Научная статья
УДК 004.056
DOI: 10.26583/bit.2025.4.05

ПРИМЕНЕНИЕ АЛГОРИТМА ГОСТ 34.11-2018 В ПРОТОКОЛЕ FRI

Владлен Д. Афонин[1], Сергей В. Запечников[2]
Национальный исследовательский ядерный университет «МИФИ», Каширское ш., 31, Москва, 115409, Россия
[1]*vladlenafonin.university@yandex.ru, https://orcid.org/0009-0003-0255-6571*
[2]*svzapechnikov@mephi.ru, https://orcid.org/0000-0002-7975-6040*

Аннотация. В статье рассматривается проблема применимости отечественных криптографических алгоритмов в современных системах доказательств с нулевым разглашением (zero-knowledge proofs, ZKP), которые находят широкое применение в блокчейнах, цифровой идентификации, медицинских и биометрических системах, а также в задачах машинного обучения и защиты конфиденциальных данных. Особое внимание уделено использованию алгоритма хэширования ГОСТ 34.11-2018 «Стрибог» в качестве случайного оракула в протоколах доказательств, где этот компонент играет ключевую роль для обеспечения корректности и безопасности вычислений. В работе проводится краткий обзор современных парадигм построения систем доказательства выполнения произвольных вычислений. В качестве объекта анализа выбран постквантовый протокол FRI, который активно применяется в агрегируемых ZKP-системах и опирается на использование хэш-функций для построения деревьев Меркля и генерации случайных элементов поля. Авторами реализована экспериментальная версия протокола FRI на языке Python с возможностью замены криптографических примитивов и проведения замеров производительности. В рамках эксперимента стандартные хэш-функции семейства Keccak были заменены на «Стрибог», что позволило провести сравнение времени работы протокола. Результаты показали, что использование ГОСТ 34.11-2018 приводит к замедлению генерации доказательства примерно в два раза, однако такое снижение производительности не является критичным для приложений, где ключевым фактором выступает соответствие национальным стандартам и регуляторным требованиям. Сделан вывод о принципиальной возможности применения алгоритма «Стрибог» в ZKP-протоколах и обозначены направления дальнейшей оптимизации его использования, включая аппаратные ускорения и адаптацию к современным моделям построения хэш-губок.

Ключевые слова: *доказательства с нулевым разглашением, криптография, алгоритмы, интерактивные доказательства с оракулом, информационная безопасность, хэш-функции.*

Для цитирования: *Афонин, Владлен Д.; Запечников, Сергей В. Применение алгоритма ГОСТ 34.11-2018 в протоколе FRI. Безопасность информационных технологий, [S.l.], т. 32, № 4, с. 65–74, 2025. ISSN 2074-7136. URL: https://bit.spels.ru/index.php/bit/article/view/1866. DOI: http://dx.doi.org/10.26583/bit.2025.4.05.*

Scientific article

USING GOST 34.11-2018 IN THE FRI PROTOCOL

Vladlen D. Afonin[1], Sergey V. Zapechnikov[2]
National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Kashirskoe sh., 31, Moscow, 115409, Russia
[1]*vladlenafonin.university@yandex.ru, https://orcid.org/0009-0003-0255-6571*
[2]*svzapechnikov@mephi.ru, https://orcid.org/0000-0002-7975-6040*

Abstract. The paper addresses the applicability of domestic cryptographic algorithms in modern zero-knowledge proof (ZKP) systems, which are increasingly used in blockchains, digital identity platforms, medical and biometric systems, as well as in machine learning and privacy-preserving applications. Particular attention is given to the use of the Russian hashing algorithm GOST 34.11-2018 “Streebog” as a random oracle within proof systems, where this component plays a crucial role in ensuring the correctness and security of computations. The paper provides a brief overview of contemporary paradigms for building proof systems for proving arbitrary computations. As the primary object of study, the post-quantum protocol FRI (Fast Reed-Solomon IOP of Proximity) was chosen. This protocol is widely employed in aggregatable ZKP systems and relies on hash functions for constructing Merkle trees and generating random field elements. An experimental implementation of the FRI protocol was developed in Python, allowing for the replacement of cryptographic primitives and the measurement of performance metrics. In the conducted experiments, standard Keccak-based hash functions were replaced with “Streebog,” enabling a comparative analysis of runtime performance. The results demonstrated that employing GOST 34.11-2018 leads to roughly a twofold slowdown in proof generation. However, this decrease in performance is not critical for practical applications where compliance with national cryptographic standards and regulatory requirements is a priority. The study concludes that the “Streebog” algorithm can be feasibly applied in ZKP protocols, and outlines potential directions for further optimization, including hardware acceleration and adaptation to modern sponge-based constructions.

