DOI: 10.15514/ISPRAS-2026-38(6)-43

Квантово-устойчивая хэш-функция BLAKE-PQ. Часть 1

А.С. Петренко, ORCID: 0000-0002-9954-4643 <fatawl932@yandex.ru>

Санкт-Петербургский Федеральный исследовательский центр Российской академии наук, Россия, 199178, г. Санкт-Петербург, 14-я линия В.О., д. 39.

Аннотация. Статья посвящена анализу угроз для симметричных криптографических примитивов в условиях появления квантовых алгоритмов, потенциально превосходящих по эффективности поиск прообраза и коллизий, и постановке задачи построения новой криптографической хэш-функции на основе алгоритма BLAKE3, обозначенной как BLAKE-PQ и предназначенной для использования в высоконагруженных системах, включая блокчейн-системы. Предлагаемая архитектура состоит из семи модификаций, в первой части настоящей работы рассмотрены: введение мультипликативного преобразования в раундовую функцию для разрушения чисто аддитивной и побитово-линейной структуры, добавление нелинейного слоя на основе таблицы замен, применение специально спроектированных неабелевых перестановок состояния между раундами, нелинейное смешивание значений в дереве Меркла при объединении дочерних узлов. Для каждого из этих компонентов сформулирован целевой класс потенциальных квантовых атак (алгоритмы скрытого сдвига и скрытой подгруппы, методы для задач типа суммы по модулю два, усовершенствованный поиск коллизий и алгебраический квантовый криптоанализ), дана оценка влияния на алгебраическую структуру и статистические свойства внутреннего состояния, а также проанализированы дополнительные накладные расходы по времени. Во второй части настоящей работы будет оценено совокупное замедление относительно исходного алгоритма BLAKE3, проведено сравнение по расчётной производительности с распространёнными на сегодняшний день хэш-функциями и сформулированы общие для совокупной работы выводы.

Ключевые слова: постквантовая криптография; симметричная криптография; хэш-функция; квантовый криптоанализ; скрытые алгебраические и групповые структуры; дерево Меркла; решёточные задачи; хаотические динамические системы; блокчейн-системы.

Для цитирования: Петренко А.С. Квантово-устойчивая хэш-функция BLAKE-PQ. Часть 1. Труды ИСП РАН, 2026, том 38, вып. 6, часть 2, стр. 363–384. DOI: 10.15514/ISPRAS–2026–38(6)–43.

Благодарности: Исследование проведено в рамках гранта Российского научного фонда № 24-41-04006 "Теоретические основы параметризуемых унифицированных способов задания конечных ассоциативных алгебр и разработка постквантовых криптографических алгоритмов с открытым ключом" (2024-2026).

Quantum-resistant hash function BLAKE-PQ. Part 1

A.S. Petrenko, ORCID: 0000-0002-9954-4643 < fatawl932@yandex.ru >

SPb Federal Research Center of the RAS,
39, 14th Line of V.I., Saint Petersburg, 199178, Russia.

Abstract. The article analyzes threats to symmetric cryptographic primitives in the face of emerging quantum algorithms that potentially outperform preimage and collision searches. It also formulates the problem of constructing a new cryptographic hash function based on the BLAKE3 algorithm, designated BLAKE-PQ, designed for use in high-load systems, including blockchain systems. The proposed architecture consists of seven modifications. The first part of this paper discusses the following: introducing a multiplicative transform into the round function to break the purely additive and bitwise linear structure, adding a nonlinear layer based on a substitution table, using specially designed non-Abelian state permutations between rounds, and nonlinear mixing of values in the Merkle tree when merging child nodes. For each of these components, a target class of potential quantum attacks is formulated (hidden shift and hidden subgroup algorithms, methods for modulo-two sum-type problems, advanced collision detection, and algebraic quantum cryptanalysis), the impact on the algebraic structure and statistical properties of the internal state is assessed, and the additional time overhead is analyzed. In the second part of this paper, the overall slowdown relative to the original BLAKE3 algorithm is estimated, the estimated performance is compared with currently common hash functions, and general conclusions for the overall work are drawn.

Keywords: post-quantum cryptography; symmetric cryptography; hash function; quantum cryptanalysis; hidden algebraic and group structures; Merkle tree; lattice problems; chaotic dynamical systems; blockchain systems.

For citation: Petrenko A.S. Quantum-resistant hash function BLAKE-PQ. Part 1. Trudy ISP RAN/Proc. ISP RAS, 2026, vol. 38, issue 6, part 2, pp. 363-384 (in Russian). DOI: 10.15514/ISPRAS-2026-38(6)-43.

Acknowledgements. The research was conducted as part of the Russian Science Foundation grant No. 24-41-04006 "Theoretical Foundations of Parameterized Unified Methods for Defining Finite Associative Algebras and the Development of Post-Quantum Public-Key Cryptographic Algorithms" (2024-2026).

1. Введение

Развитие квантовых вычислительных систем представляет фундаментальную угрозу для современной криптографической инфраструктуры. В контексте симметричной криптографии общепринятым считается, что основной квантовой угрозой является алгоритм Гровера, обеспечивающий квадратичное ускорение при поиске прообраза хэш-функций и снижающий асимптотическую сложность поиска прообраза с 2^n до порядка $2^{n/2}$, то есть до уровня стойкости $n/2$ бит. Стандартная рекомендация NIST по противодействию этой угрозе заключается в удвоении длины выхода криптографических примитивов [1-2], так переход от 256-битных хэшей к 512-битным потенциально способен обеспечить достаточный уровень криптостойкости в условиях атак с использованием алгоритма Гровера.

Однако такой подход основан на критическом допущении в виде отсутствия квантовых алгоритмов для симметричных криптопримитивов, существенно превосходящих алгоритм Гровера. Недавние исследования демонстрируют, что это допущение может оказаться ошибочным. Работы L.Grassi и X.Bonnetain [3-8] показали существование квантовых алгоритмов для так называемой «k-XOR-задачи» поиска k элементов, чья XOR-сумма равна нулю с субэкспоненциальным ускорением относительно классических методов, что имеет прямые последствия для безопасности симметричных криптопримитивов с «богатой» алгебраической структурой. Хотя эти алгоритмы требуют значительных квантовых ресурсов, их существование указывает на фундаментальную проблему – симметричные криптографические конструкции могут содержать эксплуатируемые квантовыми алгоритмами алгебраические структуры, которые не проявляются в классическом криптоанализе.

Концепция «скрытых цикличностей» – регулярных структур в пространстве состояний криптографического примитива, не обнаруживаемых классическим анализом, но потенциально доступных квантовым алгоритмам через суперпозицию состояний, остается открытым направлением исследований. G.Leurent и T.Peyrin в контексте анализа ARX-конструкций (Addition-Rotate-XOR) [9-10] отмечают, что аддитивно-линейные структуры модуля 2^{32} и XOR-линейность (линейность над полем F_2 относительно операции XOR, побитового исключающего ИЛИ) в векторном пространстве $F_{2^{32}}$ создают предсказуемые дифференциальные и линейные траектории, которые теоретически могут быть усилены квантовыми методами, объединяющими поиск Гровера с алгоритмами решения скрытых сдвигов (hidden shift problem, HSP) и квантовым линейным криптоанализом на базе квантового преобразования Фурье (QFT) или аппроксимированного квантового преобразования Фурье (AQFT) [11]. При этом A.Schrottenloher демонстрирует, что для структур, обладающих внутренней периодичностью или групповыми свойствами, квантовые алгоритмы могут достигать полиномиального или субэкспоненциального ускорения [12].

Алгоритм Саймона и его обобщения для решения задачи о скрытом сдвиге демонстрируют способность квантовых компьютеров эксплуатировать периодичности в криптографических конструкциях. Работы M.Kaplan и соавт. и H.Kuwakado и M.Morii показали, что конструкции с периодическими свойствами, такие как 3-раундовая сеть Фейстеля, шифр Эвена-Мансура и режим Encrypted-CBC-MAC для аутентификации, могут быть полностью скомпрометированы при наличии квантового доступа к криптографическому примитиву с полиномиальным числом запросов [13-15]. Хотя эти атаки требуют модели qCPA (quantum Chosen Plaintext Attack), где противник имеет доступ к криптографическому примитиву в квантовой суперпозиции, что пока остается гипотетическим сценарием, сам факт существования экспоненциального ускорения относительно классических методов ставит под сомнение долгосрочную безопасность конструкций, обладающих скрытыми периодическими структурами.

Алгоритм BHT (Brassard-Høyer-Tapp) для поиска коллизий снижает сложность атаки хэш-функции с $O(2^{n/2})$ до $O(2^{n/3})$. Хотя этот алгоритм теоретически менее эффективен, чем алгоритм Гровера для задачи поиска прообраза, он специфически нацелен на коллизионную стойкость и требует значительных квантовых ресурсов, включая квантовую память (QRAM) объемом не менее $O(2^{n/3})$ кубит. Практическая реализация BHT остается открытой задачей, однако её потенциальное существование подчеркивает необходимость увеличения длины выхода хэш-функций и разработки новых математических конструкций для обеспечения достаточной криптостойкости.

Комбинация различных квантовых алгоритмов может приводить к синергетическим эффектам. Например, работа X.Bonnetain по улучшенным алгоритмам решения задачи о скрытом сдвиге с ограниченным числом кубит демонстрирует, что даже при ограниченных квантовых ресурсах возможно достижение субэкспоненциального ускорения за счет гибридных классико-квантовых стратегий [7]. Аналогично, квантовые атаки, эксплуатирующие периодические структуры, могут быть объединены с методами так называемых «слайдовых» атак (slide attacks), что было продемонстрировано в работах по применению алгоритма Саймона к симметричным примитивам [16,17]. Это указывает на то, что даже конструкции, устойчивые к отдельным квантовым атакам, могут оказаться уязвимыми к комбинированным подходам.

