МАТЕМАТИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ

УДК 004.056 + 004.051 + 004.414.22 + 004.67 DOI 10.17223/20710410/60/4

СПОСОБЫ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ КРАТКИХ НЕИНТЕРАКТИВНЫХ АРГУМЕНТОВ С НУЛЕВЫМ РАЗГЛАШЕНИЕМ И АНАЛИЗ ДОСТИГНУТЫХ РЕЗУЛЬТАТОВ

И. В. Мартыненков

АО «КВАНТ-ТЕЛЕКОМ», г. Москва, Россия

E-mail: mivpost@yandex.ru

Рассматриваются способы повышения производительности кратких неинтерактивных аргументов с нулевым разглашением на основе полиномиальных наборов с использованием различных вычислительных методов. Проводится сравнительный анализ протоколов по размерам главных ссылочных строк и доказательств достоверности вычислений, затратам формирования доказательств и их верификации.

Ключевые слова: *краткие неинтерактивные аргументы, повышение производительности, анализ производительности, размер публичных параметров.*

WAYS TO IMPROVE THE PERFORMANCE OF ZERO-KNOWLEDGE SUCCINCT NON-INTERACTIVEARGUMENTS OF KNOWLEDGE AND THE ANALYSIS OF THE RUSULTS ACHIEVED

I. V. Martynenkov

JSC «KVANT-TELECOM», Moscow, Russia

We consider ways to improve the performance of zero-knowledge succinct non-interactive argument of knowledge (zk-SNARK) based on polynomial sets, such as quadratic arithmetic programs (QAP), square arithmetic programs (SAP), quadratic span programs (QSP), square span programs (SSP), quadratic polynomial programs (QPP), etc. To improve the performance of zk-SNARK, batch data processing methods, various modifications of exponentiation problems, bilinear pairings based on elliptic curves, etc. are used. A comparative analysis of the complexity of the common reference strings formation, the construction and verification of the calculations reliability proofs, as well as the sizes of common reference strings and proofs has been carried out.

Keywords: *succinct non-interactive arguments, performance improvement, performance analysis, size of public parameters.*

Введение

Рассматриваются способы повышения производительности кратких неинтерактивных аргументов с нулевым разглашением. Далее используется наименование «прото-

кол zk-SNARK» согласно [1]. Представлено применение ключевых/бесключевых хеш-функций, позволяющих выполнять верификацию доказательств с затратами, независимыми от размера ввода/вывода, но с издержками на дайджесты. Быстродействие может повышаться за счёт более адаптированных к протоколам zk-SNARK хеш-функций над простым полем [2, 3] или на решётках [4–6], как представлено в [7]. Появляется компромисс между сложностью вычислений доказывающего и верификатора, владеющего секретными значениями, определяемый хешированием части открытого входа.

Для деления полиномов может использоваться многоточечная оценка и интерполяция [8, 9], а для умножения — множественное возведение в степень [10, 11] на основе таблиц промежуточных степеней и др. [11–14]. Структурные свойства набора полиномов в виде квадратичных арифметических программ (Quadratic Arithmetic Program, QAP) [15] предполагают наличие нулевых многочленов, позволяющих сократить размер ключей [14]. Отдельного внимания заслуживают сокращение размера доказательства и асимметричное билинейное спаривание [16] с компонентами доказательств из разных групп [17].

Приводятся примеры разделения публичных параметров главной ссылочной строки (Common Reference String, CRS) на более компактные части, формируемые отдельно для доказывающих и верификаторов протоколов zk-SNARK [15, 17, 18]. На примере протоколов zk-SNARK из [19] представлена пакетная верификация доказательств, являющаяся более эффективной компоновкой частей доказательств для уменьшения сложности вычислений. Отмечены работы, использующие объединение набора доказательств протоколов zk-SNARK в одно заключительное доказательство.

Представлен сравнительный анализ протоколов zk-SNARK [7–9, 12–15, 17, 18, 20–36] по размерам CRS и доказательств достоверности вычислений, а также по сложностям вычислений доказывающих и верификаторов.

1. Способы применения хеш-функций

Верифицируемые вычисления (Verifiable Computation, VC) [20] предназначены для двустороннего вычисления функций по частным входным данным. Согласно [37], в схемах VC независимо от длины ввода $f(x)$ вывод доказывающего может дополнительно сокращаться. Для этого вместо $f(x)$ используются ключевые функции $\mathsf{MAC}_k(f(x))$. В результате верификация становится независимой от длины ввода/вывода, однако возникают издержки верификатора на вычисление хеш-функции. На стороне верификатора для подготовки входных данных асимметричные криптографические преобразования остаются прежними.

Вычисления верификатора протоколов zk-SNARK [15, 38] линейны по размеру открытого входа $\mathbf{u}$ с основными затратами на вычисление $g^{v_{\text{in}}(s)}$, где g — порождающий элемент группы; $v_{\text{in}}(s) = \sum\limits_{k \in I_{\text{in}}} a_k v_k(x)$ — сумма многочленов из состава полиномиального набора QAP или программ квадратичного диапазона (Quadratic Span Programs, QSP) [15]; s — секретная точка. Вычисление сокращается за счёт применения хеш-функции H для получения постоянного по размеру $\mathbf{u}' = \mathsf{H}(\mathbf{u})$. Значение $\mathbf{u}'$ становится новым открытым входом, а $\mathbf{u}$ — частью секрета $\mathbf{w}$, знание которого доказывается. Однако хеширование увеличивает объём вычислений доказывающего, так как впоследствии дополнительно необходимо верифицировать $\mathbf{u}' = \mathsf{H}(\mathbf{u})$. Компромисс между вычислениями доказывающего и верификатора определяется хешированием части входных данных $\mathbf{u}$.

В случае протокола zk-SNARK с фиксированным верификатором (Designated Verifier, DV) [15, 38] верификатор владеет секретами $\alpha, \beta_v, \beta_w \in \mathbb{Z}_p^*$, по которым с использованием $g^{v_{\text{mid}}(s)}, g^{w(s)}, g^{h(s)}$ он способен сформировать часть доказательства. Значения $v_{\text{mid}}(s), w(s), h(s)$ также основаны на полиномиальных наборах QAP/QSP [15]. Поэтому доказывающий может применить более лаконичные доказательства за счёт хеша $\mathbf{D} = \mathsf{H}(g^{\alpha v_{\text{mid}}(s)}, g^{\alpha w(s)}, g^{\alpha h(s)}, g^{\beta_v v_{\text{mid}}(s) + \beta_w w(s)})$ и отправить $\boldsymbol{\pi} = (\boldsymbol{g^{v_{\text{mid}}(s)}, g^{w(s)}, g^{h(s)}, \mathbf{D}})$. Затем верификатор выводит аналогичные недостающие значения, проверяет совпадение с $\mathbf{D}$ и проводит верификацию доказательства обычным способом. Для протокола DV zk-SNARK [39, 40] хеш-функции назначаются верификаторами на этапе формирования публичных параметров в виде CRS.