Keywords: *zero-knowledge proofs, cryptography, algorithms, interactive oracle proofs, information security, hash functions.*

For citation: *Afonin, Vladlen D.; Zapechnikov, Sergey V. Using GOST 34.11-2018 in the FRI protocol. IT Security (Russia), [S.l.], v. 32, no. 4, p. 65–74, 2025. ISSN 2074-7136. URL: https://bit.spels.ru/index.php/bit/article/view/1866. DOI: http://dx.doi.org/10.26583/bit.2025.4.05.*

Введение

В последние десятилетия доказательства с нулевым разглашением (zero-knowledge proofs, ZKP) [1] стали одним из ключевых и устоявшихся инструментов в криптографии, находя применение как в теоретических исследованиях и теории алгоритмов, так и в практически реализованных протоколах, которые используются в блокчейнах, машинном обучении, биометрических системах, медицине и многих других областях, где конфиденциальность информации имеет существенное значение. В настоящее время доступно множество инструментов, программных библиотек, платформ, фреймворков и даже языков программирования, позволяющих внедрять доказательства с нулевым разглашением в программные продукты.

Несмотря на растущую популярность идей, лежащих в основе доказательств с нулевым разглашением, в современной русскоязычной литературе вопрос применимости отечественных алгоритмов к системам доказательств с нулевым разглашением освещён недостаточно полно [2]. Настоящая работа посвящена исследованию применимости алгоритма хэширования ГОСТ 34.11-2018 «Стрибог»[1] в качестве случайного оракула в преобразовании BCS [3], которое является стандартом де-факто в современных реализациях систем доказательства выполнения произвольных вычислений, таких как Plonky3[2]. Тем не менее, все полученные результаты также относятся и к сценарию использования алгоритма «Стрибог» в схемах коммитмента многочленов, основанных на хэш-функциях. В первую очередь проанализирована производительность алгоритма.

[1]ГОСТ 34.11-2018. Информационная технология. Криптографическая защита информации. Функция хэширования. URL: https://docs.cntd.ru/document/1200161707 (дата обращения: 18.09.2025).
[2]Plonky3. A toolkit for polynomial IOPs (PIOPs). URL: https://github.com/Plonky3/Plonky3 (дата обращения: 18.09.2025).

Вопрос применимости алгоритма ГОСТ 34.11-2018 в качестве случайного оракула – предмет дискуссии в смежной области [4].

Следует отметить, что алгоритм «Стрибог» был разработан в первую очередь как национальный стандарт хэширования, обеспечивающий высокий уровень криптографической стойкости и совместимость с российскими требованиями по информационной безопасности. Однако его использование в качестве строительного блока в системах доказательства с нулевым разглашением открывает новые направления исследований. Так, ключевыми параметрами становятся не только криптографическая стойкость, но и эффективность вычислений при больших объёмах данных, а также возможность интеграции с существующими библиотеками и протоколами, где широко применяются зарубежные аналоги (SHA-2, SHA-3, Blake2 и др.) и специально сконструированные для использования в системах доказательства с нулевым разглашением хэш-функции, такие как Poseidon, Rescue и др.

Кроме того, важно адаптировать «Стрибог» к современным аппаратным платформам. В условиях роста интереса к аппаратным ускорителям, графическим процессорам и расширения возможностей построения вычислительных архитектур на базе программируемых интегральных микросхем (FPGA, ASIC) становится критически значимым вопрос оптимизации реализации алгоритма. Производительность напрямую влияет на практическую применимость систем доказательства с нулевым разглашением, так как даже незначительное увеличение времени вычислений может сделать протоколы экономически невыгодными для массового использования.

Особый интерес представляет сравнение «Стрибога» с новыми хэш-функциями, ориентированными на применение в системах доказательства с нулевым разглашением, например, Poseidon или Rescue, которые проектировались с учётом минимизации вычислительных затрат в арифметических схемах. Это позволяет взглянуть на «Стрибог» через призму требований к оптимизации протоколов с нулевым разглашением и найти баланс между нормативной базой и практической эффективностью.

Таким образом, исследование роли алгоритма ГОСТ 34.11-2018 в системах доказательства с нулевым разглашением не ограничивается вопросами его криптографической стойкости. Оно включает анализ производительности, интеграции в существующую инфраструктуру, сравнение с зарубежными аналогами и поиск сценариев, где использование отечественного стандарта может дать преимущества. Дальнейшие работы в этом направлении способны существенно расширить спектр применения доказательств с нулевым разглашением в российских и международных проектах.

