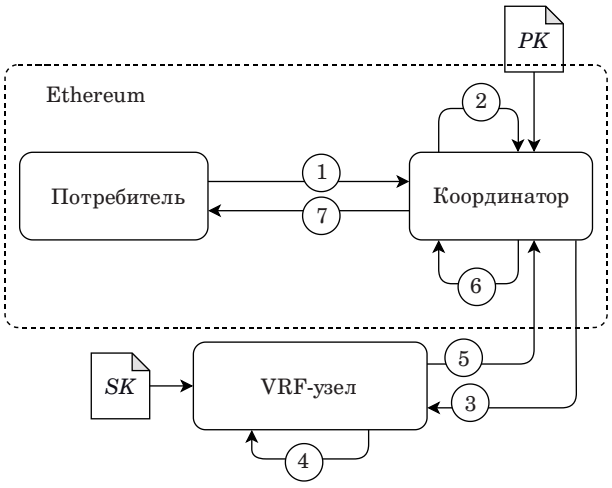
— Координатор — основной контракт. Координирует запросы к внешнему сервису и проверяет правильность результатов. Расположен в сети блокчейна EVM.

— Внешний сервис VRF — узел вне сети. Реализует генератор алгоритма VRF. Хранит секретный ключ SK на своей стороне.

Можно выделить семь основных шагов для описания полного цикла создания псевдослучайного значения. Схема протокола показана на рис. 1.

1. Запрос случайности. Вызов контракта координатора с параметрами PK и количеством запрошенных чисел.

2. Регистрация запроса в контракте координатора. Создание псевдослучайного входного значения $seed$ и регистрация запроса.



■ ***Рис. 1.*** Упрощенная схема VRF-протокола

■ ***Fig. 1.*** Simplified scheme of the VRF protocol

3. Запрос внешнего сервиса. Фактически внешний узел считывает информацию из координатора.

4. Генерация псевдослучайности. Генерация псевдослучайного значения и его доказательства с участием SK и $seed$. Результат алгоритма — это доказательство.

5. Возврат доказательства на контракт.

6. Проверка псевдослучайности.

7. Возврат результата в виде псевдослучайного числа.

Работа VRF представляет три основных шага.

1. Генерация $seed$. Операция создания псевдослучайного входного значения. Формирование значения происходит согласно определению

$$seed = hash(preSeed + blockHash),$$

где $preSeed$ — общедоступные хеш-значения пользователя, хранящиеся на контракте координатора; $blockHash$ — хеш выпущенного блока (блока, внутри которого хранится информация о запросе в службу VRF); $hash$ — операция хеширования конкатенированных значений.

2. Генерация $Gamma$. Операция создания псевдослучайного значения и его доказательств. Параметры алгоритма: входные — $seed$, SK; выходные — $Gamma$, c, s.

Опишем подробно алгоритм генерации псевдослучайности.

Функция $rand$ получает случайное число через специальное программное обеспечение, которое считывает шум с драйверов и других устройств:

$$nonce = rand(0...n);$$
$$sm \equiv nonce \bmod q.$$

Также подготавливаются ключи и псевдослучайный хеш для основного алгоритма:

$$sk \equiv SK \bmod q;$$
$$PK \equiv sk \cdot B.$$

Следующие формулы определены в рамках эллиптических кривых над конечным полем.

$$H = hashAndConvertToPnt(PK, seed), \qquad (1)$$

где $hashAndConvertToPnt$ — функция последовательного конкатенирования и создания хеш-значений переданных аргументов с последующей конвертацией в точку на кривой.

Псевдослучайное значение (псевдослучайная гамма-точка) формируется из секретного ключа и псевдослучайного хеша:

$$Gamma = sk \cdot H. \qquad (2)$$

Для контроля над *Gamma* и *PK* в процессе доказательства создаются вспомогательные переменные *U* и *V*:

$$U = sm \cdot B; \quad (3)$$

$$V = sm \cdot H. \quad (4)$$

Окончательное доказательство генерируется по следующему алгоритму:

$$c = concatAndHash\left(H, PK, Gamma, U, V\right), \quad (5)$$

где *concatAndHash* — функция последовательного конкатенирования и создания хеш-значений переданных аргументов. Вычисление функции выполняется следующим образом: для каждой точки берутся значения *x* и *y* в виде последовательностей байтов, данные последовательности объединяются. Все аргументы функции по очереди объединяются, после чего хеш вычисляется с помощью хеш-функции (в текущей версии обычно используется keccak256). Завершением работы алгоритма является создание подписи:

$$s \equiv \left(nonce + c \cdot sk\right) \bmod q. \quad (6)$$

3. Валидация *Gamma*, или проверка псевдослучайности. Параметры алгоритма: входные — *Gamma*, *c*, *s*; выходные — *true/false*.

Первым этапом проверки является проверка всех входных параметров на принадлежность к эллиптической кривой. Далее параметр *H* вычисляется таким же образом, как и в алгоритме генерации. Промежуточные значения *U′*, *V′* вычисляются как разность следующих произведений:

$$U' = s \cdot B - c \cdot PK =$$

$$= sm \cdot B + c \cdot sk \cdot B - c \cdot PK = U; \quad (7)$$

$$V' = s \cdot H - c \cdot Gamma =$$

$$= sm \cdot H + c \cdot sk \cdot H - c \cdot sk \cdot H = V. \quad (8)$$

Параметр *c′* вычисляется таким же образом, как и в алгоритме генерации. Проверка считается успешной, если параметры *c* и *c′* равны.