Другие стратегии постквантового усиления симметричных криптопримитивов, такие как увеличение числа раундов и гибридные конструкции в виде конкатенации нескольких независимых хэш-функций, обладают существенными ограничениями. Увеличение раундов повышает классический запас безопасности, но не изменяет фундаментальную алгебраическую структуру криптопримитива, если базовая раундовая функция содержит эксплуатируемую квантовым алгоритмом закономерность, большее число итераций лишь линейно увеличивает сложность атаки, сохраняя её принципиальную применимость.

Гибридные схемы (как например $H(m) = BLAKE3(m) \,||\, SHA3(m)$) защищают от компрометации одного из алгоритмов, но не решают проблему общих структурных уязвимостей, таким образом если будущий квантовый алгоритм эксплуатирует свойства, присущие широкому классу ARX-конструкций и перестановочных сетей, параллельное использование нескольких представителей этого класса не даст качественного выигрыша.

Наряду с рассмотренными подходами в литературе представлены хэш-функции, стойкость которых связывается с вычислительной трудностью математических задач, для которых на сегодняшний день неизвестны эффективные квантовые алгоритмы. К этому направлению относятся решёточные конструкции SWIFFT и SWIFFTX, связывающие коллизионную стойкость с трудными задачами на решётках, а также кодовые FSB и RFSB, связанные с задачами синдромного декодирования [18-20]. Теоретические основы решёточной криптографии, включая результаты Айтаи и связь стойкости криптографических примитивов с вычислительно трудными задачами, рассматриваются также в отечественных работах [21]. Другую математическую основу используют изогенные хэш-функции семейства CGL, где вычисление хэша соответствует последовательному движению по графу изогений суперсингулярных эллиптических кривых [22]. Ещё одно направление образуют специализированные симметричные критпопримитивы для постквантовых приложений, например Haraka v2, ориентированная преимущественно на быстрое хэширование коротких входов в хэш-ориентированных схемах подписи [23]. При этом перечисленные конструкции решают несколько иные задачи, поскольку либо строят новый криптопримитив на основе конкретного предположения о вычислительной трудности, либо оптимизируются для специализированных сценариев применения. В отличие от них, в настоящей работе рассматривается иной подход, основанный на структурном усилении уже существующей высокопроизводительной хэш-функции при сохранении её архитектурных преимуществ и возможностей параллельной обработки.

В качестве базиса для разработки квантово-устойчивой хэш-функции для потенциального использования в блокчейн-системах выбран алгоритм BLAKE3 [24]. Выбор обусловлен рядом факторов. Во-первых, это высокая производительность и развитый параллелизм, выражающаяся в том, что BLAKE3 демонстрирует пропускную способность до 6,4 ГБ/с на современных CPU благодаря древовидной архитектуре и параллельной обработке нескольких фрагментов данных одной инструкцией процессора (SIMD, Single Instruction Multiple Data), что критично для блокчейн-систем с высокой частотой хэширования [25-32]. Во-вторых, важна модульность конструкции, а трёхуровневая архитектура в виде компрессионной функции, раундовой структуры и древовидного режима позволяет вносить локализованные модификации без полной переработки криптопримитива. В-третьих, существенную роль играет устойчивость к классическим атакам, так лучшие известные результаты для BLAKE2s, предшественника BLAKE3, достигают лишь 2,5 раундов, что обеспечивает значительный запас прочности при использовании 7 раундов в BLAKE3 [33,34].

BLAKE3 представляет собой криптографическую хэш-функцию, основанную на конструкции HAIFA (HAsh Iterative FrAmework) с древовидным режимом хэширования. Алгоритм использует внутреннее состояние размером 512 бит и обрабатывает входные данные фрагментами по 1024 бита. Базовым элементом является функция компрессии, применяющая 7 раундов преобразований типа $a = a + b + m,\ d = (d \oplus a) \ll r_1,\ c = c + d,\ b = (b \oplus c) \ll r_2$, где операции включают модульное сложение по модулю 2^{32}, побитовый XOR и циклические сдвиги с константами в 16, 12, 8 и 7 бит. Расписание сообщений детерминировано и основано на фиксированной перестановке индексов входных слов между раундами.

Древовидный режим BLAKE3 можно рассматривать как вариант дерева Меркла над фрагментами по 1024 байта, в котором цепочечные значения объединяются в бинарное дерево родительских узлов, обеспечивая неограниченный параллелизм на уровне фрагментов

и использование SIMD-инструкций для одновременной обработки нескольких экземпляров функции компрессии. Выходное значение формируется через режим расширяемого выхода (XOF), позволяющий генерировать произвольное количество выходных байт из финального состояния путём последовательных применений функции компрессии с увеличением счётчика.

Несмотря на перечисленные выше достоинства, существующие исследования постквантовой безопасности BLAKE3 ограничены анализом стойкости к алгоритмам Гровера для поиска прообразов и к BHT для поиска коллизий. Вопросы устойчивости BLAKE3 к потенциальным квантовым алгоритмам, которые могли бы эксплуатировать аддитивно-мультипликативную структуру ARX-конструкций, линейность операций в древовидном режиме, детерминированность расписания сообщений и отсутствие высокостепенной нелинейности на уровне раундовой функции, остаются по сути неисследованными (рис. 1). А следовательно, квантовая устойчивость, и, в частности, постквантовая криптостойкость, BLAKE3 в широком смысле на сегодняшний день не обеспечена.

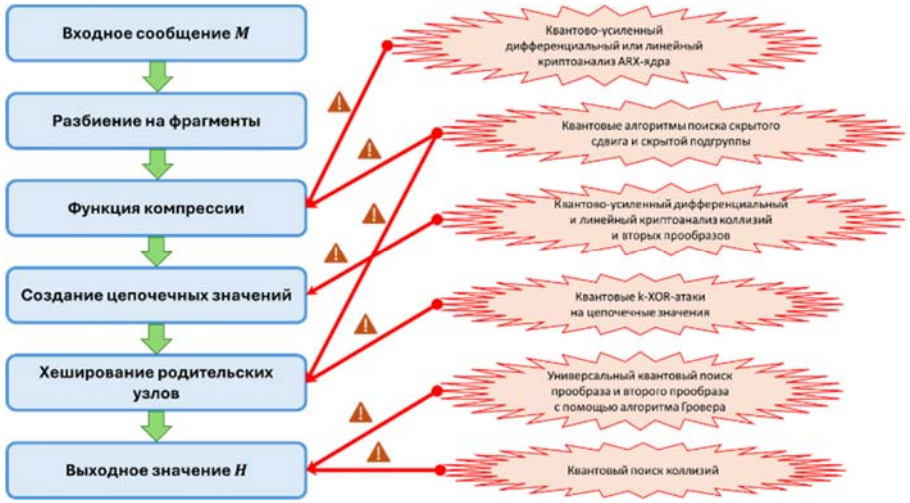


Рис. 1. Квантовые угрозы хэш-функции BLAKE3 в основных структурных компонентах.
Fig. 1. Quantum threats to the BLAKE3 hash function in the main structural components.

Целью настоящей работы является разработка архитектуры модифицированного варианта BLAKE3, обозначенного как BLAKE-PQ (BLAKE Post-Quantum), потенциально устойчивой к широкому спектру потенциальных квантовых атак, включая ранее неизвестные, эксплуатирующие скрытые алгебраические структуры, периодичности и групповые свойства симметричных криптопримитивов. В дальнейшем нас будет интересовать не абсолютная скорость хэширования, а относительная вычислительная цена постквантового усиления по сравнению с изначальными компонентами BLAKE3, поэтому дополнительные накладные расходы по времени (overhead) отдельных модификаций будут измеряться относительно BLAKE3. При этом для обеспечения практической применимости алгоритма в таких высоконагруженных системах, как современные блокчейн-платформы и экосистемы, вводится проектное ограничение временных затрат разрабатываемого криптопримитива не более чем в 10 раз относительно BLAKE3. Данный порог выбран исходя из опубликованных бенчмарков, согласно которым BLAKE3 превосходит SHA3-256 по пропускной способности примерно в 8 раз на современных CPU в зависимости от аппаратных особенностей [35, 36], с дополнительным запасом, учитывающим вариативность аппаратных платформ и реализации. Исходя из этого порога даже при десятикратном замедлении расчётная производительность

разрабатываемой схемы остаётся близкой к уровню широко используемых стандартных хэш-функций. Для достижения поставленной цели предложен комплекс из семи структурных модификаций внутренней архитектуры BLAKE3, каждая из которых нацелена на устранение определенного класса потенциальных квантовых уязвимостей. Далее поразделно рассматриваются их устройство, влияние на структурные свойства криптопримитива и вычислительные затраты, после чего приводятся итоговые выводы и направления дальнейших исследований.

2. Мультипликативный слой в раундовой функции

Базовая раундовая функция BLAKE3 наследует архитектуру ARX от семейства BLAKE2, оперируя исключительно операциями сложения по модулю 2^{32}, XOR и циклических сдвигов. Такая конструкция обеспечивает высокую производительность на современных процессорах благодаря эффективной конвейеризации и SIMD-векторизации, однако создаёт две фундаментальные алгебраические структуры: аддитивную линейность в кольце $Z_{2^{32}}$ и XOR-линейность в векторном пространстве $F_{2^{32}}$. Наличие этих структур теоретически допускает построение квантовых алгоритмов, комбинирующих поиск Гровера с методами решения задач скрытого сдвига, поскольку локальные участки раундовой функции могут демонстрировать почти аффинное поведение, то есть возможность локальной аппроксимации аффинными преобразованиями относительно определённых подпространств входных данных [37-38].

Для разрушения чисто аддитивной структуры в модифицированное ядро BLAKE-PQ внедряется мультипликативный слой непосредственно в базовую функцию смешивания G. В исходном BLAKE3 функция G принимает четыре 32-битных слова состояния $v[a], v[b], v[c], v[d]$ и два слова сообщения x, y, выполняя последовательность операций (приведено на псевдокоде):

```
G(v, a, b, c, d, x, y):
    v[a] ← v[a] + v[b] + x
    v[d] ← ROTR(v[d] ⊕ v[a], 16)
    v[c] ← v[c] + v[d]
    v[b] ← ROTR(v[b] ⊕ v[c], 12)
    v[a] ← v[a] + v[b] + y
    v[d] ← ROTR(v[d] ⊕ v[a], 8)
    v[c] ← v[c] + v[d]
    v[b] ← ROTR(v[b] ⊕ v[c], 7)
```

Здесь ROTR(w, n) обозначает циклический сдвиг вправо 32-битного слова w на n позиций, все сложения выполняются по модулю 2^{32}.