1. Подходы к построению систем доказательства

В настоящее время существует множество подходов к построению систем доказательства, однако наибольшее распространение в результате последних открытий в этой области получили два подхода:

- конструкции, основанные на комбинации полиномиального интерактивного доказательства с оракулом (Polynomial Interactive Oracle Proofs, PIOP) [3] и схемы коммитмента многочленов (Polynomial Commitment Schemes, PCS) [5];
- конструкции, основанные на BCS-преобразовании [3].

В первом случае доказательство формируется в интерактивной модели PIOP, где оно, как правило, сводится к проверке свойств многочленов (например, протокол GKR) [6]. Протоколы PIOP являются теоретико-информационными объектами и не могут быть реализованы в условиях ограниченных компьютерных ресурсов в силу вычислительной сложности алгоритмов формирования доказательства. Протоколы PIOP принципиально

интерактивны, что делает их неприменимыми в сценариях, когда доказывающий и проверяющий не могут быть онлайн одновременно. Для преодоления этих ограничений PIOP комбинируется со схемой коммитмента многочленов PCS, которая позволяет создавать специальные строки, связанные с многочленами, позволяющие проверять значения этих многочленов в произвольных точках. PCS «встраивается» в PIOP на определённых этапах протокола и делает его практически применимым и программно реализуемым, а преобразование Фиата-Шамира [7] позволяет сделать итоговый протокол неинтерактивным. Процесс превращения PIOP в конкретную эффективную практическую неинтерактивную систему доказательств в литературе часто называют *компиляцией*.

Второй подход связан с преобразованием BCS, которое позволяет преобразовать произвольное интерактивное доказательство с оракулом (Interactive Oracle Proof, IOP) [3] в неинтерактивный аргумент (Non-interactive Argument, NARG) [8]. Так, преобразование BCS лежит в основе популярной системы доказательств STARK [9]. Именно преобразование BCS будет подробнее рассмотрено далее.

Важным фактором выбора между двумя описанными подходами является компромисс между скоростью работы и универсальностью. Конструкции на основе PIOP+PCS, как правило, отличаются более компактными размерами сгенерированных доказательств и низкими затратами на проверку доказательства, что делает их подходящими для блокчейн-систем, где важна минимизация нагрузки на сеть и вычислительные ресурсы валидаторов. В то же время они часто требуют наличия доверенной третьей стороны для генерации параметров, что ограничивает их применимость в сценариях, где невозможно гарантировать честность доверенного участника. С другой стороны, протоколы, основанные на преобразовании BCS, позволяют конструировать доказательства без доверенной третьей стороны на этапе генерации параметров (это свойство часто называют *прозрачностью*), что делает их особенно актуальными в условиях распределённых вычислительных систем.

Отдельного внимания заслуживает вопрос масштабируемости. Для протоколов на основе PIOP+PCS ключевым ограничением становится сложность операций с многочленами высокой степени, что напрямую влияет на продолжительность генерации доказательства. В то же время для BCS-подхода узким местом является размер доказательств, который, несмотря на многочисленные оптимизации, остаётся существенно большим по сравнению с системами на основе эллиптических кривых. Однако благодаря применению протокола FRI [10] и множества его модификаций, удалось добиться линейного или квазилинейного времени генерации доказательств и значительно улучшить показатели масштабируемости, что открывает путь к использованию таких систем в практических приложениях с миллионами транзакций.

Немаловажную роль играет и вопрос постквантовой стойкости. PCS на основе дискретного логарифма или спариваний нестойких против атак с применением квантовых компьютеров. В связи с этим системы, использующие хэш-функции как фундаментальный примитив (такие как FRI), оказываются наиболее перспективными, так как стойкость хэш-функций в квантовой модели существенно выше. Это делает подход на основе BCS и FRI наиболее подходящим в долгосрочной перспективе, где необходимо учитывать развитие квантовых вычислительных технологий.

Кроме того, в последние годы активно развиваются гибридные схемы, комбинирующие сильные стороны обоих подходов. Например, протоколы, использующие преобразование Фиата-Шамира вместе с PIOP+PCS и элементы FRI для проверки близости многочленов, позволяют достичь баланса между компактностью доказательств и

прозрачностью генерации параметров. Такие гибридные конструкции становятся основой новых поколений систем доказательств.

Таким образом, развитие систем доказательства с нулевым разглашением сегодня происходит в сторону унификации подходов, где ключевым критерием остаётся баланс между скоростью, размером доказательства, простотой реализации и устойчивостью к атакам с использованием квантовых вычислителей. Выбор конкретного подхода определяется требованиями практической системы, но в долгосрочной перспективе именно протоколы на основе FRI и преобразования BCS представляют собой наиболее устойчивый и перспективный фундамент для построения постквантовых агрегируемых систем доказательства с нулевым разглашением.