Кроме того, хеш-функции над большим простым полем, например «POSEIDON» [2] или хеш Педерсена [3], используемые для доказательства знания прообраза, лучше адаптированы к протоколам zk-SNARK по сравнению с классическими схемами, за счёт чего повышают их быстродействие. Вариант [3] отображает последовательность бит в сжатую точку на эллиптической кривой. В [13] вычисляется хэш Меркла с использованием SHA-1, в то время как в [7] хеш-функция основана на решётке

$$\mathsf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{L}\mathbf{x} + \mathbf{R}\mathbf{y} \bmod q, \tag{1}$$

где $\mathbf{L}, \mathbf{R} \in \mathbb{Z}_q^{k \times m}$; $\mathbf{x}, \mathbf{y} \in \{1, \ldots, N\}^m$ [41]. В связи со свойством аддитивного гомоморфизма данной хеш-функции дайджест $\mathsf{d}(\mathbf{S})$ полученного хеш-дерева Меркла может выражаться в виде

$$\mathsf{d}(\mathbf{S}) = \sum_i \mathbf{S}[i] f(i) \bmod q, \tag{2}$$

где $f(i) \in \{0,1\}^m$ — функция индекса i как «частичная метка» [42]; $\mathbf{S}[i]$ — вектор в позиции i как скаляр в $\mathbb{Z}_q$. Дайджесты векторов кратчайших путей [7] вычисляются с использованием (1) и (2), которые вместе с предварительно вычисленными расстояниями и путями обрабатываются HMAC, а затем передаются на вычисление серверу в недоверенной среде. Для всех векторов разной длины, хранящих предварительно вычисленные расстояния, строится доказательство протокола zk-SNARK [13]. Возможным недостатком подходов [7] по сравнению с [13] является увеличение размера доказательства. Размер доказательства схемы [13] составляет 288 байт. Доказательство [7] содержит 256-битный (32 байта) HMAC для каждого ребра графа с кратчайшим путём, поэтому размер доказательства пропорционален количеству таких рёбер $|R|$. В общем случае доказательство [7] составляет $|R| \times 32 + 288$ байт. Примеры криптографических алгоритмов на решётках представлены в [4–6].

Например, протокол zk-SNARK [43] строится на основе симметричных хеш-функций и схем обязательств, обеспечивая «постквантовую» защиту. В [44] представлена новая схема обязательств для полиномов над конечными полями, использующая целочисленные представления многочленов и групп неизвестного порядка (Diophantine Argument of Knowledge, «DARK»). В общем виде уравнение $P(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0$ с параметрами $a_1, \ldots, a_n$ и неизвестными $x_1, \ldots, x_m$ считается разрешимым при данных $a_1, \ldots, a_n$, если существуют наборы чисел $x_1, \ldots, x_m$, при которых равенство верно. В [44] представлено преобразование интерактивных доказательств в неинтерактивные на основе «DARK» для протокола zk-SNARK «Supersonic» с формированием CRS без секретных значений. Протокол zk-SNARK «Ligero» [45] также формирует CRS без секретных значений, использует произвольные хеш-функции, симметричные примитивы и преобразуется в неинтерактивный режим эвристикой Фиата — Шамира [46].

2. Способы выполнения алгебраических операций

В общем случае алгоритмы формирования ключей и доказательств протоколов zk-SNARK имеют высокие криптографические накладные расходы, так как выполняется значительное количество возведений в степень в группах $\mathbb{G}_1$ и $\mathbb{G}_2$.

Например, доказательство протокола zk-SNARK [15, 38] состоит из семи элементов группы. Выбираются случайные $\delta_{v_{\text{mid}}}, \delta_w \in \mathbb{F}$, за счёт которых «маскируются» $v'_{\text{mid}}(x) = v_{\text{mid}}(x) + \delta_{v_{\text{mid}}} t(x)$, $w'(x) = w(x) + \delta_w t(x)$, $h'(x) = v'_{\text{mid}}(x) w'(x)/t(x)$. Используются случайные $\alpha, \beta_v, \beta_w \in \mathbb{F}$. Доказательство принимает вид

$$\begin{gathered}\boldsymbol{\pi} = (\pi'_{v_{\text{mid}}}, \pi'_w, \pi'_h, \pi'_{v'_{\text{mid}}}, \pi'_{w'}, \pi'_{h'}, \pi'_y) = \\ = (g^{v'_{\text{mid}}(s)}, g^{w'(s)}, g^{h'(s)}, g^{\alpha v'_{\text{mid}}(s)}, g^{\alpha w'(s)}, g^{\alpha h'(s)}, g^{\beta_v v'_{\text{mid}}(s) + \beta_w w'(s)}).\end{gathered} \tag{3}$$

Согласно [15], в доказательстве (3) из-за структуры многочленов QAP/QSP значения $v_0(x) + \sum\limits_k a_k v_k(x)$ и $w_0(x) + \sum\limits_k b_k w_k(x)$ могут вычисляться в поле $\mathbb{F}$ за линейное число операций, исключая $h(x)$. С помощью многоточечной оценки и интерполяции [8, 9] частное $h(x) = \big(v_0(x) + \sum\limits_k a_k v_k(x)\big)\big(w_0(x) + \sum\limits_k b_k w_k(x)\big)/t(x)$ может вычисляться со сложностью $O(s \cdot \text{poly}(\log(s)))$ и квадратичным вычислением доказательства $\boldsymbol{\pi}$.