Уязвимости протокола

Несмотря на предполагаемую случайность и непредсказуемость результатов вычислений, для конечного пользователя классическая система генерации псевдослучайного числа имеет существенный недостаток, а именно централизацию вычислений и единый, явно хранимый, ключ.

Справочная информация: MEV (максимальное извлекаемое значение) — термин, используемый больше в экономической сфере, описывает способность майнера изменять последовательность выполняемых транзакций в блоке для выполнения различных видов арбитражных операций. Эта функция также имеет чисто техническое применение, а именно возможность влиять на значение хеша путем перестановки транзакций. В рамках консенсуса PoS (Proof of Stake — доказательство доли владения) влиять на значение хеша возможно за счет перестановки подписей в разделе тела блока.

Перечислим основные уязвимости.

Сговор потребителя VRF и автономного сервиса VRF. Все значения, участвующие в генерации исходных данных, являются общедоступными и предсказуемыми: идентификатор подписки, одноразовый номер и адрес пользователя, хеш блока. Единственным камнем преткновения в этом случае может быть хеш блока, но это значение априори криптографически слабо защищено от преждевременной компрометации или подделки. Проблема стоит особенно остро, учитывая использование MEV в блокчейне Ethereum. Использование mev-boost (сеть MEV-ботов) в качестве системы, основанной на доверии, создает потенциал для централизованных вычислений и взлома хеша блоков. Можно было бы сделать замечание, что выходное значение алгоритма VRF представляет собой 256-битный хеш, и в этом случае было бы довольно сложно перебирать всевозможные комбинации транзакций, чтобы сформировать правильный блочный хеш. Однако большинству пользователей VRF нужны числа меньшего размера, например, в лотерее Pancake (смарт-контракт платформы обмена криптовалют PancakeSwap, где пользователи могут приобретать билеты за основную валюту BNB для участия в ежедневном розыгрыше с целью выиграть денежный приз) используются билеты — числа от 1 000 000 до 1 999 999, что значительно увеличивает вероятность коллизий и упрощает выбор необходимых значений.

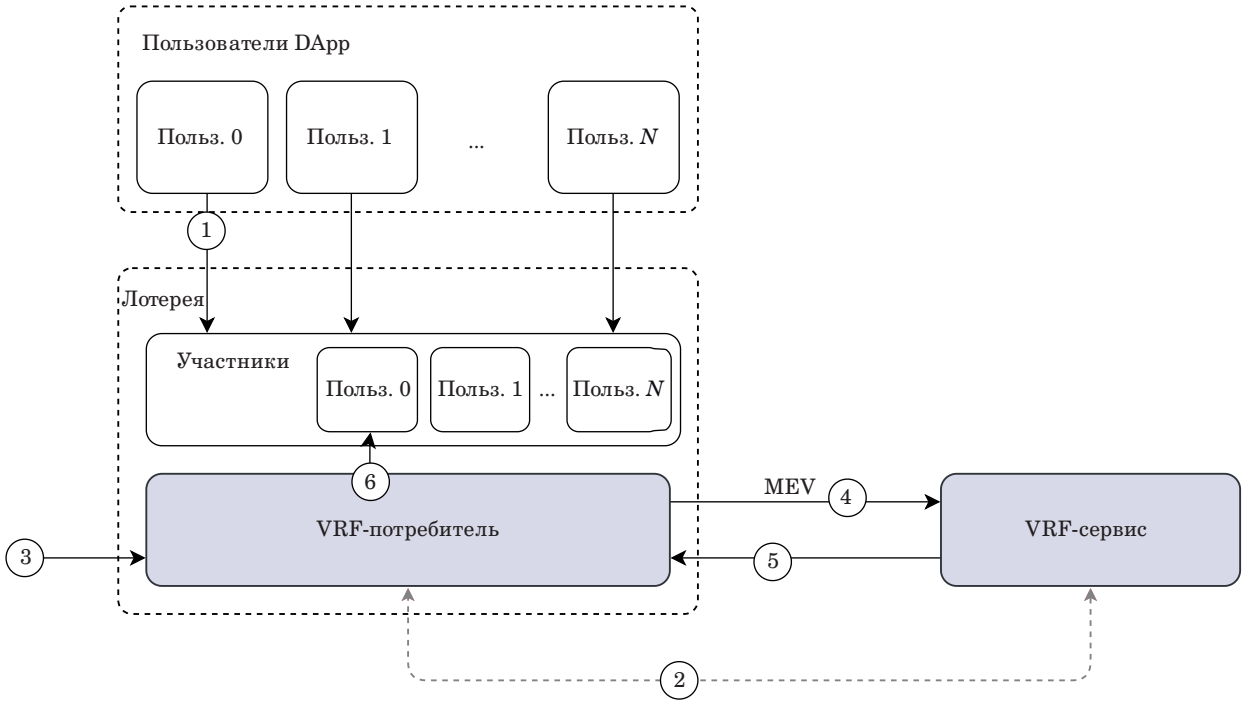
Приведем пошаговое описание схемы, показанной на рис. 2.

1. Регистрация пользователя DApp (Decentralized Application — децентрализованное приложение) или покупка пользователем лотерейного билета.

2. Сговор потребителя VRF (самой лотереи) со службой VRF по скрытому каналу связи. Потребитель должен предоставить идентификатор контракта, техническую информацию о подписке и временные характеристики (например, номер блока), в соответствии с которыми будет совершена транзакция.

3. Запуск процесса завершения лотереи.