2. Протокол FRI

Вне зависимости от выбранного способа построения системы доказательства задача во многом сводится к доказательству свойств многочленов. Для этого в современной литературе и практических реализациях используются либо схемы коммитмента многочленов, такие как KZG [5], либо доказательства близости многочленов к линейным кодам (например, к коду Рида-Соломона), такие как FRI (Fast Reed-Solomon Interactive Oracle Proof of Proximity) [10]. Важно отметить, что криптографические свойства системы доказательства зависят только от используемой в ней схемы коммитмента многочлена (в подходе PIOP+PCS) или случайного оракула (при использовании преобразования BCS). Таким образом, для достижения постквантовой стойкости системы доказательства необходимо использовать квантово-стойкую схему коммитмента многочленов. Это достигается использованием схем коммитмента многочленов, основанных на симметричных примитивах, таких как хэш-функции. Протокол FRI и ряд протоколов, основанных на нём (STIR [11], WHIR [12], BOIL [13], ARC [14]), обладают этим преимуществом, а также поддерживают агрегацию. Поэтому далее будем рассматривать именно протокол FRI как основу постквантовых агрегируемых систем доказательства с нулевым разглашением.

Протокол FRI позволяет проверять близость функций к многочленам низкой степени. Более формально, в протоколе FRI задача доказывающего – доказать, что значения функции f на множестве $\mathcal{L}$ близки к кодовому слову кода Рида-Соломона [15] $\mathrm{RS}[\mathbb{F}, \mathcal{L}, m] = \{f: \mathcal{L} \to \mathbb{F}: \exists g \in \mathbb{F}^{<2^m}[X]: \forall x\, f(x) = g(x)\}$, где $\mathcal{L} \subset \mathbb{F}^*$ – циклическая подгруппа мультипликативной группы конечного поля $\mathbb{F}$ порядка 2^n для некоторого n. Значение n определяет максимальную степень многочлена, для которого возможно выполнить протокол, накладывая ограничения на параметр m. Отношение $\rho^{-1} = \frac{2^n}{2^m} > 1$ – параметр протокола, называемый «фактором расширения». Чем выше фактор расширения, тем меньшее количество раз необходимо повторить протокол для достижения заданного уровня стойкости λ. Асимптотика количества раундов в протоколе FRI даётся формулой

$$q_{\mathrm{FRI}} = O\left(\lambda \frac{\log d}{\log \rho^{-1}}\right), \tag{1}$$

где $d < 2^m$ – степень многочлена, для которого проводится доказательство.

Протокол выполняется между двумя участниками: доказывающим и проверяющим. Доказывающий получает на вход функцию f (например, в аналитическом виде). Оба участника получают параметры протокола: поле $\mathbb{F}$, подгруппу $\mathcal{L}$, фактор расширения ρ и параметр стойкости λ. Протокол выполняется в две фазы следующим образом:

1. Фаза первоначального взаимодействия:

a. Доказывающий посылает проверяющему оракула к многочлену, для которого производится доказательство. В модели IOP доказывающий способен посылать проверяющему оракулов к произвольным функциям. Под оракулом понимается быстро вычислимая математическая функция, которую проверяющий может вычислить в произвольной точке с минимальными затратами вычислительных ресурсов. Важной метрикой протокола FRI является количество запросов проверяющего к оракулу, асимптотика которого также задаётся формулой (1).
b. Пока степень многочлена не станет нулевой, повторять (каждый раунд нумеруется значением счётчика i, начиная с 0):
   i. Проверяющий посылает доказывающему очередное случайное число $\alpha_i \in \mathbb{F}$.
   ii. Доказывающий посылает проверяющему оракула к новой функции $f_{i+i} = \frac{f_i(x)+f_i(-x)}{2} + \alpha_i \cdot \frac{f_i(x)-f_i(-x)}{2x}$, таким образом сводя задачу к проверке близости f_{i+1} к $\mathrm{RS}\left[\mathbb{F}, \mathcal{L}^{2^{i+1}}, \frac{m}{2^{i+1}}\right]$.

2. Фаза запросов:
   a. Проверяющий случайным образом выбирает элемент $\mu \in \mathcal{L}$.
   b. Для каждого раунда i (количество раундов то же, что в шаге 1.b): проверяющий проводит проверку корректности, запрашивая у доказывающего значения функций в соответствующих точках: $f_{i+1}\left(\mu^{2^{i+1}}\right) =^{?} \frac{f_i\left(\mu^{2^i}\right)+f_i\left(-\mu^{2^i}\right)}{2} + \alpha_i \cdot \frac{f_i\left(\mu^{2^i}\right)+f_i\left(-\mu^{2^i}\right)}{2\mu^{2^i}}$. Последняя проверка проводится с использованием оракула к последней присланной доказывающим функции.
   c. Фаза повторяется q_{FRI} раз для других случайных $\mu \in \mathcal{L}$, что необходимо для достижения требуемого уровня стойкости.