В модифицированной версии BLAKE-PQ после первого аддитивного шага добавляется мультипликативное преобразование (приведено на псевдокоде):

```
G_PQ(v, a, b, c, d, x, y, round):
    v[a] ← v[a] + v[b] + x

    // Вставка мультипликативного преобразования
    μ ← (v[b] ⊕ ROUND_CONST[round]) | 1
    v[a] ← (v[a] · μ) mod 2^32

    v[d] ← ROTR(v[d] ⊕ v[a], 16)
    v[c] ← v[c] + v[d]
    v[b] ← ROTR(v[b] ⊕ v[c], 12)
    v[a] ← v[a] + v[b] + y
```

```
v[d] ← ROTR(v[d] ⊕ v[a], 8)
v[c] ← v[c] + v[d]
v[b] ← ROTR(v[b] ⊕ v[c], 7)
```

Здесь множитель μ формируется как $\mu = (v[b] \oplus c_r) \mid 1$, где c_r обозначает раундовую константу ROUND_CONST[round], зависящую от номера текущего раунда, а операция побитового логического ИЛИ (OR) с единицей гарантирует нечётность множителя. Нечётность μ критична для сохранения биективности преобразования: в кольце $Z_{2^{32}}$ элемент имеет мультипликативный обратный тогда и только тогда, когда он взаимно прост с модулем, что для степени двойки эквивалентно нечётности. Формально, для нечётного μ существует μ^{-1} такое, что $(\mu \cdot \mu^{-1}) \bmod 2^{32} = 1$, что обеспечивает обратимость:

$$v'[a] = (v[a] \cdot \mu) mod 2^{32} \implies v[a] = (v'[a] \cdot \mu^{-1}) mod 2^{32}. \quad (1)$$

В противном случае умножение на чётное число $\mu = 2^k \cdot m$ (где $k \geq 1$) приводит к потере младших k битов информации и необратимой компрессии, что недопустимо для криптографического примитива.

Рассмотрим следующий пример. Пусть $v[a] = 0x12345678, v[b] = 0xABCDEF01, round = 3$, и раундовая константа ROUND_CONST = 0x9E3779B9. Тогда (приведено на псевдокоде):

```
μ = (0xABCDEF01 ⊕ 0x9E3779B9) | 1
  = 0x35FA96B8 | 1
  = 0x35FA96B9  (нечётное)

v[a] ← (0x12345678 · 0x35FA96B9) mod 2³²
```

Выбор формирования множителя из уже присутствующего в вычислении слова $v[b]$ минимизирует накладные расходы, т.к. не требуется загрузка дополнительных констант из памяти, а зависимость μ от текущего состояния вносит нелинейную обратную связь между итерациями функции G. Операция XOR с раундовой константой обеспечивает различие множителей между раундами, предотвращая вырожденные ситуации. Следует отметить, что если бы μ зависел только от $v[b]$, то идентичные значения $v[b]$ в разных раундах порождали бы идентичные мультипликативные преобразования, создавая потенциальную периодичность.

Данная модификация применяется во всех семи раундах компрессионной функции. Каждый раунд BLAKE3 содержит восемь вызовов G: четыре диагональных смешивания (позиции (0,4,8,12), (1,5,9,13), (2,6,10,14), (3,7,11,15)) и четыре колоночных смешивания (позиции (0,5,10,15), (1,6,11,12), (2,7,8,13), (3,4,9,14)). Следовательно, полная компрессия фрагмента включает $7 \times 8 = 56$ операций умножения.

На современных микроархитектурах семейства x86-64 (Intel, AMD) и ARM 32-битное умножение характеризуется следующими параметрами, представленными на табл. 1.

Табл. 1. Пропускная способность основных операций в зависимости от типа микроархитектуры.
Table 1. Bandwidth of basic operations, depending on the type of microarchitecture.

Микроархитектура	**ADD** (задержка/пропускная способность)	**XOR** (задержка/пропускная способность)	**MUL** (задержка/пропускная способность)	**ROTR** (задержка/пропускная способность)
Intel Ice Lake	1 / 4 оп./такт	1 / 4 оп./такт	3 / 1 оп./такт	1 / 2 оп./такт
AMD Zen 3	1 / 4 оп./такт	1 / 4 оп./такт	3 / 1 оп./такт	1 / 2 оп./такт
ARM Cortex-A78	1 / 4 оп./такт	1 / 2 оп./такт	4 / 1 оп./2 такта	1 / 2 оп./такт

Задержка операции MUL в 3-4 такта означает время от подачи операндов до готовности результата, однако благодаря конвейеризации процессор способен начинать новое

умножение каждый такт (пропускная способность 1 операция за такт). При оптимальном развёртывании кода компилятором операции умножения в разных вызовах G могут выполняться параллельно, сглаживая эффекты задержки.

При этом если исходная компрессия BLAKE3 требует ~200-250 тактов процессора на фрагмент в зависимости от микроархитектуры и качества оптимизации (в дальнейшем примем за среднее 220 тактов), добавление 56 умножений при пропускной способности 1 оп./такт добавляет ~ 56 тактов при близкой к идеальной конвейеризации, что даёт дополнительные накладные расходы по времени (overhead):

$$\text{overhead} = \frac{56}{220} \times 100\% \approx 25\%. \quad (2)$$

Это находится в пределах установленного ограничения общего замедления, оставляя запас для остальных модификаций.

С точки зрения криптографической стойкости внедрение мультипликативного слоя приводит к нескольким важным эффектам. Рассмотрим алгебраическую структуру преобразования. Пусть S обозначает пространство внутренних состояний, и пусть $f: S \to S$ – раундовая функция. В чистых ARX-конструкциях f может быть локально аппроксимирована аффинными преобразованиями:

$$f(x) \approx Ax + b (\mathrm{mod} 2^{32}), \quad (3)$$

где A – матрица линейных коэффициентов, b – вектор сдвига. Такие аффинные участки эксплуатируются в дифференциальном и линейном криптоанализе. После внедрения мультипликативного слоя преобразование приобретает квадратичную компоненту:

$$f_{PQ}(x) = (Ax + b) \cdot \mu(x) + r(x), \quad (4)$$

где $\mu(x) = (x_b \oplus c) \mid 1$ зависит от текущего значения слова $v[b]$, а $r(x)$ обозначает остальные нелинейные члены, возникающие из взаимодействия умножения с более глубокими слоями раундовой функции и не сводящиеся к аффинной части. Раскрывая произведение:

$$f_{PQ}(x) = A\mu(x) \cdot x + b\mu(x) + \cdots. \quad (5)$$

Видно, что коэффициенты линейных членов $A \cdot \mu(x)$ теперь нелинейно зависят от входа, нарушая аффинную структуру.

Ещё одной критической характеристикой является алгебраическая степень выходных битов относительно входных. Для n-битного умножения двух слов a и b результат $r = a \cdot b \bmod 2^n$ имеет битовое представление:

$$r_i = \bigoplus_{j+k=i,\, j<n,\, k<n} (a_j \wedge b_k), \quad (6)$$

где $\oplus$ обозначает XOR, $\wedge$ – логическое и. Степень этого полинома над F_2 составляет $deg(r_i) \leq min(i+1, 2n-i-1)$, достигая максимума порядка n в средних битах. Для 32-битного умножения степень отдельных битов результата может достигать 32, что значительно превышает степень в ARX-конструкциях, где каждая операция сложения добавляет не более одной степени к полиному.

Для наглядности рассмотрим упрощённый пример для 4-битного умножения. Пусть $a = (a_3, a_2, a_1, a_0)$ и $b = (b_3, b_2, b_1, b_0)$ – входные биты. Третий бит результата r_3 выражается как:

$$r_3 = (a_0 \wedge b_3) \oplus (a_1 \wedge b_2) \oplus (a_2 \wedge b_1) \oplus (a_3 \wedge b_0). \quad (7)$$

В реальной реализации к этому добавляются слагаемые, возникающие из переносов между младшими разрядами, что ещё повышает алгебраическую степень результата. Учитывая переносы от младших разрядов, полная формула становится полиномом степени 4-5 от

восьми входных переменных, в то время как в ARX-конструкции за один раунд степень растёт линейно с числом операций сложения.

Важным аспектом является сохранение детерминированности и отсутствие «ветвлений» в коде. Формирование μ выполняется фиксированной последовательностью (приведено на псевдокоде):

```
temp ← v[b] XOR ROUND_CONST[round]
μ ← temp OR 1
```

Обе операции XOR и OR выполняются за константное время независимо от значений данных, что исключает возможность атаки по времени выполнения (тайминг-атаки). Операция умножения в современных процессорах также выполняется с постоянным временем независимо от значений операндов, что критично для криптографических приложений.

Теоретическое обоснование устойчивости к квантовым атакам основывается на результатах работ по квантовому криптоанализу структурированных задач. Алгоритмы типа k-XOR и обобщённые методы поиска коллизий и k-кортежей существенно зависят от наличия линейных или низкостепенных соотношений между промежуточными значениями. Рассмотрим задачу k-XOR: найти k элементов из списка L, таких что их XOR равен нулю. Квантовые алгоритмы для этой задачи, как например алгоритм L.Grassi и соавт. [3], работают исходя из предположения, что целевая функция обладает линейной структурой относительно XOR. Формально, если $f(x \oplus y) = f(x) \oplus f(y)$ (соблюдается XOR-гомоморфизм), задача сводится к решению системы линейных уравнений над $\mathbb{F}_2$, для которой квантовые алгоритмы дают экспоненциальное ускорение.