4. Запрос на получение псевдослучайного числа. Аббревиатурой MEV обозначена возможность подделать хеш блока в момент выдачи запроса на генерацию псевдослучайного числа.

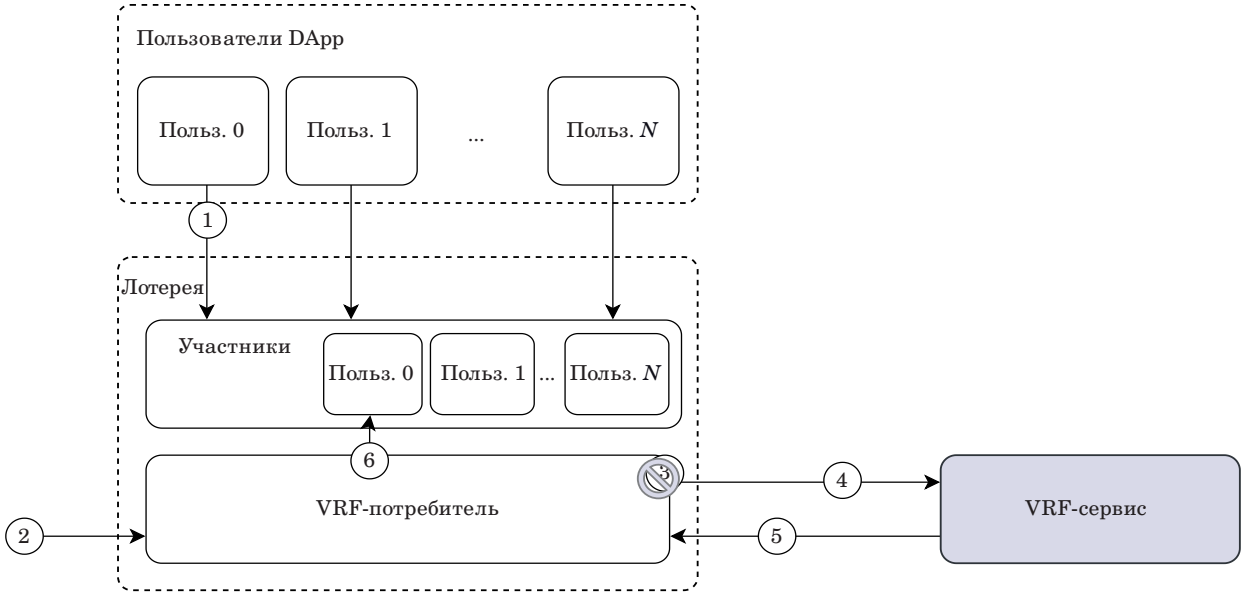


■ ***Рис. 2.*** Атака на протокол VRF через MEV (сговор VRF-потребителя и VRF-сервиса)
■ ***Fig. 2.*** Attack on the VRF protocol via MEV (collusion of the VRF of the Consumer and the VRF of the Service)

5. Возвращение псевдослучайного числа.
6. Выбор победителя.

Игнорирование транзакции. Другим возможным вектором атаки может быть игнорирование транзакции. Атака проста сама по себе: если сервис не удовлетворяет возможное число, полученное из хеша создаваемого блока, то эта транзакция просто не включается в блок и ожидает своей возможности быть обработанной в следующий раз. Символ ∅ на рис. 3 означает игнорирование транзакции.

Опишем пошагово схему, представленную на рис. 3.



■ ***Рис. 3.*** Атака на протокол VRF путем игнорирования транзакции
■ ***Fig. 3.*** Attacking the VRF protocol by ignoring the transaction

1. Регистрация пользователя/покупка лотерейного билета пользователем.

2. Начало процесса завершения лотереи.

3. Построение блока без учета выполненного вызова контракта. Игнорирование транзакции майнером.

4. Запрос псевдослучайного числа.

5. Возвращение псевдослучайного числа.

6. Выбор победителя.