3. Программная реализация

В ходе работы была реализована экспериментальная версия протокола FRI на языке Python. Программная реализация поддерживает возможность выбора хэш-функции, используемой в протоколе, и содержит в себе инструменты для проведения замеров производительности. Исходный код программы открыт и доступен на GitHub со свободной лицензией MIT[3].

Основное отличие программной реализации от математического описания протокола выше заключается в том, что в ней уже применено преобразование BCS: реализованный протокол является неинтерактивным и использует хэш-функции вместо случайных оракулов. В архитектуре программного решения это обстоятельство отражено следующими классами:

- `vc.fri.parameters.FriParameters`: модель публичных параметров протокола FRI, основные из которых:
  - используемое поле,
  - требуемый уровень стойкости протокола,
  - степень многочлена, для которого производится доказательство,
  - фактор расширения,
  - степень финального многочлена;
- `vc.fri.proof.FriProof`: модель доказательства, состоящая из

[3]Verifiable Computations. URL: https://github.com/VladlenAfonin/vc (дата обращения: 18.09.2025).

  - раундовых доказательств,
  - раскрытий дерева Меркля, представляющих собой векторные коммитменты к многочленам, полученным в ходе первой фазы протокола,
  - финального многочлена,
  - многочлена для корректировки степени – это деталь реализации, которая позволяет генерировать доказательства для многочленов, степень которых не является степенью числа 2;
- `vc.fri.prover.FriProver`: модель доказывающего, содержащая функцию `prove(f: Polynomial)`, которая генерирует и возвращает доказательство для многочлена `f`,
- `vc.fri.verifier.FriVerifier`: модель проверяющего, содержащая функцию `verify(p: FriProof)`, которая проверяет доказательство и возвращает `True` или `False` в зависимости от результата проверки.

Конструкторы классов `FriProver` и `FriVerifier` принимают на вход параметры протокола, определённые в классе `FriParameters`. Примеры использования разработанного программного продукта можно найти в тестах.

Дополнительно к библиотеке был разработан консольный интерфейс, который позволяет проводить замеры производительности, указывая параметры протокола в качестве аргументов командной строки.

4. Результаты

В протоколе FRI есть два места, где используются случайные оракулы: деревья Меркля, использующиеся для создания векторных коммитментов, и так называемые «губки» – случайные оракулы, которые используются в качестве генераторов случайных элементов поля, одинаковых для доказывающего и проверяющего. Традиционно в качестве случайных оракулов в протоколе используются хэш-функции семейства Keccak[4]. Мы заменили их на ГОСТ 34.11-2018 и сравнили производительность. Результаты сравнения производительности приведены на рис. 1. Для эксперимента использовался компьютер Apple Mac Studio 2023 с процессором Apple Silicon M1 Max и оперативной памятью 32 ГБ.

Из заметного увеличения времени работы протокола следует, что вычисление значений хэш-функции занимает существенную долю времени работы протокола. Замена зарубежной хэш-функции на отечественную замедлила выработку доказательства более чем в два раза. Тем не менее, это обстоятельство не играет особой роли на масштабах типовых времён работы протоколов, подобных FRI. Например, для многочлена степени $d = 2^{24}$ при использовании другого оборудования генерация доказательства составила 28 с, в то время как проверка заняла 3.9 мс [11].

Полученные результаты напрямую зависят от особенностей реализации. Язык Python, используемый для прототипирования, не является оптимальным с точки зрения производительности вычислений. Перенос реализации на более низкоуровневые языки (например, C или Rust) или использование аппаратных оптимизаций (SIMD-инструкции, GPU, FPGA) может существенно уменьшить разрыв в производительности между Keccak и «Стрибогом». Таким образом, сделанный вывод о двукратном замедлении скорее отражает свойства текущей реализации, чем принципиальные ограничения самого алгоритма.

[4]FIPS PUB 202 SHA-3. Standard: Permutation-Based Hash and Extendable-Output Functions. URL: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.202.pdf (дата обращения: 18.09.2025).

Более того, использование «Стрибога» может иметь преимущества в специфических сценариях. Например, в условиях, когда требуется соответствие национальным стандартам в сфере защиты информации, снижение скорости вычислений может быть оправдано ради обеспечения нормативного соответствия. В реальных протоколах доказательства с нулевым разглашением длительность генерации доказательств обычно измеряется секундами или даже минутами, поэтому увеличение продолжительности генерации на несколько десятков процентов редко является критическим фактором, особенно если это обеспечивает юридическую и регуляторную корректность.