Введение мультипликативных зависимостей нарушает это свойство. Для произведения выполняется:

$$(x \oplus \Delta) \cdot \mu \neq (x \cdot \mu) \oplus (\Delta \cdot \mu) (\mathrm{mod} 2^{32}), \tag{8}$$

поскольку умножение не дистрибутивно относительно XOR. Более того, зависимость множителя от состояния $\mu = (v[b] \oplus c) \mid 1$ создаёт нелинейную обратную связь:

$$\mu(x \oplus \Delta) = \left((x_b \oplus \Delta_b) \oplus c\right) \mid 1 \neq \mu(x) \oplus \mu(\Delta). \tag{9}$$

Это нарушает рассматриваемую XOR-линейную структуру и тем самым ограничивает прямую применимость алгоритмов, использующих данное свойство.

Альтернативные варианты внедрения мультипликативного слоя также рассматривались в процессе проектирования. Использование двух умножений на четверть функции G (приведено на псевдокоде):

```
v[a] ← (v[a] · μ₁) mod 2³²
v[c] ← (v[c] · μ₂) mod 2³²
```

было отвергнуто из-за удвоения числа операций умножения до 112 на компрессию, что приводит к дополнительным накладным расходам по времени $\sim 50\%$ только от мультипликативного слоя. Применение мультипликативного преобразования только в чётных раундах (приведено на псевдокоде):

```
if round % 2 == 0:
    v[a] ← (v[a] · μ) mod 2³²
```

снижает накладные расходы вдвое до ~ 28 умножений, но создаёт асимметрию между раундами, так нечётные раунды остаются сугубо ARX-конструкциями, потенциально позволяя строить дифференциальные характеристики через эти «слабые» раунды с последующим соединением через мультипликативные раунды. Выбранная схема с одним умножением на каждую четверть функции G во всех раундах представляет компромисс между вычислительными затратами и внесением мультипликативного преобразования во все раунды.

3. Нелинейный финальный слой на основе S-box

Несмотря на введение мультипликативного слоя в раундовую функцию, семь раундов ARX-преобразований, даже усиленных умножением, могут сохранять остаточные алгебраические структуры в выходных битах. Классический криптоанализ блочных шифров и хэш-функций демонстрирует, что многократное применение операций с низкой локальной нелинейностью (сложение, XOR) не гарантирует максимальную алгебраическую степень выходной функции относительно входа [39]. Для ARX-конструкций характерна ситуация, когда отдельные выходные биты могут быть представлены полиномами умеренной степени над полем F_2, что теоретически допускает применение алгебраических атак, усиленных квантовыми методами решения систем полиномиальных уравнений.

Для увеличения нелинейности преобразования состояния в BLAKE-PQ внедряется финальный слой подстановок на основе таблицы замен (S-box). Этот слой применяется однократно после завершения седьмого раунда, непосредственно перед процедурой финализации через XOR с цепочечным значением. Использование S-box вводит дополнительное нелинейное преобразование состояния. В качестве количественных характеристик выбранного S-box далее рассматриваются его нелинейность, дифференциальная равномерность и алгебраическая степень.

В качестве базовой таблицы замен выбран S-box алгоритма AES, определённый через композицию мультипликативного обращения в поле Галуа $GF(2^8)$ и аффинного преобразования. Этот выбор обусловлен следующими свойствами:

Во-первых, нелинейность S-box AES составляет $\mathrm{NL} = 112$, при теоретическом максимуме 120 для функций F: $\mathrm{F_2}^8 \rightarrow \mathrm{F_2}^8$. Нелинейность S-box AES при этом определяется как минимальное расстояние Хэмминга от всех аффинных функций:

$$\mathrm{NL}(S) = \min_{L \in \mathcal{A}} d_H(S, L), \tag{10}$$

где $\mathcal{A}$ – множество всех аффинных булевых функций той же размерности, а d_H обозначает расстояние Хэмминга. Значение нелинейности характеризует расстояние S-box от множества аффинных функций и используется при оценке применимости линейного криптоанализа.

Во-вторых, дифференциальная равномерность S-box AES составляет $\delta = 4$, что является оптимальным для конструкций на основе обращения в поле. Дифференциальная равномерность определяется как максимальное число входных пар с фиксированной разностью, дающих фиксированную выходную разность:

$$\delta = \max_{\Delta_{\mathrm{in}} \neq 0,\, \Delta_{\mathrm{out}}} \left| \{x : S(x) \oplus S(x \oplus \Delta_{\mathrm{in}}) = \Delta_{\mathrm{out}}\} \right|. \tag{11}$$

Для S-box AES значение $\delta = 4$ соответствует максимальной вероятности фиксированного дифференциального перехода $\frac{4}{256} = 2^{-6}$.

В-третьих, алгебраическая степень S-box AES равна 7, что является максимально возможным для обратимой 8-битной функции, т.к. теоретическим максимумом является степень $n - 1$ для n-битной подстановки.

Применение S-box выполняется на побайтовом уровне ко всем 16 словам состояния. Каждое 32-битное слово $v[i]$ разбивается на четыре байта, каждый из которых независимо преобразуется через таблицу замен следующим образом (приведено на псевдокоде):

```
S_layer(v):
    for i = 0 to 15:
        w ← v[i]
        b₀ ← SBOX[w & 0xFF]
        b₁ ← SBOX[(w >> 8) & 0xFF]
        b₂ ← SBOX[(w >> 16) & 0xFF]
        b₃ ← SBOX[(w >> 24) & 0xFF]
        v[i] ← b₀ | (b₁ << 8) | (b₂ << 16) | (b₃ << 24)
```

Здесь SBOX[...] обозначает таблицу замен размером 256 байт, содержащую предвычисленные значения S-box AES.

Рассмотрим следующий пример. Пусть $v = 0x12345678$ после седьмого раунда. Побайтовое разложение даёт (приведено на псевдокоде):

```
b₀ = 0x78,  b₁ = 0x56,  b₂ = 0x34,  b₃ = 0x12
```

Применяя таблицу S-box AES (приведено на псевдокоде):

```
SBOX[0x78] = 0x50,  SBOX[0x56] = 0xF2,
SBOX[0x34] = 0x30,  SBOX[0x12] = 0xC9
```

Получаем результат (приведено на псевдокоде):

```
v[0] ← 0xC930F250
```

Выбор байтового уровня обработки обусловлен балансом между криптографическими свойствами и вычислительной эффективностью. Альтернативные подходы включают крупноблочные S-box, как в отечественном стандарте Кузнечик [40], где используется 128-битная подстановка, разделённая на 16 независимых 8-битных S-box. Однако прямое масштабирование до 32-битной S-box для 32-битных слов BLAKE-PQ потребовало бы таблицу размером $2^{32} \times 4$ байт $= 16$ ГБ, что делает подход непрактичным. Применение 4-битных S-box при этом снижает криптографические свойства, так максимальная алгебраическая степень для 4-битных функций составляет 3, что существенно слабее требуемых характеристик.

С точки зрения производительности операция табличной подстановки характеризуется следующими параметрами. Размер таблицы SBOX составляет 256 байт, что гарантированно помещается в кэш первого уровня (L1-кэш) современных процессоров. Обращение к L1-кэшу требует 3-4 такта задержки. Для преобразования 16 слов необходимо выполнить 64 обращения к таблице, что при идеальной загрузке кэша составляет $\sim 192 - 256$ тактов. При этом дополнительно требуются операции масштабирования и побитового ИЛИ для формирования итогового 32-битного слова, что добавляет ~ 128 тактов. Суммарные дополнительные накладные расходы по времени от S-box слоя составляют $\sim 320 - 384$ такта на компрессию фрагмента, что при базовой стоимости BLAKE3 ~ 220 тактов даёт:

$$\text{overhead}_{\text{S-box}} \approx \frac{350}{220} \times 100\% \approx 159\% \quad (12)$$

Однако поскольку S-box применяется только один раз на компрессию, абсолютный вклад в общее замедление находится в приемлемых пределах.

Альтернативный подход – применение S-box только к части состояния, например, к первым 8 словам состояния v[0…7], что снижает дополнительные накладные расходы по времени вдвое. Однако такое решение создаёт асимметрию, т.к. вторая половина состояния v[8…15] не проходит через нелинейный слой, потенциально сохраняя низкостепенные полиномиальные соотношения. Учитывая, что финализация BLAKE3 выполняет XOR между половинами состояния (приведено на псевдокоде):

```
for i = 0 to 7:
    output[i] = v[i] ⊕ v[i+8] ⊕ chaining_value[i]
```

Отсутствие S-box преобразования в v[8…15] означает, что в выходное значение попадают биты, не прошедшие нелинейное преобразование. Это снижает гарантии безопасности, поэтому было принято решение применять S-box ко всем 16 словам.

С точки зрения квантовой устойчивости слой S-box решает несколько задач. Квантовые варианты алгебраических атак на симметричные криптопримитивы основываются на представлении криптографической функции в виде системы полиномиальных уравнений над F_2 с последующим решением этой системы квантовыми алгоритмами Гровера или специализированными методами вроде квантовых алгоритмов для решения систем нелинейных полиномиальных уравнений, дающими по крайней мере квадратичное ускорение по сравнению с классическими переборными методами. Эффективность таких

атак критически зависит от алгебраической степени полиномов, так системы полиномиальных уравнений низкой степени могут быть решены значительно эффективнее высокостепенных систем.

Рассмотрим упрощённую модель. Пусть состояние после семи раундов ARX с дополнительным мультипликативным слоем представимо полиномами степени $d \leq 20$ от входных битов. После применения S-box с алгебраической степенью 7 каждый выходной бит представляется композицией:

$$\text{output}_i = S_{\text{box}}\left(\text{poly}_{\deg \leq 20}(input)\right). \quad (13)$$

Степень композиции оценивается как произведение степеней:

$$\deg(\text{output}_i) \leq 7 \times 20 = 140. \quad (14)$$

Полученное значение представляет верхнюю оценку алгебраической степени рассматриваемой композиции. Увеличение степени полиномиального представления является фактором, усложняющим применение методов алгебраического криптоанализа, однако само по себе не является оценкой вычислительной сложности соответствующей атаки. При этом для сравнения, системы полиномиальных уравнений высокой степени считаются крайне трудными для решения как классическими методами, вроде алгоритмов на основе баз Грёбнера [41] и смежных алгебраических техник, так и с квантовым ускорением. Квантовые улучшения для этих алгоритмов, как правило, дают лишь квадратичный выигрыш во времени и не меняют экспоненциальный характер сложности задачи.