В [13] демонстрируются подходы повышения производительности, которые являются общими для протоколов zk-SNARK. Порождающий элемент группы g возводится в экспоненты, что ускоряется методом множественного возведения в степень [10, 11], основанным на построении таблиц промежуточных степеней. Таким образом, вычисление доказывающего вида $g^{v(s)} = \prod\limits_i g_i^{e_i}$ требует использования $g_1, \ldots, g_m \in \mathbb{G}$ и степеней $e_1, \ldots, e_m \in \mathbb{Z}$ для некоторого большого m. В данном случае применима техника построения небольшой таблицы степеней для каждой пары оснований. Например, для первых двух оснований с окном размера 1, используя одно умножение, вычисляются $\{g_1^0, g_2^0, g_1^0, g_2^1, g_1^1, g_2^0, g_1^1, g_2^1\}$. Затем рассматриваются старшие биты e_1 и e_2, указывающие на значение в таблице, которое умножается на «накопитель». Все пары оснований проходят аналогичную процедуру, после чего «накопитель» возводится в квадрат для перехода на следующий уровень. Такие таблицы могут увеличить производительность в 3–4 раза [13]. Для $g \in \mathbb{G}$ вычисление кортежей $(g^{a^i}, \ldots, g^{a^m})$ выполняется стандартной техникой предварительного вычисления таблиц степеней.

Для последующего использования верификаторам протоколов zk-SNARK целесообразно строить долговременные таблицы степеней порождающих элементов g^i и секретных точек s^j в фоновом режиме. Доказывающий вычисляет полином $h(x)$, который может храниться в виде оценки от корней QAP/QSP и др. Для этого выводится полином $p(x)$, а затем в открытом виде (без криптографических преобразований) для получения $h(x)$ выполняется деление полиномов, что ускоряется умножением на основе быстрого преобразования Фурье (БПФ) со сложностью $O(n \log n)$ и алгоритмом полиномиальной интерполяции [47] для построения двоичного дерева полиномов со сложностью $O(n \log^2 n)$. Для ускорения вычислений $h(x)$ доказывающий также может хранить не содержащие секретов таблицы возведения в степень и полиномиальное дерево, которые потенциально могут передаваться между разными доказывающими.

В протоколе zk-SNARK [14] используются структурные свойства QAP из арифметических схем, сокращающие размер ключа доказывающего, который формируется на основе порождающего элемента P группы $\mathbb{G}_1$ или $\mathbb{G}_2$ и скаляров $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$ для вычисления $\alpha_1 P, \ldots, \alpha_n P$. Арифметическая схема C с m проводами (Wires) и d вентилями (Gates) преобразуется в QAP размера m степени d. Далее строятся три

матрицы $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{F}^{(m+1)\times d}$, кодирующие топологию схемы C: j-е столбцы $\mathbf{A}/\mathbf{B}$ кодируются левыми/правыми входами j-го логического элемента, а j-е столбцы $\mathbf{C}$ — выходами логических элементов, $j \in \{1, \ldots, d\}$. Рассматривается $S \subset \mathbb{F}$ как множество размера d, $Z(z) = \prod_{\omega \in S}(z - \omega)$ и для $i \in \{1, \ldots, m\}$ полином A_i — расширение низкой степени i-й строки $\mathbf{A}$. Аналогично определяются B_i, C_i. Примечательно, что в [14] отмечается наличие нулевых строк матриц $\mathbf{A}, \mathbf{B}, \mathbf{C}$, соответствующих нулевым многочленам. Например, если i-й провод никогда не имеет ненулевого значения в качестве левого входа вентиля, то i-я строка $\mathbf{A}$ является нулевой, поэтому A_i — нулевой многочлен. Этот факт используется для сокращения размера ключа доказательства за счёт уменьшения количества ненулевых многочленов в матрицах [14], что снижает стоимость вычислений.

В протоколах zk-SNARK [12, 13] для вычисления доказательств $\boldsymbol{\pi}$ используются универсальные мультискалярные умножения. В протоколе zk-SNARK [14] доказывающий использует вектор $\boldsymbol{\alpha} \in \mathbb{F}^{d+1}$ в виде коэффициентов многочлена-частного $h(x)$ степени d или $\boldsymbol{\alpha} = (1, \mathbf{s}, \delta_1, \delta_2, \delta_3) \in \mathbb{F}^{4+m}$ для случайных $\delta_1, \delta_2, \delta_3 \in \mathbb{F}$. В первом случае $\boldsymbol{\alpha}$ представляется случайным, поэтому может использоваться алгоритм Бос-Костера [11] из-за меньших требований к памяти. Во втором случае $\mathbf{s}$ зависит от ввода схемы $C(\mathbf{x}, \mathbf{a})$ и формирователя арифметических схем C, где $\mathbf{x}$ — открытый вход; $\mathbf{a}$ — секретный вход, знание которого доказывается. Теперь $\boldsymbol{\alpha}$ может содержать $\mathbf{s}$ с дополнительными элементами. Секрет $\mathbf{s}$ рассматривается как список значений проводов схемы C при вычислении на $(\mathbf{x}, \mathbf{a})$. Разрядность проводов зависит от выбора переменной типа байт, слово и т. д., а производительность повышается предварительной группировкой элементов из $\boldsymbol{\alpha}$ по типам данных и применением метода [11].

Дополнительный вклад в быстродействие может внести асимметричное билинейное спаривание $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, где $\mathbb{G}_1$ — «базовая» группа точек эллиптической кривой [16], а $\mathbb{G}_2$ — группа точек эллиптической кривой «кручения» [16] и $\mathbb{G}_1 \neq \mathbb{G}_2$. Операции над «базовой» кривой выполняются быстрее. Если один из входов функции билинейного спаривания фиксирован, применяются методы предварительного вычисления [48–50], идейно схожие с пакетной верификацией [19]. Вычислительные методы, обеспечивающие производительное билинейное спаривание, также представлены в [51–60]. Например, доказывающий может вычислять $g_w^{w(s)}$ над кривой «кручения», а остальные значения — над «базовой» кривой. Такая асимметрия предполагает, что элемент поля может отображаться в виде значения первой, второй или обеих групп. Возможно использование сокращённого преобразования Тейта [61, 62] для эллиптической кривой E над $\mathbb{F}_q$ для простого числа q с группами $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, эквивалентными подгруппам в $(E(\mathbb{F}_q), E(\mathbb{F}_q^k), \mathbb{F}_{q^k}^*)$ соответственно. Эллиптические кривые, удобные для билинейного спаривания, имеют размер элементов группы $\mathbb{G}_1$ меньший, чем размер элементов группы $\mathbb{G}_2$ [63]. Поэтому в протоколе zk-SNARK [17] для повышения производительности $\mathbf{A}, \mathbf{C} \in \mathbb{G}_1$ и $\mathbf{B} \in \mathbb{G}_2$. Выбор групп позволяет гибко настраивать уровень защищённости и производительности операций.

