ПРОГРАММИРОВАНИЕ, 2024, № 4, с. 3–12

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

УДК 004.75

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ ХЭШИРОВАНИЯ С ТОЧКИ ЗРЕНИЯ ПРИМЕНЕНИЯ В СХЕМАХ ZK-SNARK В РАСПРЕДЕЛЕННЫХ РЕЕСТРАХ

© 2024 г. Д. О. Кондырев[a,*]

[a] *Новосибирский государственный университет*
630090 Новосибирск, ул. Пирогова, д. 2, Россия
**E-mail: dkondyrev@gmail.com*

Поступила в редакцию 26.05.2023
После доработки 20.11.2023
Принята к публикации 22.01.2024

В работе представлен сравнительный анализ эффективности алгоритмов хэширования с точки зрения применимости в системах на основе протокола неинтерактивного доказательства знания с нулевым разглашением zk-SNARK. Были рассмотрены хэш-функции sha256, sha3, poseidon, mime, blake2, которые находят наибольшее применение в современных распределенных реестрах. Для проведения экспериментов с замером параметров была разработана инфраструктура на основе набора инструментов ZoKrates. На основе полученных результатов определены границы практической применимости алгоритмов для задачи доказательства знания прообраза хэш-функции с помощью схем zk-SNARK в распределенных реестрах, а также выявлены возникающие проблемы эффективности.

Ключевые слова: распределенные реестры, доказательство с нулевым разглашением, zk-SNARK, R1CS, хеш-функции, эффективность алгоритма

DOI: 10.31857/S0132347424040012, **EDN:** PTNTQN

1. ВВЕДЕНИЕ

Распределенные реестры находят все большее применение в различных задачах. Помимо чисто исследовательских проектов начинают появляться первые промышленные системы, построенные на базе технологии распределенного реестра. При этом одним из факторов, существенно ограничивающих применение подобных технологий, является безопасность информации, которая должна обеспечиваться такими системами. В связи с этим активно ведутся исследования, направленные на создание различных протоколов и способов защиты информации для распределенных реестров [1].

Существенным прорывом в этом направлении можно считать появление в 2013 г. zk-SNARK [2], который сделал возможным эффективное использование неинтерактивных протоколов доказательства с нулевым разглашением в распределенных реестрах.

При том что теоретически zk-SNARK позволяет задавать произвольные ограничения [2], на практике возникают ограничения по времени работы и по памяти. При этом традиционные алгоритмы, оптимизированные для эффективного программного вычисления или реализации в виде аппаратного обеспечения, оказываются не настолько эффективными для использования в zk-SNARK. В связи с этим встает задача разработки специальных алгоритмов, оптимизированных под zk-SNARK.

Отдельным важным классом криптографических алгоритмов, которые находят широкое применение в распределенных реестрах и, в частности, в сокрытии информации с помощью доказательства с нулевым разглашением, являются хэш-функции. Задача доказательства знания прообраза хэш-функции возникла уже в первых блокчейн-системах на основе zk-SNARK [3]. После этого применение хэш-функций в системах ограничений, накладываемых на скрываемые данные, расширялось.

При том что рассматриваемые криптографические хэш-функции довольно давно используются в различных блокчейн-приложениях zk-SNARK, не было проведено полноценного исследования производительности для конкретных схем.

Целью данной работы было провести полный сравнительный анализ производительности наиболее часто применяемых хэш-функций и определить для них границы практической применимости в системах на базе технологии распределенного реестра. В работе рассматриваются хэш-функции sha256 [4], sha3 [5], blake2 [6], poseidon [7], mimc [8]. Для проведения экспериментов с замером параметров была разработана инфраструктура на основе ZoKrates.

2. ДОКАЗАТЕЛЬСТВО С НУЛЕВЫМ РАЗГЛАШЕНИЕМ

Доказательство с нулевым разглашением – криптографический протокол, в котором принимают участие две стороны – доказывающая и проверяющая (верификатор).

Цель протокола заключается в том, чтобы верификатор мог убедиться, что доказывающая сторона обладает знанием секретного параметра. При этом сам секретный параметр не должен раскрываться верификатору или кому-либо еще [9].

Доказательство с нулевым разглашением по определению должно удовлетворять следующим трем свойствам:

• **Полнота:** если утверждение верно и обе стороны следуют одному и тому же протоколу, то верификатор может убедиться в истинности утверждения.

• **Устойчивость:** если утверждение ложно, верификатор с большой вероятностью не будет убежден в его истинности.

• **Нулевое разглашение:** верификатор не получает дополнительной информации.

Доказательства с нулевым разглашением применяются для решения широкого круга задач. Примерами практического применения могут служить системы анонимного проверяемого голосования, безопасные аукционы, протоколы аутентификации [10].

В распределенных реестрах основное применение протоколы доказательства с нулевым разглашением нашли в решении проблемы приватности транзакций и смарт-контрактов.