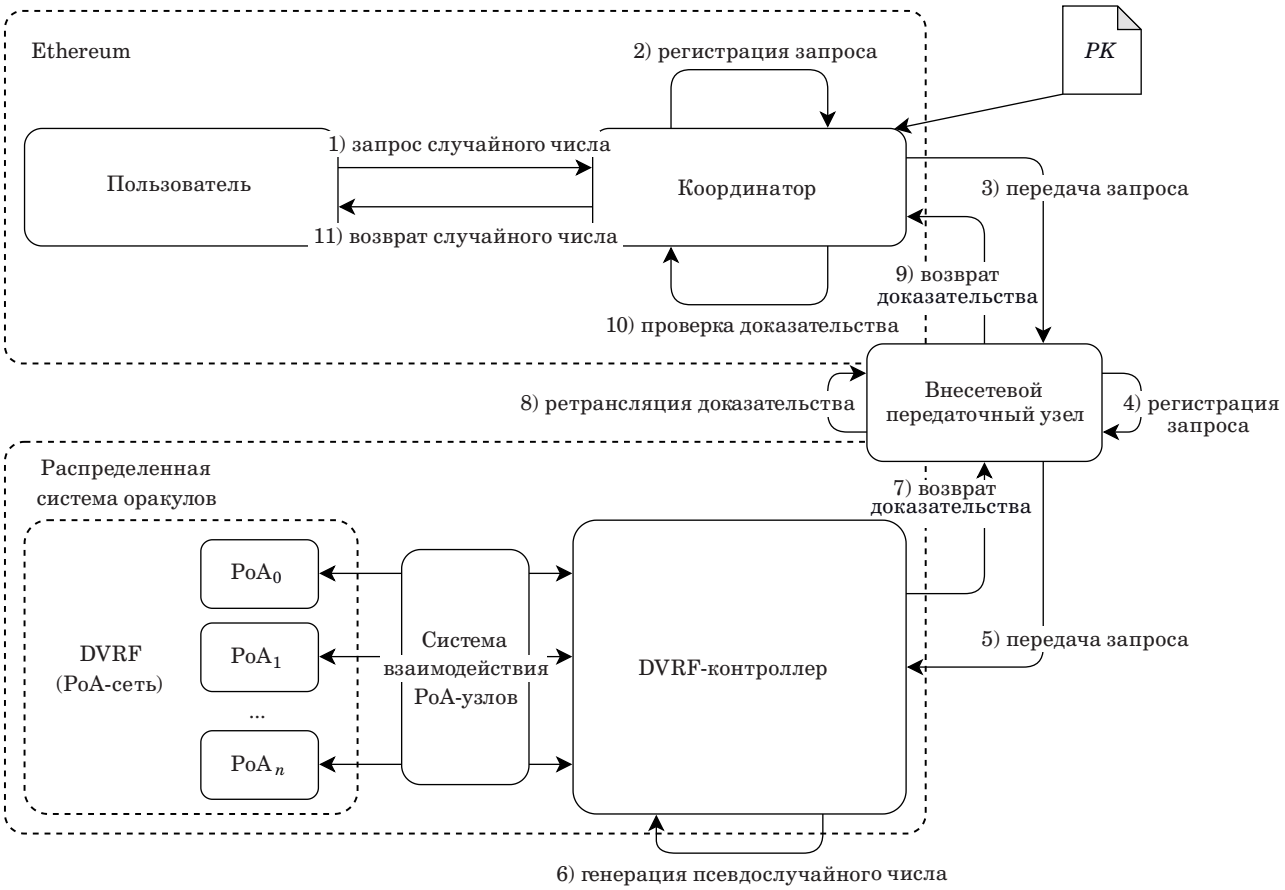
Как итог: проблема классического алгоритма заключается в том, что начальное входное значение хранится публично. Это обстоятельство предоставляет злоумышленникам возможность подделать секретное значение, поскольку доступ к начальному входному значению позволяет им воспроизвести или предсказать секретный ключ. Таким образом, отсутствие мер по защите начального входного значения существенно снижает безопасность всей системы.

Распределенный VRF-протокол

Если нет способа предотвратить подделку начального входного значения, то необходимо найти способ скрыть секрет *SK*. Изложенная выше проблема приводит к необходимости создания системы, в которой значение секретного ключа хранится не явно, а в виде отдельных проекций (точек на эллиптической кривой), совместно используемых между участниками. Мы воспользовались элементами схемы Шамира для обеспечения безопасного распределения секрета между участниками протокола [9]. В данной работе представлен вариант решения, основанный на алгоритме обмена ключами без участия дилера [10–14]. Работа протокола описана в соответствии с абстрактной Ethereum-подобной блокчейн-моделью. Поначалу предполагается использование узлов PoA (Proof of Activity) из-за большей распространенности и простоты использования обычными пользователями.

По сравнению с предыдущей версией системы (см. рис. 1) эта схема (рис. 4) вносит изменения в алгоритм генерации псевдослучайности путем создания протокола на основе готовой блокчейн-сети. Система разделена на три основных компонента:

1) алгоритм проверки псевдослучайных чисел по цепочке (предлагаемая хостинговая сеть Ethereum или другой EVM блокчейн);



■ ***Рис. 4.*** Абстрактное представление распределенного VRF-протокола (или DVRF)

■ ***Fig. 4.*** Abstract representation of the distributed VRF protocol (or DVRF)

2) встроенный алгоритм генерации псевдослучайных чисел;

3) автономный ретранслятор данных между двумя сетями.

На стороне генератора PoA-узлы участвуют в создании псевдослучайных чисел. Каждый узел имеет право «подать заявку на участие». Количество участников ограничено. На этапе регистрации каждый PoA участвует в раунде обмена проекциями секретного ключа, генерируя индивидуальные секретные ключи *SSK* и генерируя общий ключ *PK*. Принятие решения группой валидаторов PoA происходит в соответствии с заранее определенным пороговым значением в системе. Все взаимодействие узлов PoA осуществляется посредством обмена информацией по контракту с контроллером DVRF.

В качестве примера контракта с контроллером DVRF возьмем *VRFKeyCenter*, показанный на рис. 5, 6, он является представлением контракта координатора узлов PoA, расположенного во внешней сети.

Любые нарушения операций верификации (например, проверка порогового значения пользователей на рис. 5) приводят к прерыванию выполнения процессов, расположенных после условия, ниже по диаграмме.

Опишем основные функции распределенной системы генерации псевдослучайных чисел.

Регистрация свободных узлов PoA.

Для инициализации схемы необходимо не менее k участников с уникальным идентификатором, каждый из которых случайным образом формирует многочлен $f_i(x)$ над полем $GF(q)$ степени

$$deg\left(f_i(x)\right) = k-1,$$

где q — порядок группы точек эллиптической кривой. Узлы инициализируются в соответствии со схемой на рис. 5.

Распределенная генерация пары PK, SK.

1. Каждый из участников вычисляет значение своего многочлена $f_i(x)$ в $id_j = x,\ j = \{1, k\}$ и отправляет каждому участнику значение $f_i(id_j)$ по скрытому каналу связи.