Отдельно стоит обратить внимание на потенциальную возможность сокращения размера доказательств. Например, по сравнению с протоколом zk-SNARK [17] на основе QAP из арифметических схем с выходом в виде набора бит и доказательством из трёх элементов поля протокол zk-SNARK [18] на основе SSP из логических схем с выходом один бит имеет доказательство из двух элементов поля. Доказательство протокола zk-SNARK [17] также сокращается до двух элементов поля за счёт переписывания схемы с использованием только вентилей возведения в квадрат. Детальнее,

каждый элемент умножения $a \cdot b = c$ переписывается как $(a+b)^2 - (a-b)^2 = 4c$. Если используются только элементы возведения в квадрат, то полиномы QAP, соответствующие левым и правым входам логических элементов, равны $(u_i(x) = v_i(x)$ для всех $i)$. Зафиксировав значения секретных рандомизаторов $r = s \in \mathbb{Z}_p$, отвечающих за введение свойства нулевого разглашения, и используя компоненты ключа верификации $\beta, \alpha \in \mathbb{Z}_p^*$, формируем $\mathbf{B} = \mathbf{A} + \beta - \alpha$ [17]. Поэтому необходимо построить только два элемента доказательства $\mathbf{A}$ и $\mathbf{C}$ [17]. Однако описанный метод может удвоить количество вентилей и требует дополнительных проводов для вычитания квадратов. Поэтому уменьшение размера доказательства связано с дополнительными вычислительными затратами.

3. Разделение публичных параметров главной ссылочной строки

В протоколах zk-SNARK [15, 18] экономия ресурсов достигается за счёт предварительного вычисления наборов $\{g_1^{v_i(s)}\}_{i>l_u}$, $g_1^{t(s)}$ и $\{g_1^{v_i(s)}\}_{i=0}^{m}$, $g_2^{t(s)}$, где $v_i(x), t(x)$ — многочлены QAP/QSP; s — секретная точка; l_u — разрядность открытого входа; l_w — разрядность секретного входа. Для оценки многочлена-частного $h(x)$ в m-корнях $r_1, \ldots, r_m$ целевого полинома $t(x) = \prod\limits_{i=1}^{m}(x - r_i)$ доказывающий может применять дискретное преобразование Фурье, как представлено в [18]. В результате повышение производительности протокола zk-SNARK [18] достигается модификацией CRS доказывающего:

$$\boldsymbol{\sigma}_P = \Big(r, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, \{g_1^{v_i(s)}\}_{i>l_u}, \{g_2^{v_i(s)}\}_{i>l_u}, \{g_1^{\beta v_i(s)}\}_{i>l_u},$$
$$g_1^{\beta t(s)}, g'_2, g'^{\beta}_2, \gamma, \{t(r'_j)^{-1}\}_{j=1}^{d}, \{g_1^{l'_j(s)}\}_{j=1}^{d}, g_1^{t(s)}, Q\Big).$$

Верификатор протокола zk-SNARK [18] вычисляет $V = g_1^{v_0(s)+\sum\limits_{i=1}^{l_u} a_i v_i(s)} V_w$ и проводит верификацию, при этом полная CRS заменяется компактным вариантом с $l_u + 6$ элементами группы:

$$\boldsymbol{\sigma}_V = (r, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, \{g_1^{v_i(s)}\}_{i=0}^{l_u}, g_2, g_2^{t(s)}, g'_2, g'^{\beta}_2).$$

Если CRS формируется верификатором, то за счёт знания применяемых секретов, например β и τ в [18], сложность верификации дополнительно сокращается с помощью соответствующих предварительных вычислений.

Другим примером является протокол zk-SNARK [17], в котором если каждый вентиль соединён с постоянным числом проводов, то набор QAP является разреженным, а сложность вычисления линейной — $O(n)$, где $n = \deg(t(x))$, или зависит от параметра безопасности. При этом для $q = 1, \ldots, n$ выполняются следующие оценки полиномов:

$$\sum_{i=0}^{m} a_i u_i(r_q) = \sum_{i=0}^{m} a_i u_{i,q}, \quad \sum_{i=0}^{m} a_i v_i(r_q) = \sum_{i=0}^{m} a_i v_{i,q}, \quad \sum_{i=0}^{m} a_i w_i(r_q) = \sum_{i=0}^{m} a_i w_{i,q}. \tag{4}$$

Если $r_1, \ldots, r_m$ — корни $t(x)$ при подходящем простом p, то $h(x)$ вычисляется с использованием БПФ в $\mathbb{Z}_p$ за $O(n \log n)$ операций. Вычисление коэффициентов в (4) возможно за счёт БПФ, на основе которых доказывающий выполняет $m + 3n - l_u + 3$ возведений в степень в $\mathbb{G}_1$, $n+1$ — в $\mathbb{G}_2$. С увеличением параметра (уровня) безопасности порядок групп растёт и возведение в степень становится трудозатратнее. В таком случае CRS [17]

$$\boldsymbol{\sigma}_1 = \begin{pmatrix} \alpha, \beta, \delta, \{x^i\}_{i=0}^{n-1}, \{(\beta u_i(x) + \alpha v_i(x) + w_i(x))/\gamma\}_{i=0}^{l_u}, \\ \{(\beta u_i(x) + \alpha v_i(x) + w_i(x))/\delta\}_{i=l_u+1}^{m}, \{x^i t(x)/\delta\}_{i=0}^{n-2} \end{pmatrix}, \boldsymbol{\sigma}_2 = \big(\beta, \gamma, \delta, \{x^i\}_{i=0}^{n-1}\big) \tag{5}$$

расширяется такими элементами $[u_i(x)]_1, [v_i(x)]_1, [w_i(x)]_2$ для $i \in \{1, \ldots, m\}$ при $[z_i(x)]_j = g_j^{z_i(x)}$, что $\mathbf{A}$ и $\mathbf{B}$ доказательства $\boldsymbol{\pi}$ [17], а именно

$$\begin{gathered}\mathbf{A} = \alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta, \quad \mathbf{B} = \beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta, \\ \mathbf{C} = \left(\sum_{i=l_u+1}^{m} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)) + h(x)t(x) \right) / \delta + \mathbf{A}s + \mathbf{B}r - rs\delta,\end{gathered} \tag{6}$$

становится возможным строить без вычисления коэффициентов $\sum\limits_{i=0}^{m} a_i u_i(x)$ и $\sum\limits_{i=0}^{m} a_i v_i(x)$ и последующего возведения в степень. В результате в случае QSP/SSP коэффициенты $a_i \in \{0, 1\}$ расширяют CRS и доказывающий может выполнить m умножений для каждой компоненты $\mathbf{A}$ и $\mathbf{B}$. Модифицированная CRS, предполагающая более быстрое вычисление доказательств, формируется из исходной CRS, поэтому свойства защищённости протокола не нарушаются [17].