Одним из важных примеров использования протоколов неинтерактивного доказательства знания с нулевым разглашением (NIZK) является подтверждение права собственности на цифровые активы, хранящиеся в виде токенов в блокчейне [11]. Цифровые активы представляют собой набор двоичных данных, которые являются уникально идентифицируемыми и обладают определенной ценностью. Если два пользователя хотят обменять свои цифровые активы, в процессе обмена может произойти утечка информации о конфиденциальности пользователя, которая включает в себя идентификационные данные и содержимое обмененных цифровых активов. Благодаря использованию протоколов доказательства с нулевым разглашением можно обмениваться цифровыми активами без утечки конфиденциальной информации пользователя. Кроме того, они позволяют сгенерировать проверяемое доказательство, которое подтверждает корректность процесса обмена активами [10].

Доказательства с нулевым разглашением применяются также для решения проблемы масштабируемости в механизмах ZK Rollups [12].

Более подробно применение протоколов доказательства с нулевым разглашением в распределенных реестрах рассматривается в статьях [10–14].

3. ТРЕБОВАНИЯ К ПРОТОКОЛАМ ДОКАЗАТЕЛЬСТВА С НУЛЕВЫМ РАЗГЛАШЕНИЕМ В РАСПРЕДЕЛЕННЫХ РЕЕСТРАХ

Технология распределенных реестров позволяет построить децентрализованную систему хранения информации. Данные в таких системах хранятся распределенно на узлах сети, история транзакций не может быть изменена или удалена. При этом в большинстве реализаций каждый узел системы поддерживает в актуальном состоянии полную копию всего реестра. Такой подход позволяет повысить надежность и отказоустойчивость, но ценой дополнительных накладных расходов по памяти.

Особенности технологии распределенных реестров накладывают ряд ограничений на используемые протоколы доказательства с нулевым разглашением. Для протоколов, используемых в распределенных реестрах, следующие требования являются критическими:

• **Ограничение на количество раундов взаимодействия участников.** Это свойство вытекает из децентрализованного характера системы. Протоколы должны учитывать тот факт, что пользователи могут не быть в сети одновременно.

• **Ограничения по памяти.** Вся история транзакций может сохраняться на каждом узле. Это накладывает серьезные ограничения на размер транзакций (для большинства систем он не должен превышать нескольких сотен байт). Поэтому любые дополнительные данные, которые необходимо хранить в распределенном реестре, включая

данные криптографических протоколов, должны соответствовать этому требованию.

• **Вычислительная эффективность.** Проверка корректности транзакций в сети происходит на каждом узле. Поэтому вычислительная эффективность отдельных операций, выполняемых в ходе проверки, крайне важна. Если протокол предполагает вычисления, выполняемые во время валидации транзакций, то они должны быть максимально оптимизированы по времени выполнения. Иначе создание новых блоков будет происходить медленно. В итоге пропускная способность системы (количество транзакций в секунду) упадет, что может сделать применение протокола практически нецелесообразным [11, 15].

Стоит отметить, что перечисленные ограничения являются довольно серьезными и существенно ограничивают множество протоколов, которые могут быть использованы в системах на базе технологии распределенного реестра.

В последние годы активно ведутся исследования по разработке новых криптографических протоколов для сохранения приватности и анонимности в таких системах, а также адаптации и модификации существующих протоколов.

4. ПРОТОКОЛ ZK-SNARK

zk-SNARK (zero-knowledge Succinct Non-Interactive Argument of Knowledge) – это криптографический протокол неинтерактивного доказательства знания с нулевым разглашением. Он позволяет доказывать, что некоторые приватные данные удовлетворяют системе ограничений, выраженной в виде арифметической схемы C, не раскрывая эти данные.

zk-SNARK представляет собой тройку алгоритмов полиномиального времени выполнения (Gen, P, V):

• $Gen(\lambda, C) \to (pk, vk)$. Этот алгоритм принимает в качестве входных данных параметр безопасности λ и арифметическую схему C. На их основе генератор Gen создает пару ключей – ключ доказательства (pk, proving key) и ключ верификации (vk, verification key). Оба ключа публикуются как открытые параметры и могут использоваться любое количество раз для создания доказательства и проверки его корректности.

• $P(pk, x, a) \to \pi$. Принимая на вход ключ доказательства pk и любые (x, a), где x – публичные данные, a – секретный параметр, алгоритм P выводит неинтерактивное доказательство π.

• $V(vk, x, \pi) \to b$. Принимая на вход ключ верификации vk, публичные данные x и доказательство π, верификатор V выдает $b = 1$, если доказательство является корректным, и 0 иначе.

Данная конструкция удовлетворяет всем требованиям, предъявляемым к алгоритмам доказательства с нулевым разглашением [3].

Преимущество zk-SNARK над другими протоколами доказательства с нулевым разглашением заключается в гарантиях эффективности: длина доказательства зависит только от параметра безопасности, а время проверки не зависит от размера схемы и секретного параметра.

Таким образом, zk-SNARK можно рассматривать как неинтерактивный протокол с коротким доказательством и быстрым временем верификации, что сделало его оптимально подходящим для использования в распределенных реестрах [15].

Однако, при том что zk-SNARK подходит для применения в распределенных реестрах, из-за ограничений на память и вычислительную эффективность, про которые рассказывалось в предыдущем разделе, на практике может быть реализована не любая по сложности схема C. Поэтому не любой криптографический алгоритм может быть использован для описания ограничений, накладываемых на скрываемые с помощью zk-SNARK данные.