2. Получив от всех k участников схемы набор значений k многочленов в точке $id_j = x$, каждый участник вычисляет значение своей проекции $sk_i = \sum_{j=1}^{k} f_j(id_i)$ секретного ключа *SK*. Результатом будет формирование $sk = \sum_{i=1}^{k} f_i(0)$, о котором ни у кого из участников протокола не будет никакой информации.

3. Для формирования *PK* каждый участник публикует свою собственную проекцию открытого ключа

$$PK_i = sk_i \cdot B,$$

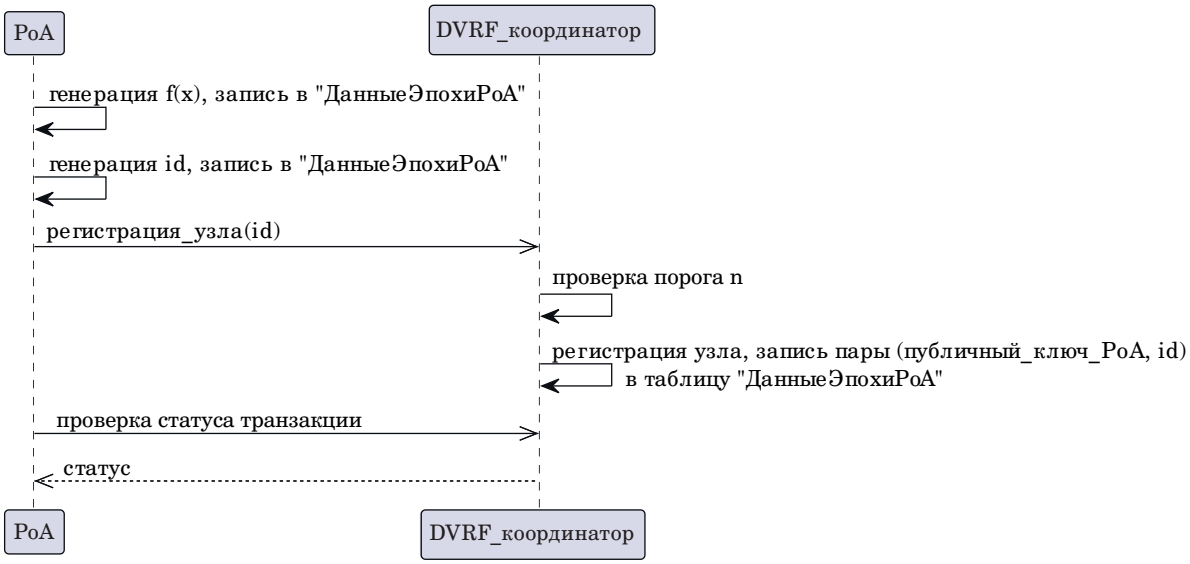
где B — генератор группы.

4. Открытый ключ собирается в соответствии с

$$PK = \sum_{i=1}^{k} \lambda_i \cdot PK_i = \sum_{i=1}^{k} \lambda_i \cdot sk_i \cdot B = sk \cdot B,$$

где λ_i — соответствующий множитель Лагранжа:

$$\lambda_i = \prod_{\substack{j=0 \\ j \neq i}}^{k} \frac{0 - id_j}{id_i - id_j}.$$



■ ***Рис. 5.*** Диаграмма регистрации PoA-узлов

■ ***Fig. 5.*** PoA node registration diagram

5. Чтобы сделать порог схемы k из n, после инициализации ключей первоначальные участники могут выдавать дополнительные проекции секрета другим участникам, вычисляя их многочлены $f_i(x)$ в новых точках id_j и отправляя по секретному каналу.

Все вышеперечисленные шаги представлены на рис. 6.

Генерация распределенного псевдослучайного числа.

Чтобы сгенерировать случайное число, каждый участник генерирует одноразовый номер и аналогично исходному алгоритму (1) параметр H:

$$nonce_i = rand(0...n);$$
$$sm_i \equiv nonce_i \bmod q.$$

Начальное значение формируется аналогично классической схеме VRF путем объединения и хеширования общедоступных значений пользователя протокола VRF:

$$Gamma_i = sk_i \cdot H;$$
$$U_i = sm_i \cdot B;$$
$$V_i = sm_i \cdot H.$$

Создание компонент U и V производится следующим образом:

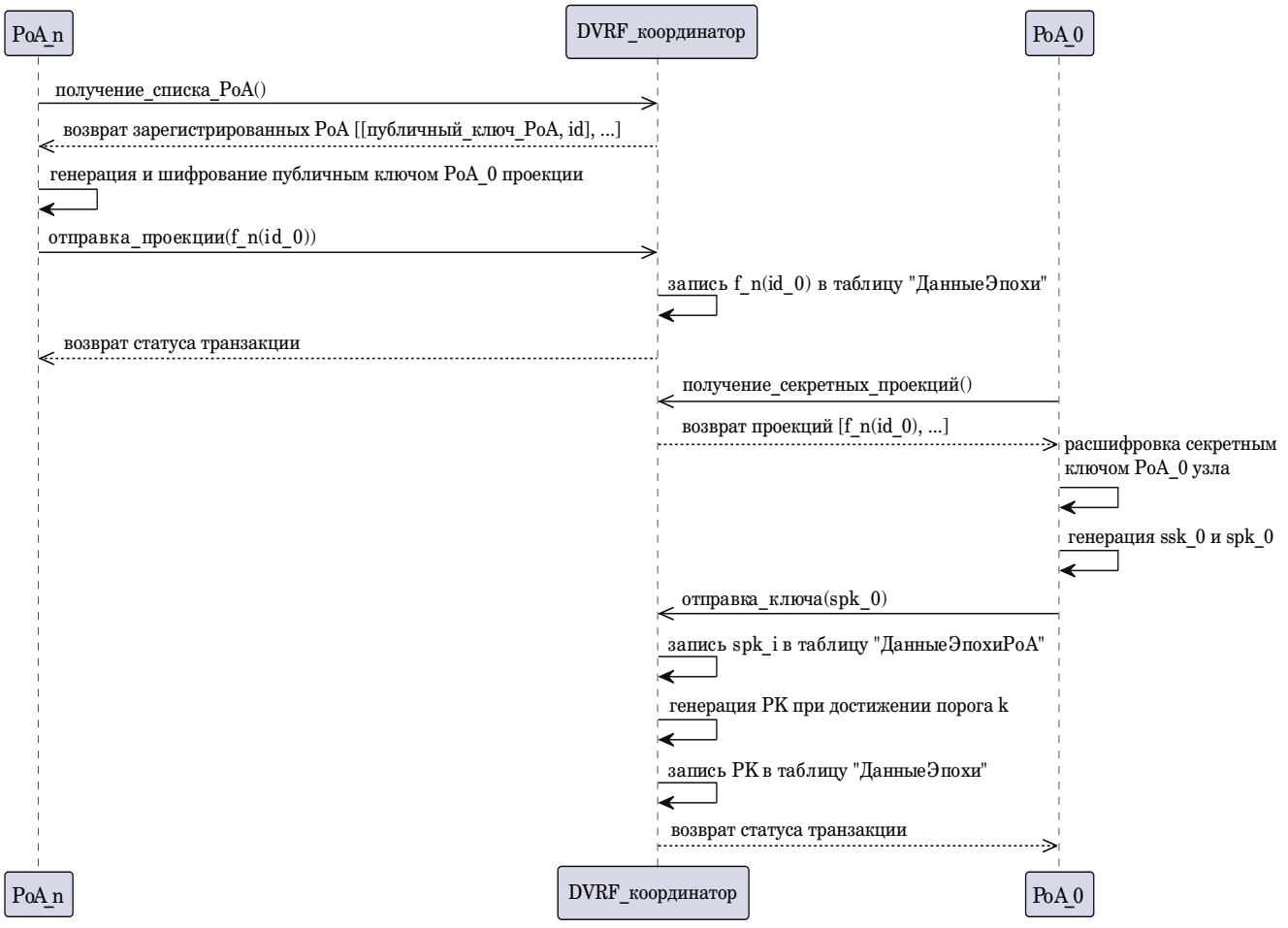
$$U = \sum_{i=1}^{k} \lambda_i \cdot U_i = \sum_{i=1}^{k} \lambda_i \cdot sm_i \cdot B = sm' \cdot B;$$
$$V = \sum_{i=1}^{k} \lambda_i \cdot V_i = \sum_{i=1}^{k} \lambda_i \cdot sm_i \cdot H = sm'H.$$

Далее выполняется формирование частичных подписей. Хеш-функция вычисляется аналогично исходной схеме по формуле (5). Проекция подписи формируется как

$$s_i \equiv \left(nonce_i + c \cdot sk_i\right) \bmod q. \quad (9)$$

Сборка подписи представлена формулой

$$s = \sum_{i=1}^{k} \lambda_i \cdot s_i = \sum_{i=1}^{k} \lambda_i \cdot sm_i + c \cdot \lambda_i \cdot sk_i =$$



■ ***Рис. 6.*** Диаграмма последовательности генерации распределенного ключа
■ ***Fig. 6.*** Diagram of the sequence of distributed key generation

$$= \sum_{i=1}^{k} \lambda_i \cdot sm_i + \sum_{i=1}^{k} c \cdot \lambda_i \cdot sk_i = sm' + c \cdot sk. \quad (10)$$

Проверка корректности приведенного псевдослучайного числа производится в соответствии с оригинальной схемой.

Заключение

По результатам работы можно сделать вывод, что на данный момент существующие решения генерации псевдослучайных чисел для блокчейн-систем имеют очевидные недостатки, связанные с вероятностью манипулирования техническими узлами самих систем. Предложенная и описанная выше система распределенных вычислений, основанная на алгоритме VRF, позволяет избежать уязвимостей классического протокола, а именно нивелировать возможность сговора с держателем секретного ключа централизованного VRF-сервиса. Хотя распределенный подход имеет собственные риски централизации при малых размерностях системы из-за сговора участников протокола, это определенно шаг вперед по сравнению с централизованным сервисом. С другой стороны, существует множество известных и зарекомендовавших себя схем, способных улучшить безопасность децентрализованных протоколов, вроде введения проверяемого разделения секрета Фельдмана или введение систем штрафов и поощрений [15–20]. Эти улучшения не рассматриваются в данной работе и должны быть обоснованы с учетом конкретных условий на практике.