Кроме того, верификатору протокола zk-SNARK [17] не требуется полная CRS (5). Достаточно знать $l_u + 2$ элементов $\mathbb{G}_1$, три элемента $\mathbb{G}_2$ и один элемент $\mathbb{G}_T$. В данном случае формирование ключей преобразуется к виду $(\sigma_P, \sigma_V, \tau) \leftarrow \mathsf{Setup}(R)$ с компонентом

$$\sigma_V = \left(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, \left\{ \left[\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma} \right]_1 \right\}_{i=0}^{l_u}, [1]_2, [\gamma]_2, [\delta]_2, [\alpha\beta]_T \right). \tag{7}$$

Верификация подтверждает, что доказательство (6) состоит из трёх корректных элементов $\mathbf{A}, \mathbf{B}, \mathbf{C}$:

$$[\mathbf{A}]_1 \cdot [\mathbf{B}]_2 = [\alpha]_1 \cdot [\beta]_2 + \sum_{i=0}^{l_u} a_i \left[\tfrac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma} \right]_1 \cdot [\gamma]_2 + [\mathbf{C}]_1 \cdot [\delta]_2.$$

Верификатор вычисляет l_u возведений в степень в $\mathbb{G}_1$, ряд умножений и три билинейных спаривания, где предполагается, что $[\alpha\beta]_T = [\alpha]_1 \cdot [\beta]_2$ предварительно вычисляются в σ_V (7).

4. Пакетная верификация доказательств

Пакетная верификация ускоряет работу протоколов zk-SNARK. Например, для всех протоколов zk-SNARK [19] вместо проверки P равенств вида $X_i = Y_i$ для $i \in [1, \ldots, P]$ верификатор может сгенерировать случайные $\tau_i \in \mathbb{Z}_p$ для $i \in [1, \ldots, P-1]$, а затем проверить равенство $\prod\limits_{i=1}^{P-1} X_i^{\tau_i} \cdot X_P = \prod\limits_{i=1}^{P-1} Y_i^{\tau_i} \cdot Y_P$. Если X_i и Y_i имеют подходящую структуру, то пакетная верификация экономит ресурсы. Например, если $X_i = e(U_i, V)$, где V не зависит от i, то при вычислении $\prod\limits_{i=1}^{P-1} X_i^{\tau_i} \cdot X_P = e\left(\prod\limits_{i=1}^{P-1} U_i^{\tau_i} \cdot U_P, V \right)$ выполняется одно билинейное спаривание и $(P-1)$ возведений в степень.

В [19] представлено пять видов пакетной верификации, соответствующих разным протоколам zk-SNARK. Используются случайные секретные значения $\chi, \gamma, \delta \in \mathbb{Z}_p$. Значения вида π_{ij} являются ранее сформированными доказательствами [19].

Пакетная верификация достоверности обязательств. Проверка выполнения равенств вида $e(A_{i1}, g_2^{\gamma}) = e(g_1, A_{i2}^{\gamma})$ для $i \in [1, \ldots, P]$ требует $2P$ билинейных спарива-

ний. Пакетная версия $\prod\limits_{i=1}^{P} e(A_{i1}, g_2^{\gamma})^{\tau_i} = \prod\limits_{i=1}^{P} e(g_1, A_{i2}^{\gamma})^{\tau_i}$ или ещё производительнее:

$$e\left(\prod_{i=1}^{P} A_{i1}^{\tau_i}, g_2^{\gamma}\right) = e\left(g_1, \prod_{i=1}^{P} A_{i2}^{\gamma\tau_i}\right),$$

при $\tau_i \in \mathbb{Z}_p$ и $\tau_P = 1$ требует двух билинейных спариваний, по $1 \times (P-1)$ возведений в степень в $\mathbb{G}_2$ и $\mathbb{G}_1$ для левой и правой частей.

Пакетная верификация доказательств знания результата произведения. Если верификатор получает P доказательств $\pi_i = ((A_{i1}, A_{i2}^{\gamma}), (B_{i1}, B_{i2}^{\gamma}), (C_{i1}, C_{i2}^{\gamma})$, то наивная верификация уравнений $e(A_{i1}, B_{i2}^{\gamma}) = e(g_1, C_{i2}^{\gamma}) e(\pi_i, g_2^{\gamma Z(\chi)})$ требует $3P$ билинейных спариваний. Тогда рассматривается $Z(X) = \prod\limits_{i=1}^{n}(X - w^{i-1}) = X^n - 1$ — единственный многочлен n-й степени, для которого $Z(w^{i-1}) = 0$ при $i \in [1, \ldots, n]$, и пакетная версия

$$\prod_{i=1}^{P} e(A_{i1}, B_{i2}^{\gamma})^{\tau_i} = e\left(g_1, \prod_{i=1}^{P} C_{i2}^{\gamma\tau_i}\right) e\left(\prod_{i=1}^{P} \pi_i^{\tau_i}, g^{\gamma Z(\chi)}\right)$$

требует только $(P+2)$ билинейных спариваний, $1 \times P$ возведений в степень в $\mathbb{G}_1$, по $1 \times (P-1)$ возведений в степень в $\mathbb{G}_2$ и $\mathbb{G}_T$. Если $A_{i1} = A_1$ фиксировано, то за счёт изменения первого преобразования на $e\left(A_1, \prod\limits_{i=1}^{P} B_{i2}^{\gamma}\right)^{\tau_i}$ вычисления сокращаются до трёх билинейных спариваний, $1 \times (P-1)$ возведений в степень в $\mathbb{G}_1$ и $2 \times (P-1)$ возведений в степень в $\mathbb{G}_2$. Случай фиксированного $B_{i2}^{\gamma} = B_2$ аналогичен.