Дополнительно применение S-box изменяет статистические зависимости в промежуточном состоянии и может затруднять сохранение отдельных структурных смещений через финальный нелинейный слой. При этом оценка статистической неотличимости полного преобразования BLAKE-PQ от случайного отображения требует отдельного экспериментального анализа.

Важным аспектом такой реализации является защита от атак по побочным каналам. Табличная реализация S-box потенциально уязвима к кэш-тайминговым атакам, то есть атакам, восстанавливающим секретную информацию по различиям во времени доступа к данным в кэше процессора. Поскольку время доступа к элементу таблицы может зависеть от того, находится ли соответствующая кэш-линия в L1, L2 или требует обращения к основной памяти. В современных процессорах с механизмами аппаратной предвыборки данных (hardware prefetching) эта проблема значительно уменьшает количество потенциальных утечек через побочные каналы кэширования для небольших таблиц в 256 байт, постоянно находящихся в L1. Альтернативным подходом является «битслайсная» (bitsliced) реализация S-box, при которой таблица замен реализуется через набор логических операций без обращений к памяти, однако для AES S-box такая реализация требует ~ 115 логических операций на байт, что значительно медленнее табличного варианта и выходит за рамки обозначенного ограничения на увеличение временных затрат.

В результате однократное применение побайтового S-box слоя на основе AES ко всем 16 словам состояния после завершения раундовых преобразований вводит дополнительную алгебраическую нелинейность выходной функции при умеренных вычислительных затратах, внося критический вклад в постквантовую устойчивость конструкции BLAKE-PQ.

4. Неабелева групповая перестановка состояния

Алгебраическая структура многих симметричных криптографических примитивов обладает скрытой коммутативностью или абелевостью на уровне операций над подмножествами внутреннего состояния. В классическом криптоанализе эти свойства эксплуатируются через линейные и дифференциальные методы, однако в квантовом контексте наличие абелевой групповой структуры создаёт принципиально иную угрозу в виде возможности применения

алгоритмов решения задачи скрытой подгруппы (Hidden Subgroup Problem, HSP) [42]. Квантовые алгоритмы для задачи HSP на абелевых группах, включая классический алгоритм Шора для факторизации и дискретного логарифма, демонстрируют экспоненциальное ускорение относительно известных классических методов. И хотя на текущий момент обобщение на неабелевы группы остаётся открытой проблемой квантовых вычислений, установлено, что для абелевых групп эффективные квантовые алгоритмы существуют и работают за полиномиальное время.

Структура состояния BLAKE3 представляет собой вектор из 16 слов по 32 бита, организованных в матрицу 4×4. Раундовая функция выполняет смешивания через функцию G, применяемую последовательно к колонкам и диагоналям этой матрицы. Важно отметить, что сами индексы состояния фиксированы, колонковые смешивания всегда обрабатывают четвёрки (0,1,2,3), (4,5,6,7), (8,9,10,11), (12,13,14,15), а диагональные – четвёрки (0,5,10,15), (1,6,11,12), (2,7,8,13), (3,4,9,14). Эта детерминированная структура создаёт тензорное разложение пространства состояний, т.к. состояние можно представить как прямое произведение четырёх независимых подпространств, каждое из которых эволюционирует относительно независимо до диагональных смешиваний.

Математически это означает, что групповая структура преобразований состояния содержит абелевы подгруппы. Рассмотрим упрощённую модель: пусть σ_i – преобразование i-й колонки, действующее только на слова $\{4i, 4i+1, 4i+2, 4i+3\}$. Тогда σ_i и σ_j при $i \neq j$ коммутируют:

$$\sigma_i \circ \sigma_j = \sigma_j \circ \sigma_i, \tag{15}$$

поскольку они действуют на непересекающиеся подмножества состояния. Композиция таких преобразований образует абелеву группу $(\mathrm{Z}_2)^4$, где каждая компонента соответствует одной колонке. Даже после введения диагональных смешиваний полная группа преобразований может содержать нетривиальные абелевы факторы, то есть компоненты, поведение которых описывается коммутативными подгруппами, потенциально эксплуатируемые неким квантовым алгоритмом HSP.

Для нарушения абелевой структуры в BLAKE-PQ внедряется слой неабелевых перестановок индексов состояния, применяемый после каждого раунда. Перестановки спроектированы таким образом, чтобы их композиции порождали неабелеву группу – а именно, подгруппы симметрических групп S_n с $n \geq 3$, которые являются неабелевыми. Простейшие неабелевы операции – транспозиции (обмены двух элементов). Хотя отдельные транспозиции коммутируют, если действуют на непересекающиеся пары, правильный выбор паттерна обменов гарантирует некоммутативность последовательных применений.

Такая схема перестановок основана на фиксированном наборе обменов, применяемом после завершения смешиваний по диагоналям и колонкам в каждом раунде (приведено на псевдокоде):

```
permutation_layer(v, round):
    // Группа 0-4: перестановки в первом квинтете
    swap(v[0], v[1])    // транспозиция (0 1)
    swap(v[2], v[4])    // транспозиция (2 4)

    // Группа 5-9: перестановки во втором квинтете
    swap(v[5], v[7])    // транспозиция (5 7)
    swap(v[6], v[8])    // транспозиция (6 8)

    // Перестановка оставшихся слов состояния
    swap(v[10], v[12])  // транспозиция (10 12)
    swap(v[11], v[13])  // транспозиция (11 13)
    swap(v[14], v[15])  // транспозиция (14 15)
```


Здесь $swap(a, b)$ обозначает базовую операцию обмена значений двух переменных (приведено на псевдокоде):

```
swap(a, b):
    temp ← a
    a ← b
    b ← temp
```

Рассмотрим алгебраические свойства этой конструкции. Каждая транспозиция $(i\ j)$ является элементом симметрической группы S_{16}. Композиция транспозиций порождает подгруппу S_{16}. Проверим некоммутативность. Пусть π_1 – паттерн раунда r, π_2 – паттерн раунда $r+1$ (в данной схеме они идентичны, но допускается использовать раундозависимые варианты). Рассмотрим действие на конкретный элемент. Обозначим четыре выбранных слова состояния как a, b, c, d вместо $v[i_0], v[i_1], v[i_2], v[i_3]$. После первого применения π_1 (приведено на псевдокоде):

```
swap(v[0], v[1]) → v[0] = b, v[1] = a
swap(v[2], v[4]) → v[2] = d, v[4] = c
```

Получаем состояние $(b, a, d, x, c, \dots)$, где x – неизменённый элемент. Применяем π_1 второй раз (приведено на псевдокоде):

```
swap(v[0], v[1]) → v[0] = a, v[1] = b
swap(v[2], v[4]) → v[2] = c, v[4] = d
```

Получаем исходное состояние $(a, b, c, x, d, \dots)$, то есть ${\pi_1}^2 = id$, где id обозначает тождественное отображение. Однако рассмотрим композицию с другим паттерном. Пусть π_2 включает $swap(v, v)$ (приведено на псевдокоде):

```
Начальное: (a, b, c, _, d, ...)
π1: (b, a, d, _, c, ...)
π2 (swap(v[1], v[2])): (b, d, a,  , c, ...)
```

Теперь применим в обратном порядке (приведено на псевдокоде):

```
Начальное: (a, b, c, _, d, ...)
π2 (swap(v[1], v[2])): (a, c, b, _, d, ...)
π1 (swap(v[0], v[1])): (c, a, b, _, d, ...)
   (swap(v[2], v[4])): (c, a, d,  , b, ...)
```

Результаты различны: $(b, d, a, _, c, \dots) \neq (c, a, d, _, b, \dots)$, следовательно $\pi_1 \circ \pi_2 \neq \pi_2 \circ \pi_1$, что доказывает некоммутативность.

В используемой фиксированной схеме семь транспозиций изменяют расположение соответствующих пар слов состояния. Математически группа, порождённая набором транспозиций $T = \{(0\ 1), (2\ 4), (5\ 7), (6\ 8), (10\ 12), (11\ 13), (14\ 15)\}$, является подгруппой S_{16}. После применения перестановки следующий раунд функции G обрабатывает изменённое расположение слов состояния, вследствие чего изменяются позиционные зависимости между последовательными раундами. После перестановки индексов следующий раунд применяет функцию G к новым позициям слов, что создаёт нелинейную связь между исходными и конечными позициями данных.

Критическое свойство такой модификации заключается в следующем – даже если группа перестановок сама по себе абелева, её композиция с нелинейными раундовыми преобразованиями нарушает абелеву структуру полной системы перестановок состояния. Пусть R обозначает раундовую функцию ARX-конструкции, дополненной мультипликативным слоем, P – слой перестановок. Полное преобразование одного раунда:

$$F = P \circ R. \tag{16}$$

Даже если P абелева, композиция некоммутативна $R(P(x)) \neq P(R(x))$ в общем случае, поскольку R зависит от конкретных позиций слов в состоянии. Рассмотрим два последовательных раунда:

$$F^{(2)} = (P \circ R) \circ (P \circ R) = P \circ R \circ P \circ R. \tag{17}$$

Промежуточный член $R \circ P$ не упрощается, поскольку раундовая функция нелинейно зависит от расположения слов. Это приводит к эффективно неабелевой динамике всей системы.