5. СИСТЕМЫ ОГРАНИЧЕНИЙ РАНГА 1 (R1CS)

zk-SNARK позволяет доказывать произвольные NP-утверждения. Следовательно, чтобы быть доказуемым, вычисление C должно быть сформулировано как пример NP-полной задачи, например, как задача о выполнимости арифметической схемы или удовлетворении ограничений.

Арифметическая схема C над полем F – это направленный ациклический граф, где каждой вершине соответствует операция сложения или умножения над полем F, а каждому ребру – значение из F.

Возьмем в качестве примера следующую арифметическую схему с входными значениями $i_0, ..., i_3$, выходами o_0, o_1 и промежуточными выходами w_0, w_1, w_2 (рис. 1).

Алгебраическое представление такой арифметической схемы задается следующими ограничениями:

$$w_0 = i_0 + i_1, \qquad o_0 = i_0 * w_0,$$
$$w_1 = w_0 * i_2, \qquad o_1 = w_1 + w_2.$$
$$w_2 = i_2 * i_3,$$

Системы ограничений ранга 1 (rank-1 constraint systems, R1CS) можно рассматривать как краткое алгебраическое представление арифметических

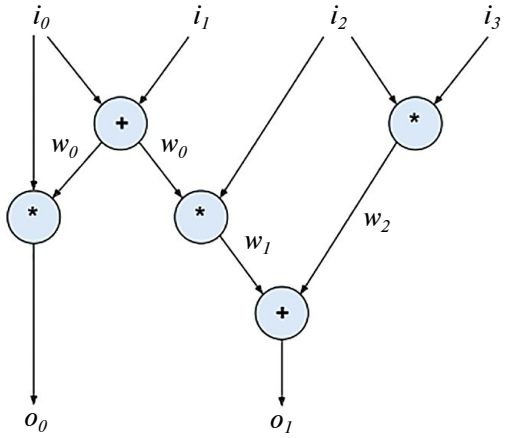

Рис. 1. Арифметическая схема.

схем. R1CS позволяют представить схему C в виде, удобном для дальнейшего эффективного доказательства выполнимости этой схемы с помощью zk-SNARK.

Чтобы привести арифметическую схему к такому представлению, ограничения умножения можно представить в виде

$$w_1 = w_0 * i_2 \Leftrightarrow$$

$$\Leftrightarrow (1 * w_1) = (1 * w_0) \times (1 * i_2) \Leftrightarrow$$

$$\Leftrightarrow \left\langle \begin{pmatrix} 0\\0\\0\\0\\0\\1\\0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 1\\i_0\\i_1\\i_2\\i_3\\w_0\\w_1\\w_2\\o_0\\o_1 \end{pmatrix} \right\rangle * \left\langle \begin{pmatrix} 0\\0\\0\\1\\0\\0\\0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 1\\i_0\\i_1\\i_2\\i_3\\w_0\\w_1\\w_2\\o_0\\o_1 \end{pmatrix} \right\rangle = \left\langle \begin{pmatrix} 0\\0\\0\\0\\0\\1\\0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 1\\i_0\\i_1\\i_2\\i_3\\w_0\\w_1\\w_2\\o_0\\o_1 \end{pmatrix} \right\rangle$$

Если перейти к выражению всей схемы через матричное произведение, то получим систему ограничений ранга 1:

$$\begin{pmatrix} 0&1&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&1&0&0 \end{pmatrix} \begin{pmatrix} 1\\i_0\\i_1\\i_2\\i_3\\w_0\\w_1\\w_2\\o_0\\o_1 \end{pmatrix}$$

$$\odot \begin{pmatrix} 1&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&0 \end{pmatrix} \begin{pmatrix} 1\\i_0\\i_1\\i_2\\i_3\\w_0\\w_1\\w_2\\o_0\\o_1 \end{pmatrix}$$

$$= \begin{pmatrix} 0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&1 \end{pmatrix} \begin{pmatrix} 1\\i_0\\i_1\\i_2\\i_3\\w_0\\w_1\\w_2\\o_0\\o_1 \end{pmatrix}$$

Далее, имея систему ограничений в R1CS-представлении, можно сформулировать задачу выполнимости в следующем виде:

Можно ли дополнить вектор, состоящий из входных и выходных значений (i, o)*, вектором* w *таким образом, что*

$$\mathbf{A}(1, i, o, w)^{\mathrm{T}} \odot \mathbf{B}(1, i, o, w)^{\mathrm{T}} = \mathbf{C}(1, i, o, w)^{\mathrm{T}},$$

где $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ *– заданные матрицы.*

Благодаря краткости и лаконичности R1CS превратилась в де-факто стандартный формат ввода для реализаций zk-SNARK. Поскольку в R1CS используется стандартная линейная алгебра, они также хорошо подходят для описания криптографических протоколов и теоретического анализа [16].

Эффективность алгоритмов установочной фазы *Gen* и генерации доказательства *P* по времени работы и по памяти напрямую зависит от количества ограничений в R1CS-представлении. Таким образом, оптимизация, которая уменьшает количество ограничений в R1CS без изменения ее выполнимости, имеет решающее значение для эффективности.