Пакетная верификация доказательств знания циклического сдвига вектора. Если верификатор получает P доказательств $\pi_i{=}(\pi_{i1}, \pi_{i2}^{\delta}){=}((A_{i1}, A_{i2}^{\gamma}), (B_{i1}, B_{i2}^{\gamma}))$, то наивная верификация требует $4P$ билинейных спариваний. Пакетная версия

$$e\left(\prod_{i=1}^{P} \pi_{i1}^{\tau_i} \prod_{i=P+1}^{2P} (B_{i1}\pi_{i1})^{\tau_i}, g_2^{\delta Z(\chi)}\right) = e\left(g_1^{Z(\chi)}, \prod_{i=1}^{P} \pi_{i2}^{\delta\tau_i}\right) e\left(\prod_{i=P+1}^{2P} A_{i1}^{\tau_i}, g_2^{\delta Z(\chi) Z^*(\chi)}\right)$$

требует три билинейных спаривания, $1 \times (P-1)$ и $1 \times (2P-1)$ возведений в степень в $\mathbb{G}_1$, $1 \times P$ возведений в степень в $\mathbb{G}_2$.

Пакетная верификация доказательств знания суммы элементов множеств. Протокол zk-SNARK доказательства знания суммы элементов, входящих в конечное множество, рассматривает такой набор $\mathbf{S} = ((S_1, \ldots, S_n), s)$ при $S_i, s \in \mathbb{Z}_p$, для которого существует вектор $\mathbf{b} \in \{0,1\}^n$ и выполняется равенство $\sum\limits_{i=1}^{n} S_i b_i = s$. Доказывается знание открытия обязательства (B_1, B_2^{γ}) к вектору $\mathbf{b}$, при котором выполняется указанное равенство.

Таким образом, верификатор генерирует случайные τ_i, $i \in \{1, \ldots, 8\}$, $\tau_6 = \tau_8 = 1$, и строит $l_i(X) = \prod\limits_{j \neq i}((X - w^{j-1})/(w^{i-1} - w^{j-1}))$ — i-й базисный многочлен Лагранжа, являющийся единственным многочленом степени $n-1$, для которого $l_i(w^{i-1}) = 1$ и $l_i(w^{j-1}) = 0$ при $j \neq i$. Выполняется две проверки:

$$e(B_1^{\tau_1} C_1^{\tau_2} D_1^{\tau_3}, g_2^{\gamma}) e((B_1/g_1)^{\tau_4} (S_1')^{\tau_5}, B_2^{\gamma}) e((g_1^{l_1(\chi)})^{\tau_6}, D_2^{\gamma}/(g_2^{\gamma l_1(\chi)})^{s}) =$$
$$= e(g_1, B_2^{\gamma\tau_1} C_2^{\gamma(\tau_2+\tau_5)} D_2^{\gamma\tau_3}) e(\pi_1^{\tau_4} \pi_2^{\tau_5} \pi_4^{\tau_6}, g_2^{\gamma Z(\chi)}),$$
$$e(\pi_{31}^{\tau_7}(D_1/C_1\pi_{31})^{\tau_8}, g_2^{\delta Z(\chi)}) = e((g_1^{Z(\chi)})^{\tau_7}, \pi_{32}^{\delta}) e(D_1^{\tau_8}, g_2^{\delta Z(\chi) Z^*(\chi)}).$$

В результате требуется только восемь билинейных спариваний, 2×3 и 2×2 возведений в степень в $\mathbb{G}_1$, три дополнительных возведения в степень в $\mathbb{G}_1$, 1×3 возведений в степень в $\mathbb{G}_2$ и одно дополнительное возведение в степень в $\mathbb{G}_2$.

Пакетная верификация доказательств принадлежности диапазону. В протоколе zk-SNARK подтверждения принадлежности элемента общедоступному диапазону $[L,\ldots,H]$ доказывается знание открытия обязательства (A_1, A_2^{γ}) к значению $a\in[L,\ldots,H]$. Это будет означать, что публичный вход (A_1, A_2^{γ}) корректно привязан к вектору $\mathbf{a}$ с $a_1 = a$ и $a_i = 0$ для $i > 1$. Диапазон $[L,\ldots,H]$ можно заменить на $[0,\ldots,H-L]$, а затем использовать гомоморфные свойства схемы формирования обязательств, чтобы добавить L к зафиксированному значению. Поэтому предполагается, что диапазон равен $[0,\ldots,H]$ при $H\geqslant 1$. Данный вариант похож на протокол zk-SNARK доказательства знания суммы элементов, входящих в конечное множество, где дополнительно фиксируется значение $a\in[0,\ldots,H]$ и используется разреженный вектор $\mathbf{S}$ с $S_i = \lfloor (H+2^{i-1})/2^i \rfloor$. Доказываетя, что для обязательств (A_1, A_2^{γ}) зафиксированного значения a выполняется равенство $\sum\limits_{i=1}^{n} S_i b_i = a$.

Таким образом, верификатор генерирует случайные τ_i, $i\in\{0,\ldots,8\}$, $\tau_6=\tau_8=1$, и выполняет две проверки:

$$e(A_1^{\tau_0}B_1^{\tau_1}C_1^{\tau_2}D_1^{\tau_3}, g_2^{\gamma})\, e(g_1^{-\tau_1}(B_1/g_1)^{\tau_4}(S_1')^{\tau_5}, B_2^{\gamma})\, e((g_1^{l_1(\chi)})^{\tau_6}, D_2^{\gamma}/A_2^{\gamma}) =$$
$$= e(g_1, A_2^{\gamma\tau_0}C_2^{\gamma(\tau_2+\tau_5)}D_2^{\gamma\tau_3})\, e(\pi_1^{\tau_4}\pi_2^{\tau_5}\pi_4^{\tau_6}, g_2^{\gamma Z(\chi)}),$$
$$e(\pi_{31}^{\tau_7}(D_1/C_1\pi_{31})^{\tau_8}, g_2^{\delta Z(\chi)}) = e((g_1^{Z(\chi)})^{\tau_7}, \pi_{32}^{\delta})\, e(D_1^{\tau_8}, g_2^{\delta Z(\chi)Z^*(\chi)}).$$

В результате требуется только восемь билинейных спариваний, 1×4, 2×3 и 1×2 возведений в степень в $\mathbb{G}_1$, три дополнительных возведения в степень в $\mathbb{G}_1$, 1×3 возведений в степень в $\mathbb{G}_2$.