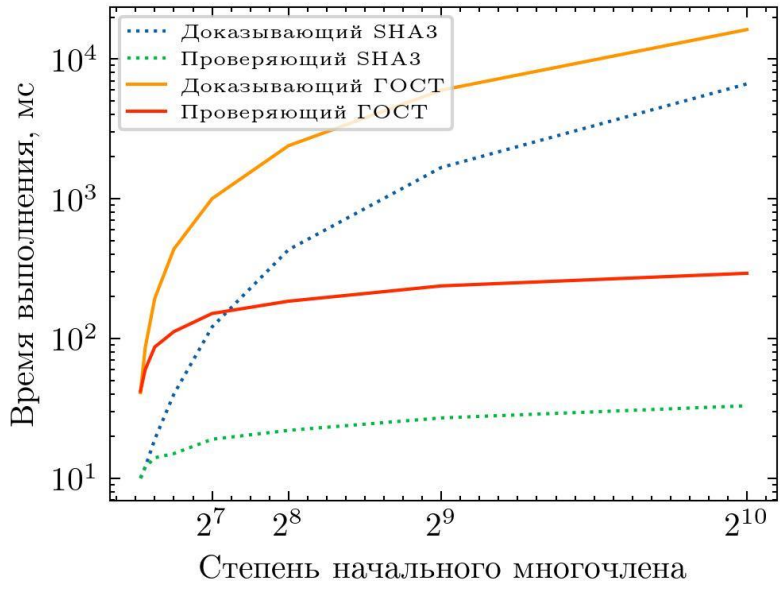

Рис. 1. Сравнение производительности двух реализаций протокола FRI: с использованием хэш-функций семейства Keccak и с использованием отечественной хэш-функции «Стрибог»

Ещё одним направлением для исследования является адаптация «Стрибога» под модель губки (sponge construction), которая активно используется в современных системах доказательств. Если удастся эффективно преобразовать ГОСТ 34.11-2018 для подобных сценариев, это может значительно повысить его применимость в современных протоколах и уменьшить отставание по производительности.

Таким образом, проведённый эксперимент показывает, что замена криптографических примитивов на отечественные возможна без фундаментального ущерба для корректности и устойчивости протокола. Несмотря на снижение скорости вычислений, протокол FRI с использованием «Стрибога» остаётся практически применимым, а дальнейшие исследования могут привести к появлению оптимизированных реализаций, которые нивелируют текущие ограничения.

Заключение

В работе проведён анализ применимости отечественного алгоритма ГОСТ 34.11-2018 «Стрибог» в качестве случайного оракула в системах доказательства с нулевым разглашением на примере широко распространенного квантово-стойкого протокола FRI доказательства ограниченности степени многочлена. Получена экспериментальная зависимость времени работы компонентов протокола от степени начального многочлена, для которого проводится доказательство.

Исследование показало, что ГОСТ 34.11-2018 может быть использован в качестве случайного оракула в системах доказательства с нулевым разглашением без критического снижения производительности, хотя в приложениях, где скорость играет особо важную роль, таких как системы реального времени, рекомендуется использование уже устоявшихся хэш-функций семейства Keccak.

Следует подчеркнуть, что даже при замедлении вычислений в два раза по сравнению с зарубежными аналогами, использование «Стрибога» не выводит протокол FRI за пределы практической применимости. Это особенно актуально в сценариях, где ключевым фактором является не скорость, а соответствие национальным стандартам и нормативным требованиям в области информационной безопасности. В таких случаях использование ГОСТ 34.11-2018 может быть оправданным и даже предпочтительным решением.

Перспективным направлением для дальнейших исследований является оптимизация реализации алгоритма «Стрибог» для применения в системах доказательства с нулевым разглашением. Перенос вычислений на низкоуровневые языки программирования, использование аппаратных ускорителей (GPU, FPGA, ASIC) и исследование возможностей адаптации структуры губки для более эффективного использования в протоколах, подобных FRI, могут существенно сократить наблюдаемый разрыв в производительности.