6. ПРОБЛЕМА ЭФФЕКТИВНОСТИ

Сложность доказательства выполнения арифметической схемы с помощью zk-SNARK составляет $O(n \log n)$, где n – число операций умноже-

ния. Таким образом, эффективность напрямую определяется n.

Схема C более эффективна, чем C', тогда и только тогда, когда обе схемы вычисляют одну и ту же функцию и $n_C < n_{C'}$ [16].

В контексте реальных приложений zk-SNARK асимптотические результаты менее актуальны, и конкретная эффективность имеет решающее значение для обеспечения практической применимости. Особенно это актуально в распределенных реестрах с их дополнительными ограничениями по времени работы и по памяти.

Поэтому для конкретных алгоритмов важно понимать, насколько они эффективны с точки зрения количества ограничений.

7. ХЭШ-ФУНКЦИИ В РАСПРЕДЕЛЕННЫХ РЕЕСТРАХ

В ходе работы было проведено сравнение эффективности различных криптографических хэш-функций с точки зрения реализации в R1CS.

Хэш-функция H – это функция, которая принимает на вход данные произвольного размера и сопоставляет их с выходными данными фиксированного размера. Криптографические хэш-функции должны обладать дополнительными свойствами:

- стойкость к коллизиям – трудно найти такие a и b, что $H(a) = H(b)$;
- стойкость к поиску первого прообраза – для заданного y трудно найти такое входное значение a, что $H(a) = y$;
- стойкость к поиску второго прообраза – для заданных входных данных a и выходного значени $y = H(a)$ трудно найти вторые входные данные b такие, что $H(b) = y$ [13].

В распределенных реестрах хэш-функции применяются при решении многих задач, таких как:

- формирование адресов аккаунтов;
- защита данных в блоках транзакций;
- подпись транзакций;
- обеспечение работы алгоритмов консенсуса (например, нахождение подходящего значения в алгоритме на основе доказательства выполнения работы (Proof-of-Work)) [13, 17].

Отдельной категорией задач, в которых применяются хэш-функции, стало формирование доказательств с нулевым разглашением. Как было упомянуто выше, важным применением таких доказательств в распределенных реестрах является задача подтверждения права собственности на цифровые активы.

Впервые хэш-функции для решения подобной задачи были применены в системе Zerocash. Zerocash реализует схему децентрализованных анонимных платежей, которая позволяет пользователям осуществлять переводы, сохраняя приватность. Транзакции переводов скрывают отправителя платежа, адресата и сумму перевода. Для доказательства корректности проведения транзакций используются схемы zk-SNARK, которые строятся с использованием хэш-функции sha256 [3].

В более общем виде задачу можно сформулировать как доказательство владения определенными данными без необходимости их раскрытия какой-либо стороне. Один из вариантов решения – применить хэш-функцию к данным и с помощью схемы zk-SNARK сгенерировать доказательство знания прообраза. Это позволяет верификатору убедиться, что доказывающая сторона обладает определенными данными таким образом, чтобы не размещать сами эти данные в открытом виде в транзакциях распределенного реестра.

При таком подходе возрастает важность оптимизации хэш-функций для применения в схемах zk-SNARK, поскольку именно вычисление хэш-функции, как самая затратная операция, определяет сложность всей схемы.

В первых распределенных реестрах наибольшее применение нашли функции sha256. В более поздних системах стали применяться алгоритмы sha3 и blake2 [13, 17].

Однако стало понятно, что классические алгоритмы не максимально эффективно решают возникающие задачи. В связи с этим стали разрабатываться специально оптимизированные для применения в zk-SNARK алгоритмы хэширования, такие как mimc [8] и poseidon [7].

8. КРИПТОГРАФИЧЕСКАЯ СТОЙКОСТЬ И ЭФФЕКТИВНОСТЬ ХЭШ-ФУНКЦИЙ

В табл. 1 приведен сравнительный анализ рассматриваемых хэш-функций по уровню криптографической стойкости (security level) и производительности для решения стандартных задач. В качестве задач для сравнения взяты вычисление хэш-функции от данных размером не менее 512 бит и построение дерева Меркла с 2^{20} элементами.

Указанные значения скорости работы хэш-функций, а также более подробные данные о производительности приведены в статье [18].

В работах [7, 8] показано, что с помощью конструкций poseidon и mimc можно достичь

Таблица 1. Сравнительный анализ криптографической стойкости и производительности хэш-функций

	Уровень криптографической стойкости (бит)	Вычисление хэш-функции от данных размером 512 бит (мс)	Построение дерева Меркла с 2^{20} элементами (с)
sha256	128	0.32	0.624
sha3	128	0.42	0.439
blake2	128	0.21	0.222
poseidon	128	20	22.6
mimc	128	38	42.2

необходимого уровня криптографической стойкости. В рамках работы рассматривались варианты poseidon и mimc, которые обеспечивают стойкость 128 бит, что соответствует уровню алгоритмов sha256, sha3 и blake2.