В [21] представлен протокол zk-SNARK «Dory» с публичным формированием CRS без использования секретных значений, который строит доказательства знания результата сопряжения векторов двух групп и использует схему доказательств на основе преобразования Фиата — Шамира [46]. Протокол также объединяет доказательства в пакет, повышая производительность верификации. Кроме того, производительность [21] достигается за счёт симметрии сообщений и ключей, применяемых для формирования доказательств: если сообщение принадлежит $\mathbb{G}_1$, то ключ для формирования доказательства принадлежит $\mathbb{G}_2$, и наоборот.

5. Рекурсивная композиция доказательств

Объединение цепочки доказательств протоколов zk-SNARK в одно заключительное доказательство, которое имеет размер и сложность верификации, аналогичные отдельным доказательствам, является важной характеристикой систем достоверных вычислений. Например, протокол zk-SNARK [64] представляет развитие идей поэтапных вычислений с верификацией результатов на каждом промежуточном шаге (инкрементные доказываемые вычисления) [65–67] с произвольным компонентным протоколом zk-SNARK, например [8, 9] и др. Применяется рекурсивная композиция протокола zk-SNARK с фиксированным количеством шагов, что сокращает трудозатраты на формирование CRS за счёт её повторного использования. В результате понятие zk-SNARK расширяется до распределённой среды. После каждого этапа выводится текущее состояние и доказательство его корректности, а вычисления представляются в виде ориентированных ациклических графов, разворачивающихся во времени. В кон-

це цепочки верификатор протокола zk-SNARK получает единственное доказательство достоверности всех шагов.

Протокол zk-SNARK «FRACTAL» [68] представляет новую методологию рекурсивной композиции доказательств с публичным формированием CRS без использования секретных значений. Ядром схемы является постквантовый протокол zk-SNARK с системой ограничений ранга 1 (Rank-1 Constraint System, R1CS). Подобно [69, 70], схема [68] также использует коды Рида — Соломона при работе с R1CS и построении вероятностно-проверяемых доказательств.

Протоколы zk-SNARK семейства «Spartan» [71] также формируют CRS без использования секретных значений. В частности, реализация протокола zk-SNARK «Halo» [72] представляет рекурсивную композицию доказательств, в которой для группы $\mathbb{G}$ простого порядка p, случайного вектора $\mathbf{G} \in \mathbb{G}^d$, $d \in \mathbb{Z}$, значений $r, H \in \mathbb{G}$, коэффициентов $a_i \in \mathbb{F}_p$, являющихся членами i-й степени QAP-полинома $p(X)$, формируется CRS в виде $\boldsymbol{\sigma} = (\mathbb{G}, \mathbb{F}_p, \mathbf{G}, H)$, а схема формирования обязательств имеет следующий вид:

$$\mathsf{Com}(\boldsymbol{\sigma}, p(X); r) = \langle \mathbf{a}, \mathbf{G} \rangle + [r]H.$$

Здесь $\langle \mathbf{a}, \mathbf{G} \rangle = a_0 G_0 + a_1 G_1 + a_2 G_2 + \ldots$ — скалярное произведение векторов; $[r]H$ — r-кратное сложение точки эллиптической кривой. В качестве секретов выступают $\mathbf{a}, r$.

Источник [73] обобщает конструкцию «Halo» [72], продолжая развитие рекурсивных доказательств с использованием компактных схем обязательств для доказательства достоверности оценки полиномов в секретных точках. «Halo Infinite» [73] демонстрирует новую методологию построения систем доказательств знания «Bulletproofs» без протокола zk-SNARK, используя свойство агрегации внутренних доказательств. Версия «Halo Infinite» [73] объединяет линейные комбинации обязательств в краткое обязательство, открываемое при необходимости.

6. Сравнительный анализ производительности протоколов zk-SNARK

В таблице приведено сравнение протоколов zk-SNARK [7–9, 12–15, 17, 18, 20–36] по размерам CRS и доказательств, а также по сложности вычислений доказывающих и верификаторов. Используются следующие обозначения: n — количество логических элементов дискретных функций (вентилей арифметической/логической схемы); λ — параметр безопасности; $\mathbb{G}$ — размер элемента группы; A, M, E, P — затраты на сложение, умножение, возведение в степень и билинейное спаривание; l_u, l_w, l_o — разрядность открытого и секретного входов и выхода; m — количество переменных дискретной функции (проводов арифметической/логической схемы); n_M — количество элементов умножения дискретной функции (арифметической/логической схемы); l — размер открытого входа в элементах группы. Для протокола zk-SNARK [30] на основе полиномиальных наборов QPP введены дополнительные обозначения: d — количество корней полиномов набора QPP; $v = \max\limits_{i=1,\ldots,m} \{n_i\}$ — максимальная степень полиномов набора QPP; n_i — степень выходного полинома $c_i(z)$ набора QPP для $i = 1, \ldots, m$; m — количество выходных полиномов; T — сложность вычисления всех полиномов $c_i(z)$.

Анализ таблицы показывает, что с точки зрения практической реализации наиболее приемлемыми являются протоколы zk-SNARK [12–14, 17, 18, 32–35], которые для произвольных дискретных функций всегда имеют фиксированный размер доказательств и постоянное количество уравнений верификации.

Выделяются работы [15, 30, 31], которые также имеют фиксированные размеры доказательств. Тем не менее сложность верификации протокола zk-SNARK [15] зави-