СПИСОК ЛИТЕРАТУРЫ:

1. Goldwasser S., Micali S., and Rackoff C. (1989). The knowledge complexity of interactive proof systems. In: SIAM Journal on Computing, 18(1):186–208, 1989. DOI: http://dx.doi.org/10.1137/0218012.
2. Афонин, Владлен Д.; Запечников, Сергей В.; Простов, Игорь А. Анализ возможностей применения алгоритма гост 34.11-2018 в системах доказательства с нулевым разглашением. Безопасность информационных технологий, [S.l.], т. 31, № 2, с. 81–89, 2024. ISSN 2074-7136. URL: https://bit.spels.ru/index.php/bit/article/view/1643. DOI: http://dx.doi.org/10.26583/bit.2024.2.05.
3. Ben-Sasson E., Chiesa A., and Spooner N. (2016). Interactive oracle proofs. In Proceedings, part ii, of the 14th international conference on theory of cryptography. V. 9986, p. 31–60, Berlin, Heidelberg, 2016. Springer-Verlag. DOI: http://dx.doi.org/10.1007/978-3-662-53644-5_2.
4. Akhmetzyanova L., Babueva A., and Bozhko A. (2023). Streebog as a random oracle. Cryptology ePrint Archive, Paper 2023/1075. 2023. URL: https://eprint.iacr.org/2023/1075 (дата обращения: 18.09.2025).
5. Kate A., Zaverucha G.M., and Goldberg I. (2010). Constant-size commitments to polynomials and their applications. In M. Abe, editors, Advances in cryptology – asiacrypt 2010, p. 177–194, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN: 978-3-642-17373-8. DOI: http://dx.doi.org/10.1007/978-3-642-17373-8_11.
6. Goldwasser S., Kalai Y.T., and Rothblum G.N. (2008). Delegating computation: interactive proofs for muggles. J. ACM, 62(4), September 2015. DOI: http://dx.doi.org/10.1145/2699436.
7. Fiat A. and Shamir A. (1986). How to prove yourself: practical solutions to identification and signature problems. In A. M. Odlyzko, editors, Advances in cryptology – crypto' 86, p. 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg. ISBN: 978-3-540-47721-1. DOI: http://dx.doi.org/10.1007/3-540-47721-7_12.
8. Chiesa A. and Yogev E. (2024). Building cryptographic proofs from hash functions. URL: https://github.com/hash-based-snargs-book (дата обращения: 18.09.2025).
9. Ben-Sasson E., Bentov I., Horesh Y., and Riabzev M. (2018). Scalable, transparent, and post-quantum secure computational integrity. In: Cryptology ePrint Archive, Paper 2018/046. 2018. URL: https://eprint.iacr.org/2018/046 (дата обращения: 18.09.2025).
10. Ben-Sasson E., Bentov I., Horesh Y., and Riabzev M. (2018). Fast Reed-Solomon Interactive Oracle Proofs of Proximity. n I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella, editors, 45Th international colloquium on automata, languages, and programming (icalp 2018), v. 107 of Leibniz International Proceedings in Informatics (LIPIcs), 14:1–14:17, Dagstuhl, Germany, 2018, Schloss Dagstuhl – Leibniz-Zentrum für Informatik. DOI: http://dx.doi.org/10.4230/LIPIcs.ICALP.2018.14.
11. Arnon G., Chiesa A., Fenzi G., and Yogev E. (2024). STIR: Reed--Solomon Proximity Testing with Fewer Queries. In: Cryptology {ePrint} Archive, Paper 2024/390. 2024. URL: https://eprint.iacr.org/2024/390 (дата обращения: 18.09.2025).

12. Arnon G., Chiesa A., Fenzi G., and Yogev E. (2024). WHIR: Reed--Solomon Proximity Testing with Super-Fast Verification. In: Cryptology ePrint Archive, Paper 2024/1586. 2024. URL: https://eprint.iacr.org/2024/1586 (дата обращения: 18.09.2025).
13. Kohrita T., Nikolaev M., and Silva J. (2024). BOIL: Proof-Carrying Data from Accumulation of Correlated Holographic IOPs. Cryptology ePrint Archive, Paper 2024/1993. URL: https://eprint.iacr.org/2024/1993 (дата обращения: 18.09.2025).
14. Bünz B., Mishra P., Nguyen W., and Wang W. (2024). Arc: Accumulation for Reed--Solomon Codes. Cryptology ePrint Archive, Paper 2024/1731. URL: https://eprint.iacr.org/2024/1731 (дата обращения: 18.09.2025).
15. Reed I.S. and Solomon G. (1960) Polynomial codes over certain finite fields. Journal of the Society for Industrial and Applied Mathematics, 8(2):300-304, 1960. DOI: http://dx.doi.org/10.1137/0108018.