Литература

1. **Boneh D., Waters B.** Constrained pseudorandom functions and their applications. *Lecture Notes in Computer Science*, 2013, vol. 7922, pp. 280–300. doi:10.1007/978-3-642-42045-0_15
2. **Micali S., Rabin M., Vadhan S.** Verifiable random functions. *Proc. 40th Int. Annual IEEE Symp. on Foundations of Computer Science*, New York, USA, 1999, pp. 120–130. doi:10.1109/SFFCS.1999.814584
3. *Verifiable Random Functions (VRFs).* https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-vrf (дата обращения: 05.03.2024).
4. **Dodis Y., Yampolskiy A.** A verifiable random function with short proofs and keys. *Proc. 8th Intern. Conf. on Theory and Practice of Public Key Cryptography*, Le Diableret, Switzerland, 2005, pp. 416–431. doi:10.1007/978-3-540-30580-4_28
5. *Pseudo-random Generators and Pseudo-random Functions: Cryptanalysis and Complexity Measures.* https://inria.hal.science/tel-01667124v1 (дата обращения: 23.05.2023).
6. *SEC 2: Recommended Elliptic Curve Domain Parameters.* https://www.secg.org/SEC2-Ver-1.0.pdf (дата обращения: 10.02.2023).
7. *Ethereum: A Secure Decentralised Generalised Transaction Ledger.* https://ethereum.github.io/yellowpaper/paper.pdf (дата обращения: 15.12.2023).
8. **David B., Gaži P., Kiayias A., Russell A.** Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake protocol. *Proc. 37th Annual Intern. Conf. on the Theory and Applications of Cryptographic Techniques*, Tel Aviv, Israel, 2018, pp. 66–98. doi:10.1007/978-3-319-78375-8_3
9. **Shamir A.** How to share a secret. *Communications of the ACM*, 1979, vol. 22, pp. 612–613. doi:10.1145/359168.359176
10. **Zhang Q., Zhihui L., Xiong L.** A verifiable secret sharing scheme without dealer in vector space. *Proc. 8th Intern. Conf. on Fuzzy Systems and Knowledge Discovery*, Shanghai, China, 2011. doi:10.1109/FSKD.2011.6019953
11. **Sun Y.** A completely fair secret sharing scheme without dealer. *Proc. 29th IEEE Intern. Conf. on Consumer Electronics-Taiwan*, Puli, Taiwan, 2016. doi:10.1109/ICCE-TW.2016.7520905
12. **Pedersen T.** A threshold cryptosystem without a trusted party. *Proc. 10th Intern. Conf. on the Theory and Application of Cryptographic Techniques*, Brighton, United Kingdom, 1991, pp. 522–526. doi:10.1007/3-540-46416-6_47
13. **Blundo C., De Santis A., Vaccaro U.** Randomness in distribution protocols. *Proc. 21th Intern. Colloquium on Automata, Languages and Programming*, Jerusalem, Israel, 1994. doi:10.1007/3-540-58201-0_99
14. **Popov S.** On a decentralized trustless pseudo-random number generation algorithm. *Journal of Mathematical Cryptology*, 2017, vol. 11, pp. 37–43. doi:10.1515/jmc-2016-0019
15. **Gennaro R., Rabin M. O., Rabin T.** Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. *Proc. 17th ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, Puerto Vallarta, Mexico, 1998, pp. 110–111. doi:10.1145/277697.277716
16. **Chor B., Goldwasser S., Micali S., Awerbuch B.** Verifiable secret sharing and achieving simultaneity in the presence of faults. *Proc. 26th Annual Symp. on Foundations of Computer Science*, Portland, USA, 1985, pp. 383–395. doi:10.1109/SFCS.1985.64
17. **Tomescu A., Chen R.** Towards scalable threshold cryptosystems. *Proc. 41th IEEE Symp. on Security and Privacy*, San Francisco, USA, 2020, pp. 877–893. doi:10.1109/SP40000.2020.00059
18. **Gueta G. G., Abraham I., Grossman S., Malkhi D.** SBFT: A scalable and decentralized trust infrastructure. *Proc. 49th Annual IEEE/IFIP Intern.Conf. on Dependable Systems and Networks*, Portland, USA, 2019, pp. 568–580. doi:10.1109/DSN.2019.00063

19. **Gilad Y., Hemo R., Micali C., Vlachos G., Zeldovich N.** Algorand: Scaling byzantine agreements for cryptocurrencies. *Proc. 26th Symp. on Operating Systems Principles*, Shanghai, China, 2017, pp. 51–68. doi:10.1145/3132747.3132757

20. **Papadopoulos D., Wessels D.** Can NSEC5 be practical for DNSSEC deployments? *IACR Cryptology ePrint Archive*, 2017, vol. 2017, pp. 99.

UDC 004.056.55
doi:10.31799/1684-8853-2024-3-32-40
EDN: FNYMEW

Distributed pseudorandom generation protocol based on verifiable random function algorithm

I. S. Velichko[a], Post-Graduate Student, orcid.org/0009-0005-6662-5606
A. V. Afanasieva[a], Senior Lecturer, orcid.org/0000-0003-3001-0990
S. V. Bezzateev[a], Dr. Sc., Tech., Professor, orcid.org/0000-0002-0924-6221, bsv@vu.spb.ru
[a]Saint-Petersburg State University of Aerospace Instrumentation, 67, B. Morskaia St., 190000, Saint-Petersburg, Russian Federation

Introduction: Verifiable Random Function stands out as one of the solutions for generating random numbers within the realm of smart contract security. Current centralized solutions, relying on this algorithm, do not offer transparency to system participants, thereby raising security concerns. **Purpose:** To develop a protocol for a system based on a verifiable pseudorandom function for a decentralized blockchain system with a high level of data protection against falsification. **Results:** To address the issue of falsifying initial input values, we propose an approach that replaces the classical centralized system based on a single oracle with a decentralized one. To ensure secure operation, we have developed a system for generating a shared distributed secret key and methods for generating pseudorandom values based on the obtained secret. This method is implemented using a dealer-free key exchange algorithm. We describe the operation of the protocol in the context of an abstract Ethereum-like blockchain model, initially utilizing Proof of Activity nodes to enhance accessibility and user-friendliness. **Practical relevance:** The developed method offers an effective solution to protect the system for generating pseudorandom numbers from falsification of input values generated by smart contracts. The introduction of a secret-sharing system without a dealer and the option for cyclic rounds of participant registration enhance the security, flexibility, and scalability of the system.