С точки зрения производительности слой перестановок является исключительно «дешёвым». Операция $swap$ реализуется тремя операциями XOR или тремя операциями пересылки регистров (приведено на псевдокоде):

```
// XOR-swap (без временной переменной)
a ← a ⊕ b
b ← a ⊕ b
a ← a ⊕ b

// Register swap (на современных архитектурах)
temp ← a
a ← b
b ← temp
```

На архитектурах с достаточным числом регистров компилятор выполняет $swap$ через переименование регистров без фактических операций пересылки данных. Семь транспозиций требуют максимум 21 элементарную операцию (3 на $swap$), что при задержке в один такт на операцию составляет ~ 21 такт. Это применяется 7 раз после каждого раунда, давая суммарные дополнительные накладные расходы по времени ~ 150 тактов на компрессию фрагмента, что составляет:

$$\text{overhead}_{\text{perm}} \approx \frac{150}{220} \times 100\% \approx 68\%. \quad (18)$$

Однако эта оценка может быть пересмотрена, т.к. на практике $swap$-операции могут быть выполнены параллельно с другими вычислениями благодаря механизмам внеочередного исполнения, снижая реальные дополнительные накладные расходы по времени до $\sim 30 - 40$ тактов ($\sim 15\%$ замедления) [43].

Теоретическое обоснование защиты от квантовых алгоритмов, решающих задачу HSP базируется на сложности этих алгоритмов для неабелевых групп. Для абелевой группы G квантовый алгоритм HSP работает следующим образом: задан квантовый оракул $f: G \to X$ и выполняется условие, что f постоянна на смежных классах некоторой скрытой подгруппы $H \subset G$, требуется найти систему порождающих элементов подгруппы H. Алгоритм использует квантовое преобразование Фурье (QFT) или же аппроксимированное квантовое преобразование Фурье (AQFT) над группой G, которое для абелевых групп эффективно реализуется за полиномиальное время [11]. После полиномиального числа запросов к оракулу выполняется квантовое преобразование Фурье по группе G и измерение в соответствующем Фурье-базисе. Это измерение даёт случайный элемент двойственной группы $\hat{G}$, ортогональный скрытой подгруппе H. Собирая достаточно таких образцов, можно по их общему ортогональному подпространству восстановить порождающие элементы H.

Для неабелевых групп QFT заменяется на нередуцированное представление группы через матрицы, и эффективность алгоритма зависит от структуры представлений группы. Для симметрической группы S_n нередуцированные представления имеют размерность, растущую факториально по n, что делает прямое применение потенциального HSP-алгоритма экспоненциально «дорогим». Существуют частичные результаты для специальных классов неабелевых групп, но общий случай на данный момент остаётся открытой проблемой.

Внедряя неабелеву перестановочную структуру, алгоритм BLAKE-PQ переводит потенциальную задачу HSP из класса абелевых групп в класс неабелевых, для которых эффективных алгоритмов не существует. Даже если будущий квантовый алгоритм сможет эксплуатировать какие-либо групповые свойства криптопримитива, задача взлома существенно усложняется, требуя экспоненциально больших квантовых ресурсов.

Альтернативные схемы перестановок рассматривались в процессе проектирования. Использование полной случайной перестановки 16 элементов (слов) потребовало бы хранения таблицы индексов и её применения через косвенную адресацию (приведено на псевдокоде):

```
for i = 0 to 15:
    temp[i] ← v[perm_table[i]]
v ← temp
```

Это требует 16 обращений к памяти для загрузки индексов и 16 операций пересылки, что значительно «дороже» фиксированных операций $swap$. Более того, косвенная адресация может создавать зависимости по данным, препятствующие конвейеризации.

Другой вариант – раундозависимые перестановки, где паттерн обменов меняется от раунда к раунду (приведено на псевдокоде):

```
permutation_layer(v, round):
    pattern ← PERM_PATTERNS[round]
    apply_swaps(v, pattern)
```

Это увеличивает сложность потенциального анализа для криптоаналитика, но требует хранения 7 различных паттернов и выборки по номеру раунда, добавляя накладные расходы без существенного выигрыша в безопасности, т.к. уже одна фиксированная схема перестановок, применённая 7 раз в композиции с нелинейными раундами, создаёт достаточно сложную неабелеву структуру.

Выбранная же схема с семью фиксированными транспозициями изменяет позиционные зависимости между последовательными раундами при дополнительных вычислительных затратах порядка одной-двух сотен тактов на компрессию фрагмента, затрудняя прямое применение HSP-подходов к структуре преобразований состояния.

5. Нелинейное смешивание состояний в дереве Меркла

Архитектура хэширования BLAKE3 принципиально отличается использованием древовидного режима с неограниченным ветвлением от традиционных конструкций Меркла-Дамгора, в которых произвольное по длине сообщение разбивается на фрагменты фиксированного размера, последовательно обрабатывается компрессионной функцией, а каждый следующий шаг использует в качестве входа предыдущее цепочечное значение. Входное сообщение разбивается на фрагменты (чанки, chunks) по 1024 байта, каждый из которых отдельно обрабатывается в дерево высоты порядка $log_{16}(chunk_size/64)$, после чего результаты объединяются в бинарное дерево родительских узлов до получения корневого хэша. Эта структура обеспечивает массивный параллелизм и является ключевым фактором высокой производительности BLAKE3 на многоядерных системах. Однако древовидная структура вводит новую специфическую уязвимость – операция объединения двух дочерних узлов в родительский выполняется через стандартную компрессионную функцию с конкатенацией цепочечных значений «детей».

Формально, пусть CV_L и CV_R обозначают 256-битные цепочечные значения левого и правого дочерних узлов. В оригинальном BLAKE3 родительский узел вычисляется как:

$$\text{CV}_{\text{parent}} = \text{compress}(k, \text{CV}_L \parallel \text{CV}_R, t, \text{PARENT}), \quad (19)$$

где k – ключ или начальное значение, t – счётчик фрагментов, PARENT – флаг типа узла. Критически важно, что операция конкатенации $CV_L \parallel CV_R$ является линейной относительно битовых операций, она просто помещает биты в последовательные позиции входного фрагмента без перемешивания. Хотя последующая компрессионная функция нелинейна, начальное расположение входных данных сохраняет структурное разделение между левой и правой половинами.

Эта особенность создаёт потенциальную уязвимость к обобщённым k-XOR атакам на древовидной структуре. В контексте рассматриваемой атаки на древовидную структуру используется обобщённая k-списочная формулировка задачи k-XOR: даны k списков $L_1, L_2, \ldots, L_k$ по N элементов каждый, необходимо найти по одному элементу из каждого списка такому, чтобы их XOR был равен целевому значению:

$$x_1 \oplus x_2 \oplus \ldots \oplus x_k = \text{target}, \qquad x_i \in L_i, \tag{20}$$

Классическая сложность решения составляет $O\left(N^{\frac{k}{k+1}}\right)$, но квантовые алгоритмы типа алгоритма L.Grassi и соавт. [3-4] достигают сложности $O\left(N^{\frac{k}{2k-1}}\right)$ с использованием квантовой оперативной памяти (QRAM), что даёт субэкспоненциальное ускорение для $k \geq 3$. Под QRAM мы понимаем абстрактный ресурс, который позволяет по суперпозиции адресов $\sum_i \alpha_i \mid i\rangle$ за один логический запрос получить суперпозицию соответствующих содержимых ячеек $\sum_i \alpha_i \mid i, x_i\rangle$. Такой ресурс необходим большинству известных квантовых алгоритмов для k-XOR и связанных задач, поскольку они предполагают одновременный доступ ко многим элементам больших списков.

В контексте древовидного хэширования атака может быть сформулирована следующим образом. Предположим, что структура дерева позволяет противнику выбирать множество входных фрагментов и наблюдать соответствующие промежуточные узлы дерева. Если операция объединения узлов обладает линейной компонентой, противник может построить систему уравнений вида:

$$f\left(\mathrm{CV}_{i_1} \oplus \mathrm{CV}_{i_2} \oplus \ldots \oplus \mathrm{CV}_{i_k}\right) = \text{target}, \tag{21}$$

где f – компрессионная функция, ограниченная на подпространство входов, допускающее линейную аппроксимацию. Квантовый алгоритм k-XOR может эффективно найти такой набор индексов, удовлетворяющий этому условию, потенциально позволяя построить коллизии или прообразы для полного дерева быстрее универсальных атак, не зависящих от внутренней структуры примитива.

Для нарушения линейности операции объединения узлов в BLAKE-PQ внедряется нелинейный слой смешивания дочерних значений. Модифицированная процедура вычисления родительского узла выполняется в три этапа.

Первый этап – стандартная компрессия конкатенированных значений (приведено на псевдокоде):

```
parent_step1(CV_L, CV_R, key, counter):
    input = CV_L || CV_R
    CV1 = compress(key, input, counter, PARENT_FLAG)
    return CV1
```

Второй этап – нелинейное смешивание исходных дочерних значений (приведено на псевдокоде):

```
nonlinear_mix(CV_L, CV_R):
    mix = array[8]
    for i = 0 to 7:
        L = CV_L[i]
        R = CV_R[i]

        M = (L * R) mod 2^32
        M = M XOR ROTR(L, 13) XOR ROTL(R, 17)
        M = M + MIX_CONST[i]

        mix[i] = M
    return mix
```


Здесь MIX_CONST[i] – массив предвычисленных констант. Рассмотрим детальнее каждую компоненту.

Мультипликативное смешивание $M = (L \cdot R) \bmod 2^{32}$ создаёт квадратичную нелинейность. Результат зависит от произведения двух входных слов, что нарушает XOR-линейность. Для двух пар входов (L_1, R_1) и (L_2, R_2) рассмотрим их XOR-комбинацию:

$$(L_1 \oplus \Delta L) \cdot (R_1 \oplus \Delta R) = L_1 \cdot R_1 \oplus L_1 \cdot \Delta R \oplus R_1 \cdot \Delta L \oplus \Delta L \cdot \Delta R, \tag{22}$$

где $\Delta L = L_1 \oplus L_2$ и $\Delta R = R_1 \oplus R_2$. Раскрытие произведения даёт перекрёстные члены $L_1 \cdot \Delta R$ и $R_1 \cdot \Delta L$, которые не могут быть представлены линейной комбинацией исходных разностей. Это принципиально нарушает свойство:

$$(L_1 \oplus L_2) \cdot (R_1 \oplus R_2) \neq (L_1 \cdot R_1) \oplus (L_2 \cdot R_2) \qquad (\mathrm{mod} 2^{32}), \tag{23}$$

необходимое для эффективной работы k-XOR алгоритмов. Смешивание на основе циклических вращений добавляет диффузию битов:

$$M \leftarrow M \oplus \mathrm{ROTR}(L, 13) \oplus \mathrm{ROTL}(R, 17). \tag{24}$$

Константы 13 и 17 выбраны взаимно простыми с 32 для обеспечения хорошего рассеяния. Комбинация сдвигов в противоположных направлениях создаёт сложные битовые зависимости, предотвращающие локализацию влияния отдельных битов входа.