Poseidon и mimc представляют из себя узкоспециализированные хэш-функции и, как видно из таблицы, для решения стандартных задач их результаты оказываются существенно слабее. Так происходит, поскольку каждый раунд этих схем требует выполнения операций умножения в конечном поле, что является относительно дорогой операцией, требующей сотни циклов CPU, по сравнению с битовыми операциями, используемыми в традиционных хэш-функциях [18]. По этой же причине эти алгоритмы оказываются эффективнее в схемах zk-SNARK.

9. ПРИМЕНИМОСТЬ АЛГОРИТМОВ ДЛЯ ПРОТОКОЛА ZK-SNARK

Теоретический анализ большей или меньшей применимости в zk-SNARK классических хэш-функций не проводился, поскольку при компиляции в R1CS-представление получаются системы ограничений большого размера. Поэтому сравнения производительности проводят на конкретных реализациях отдельных схем, замеряя количество ограничений R1CS [7, 18].

Одной из основных причин, почему традиционные алгоритмы плохо оптимизированы для zk-SNARK является тот факт, что доказательства создаются для утверждений, сформулированных в виде операций над конечными полями, тогда как классические алгоритмы работают над битовыми строками. Необходимый перевод кода, оперирующего битами, в арифметику конечных полей требует значительных накладных расходов [18].

Специально разработанные для zk-SNARK алгоритмы имеют конструкцию, в которую заложены операции, оптимальные с точки зрения внутреннего представления R1CS. Так, mimc и poseidon в качестве нелинейной функции в раундах используют x^{α} [7, 8]. Для рассматриваемой в работе реализации poseidon S-блоки имеют вид $S-box(x)=x^5$, что выражается с помощью трех ограничений R1CS.

10. ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

Для проведения вычислительных экспериментов использовался ZoKrates – набор программных инструментов для создания доказательств знания с нулевым разглашением, использующий zk-SNARK в качестве системы проверки.

ZoKrates скрывает значительную сложность, присущую доказательствам с нулевым разглашением, и предоставляет разработчикам более высокоуровневые программные абстракции для реализации системы ограничений. Для этого он определяет предметно-ориентированный язык, который позволяет разработчикам задавать доказуемые вычисления без необходимости разбираться в низкоуровневых деталях реализации системы доказательств.

Компилятор ZoKrates транслирует код, написанный на этом языке, в доказуемые системы ограничений. Кроме того, ZoKrates позволяет выполнять реализованные программы, генерировать доказательства и выполнять проверку корректности доказательств [19].

В качестве задачи для замера эффективности работы различных алгоритмов использовалась задача проверки знания прообраза хэш-функций – чтобы исключить дополнительные накладные расходы и сравнивать непосредственно эффективность реализации хэш-функций.

Реализации сравниваемых хэш-функций взяты из стандартной библиотеки ZoKrates [20]. В качестве протокола zk-SNARK используется система доказательства Groth16 [21] с эллиптической кривой ALT_BN128.

Программа принимает значение входной строки в виде секретного параметра, далее в коде вычисляется хэш-функция от этих секретных данных, после чего происходит сравнение с заранее вычисленным правильным значением. Пример кода для функции sha3 приведен в листинге 1.

Листинг 1: Код функции проверки знания прообраза хэша на предметно-ориентированном языке ZoKrates

```
import "hashes/sha3/256bit" as sha3

def main(private u64[32] a):
  u64[4] h = sha3(a)
  assert(h[0] == 18011198171195110515)
  assert(h[1] == 11930925129043369621)
  assert(h[2] == 7871165761403032144)
  assert(h[3] == 16761886150217802019)
  return
```

zk-SNARK гарантирует, что размер доказательства и время его проверки не зависят от сложности системы ограничений. Однако, другие параметры могут отличаться и именно по ним можно определить эффективность того или иного алгоритма и возможность его применения для определенной задачи. В проведенных экспериментах измерялись следующие параметры сравниваемых алгоритмов:

• длина ключа доказательства и ключа верификации;

• время установочной фазы протокола (алгоритма генерации ключей *Gen*);

• количество ограничений в R1CS-представлении алгоритма;

• время генерации доказательства.

Для повышения точности каждый эксперимент с замером времени проводился $N = 10$ раз, после чего полученные значения времени усреднялись.

Инфраструктура для измерения была реализована на Python 3.10. Она включает в себя:

• автоматический генератор кода функций проверки прообраза для различных входных значений на основе шаблонов;

• автоматизацию запуска различных команд ZoKrates (компиляция кода, генерация параметра безопасности, генерация доказательства);

• функции многократного выполнения экспериментов и замера времени.

Вычислительные эксперименты проводились на компьютере с процессором Intel Core i5-11600K, 32 Гб оперативной памяти и SSD 500 Гб.

11. РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТА

Ниже приведены графики, на которых отображены полученные результаты для различных хэш-функций в зависимости от размера входных данных (рис. 2–5).

Исходя из полученных экспериментальных данных можно сделать следующие выводы:

• Количество ограничений во всех алгоритмах растет линейно в зависимости от входа. При этом наибольшую скорость роста демонстрируют sha3, sha256 и blake2.

• Как следствие количества ограничений в схеме с ростом размера входных данных увеличивается время выполнения установочной фазы и генерации доказательства. Алгоритмы sha3 и blake2 показывают линейный рост, при этом довольно быстрый. При значениях в 2 Кб генерация доказательства превышает 60 и 20 с соответственно. Алгоритм sha256 оказывается еще менее эффективным, поскольку показывает нелинейный рост времени генерации доказательства и на входных данных в 2 Кб превышает 10 мин.