Keywords — Verifiable Random Function, blockchain, smart contracts, distributed systems, electronic signature, Schnorr scheme, secret sharing, pseudorandom number generation algorithms.

For citation: Velichko I. S., Afanasieva A. V., Bezzateev S. V. Distributed pseudorandom generation protocol based on verifiable random function algorithm. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2024, no. 3, pp. 32–40 (In Russian). doi:10.31799/1684-8853-2024-3-32-40, EDN: FNYMEW

References

1. Boneh D., Waters B. Constrained pseudorandom functions and their applications. *Lecture Notes in Computer Science*, 2013, vol. 7922, pp. 280–300. doi:10.1007/978-3-642-42045-0_15
2. Micali S., Rabin M., Vadhan S. Verifiable random functions. *Proc. 40th Intern. Annual IEEE Symp. on Foundations of Computer Science*, New York, USA, 1999, pp. 120–130. doi:10.1109/SFFCS.1999.814584
3. *Verifiable Random Functions (VRFs).* Available at: https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-vrf (accessed 05 March 2024).
4. Dodis Y., Yampolskiy A. A verifiable random function with short proofs and keys. *Proc. 8th Intern. Conf. on Theory and Practice of Public Key Cryptography*, Le Diableret, Switzerland, 2005, pp. 416–431. doi:10.1007/978-3-540-30580-4_28
5. *Pseudo-random Generators and Pseudo-random Functions: Cryptanalysis and Complexity Measures.* Available at: https://inria.hal.science/tel-01667124v1 (accessed 23 May 2023).
6. *SEC 2: Recommended Elliptic Curve Domain Parameters.* Available at: https://www.secg.org/SEC2-Ver-1.0.pdf (accessed 10 February 2023).
7. *Ethereum: A Secure Decentralised Generalised Transaction Ledger.* Available at: https://ethereum.github.io/yellowpaper/paper.pdf (accessed 15 December 2023).
8. David B., Gaži P., Kiayias A., Russell A. Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake protocol. *Proc. 37th Annual Intern. Conf. on the Theory and Applications of Cryptographic Techniques*, Tel Aviv, Israel, 2018, pp. 66–98. doi:10.1007/978-3-319-78375-8_3
9. Shamir A. How to share a secret. *Communications of the ACM*, 1979, vol. 22, pp. 612–613. doi:10.1145/359168.359176
10. Zhang Q., Zhihui L., Xiong L. A verifiable secret sharing scheme without dealer in vector space. *Proc. 8th Intern. Conf. on Fuzzy Systems and Knowledge Discovery*, Shanghai, China, 2011. doi:10.1109/FSKD.2011.6019953
11. Sun Y. A completely fair secret sharing scheme without dealer. *Proc. 29th IEEE Intern. Conf. on Consumer Electronics-Taiwan*, Puli, Taiwan, 2016. doi:10.1109/ICCE-TW.2016.7520905
12. Pedersen T. A threshold cryptosystem without a trusted party. *Proc. 10th Intern. Conf. on the Theory and Application of Cryptographic Techniques*, Brighton, United Kingdom, 1991, pp. 522–526. doi:10.1007/3-540-46416-6_47
13. Blundo C., De Santis A., Vaccaro U. Randomness in distribution protocols. *Proc. 21th Intern. Colloquium on Automata, Languages and Programming*, Jerusalem, Israel, 1994. doi:10.1007/3-540-58201-0_99
14. Popov S. On a decentralized trustless pseudo-random number generation algorithm. *Journal of Mathematical Cryptology*, 2017, vol. 11, pp. 37–43. doi:10.1515/jmc-2016-0019
15. Gennaro R., Rabin M. O., Rabin T. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. *Proc. 17th ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, Puerto Vallarta, Mexico, 1998, pp. 110–111. doi:10.1145/277697.277716
16. Chor B., Goldwasser S., Micali S., Awerbuch B. Verifiable secret sharing and achieving simultaneity in the presence of faults. *Proc. 26th Annual Symp. on Foundations of Computer Science*, Portland, USA, 1985, pp. 383–395. doi:10.1109/SFCS.1985.64
17. Tomescu A., Chen R. Towards scalable threshold cryptosystems. *Proc. 41th IEEE Symp. on Security and Privacy*, San Francisco, USA, 2020, pp. 877–893. doi:10.1109/SP40000.2020.00059
18. Gueta G. G., Abraham I., Grossman S., Malkhi D. SBFT: A scalable and decentralized trust infrastructure. *Proc. 49th Annual IEEE/IFIP Intern. Conf. on Dependable Systems and Networks*, Portland, USA, 2019, pp. 568–580. doi:10.1109/DSN.2019.00063
19. Gilad Y., Hemo R., Micali C., Vlachos G., Zeldovich N. Algorand: Scaling byzantine agreements for cryptocurrencies. *Proc. 26th Symp. on Operating Systems Principles*, Shanghai, China, 2017, pp. 51–68. doi:10.1145/3132747.3132757
20. Papadopoulos D., Wessels D. Can NSEC5 be practical for DNSSEC deployments? *IACR Cryptology ePrint Archive*, 2017, vol. 2017, pp. 99.