Аддитивная компонента M ← M + MIX_CONST[i] вносит позиционную специфичность и предотвращает вырожденные случаи нулевых входов, где мультипликативное смешивание давало бы нулевой результат.

Третий этап – вторая компрессия смешанного значения (приведено на псевдокоде):

```
parent_step2(mix, key, counter):
    input = mix || mix
    CV2 = compress(key, input, counter XOR 0xAAAAAAAA, PARENT_FLAG)
    return CV2
```

Дублирование функции mix в двух параллельных ветвях заполняет полный 512-битный входной фрагмент. Модификация счётчика $\mathrm{counter} \oplus 0\mathrm{xAAAAAAAA}$ гарантирует независимость контекста от первой компрессии.

Финальное значение формируется как XOR результатов обоих ветвей (приведено на псевдокоде):

```
parent_final(CV_L, CV_R, key, counter):
    CV1 = parent_step1(CV_L, CV_R, key, counter)
    mix = nonlinear_mix(CV_L, CV_R)
    CV2 = parent_step2(mix, key, counter)

    for i = 0 to 7:
        CV_parent[i] = CV1[i] XOR CV2[i]

    return CV_parent
```

С точки зрения производительности нелинейное дерево создаёт дополнительные накладные расходы по времени только на родительских узлах. В типичном сценарии хэширования длинного сообщения количество листовых узлов дерева, а именно узлов, соответствующих фрагментам исходного сообщения, на порядки превосходит количество родительских. Для сообщения из N фрагментов требуется примерно N вычислений на листовом уровне дерева и $log_2(N)$ уровней родительских узлов. Суммарное число родительских узлов составляет примерно N, но поскольку листовые узлы вносят основной вклад во время выполнения, относительный вклад родительских узлов в общее время составляет приблизительно $10 - 20$ процентов.

Каждый родительский узел в BLAKE-PQ требует две компрессии вместо одной плюс операции нелинейного смешивания. Если стандартная компрессия стоит C тактов, где $C \approx 230$, дополнительные накладные расходы по времени родительского узла составляет:

$$\text{cost}_{\text{parent}} = 2C + 8 \times \text{cost}_{\text{mul}} + 16 \times \text{cost}_{\text{rot}} + 16 \times \text{cost}_{\text{xor}} + 8 \times \text{cost}_{\text{add}}, \quad (25)$$

где $cost_{mul} \approx$ 3 такта, $cost_{rot} \approx$ 1 такт, $cost_{xor} \approx$ 1 такт, $cost_{add} \approx$ 1 такт. Суммарно:

$$\text{cost}_{\text{parent}} \approx 2 \times 230 + 24 + 16 + 16 + 8 \approx 524 \text{ тактов}. \quad (26)$$

Это замедление примерно в 2.3 раза для родительских узлов. В пересчёте на полное дерево с учётом преобладания временных затрат листовых узлов общее замедление составляет:

$$\text{overhead}_{\text{total}} \approx 0.15 \times (2.3 - 1) \times 100\% \approx 20\%, \quad (27)$$

что находится в приемлемых пределах.

Теоретическое обоснование защиты от k-XOR атак базируется на нарушении линейной структуры задачи. Классическая и квантовая эффективность k-XOR алгоритмов критически зависит от наличия XOR-гомоморфизма или его аппроксимации:

$$f(x_1 \oplus x_2 \oplus \ldots \oplus x_k) \approx f(x_1) \oplus f(x_2) \oplus \ldots \oplus f(x_k), \quad (28)$$

После внедрения нелинейного смешивания это свойство нарушается уже на этапе формирования входа для компрессии. Рассмотрим две пары узлов $\left(CV_L^{(1)}, CV_R^{(1)}\right)$ и $\left(CV_L^{(2)}, CV_R^{(2)}\right)$. Для иллюстрации достаточно рассмотреть только мультипликативный компонент функции mix. Полное выражение содержит дополнительные члены, которые лишь увеличивают степень нелинейности. Для фиксированного индекса i нелинейное смешивание задаётся выражениями

$$\text{mix}_i^{(1)} = \left(CV_{L,i}^{(1)} \cdot CV_{R,i}^{(1)}\right) \oplus \text{ROTR}\left(CV_{L,i}^{(1)}, 13\right) \oplus \text{ROTL}\left(CV_{R,i}^{(1)}, 17\right) \quad (29)$$

и

$$\text{mix}_i^{(2)} = (CV_{L,i}^{(2)} \cdot CV_{R,i}^{(2)}) \oplus \text{ROTR}(CV_{L,i}^{(2)}, 13) \oplus \text{ROTL}(CV_{R,i}^{(2)}, 17). \quad (30)$$

Для XOR этих пар узлов получаем:

$$\text{mix}\left(CV_L^{(1)} \oplus CV_L^{(2)}, CV_R^{(1)} \oplus CV_R^{(2)}\right) \neq \text{mix}^{(1)} \oplus \text{mix}^{(2)}. \quad (31)$$

Это неравенство следует из мультипликативного расширения:

$$\begin{gathered}(CV_L^{(1)} \oplus CV_L^{(2)}) \cdot (CV_R^{(1)} \oplus CV_R^{(2)}) = \\ CV_L^{(1)} \cdot CV_R^{(1)} \oplus CV_L^{(1)} \cdot CV_R^{(2)} \oplus CV_L^{(2)} \cdot CV_R^{(1)} \oplus CV_L^{(2)} \cdot CV_R^{(2)},\end{gathered} \quad (32)$$

в то время как XOR отдельных смешиваний даёт:

$$\left(CV_L^{(1)} \cdot CV_R^{(1)}\right) \oplus \left(CV_L^{(2)} \cdot CV_R^{(2)}\right). \quad (33)$$

XOR двух путей в BLAKE3 приводит к взаимной компенсации перекрёстных членов $CV_L^1 \cdot CV_R^2$ и $CV_L^2 \cdot CV_R^1$, создавая линейную структуру выходного значения относительно входных состояний. Именно эта линейность относительно XOR делает древовидный режим уязвимым для квантовых k-XOR алгоритмов. Мультипликативная модификация разрушает эту компенсацию, делая выход нелинейным по отношению к XOR входных значений. Дополнительно двухэтапная схема с отдельными компрессиями создаёт структуру, где финальное цепочечное значение:

$$\text{CV}_{\text{parent}} = \text{CV}_1 \oplus \text{CV}_2. \quad (34)$$

представляет собой XOR двух отдельно вычисленных путей. Даже если одна из ветвей, как например CV_1 от прямой конкатенации или CV_2 от нелинейного смешивания, допускает структурную атаку, противник должен одновременно контролировать обе компоненты состояния. Формально, для успешной коллизии или прообраза необходимо выполнение:

$$\left(\text{CV}_1^{(a)} \oplus \text{CV}_2^{(a)}\right) = \left(\text{CV}_1^{(b)} \oplus \text{CV}_2^{(b)}\right). \quad (35)$$

что эквивалентно:

$$\text{CV}_1^{(a)} \oplus \text{CV}_1^{(b)} = \text{CV}_2^{(a)} \oplus \text{CV}_2^{(b)}. \quad (36)$$

Это требует согласованной атаки на оба пути одновременно, что существенно повышает сложность по сравнению с атакой на один из путей.

Альтернативные конструкции также рассматривались, включая применение только нелинейного смешивания без второй компрессии или использование более простых нелинейных функций. Однако упрощённые варианты не дают преимущества обработки в двух параллельных ветвях и могут быть уязвимы к специализированным атакам. Выбранная двухкомпрессионная схема с мультипликативным слоем, последующим вращательным смешиванием и финальным XOR-объединением нарушает исходную XOR-линейную структуру объединения родительских узлов при умеренном увеличении вычислительных затрат.

6. Заключение

В первой части настоящей работы была представлена концепция квантово-устойчивой хэш-функции BLAKE-PQ на основе архитектуры BLAKE3 и проведён детальный анализ существующих квантовых атак на криптографические примитивы, включая алгоритмы BHT, k-XOR атаки и методы криптоанализа на основе задачи скрытого сдвига. Рассмотрены базовые модификации раундовой функции через введение нелинейных преобразований, предложен метод усиления структуры посредством встраивания S-блоков на основе S-box AES, разработан перестановочный слой на базе последовательности транспозиций для противодействия HSP-атакам, а также представлен нелинейный смешивающий состояния слой. Во второй части будут рассмотрены оставшиеся три модификации, проведён комплексный анализ вычислительных затрат всех введённых модификаций и выполнена оценка общей криптографической стойкости цельной конструкции BLAKE-PQ.

Список литературы / References

[1]. Grover L.K. A fast quantum mechanical algorithm for database search. Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (STOC '96). New York: ACM, 1996, pp. 212-219.

[2]. Barker E., Roginsky A. Recommendation for applications using approved hash algorithms (NIST Special Publication 800 107 Revision 1). Gaithersburg, MD: National Institute of Standards and Technology, 2012. [Отозван в 2022 г.]

[3]. Grassi L., Naya-Plasencia M., Schrottenloher A. Quantum algorithms for the k xor problem // Advances in Cryptology – ASIACRYPT 2018. Cham: Springer, 2018, pp. 527-559.

[4]. Eichlseder M., Grassi L., Lüftenegger R., Øygarden M., Rechberger C., Schofnegger M., Wang Q. An algebraic attack on ciphers with low-degree round functions: application to full MiMC. Advances in Cryptology – ASIACRYPT 2020. – Cham: Springer, 2020. – P. 477–506.

[5]. Grassi L., Khovratovich D., Schofnegger M. Poseidon2: A faster version of the Poseidon hash function // Progress in Cryptology – AFRICACRYPT 2023. – Cham: Springer, 2023, pp. 177-203.