Время установочной фазы начинает насчитывать несколько минут у всех трех алгоритмов уже при значениях больше 1 Кб. Однако это на так существенно влияет на применимость, поскольку эта часть протокола отрабатывает только один раз для каждой новой схемы.

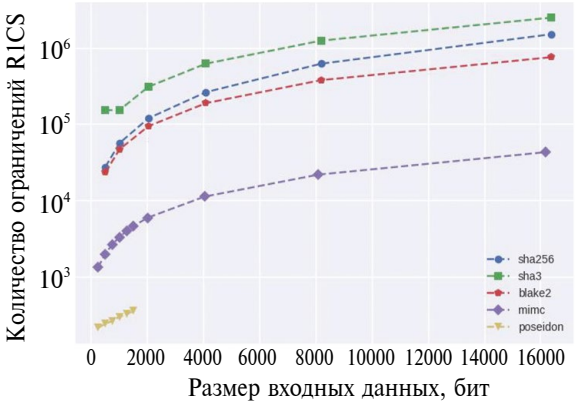

Рис. 2. Количество ограничений в зависимости от размера входных данных.

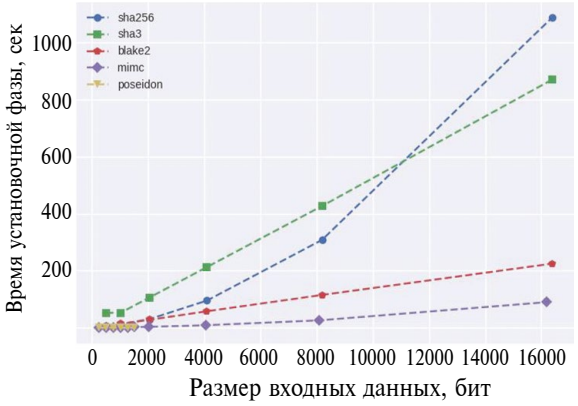

Рис. 3. Время работы алгоритма установочной фазы в зависимости от размера входных данных.

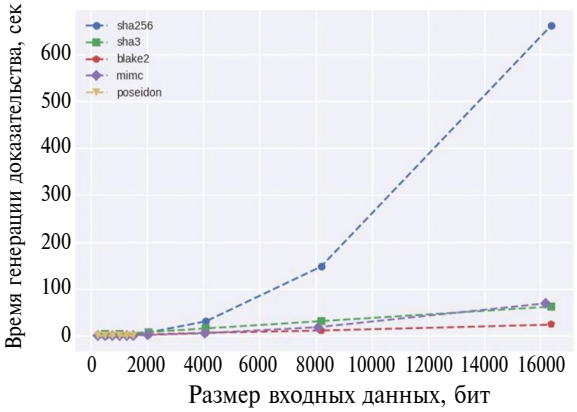

Рис. 4. Время работы алгоритма генерации доказательства в зависимости от размера входных данных.

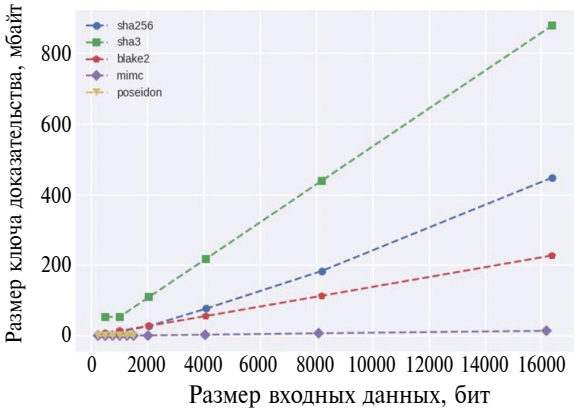

Рис. 5. Размер ключа доказательства в зависимости от размера входных данных.

Таким образом, классические алгоритмы плохо подходят для распределенных реестров. Уже при входных данных размером в несколько килобайт размер и время становятся слишком большими для практических задач, которые предполагают частую генерацию доказательств (сопоставимую со временем появления новых блоков транзакций в системе). Отсюда можно сделать вывод, что в таких системах они могут только ограниченно применяться для доказательства знания прообраза хэш-функции от небольшого объема данных.

Хэш-функции mimc и poseidon лучше справляются с этой задачей, также показывая линейный рост времени установочной фазы и генерации доказательства, но значительно более медленный.

• С увеличением размера входных данных и сложности схемы растет и размер ключа доказательства.

Даже для алгоритма mimc при входных данных в 1.5 Кб размер ключа доказательства превышает 10 Мб. У sha256 и blake2 он растет еще быстрее и при таких же входных данных занимает уже сотни Мб. sha3 оказывается наименее эффективным по памяти – при входных данных более 2.5 Кб получаем размер ключа больше 1 Гб.

Поскольку на размер блока в наиболее распространенных распределенных реестрах накладываются ограничения и он как правило не превышает нескольких мегабайт, это не позволяет эффективно хранить ключи доказательства непосредственно в распределенном реестре.