REFERENCES:

[1] Goldwasser S., Micali S., and Rackoff C. (1989). The knowledge complexity of interactive proof systems. In: SIAM Journal on Computing, 18(1):186–208, 1989. DOI: http://dx.doi.org/10.1137/0218012.
[2] Afonin, Vladlen D.; Zapechnikov, Sergey V.; Prostov, Igor A. GOST 34.11-2018 Analysis in The Context of Zero-Knowledge Proofs. IT Security (Russia), [S.l.], v. 31, no. 2, p. 81–89, 2024. ISSN 2074-7136. URL: https://bit.spels.ru/index.php/bit/article/view/1643. DOI: http://dx.doi.org/10.26583/bit.2024.2.05 (in Russian).
[3] Ben-Sasson E., Chiesa A., and Spooner N. (2016). Interactive oracle proofs. In Proceedings, part ii, of the 14th international conference on theory of cryptography. V. 9986, p. 31–60, Berlin, Heidelberg, 2016. Springer-Verlag. DOI: http://dx.doi.org/10.1007/978-3-662-53644-5_2.
[4] Akhmetzyanova L., Babueva A., and Bozhko A. (2023). Streebog as a random oracle. Cryptology ePrint Archive, Paper 2023/1075. 2023. URL: https://eprint.iacr.org/2023/1075 (accessed: 18.09.2025).
[5] Kate A., Zaverucha G.M., and Goldberg I. (2010). Constant-size commitments to polynomials and their applications. In M. Abe, editors, Advances in cryptology – asiacrypt 2010, p. 177–194, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN: 978-3-642-17373-8. DOI: http://dx.doi.org/10.1007/978-3-642-17373-8_11.
[6] Goldwasser S., Kalai Y.T., and Rothblum G.N. (2008). Delegating computation: interactive proofs for muggles. J. ACM, 62(4), September 2015. DOI: http://dx.doi.org/10.1145/2699436.
[7] Fiat A. and Shamir A. (1986). How to prove yourself: practical solutions to identification and signature problems. In A. M. Odlyzko, editors, Advances in cryptology – crypto' 86, p. 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg. ISBN: 978-3-540-47721-1. DOI: http://dx.doi.org/10.1007/3-540-47721-7_12.
[8] Chiesa A. and Yogev E. (2024). Building cryptographic proofs from hash functions. URL: https://github.com/hash-based-snargs-book (accessed: 18.09.2025).
[9] Ben-Sasson E., Bentov I., Horesh Y., and Riabzev M. (2018). Scalable, transparent, and post-quantum secure computational integrity. In: Cryptology ePrint Archive, Paper 2018/046. 2018. URL: https://eprint.iacr.org/2018/046 (accessed: 18.09.2025).
[10] Ben-Sasson E., Bentov I., Horesh Y., and Riabzev M. (2018). Fast Reed-Solomon Interactive Oracle Proofs of Proximity. n I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella, editors, 45Th international colloquium on automata, languages, and programming (icalp 2018), v. 107 of Leibniz International Proceedings in Informatics (LIPIcs), 14:1–14:17, Dagstuhl, Germany, 2018, Schloss Dagstuhl – Leibniz-Zentrum für Informatik. DOI: http://dx.doi.org/10.4230/LIPIcs.ICALP.2018.14.
[11] Arnon G., Chiesa A., Fenzi G., and Yogev E. (2024). STIR: Reed--Solomon Proximity Testing with Fewer Queries. In: Cryptology {ePrint} Archive, Paper 2024/390. 2024. URL: https://eprint.iacr.org/2024/390 (accessed: 18.09.2025).
[12] Arnon G., Chiesa A., Fenzi G., and Yogev E. (2024). WHIR: Reed--Solomon Proximity Testing with Super-Fast Verification. In: Cryptology ePrint Archive, Paper 2024/1586. 2024. URL: https://eprint.iacr.org/2024/1586 (accessed: 18.09.2025).
[13] Kohrita T., Nikolaev M., and Silva J. (2024). BOIL: Proof-Carrying Data from Accumulation of Correlated Holographic IOPs. Cryptology ePrint Archive, Paper 2024/1993. URL: https://eprint.iacr.org/2024/1993 (accessed: 18.09.2025).
[14] Bünz B., Mishra P., Nguyen W., and Wang W. (2024). Arc: Accumulation for Reed--Solomon Codes. Cryptology ePrint Archive, Paper 2024/1731. URL: https://eprint.iacr.org/2024/1731 (accessed: 18.09.2025).
[15] Reed I.S. and Solomon G. (1960) Polynomial codes over certain finite fields. Journal of the Society for Industrial and Applied Mathematics, 8(2):300-304, 1960. DOI: http://dx.doi.org/10.1137/0108018.

Статья поступила в редакцию 18.09.2025; одобрена после рецензирования 30.09.2025; принята к публикации 22.10.2025
The article was submitted 18.09.2025; approved after reviewing 30.09.2025; accepted for publication 22.10.2025