[6]. Bonnetain X. Hidden Structures and Quantum Cryptanalysis. PhD thesis. Paris: Université Paris-Saclay, 2019.

[7]. Bonnetain X., Naya-Plasencia M. Hidden shift quantum cryptanalysis and implications. Advances in Cryptology – ASIACRYPT 2018. Cham: Springer, 2018, pp. 560-592.

[8]. Bonnetain X., Hosoyamada A., Naya-Plasencia M., Sasaki Y., Schrottenloher A. Quantum attacks without superposition queries: the offline Simon's algorithm. Advances in Cryptology – ASIACRYPT 2019. Cham: Springer, 2019, pp. 552-583.
[9]. Leurent G., Peyrin T. Construction of differential characteristics in ARX designs – application to Skein. Advances in Cryptology – CRYPTO 2013. Berlin: Springer, 2013, pp. 524-541.
[10]. Leurent G., Peyrin T. From collisions to chosen prefix collisions – Application to full SHA 1. 29th USENIX Security Symposium. – USENIX Association, 2020, pp. 1939-1956.
[11]. Petrenko A., Petrenko S., Gnidko K. et al. Application of quantum fourier transform in cryptology. Journal of Computer Virology and Hacking Techniques, 2025, vol. 21, no. 1. DOI: 10.1007/s11416-025-00563-3. EDN DFZWUC.
[12]. Schrottenloher A. Quantum algorithms for cryptanalysis and quantum-safe symmetric cryptography. PhD thesis. Paris: Université Paris Saclay, 2021.
[13]. Kaplan M. Quantum attacks against iterated block ciphers. In QCRYPT 2016, poster abstract. 2016.
[14]. Kuwakado H., Morii M. Quantum distinguisher between the 3 round Feistel cipher and the random permutation. 2010 IEEE International Symposium on Information Theory (ISIT). IEEE, 2010, pp. 2682-2685.
[15]. Kuwakado H., Morii M. Security on the quantum-type Even-Mansour cipher. 2012 International Symposium on Information Theory and its Applications (ISITA 2012). IEEE/IEICE, 2012, pp. 312-316.
[16]. Kaplan M., Leurent G., Leverrier A., Naya-Plasencia M. Breaking symmetric cryptosystems using quantum period finding. Advances in Cryptology – CRYPTO 2016. Berlin: Springer, 2016, pp. 213-241.
[17]. Schrottenloher A. New algorithms for quantum (symmetric) cryptanalysis. Mathematical Cryptology, 2021, vol. 1, no. 1, pp. 1-18.
[18]. Lyubashevsky V., Micciancio D., Peikert C., Rosen A. SWIFFT: A Modest Proposal for FFT Hashing. Fast Software Encryption – FSE 2008. Lecture Notes in Computer Science. – Berlin, Heidelberg: Springer, 2008, vol. 5086, pp. 54-72. DOI: 10.1007/978-3-540-71039-4_4.
[19]. Arbitman Y., Dogon G., Lyubashevsky V., Micciancio D., Peikert C., Rosen A. SWIFFTX: A Proposal for the SHA-3 Standard. – Submission to NIST SHA-3 Competition, 2008.
[20]. Bernstein D.J., Lange T., Peters C., Schwabe P. Really Fast Syndrome-Based Hashing. Progress in Cryptology – AFRICACRYPT 2011. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, vol. 6737, pp. 134-152. DOI: 10.1007/978-3-642-21969-6_9.
[21]. Шокуров А.В., Кузюрин Н.Н., Фомин С.А. Решетки, алгоритмы и современная криптография: курс лекций. 2008. 127 с. Доступно по ссылке: https://www.ispras.ru/courses/book-lattice-cryptography.pdf, дата обращения: 08.08.2026.
[22]. Charles D.X., Goren E.Z., Lauter K.E. Cryptographic Hash Functions from Expander Graphs. Journal of Cryptology, 2009, vol. 22, no. 1, pp. 93-113. DOI: 10.1007/s00145-007-9002-x.
[23]. Kölbl S., Lauridsen M.M., Mendel F., Rechberger C. Haraka v2 – Efficient Short-Input Hashing for Post-Quantum Applications. IACR Transactions on Symmetric Cryptology, 2016, vol. 2016, no. 2, pp. 1-29. DOI: 10.46586/tosc.v2016.i2.1-29.
[24]. Aumasson J.-P., Neves S., O'Connor J., Wilcox Z. The BLAKE3 Hashing Framework // Internet-Draft draft-aumasson-blake3-00. Internet-Draft draft-aumasson-blake3-00. – Internet Engineering Task Force, 2024. Available at: https://www.ietf.org/archive/id/draft-aumasson-blake3-00.html, accessed 08.03.2026.
[25]. Petrenko A. Applied Quantum Cryptanalysis. Aalborg: River Publishers, 2023. 222 p. DOI: 10.1201/9781003392873.
[26]. Petrenko S., Petrenko A., Kazak A., Makoveichuk K., Oleinikov N. Directions for enhancing quantum resilience of blockchain platforms and ecosystems. International Journal of Information Management Data Insights, 2026, vol. 6, no. 1. Art. 100390. DOI: 10.1016/j.ijimei.2026.100390.
[27]. Ishchukova E., Petrenko S., Petrenko A., Gnidko K., Nekrasov A. Potential Vulnerabilities of Cryptographic Primitives in Modern Blockchain Platforms. Sci. 2025, vol. 7, no. 3, pp. 112. DOI: 10.3390/sci7030112.
[28]. Petrenko A.S., Petrenko S.A., Taran V.N. Universal Quantum Gate as a Tool for Modeling Quantum Cryptanalysis Algorithms on a Quantum Circuit. CEUR Workshop Proceedings, 2022, vol. 3094, pp. 143-150.
[29]. Skiba V.Yu., Petrenko S.A., Gnidko K.O., Petrenko A.S. Concept of ensuring the resilience of operation of national digital platforms and blockchain ecosystems under the new quantum threat to security. Computing, Telecommunications and Control, 2025, vol. 18, no. 2, pp. 56-73. DOI: 10.18721/JCSTCS.18205.

[30]. Petrenko A.S., Petrenko S.A., Makoveichuk K.A., Olifirov A.V., Krachunov H. Security threat model based on analysis of foreign national quantum programs // CEUR Workshop Proceedings, 2021, vol. 3057, pp. 11-25. Available at: https://ceur-ws.org/Vol-3057/paper2.pdf, accessed 08.03.2026.
[31]. Dinh K.L., Do T.B., Moldovyan N.A., Petrenko A.S. Typical Algebraic Signature Schemes with Two Hidden Groups // Future Data and Security Engineering. FDSE 2025: Proceedings of the 11th International Conference, edited by T.K. Dang, J. Küng, T.M. Chung. Singapore: Springer, 2026. Communications in Computer and Information Science, vol. 2709, pp. 7-27. DOI: 10.1007/978-981-95-4724-1_7.
[32]. Петренко А.С. Квантово-устойчивый консенсус с адаптивной иерархической архитектурой для национальной блокчейн-системы. Известия РАН. Серия физическая, 2026, т. 90, № 7.
[33]. Ji L., Liangyu W. Preimage attacks against 2.5-round BLAKE-256 and BLAKE-512. Proceedings of the 4th International Conference on Network and System Security (NSS 2010). IEEE, 2010, pp. 259-266.
[34]. Aumasson J.-P., Neves S., Wilcox-O'Hearn Z., Winnerlein C. BLAKE2: simpler, smaller, fast as MD5. Applied Cryptography and Network Security – ACNS 2013. Berlin: Springer, 2013, pp. 119-135.
[35]. BLAKE3: a faster and more secure hash function: Official implementation and benchmarks. BLAKE3 team. Available at: https://github.com/BLAKE3-team/BLAKE3, accessed 08.03.2026.
[36]. Kerkour A. Choosing a hash function for 2030 and beyond: SHA 2 vs SHA 3 vs BLAKE3. Kerkour Security, 2024. Available at: https://kerkour.com/fast-secure-hash-function-sha256-sha512-sha3-blake3, accessed 08.03.2026.
[37]. Bonnetain X., Naya-Plasencia M., Schrottenloher A. Quantum algorithms for attacking hardness assumptions in classical and post-quantum cryptography. IET Information Security, 2022. DOI: 10.1049/ise2.12081.
[38]. Childs A.M., van Dam W. Quantum algorithm for a generalized hidden shift problem. Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007). SIAM, 2007, pp. 1225-1234.
[39]. Khovratovich D., Naya-Plasencia M., Röck A., Schlünzen N. Cryptanalysis of low algebraic degree hash functions and applications to Skein, Hamsi, and SIMD family of hash functions. Advances in Cryptology – EUROCRYPT 2013. Cham: Springer, 2013, pp. 1-22.
[40]. ГОСТ Р 34.12–2015. Информационная технология. Криптографическая защита информации. Блочные шифры. – Введ. 01.01.2016. М.: Стандартинформ, 2015. 16 с.
[41]. Cox D., Little J., O'Shea D. Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra. 4th ed. Cham: Springer, 2015. 646 p.
[42]. Nielsen M.A., Chuang I.L. Quantum Computation and Quantum Information. 10th anniversary ed. Cambridge: Cambridge University Press, 2010. 710 p.
[43]. Fog A. The microarchitecture of Intel, AMD and VIA CPUs: An optimization guide for assembly programmers and compiler makers. Copenhagen: Agner Fog, 2025. Available at: https://www.agner.org/optimize/microarchitecture.pdf, accessed 08.03.2026.

Информация об авторе / Information about author

Алексей Сергеевич ПЕТРЕНКО – младший научный сотрудник Федерального государственного бюджетного учреждения науки «Санкт Петербургский Федеральный исследовательский центр Российской академии наук» с 2023 года. Сфера научных интересов: информационная безопасность, криптография, постквантовая криптография, квантовые вычисления, блокчейн.

Alexey Sergeevich PETRENKO – junior research assistant of SPb FRC RAS since 2023. Research interests: information security, cryptography, post-quantum cryptography, quantum computing, blockchain.