• Наилучшие результаты по всем параметрам показал алгоритм poseidon, однако его удалось проверить только на небольших размерах входных данных из-за ограничений реализации.

12. ЗАКЛЮЧЕНИЕ

В результате работы был проведен сравнительный анализ эффективности алгоритмов хэширования с точки зрения применимости в протоколах неинтерактивного доказательства знания с нулевым разглашением zk-SNARK в распределенных реестрах. Проведенные эксперименты подтвердили актуальность проблемы оптимизации криптографических алгоритмов для использования в схемах zk-SNARK. Лучшие результаты показали специально разработанные для использования в zk-SNARK алгоритмы poseidon и mimc. Классические алгоритмы sha256, sha3 и blake2 оказываются ограниченно применимыми для доказательства знания прообраза хэш-функции только от небольшого объема данных. При этом определенные проблемы характерны для всех исследованных алгоритмов, поскольку с ростом размера входных данных растет размер ключа доказательства, что при достаточно небольших объемах входных данных уже не позволяет эффективно хранить ключ в распределенном реестре.

БЛАГОДАРНОСТИ

Работа выполнена при поддержке Математического Центра в Академгородке, соглашение с Министерством науки и высшего образования Российской Федерации № 075-15-2022-282.

СПИСОК ЛИТЕРАТУРЫ

1. *Kondyrev D.O.* Overview of privacy preserving technologies for distributed ledgers // Eurasian Journal of Mathematical and Computer Applications. 2021. V. 9. № 1. P. 55–68.
2. *Ben-Sasson E., Chiesa A., Genkin D., Tromer E., Virza M.* SNARKs for C: Verifying program executions succinctly and in zero knowledge // CRYPTO'2013, LNCS. 2013. V. 8043. P. 90–108.

3. *Ben-Sasson E., Chiesa A., Garman C., Green M., Miers I., Tromer E., Virza M.* Zerocash: Decentralized anonymous payments from bitcoin // IEEE Symp. Security and Privacy, San Jose, USA. 2014. P. 459–474.
4. NIST. FIPS PUB180–2, Secure hash standard. 2002.
5. NIST. FIPS PUB202, SHA-3 standard: permutation-based hash and extendable-output functions. 2015.
6. *Aumasson J.P., Neves S., Wilcox-O'Hearn Z., Winnerlein C.* BLAKE2: simpler, smaller, fast as MD5 // ACNS2013, Proceedings of the 11th International Conference Applied Cryptography and Network Security, Banff, AB, Canada. 2013. V. 7954 of LNCS, P. 119–135.
7. *Grassi L., Khovratovich D., Rechberger C., Roy A., Schofnegger M.* Poseidon: A New Hash Function for Zero-Knowledge Proof Systems // Proceedings of the 30th USENIX Security Symposium. 2021. V. 2021.
8. *Albrecht M., Grassi L., Rechberger C., Roy A., Tiessen T.* MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity // ASIACRYPT 2016. Proceedings of the 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam. 2016. Part I.V. 10031 of LNCS. P. 191–219.
9. *Шнайер Б.* Прикладная криптография: протоколы, алгоритмы и исходные коды на языке С. 2-е изд. СПб.: Альфа-книга, 2018. 1040 с.
10. *Sun X., Yu F.R., Zhang P., Sun Z., Xie W., Peng X.* A survey on zero-knowledge proof in blockchain // IEEE network. 2021. V. 35. № 4. P. 198–205.
11. *Konkin A., Zapechnikov S.* Zero knowledge proof and ZK-SNARK for private blockchains // Journal of Computer Virology and Hacking Techniques. 2023. P. 1–7.
12. *Thibault L.T., Sarry T., Hafid A.S.* Blockchain scaling using rollups: A comprehensive survey // IEEE Access. 2022.
13. *Raikwar M., Gligoroski D., Kralevska K.* SoK of Used Cryptography in Blockchain // IEEE Access. 2019. V. 7. P. 148550–148575.
14. *Wang L., Shen X., Li J., Shao J., Yang Y.* Cryptographic primitives in blockchains // Journal of Network and Computer Applications. 2019. V. 127. P. 43–58.
15. *Virza M.* On Deploying Succinct Zero-Knowledge Proofs // PhD Thesis. Massachusetts Institute of Technology. 2017. 131 p.
16. *Eberhardt J.* Scalable and Privacy-preserving Off-chain Computations // PhD Thesis. Technical University of Berlin. 2021. 284 p.
17. *Yaga D., Mell P., Roby N., Scarfone K.* Blockchain technology overview // NIST Interagency/Internal Report (NISTIR) – 8202. 2018.
18. *Grassi L., Khovratovich D., Luftenegger R., Rechberger C., Schofnegger M., Walch R.* Reinforced concrete: A fast hash function for verifiable computation // Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. 2022. P. 1323–1335.
19. *Eberhardt J., Tai S.* ZoKrates – scalable privacy-preserving off-chain computations // IEEE Intern. Conf. Blockchain. Halifax, Canada. 2018. P. 1084–1091.
20. https://github.com/Zokrates/ZoKrates – ZoKrates.
21. *Groth J.* On the size of pairing-based non-interactive arguments // EUROCRYPT 2016. Proceedings of the 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria. 2016. Part II 35. P. 305–326.

COMPARATIVE EFFICIENCY ANALYSIS OF HASHING ALGORITHMS FOR APPLICATIONS IN ZK-SNARK CIRCUITS IN DISTRIBUTED LEDGERS

© 2024 D. O. Kondyrev[a]

[a] *Novosibirsk State University*
Pirogova st. 2, Novosibirsk, 630090 Russia

The paper presents a comparative efficiency analysis of hashing algorithms in terms of applicability in zk-SNARK based systems. We have considered the hash functions sha256, sha3, blake2, mimc and poseidon, which are most widely used in modern distributed ledgers. To conduct experiments with measuring parameters, an infrastructure based on the ZoKrates toolbox was developed. A series of measurements with different input data was carried out for each algorithm. The number of constraints in the R1CS representation of the algorithm, the length of the proof key and the verification key, the running time of the setup phase of the protocol, and the proof generation time were measured. Based on the obtained results, we determined the boundaries of the practical applicability of algorithms for the problem of proving the knowledge of the preimage of a hash function using zk-SNARK circuits in distributed ledgers, and also identified emerging efficiency problems.

Keywords: distributed ledgers, zero-knowledge proof, zk-SNARK, R1CS, hash functions, algorithm efficiency

REFERENCES

1. *Kondyrev D.O.* Overview of privacy preserving technologies for distributed ledgers // Eurasian Journal of Mathematical and Computer Applications. 2021. V. 9. № 1. P. 55–68.
2. *Ben-Sasson E., Chiesa A., Genkin D., Tromer E., Virza M.* SNARKs for C: Verifying program executions succinctly and in zero knowledge // CRYPTO'2013, LNCS. 2013. V. 8043. P. 90–108.
3. *Ben-Sasson E., Chiesa A., Garman C., Green M., Miers I., Tromer E., Virza M.* Zerocash: Decentralized anonymous payments from bitcoin // IEEE Symp. Security and Privacy, San Jose, USA. 2014. P. 459–474.
4. NIST. FIPS PUB180–2, Secure hash standard. 2002.
5. NIST. FIPS PUB202, SHA-3 standard: permutation-based hash and extendable-output functions. 2015.
6. *Aumasson J.P., Neves S., Wilcox-O'Hearn Z., Winnerlein C.* BLAKE2: simpler, smaller, fast as MD5 // ACNS2013, Proceedings of the 11th International Conference Applied Cryptography and Network Security, Banff, AB, Canada. 2013. V. 7954 of LNCS, P. 119–135.
7. *Grassi L., Khovratovich D., Rechberger C., Roy A., Schofnegger M.* Poseidon: A New Hash Function for Zero-Knowledge Proof Systems // Proceedings of the 30th USENIX Security Symposium. 2021. V. 2021.
8. *Albrecht M., Grassi L., Rechberger C., Roy A., Tiessen T.* MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity // ASIACRYPT 2016. Proceedings of the 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam. 2016. Part I. V. 10031 of LNCS. P. 191–219.
9. *Schneier B.* Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley & Sons Inc., 1996, 2nd ed.
10. *Sun X., Yu F.R., Zhang P., Sun Z., Xie W., Peng X.* A survey on zero-knowledge proof in blockchain // IEEE network. 2021. V. 35. № 4. P. 198–205.
11. *Konkin A., Zapechnikov S.* Zero knowledge proof and ZK-SNARK for private blockchains // Journal of Computer Virology and Hacking Techniques. 2023. P. 1–7.
12. *Thibault L.T., Sarry T., Hafid A.S.* Blockchain scaling using rollups: A comprehensive survey // IEEE Access. 2022.
13. *Raikwar M., Gligoroski D., Kralevska K.* SoK of Used Cryptography in Blockchain // IEEE Access. 2019. V. 7. P. 148550–148575.
14. *Wang L., Shen X., Li J., Shao J., Yang Y.* Cryptographic primitives in blockchains // Journal of Network and Computer Applications. 2019. V. 127. P. 43–58.
15. *Virza M.* On Deploying Succinct Zero-Knowledge Proofs // PhD Thesis. Massachusetts Institute of Technology. 2017. 131 p.
16. *Eberhardt J.* Scalable and Privacy-preserving Off-chain Computations // PhD Thesis. Technical University of Berlin. 2021. 284 p.
17. *Yaga D., Mell P., Roby N., Scarfone K.* Blockchain technology overview // NIST Interagency/Internal Report (NISTIR) – 8202. 2018.
18. *Grassi L., Khovratovich D., Luftenegger R., Rechberger C., Schofnegger M., Walch R.* Reinforced concrete: A fast hash function for verifiable computation // Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. 2022. P. 1323–1335.
19. *Eberhardt J., Tai S.* ZoKrates – scalable privacy-preserving off-chain computations // IEEE Intern. Conf. Blockchain. Halifax, Canada. 2018. P. 1084–1091.
20. https://github.com/Zokrates/ZoKrates – ZoKrates.
21. *Groth J.* On the size of pairing-based non-interactive arguments // EUROCRYPT 2016. Proceedings of the 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria. 2016. Part II 35. P. 305–326.