Сравнительный анализ производительности протоколов zk-SNARK

Протокол	CRS	Доказ-во	Доказывающий	Верификатор
Й. Грот и др. [22]	$O(1)\ \mathbb{G}$	$O(n)\ \mathbb{G}$	$O(n)\ E$	$O(n)\ P$
Й. Грот и др. [23]	$O(1)\ \mathbb{G}$	$O(n)\ \mathbb{G}$	$O(n)\ E$	$O(n)\ E$
М. Эйб и др. [24]	$O(1)\ \mathbb{G}$	$O(n)\ \mathbb{G}$	$O(n)\ E$	$O(n)\ E$
Дж. Джентри [25]	$O(1)\ \mathbb{G}$	$l_w\lambda^{O(1)}\ \mathbb{G}$	$n\lambda^{O(1)}\ M$	$n\lambda^{O(1)}\ M$
Й. Грот [26]	$O(n^{1/2})\ \mathbb{G}$	$O(n^{1/2})\ \mathbb{G}$	$O(n)\ M$	$O(n)\ M$
Р. Дженнаро и др. [20]	$O(n\ poly(\lambda))\ \mathbb{G}$	$O(l_o)\ \mathbb{G}$	$O(n\ poly(\lambda))\ M$	$O(l_o\ poly(\lambda))\ M$
Й. Грот [27]	$O(n)\ \mathbb{G}$	$O(n)\ \mathbb{G}$	$O(n)\ E$	$O(n)\ M$
Й. Грот [27]	$O(n)\ \mathbb{F}_2$	$O(n)\ \mathbb{F}_2$	$O(n)\ M$	$O(n)\ M$
Й. Грот [8]	$O(n^{2/3})\ \mathbb{G}$	$O(n^{2/3})\ \mathbb{G}$	$O(n^{4/3})\ E$	$O(n)\ M$, $O(n^{2/3})\ P$
Х. Липмаа [9]	$n^{1/3+O(1)}\ \mathbb{G}$	$O(n^{2/3})\ \mathbb{G}$	$O(n^{4/3})\ A, n^{1+O(1)}\ E$	$O(n)\ M, O(n^{2/3})\ P$
Х. Липмаа [28]	$O(n)\ \mathbb{G}$	$O(1)\ \mathbb{G}$	$O(n\log n)\ E$	$O(1)\ P$
Э. Бен-Сассон и др. [12]	$O(11n+3l+2)\ \mathbb{G}_1$, $6\ \mathbb{G}_2$	$11\ \mathbb{G}_1, 1\ \mathbb{G}_2$	$O(n)\ \mathbb{G}_1, \mathbb{G}_2$	$O(l)\ E, 21\ P$
Р. Дженнаро и др. [15]	$O(n)\ \mathbb{G}$	$9\ \mathbb{G}$	$O(n+n\log^2 n)\ E$	$O(l_u+l_w)\ E/P$
Б. Парно и др. [13]	$O(7m+n_M-2l)\ \mathbb{G}$	$8\ \mathbb{G}$	$O(7m+n_M-2l)\ E$	$O(l)\ E, 11\ P$
П. Фаузи и др. [29]	$n^{(1+O(1))/2}\ \mathbb{G}$	$O(n^{1/2})\ \mathbb{G}$	$n^{(1+O(1))/2} \times$ $\times\ O(n^{1/2}(1+\log n))\ M$	$O(n)\ M$, $O(n^{1/2})\ P$
Г. Данезис и др. [18]	$O(2m+n-2l)\ \mathbb{G}_1$, $O(m+n-l)\ \mathbb{G}_2$	$3\ \mathbb{G}_1, 1\ \mathbb{G}_2$	$O(m+n-l)\ E_1$	$O(l)\ M_1, 6\ P$
Ю. Чжан и др. [7]	$O(m)\ \mathbb{G}$	$O(l_w)\ \mathbb{G}$	$O(m\log m)\ E$	$O(l_w)\ E, P$
А. Косба и др. [30]	$O(dl)\ \mathbb{G}$	$8\ \mathbb{G}$	$O(T+dv \times$ $\times\ (\log(dv)+m))\ E$	$O(\sum_{i\in[l]} n_i)\ M$
Х. Липмаа [31]	$O(n\log n)\ \mathbb{G}$	$O(1)\ \mathbb{G}$	$O(n\log^2 n)\ E$, $O(n\log n)\ M$	$O(\log n)\ P$
Э. Бен-Сассон и др. [32]	$O(6m+n_M+l)\ \mathbb{G}_1$, $O(m)\ \mathbb{G}_2$	$7\ \mathbb{G}_1, 1\ \mathbb{G}_2$	$O(6m+n_M-l)\ E_1$, $O(m)\ E_2$	$O(l)\ E_1, 12\ P$
К. Костелло и др. [33]	$O(7m+n_M-2l)\ \mathbb{G}$	$8\ \mathbb{G}$	$O(7m+n_M-2l)\ E$	$O(l)\ E, 11\ P$
М. Бакес и др. [34]	$O(1+7m+n-2l)\ \mathbb{G}$	$3+8\ \mathbb{G}$	$O(3l+7m+n-2l)\ E$	$6+11\ P, O(l)\ M$, $O(1+l)\ E$
Й. Грот [17]	$O(m+2n_M)\ \mathbb{G}_1$, $O(n_M)\ \mathbb{G}_2$	$2\ \mathbb{G}_1, 1\ \mathbb{G}_2$	$O(m+3n_M-l)\ E_1$, $O(n_M)\ E_2$	$O(l)\ E_1, 3\ P$
Й. Грот и др. [35]	$O(m+4n_M+5)\ \mathbb{G}_1$, $O(2n_M+3)\ \mathbb{G}_2$	$2\ \mathbb{G}_1, 1\ \mathbb{G}_2$	$O(m+4n_M-l)\ E_1$, $O(2n_M)\ E_2$	$O(l)\ E_1, 6\ P$
Э. Бен-Сассон и др. [14]	$O(6m+n_M-l)\ \mathbb{G}_1$, $O(m)\ \mathbb{G}_2$	$7\ \mathbb{G}_1, 1\ \mathbb{G}_2$	$O(6m+n_M-l)\ E_1$, $O(m)\ E_2$	$O(l)\ E_1, 12\ P$
А. Кьеза и др. [36]	$O(4n+2)\ \mathbb{G}_1$, $2\ \mathbb{G}_2$	$13\ \mathbb{G}_1, 8\ \mathbb{F}_q$	$O(22n)\ \mathbb{G}_1$, $O(n\log m)\ \mathbb{F}_q$	$2\ P\ \mathbb{G}_1$, $O(l_u+\log n)\ \mathbb{F}_q$
Дж. Ли [21]	–	$1\ \mathbb{G}_T$, $O(6\log d)\ \mathbb{G}_T$	$O(d^{1/2})\ P$	$O(\log d)\ M$, $1\ P$

сит от разрядностей открытого и секретного входов дискретной функции, в протоколе zk-SNARK [31] сложность верификации зависит от количества логических элементов дискретной функции, а сложность верификации протокола zk-SNARK [30] — от степеней полиномов, соответствующих выходам дискретной функции.

Характерно, что многие рассмотренные протоколы имеют структуру, унаследованную или соответствующую развитию исторически основополагающих криптографических протоколов zk-SNARK [15, 13].

Отдельного внимания заслуживают протоколы zk-SNARK [22, 23], которые характеризуются простотой описания и могут быть реализованы с меньшими трудозатратами. Однако у данных протоколов размер доказательств и количество билинейных спариваний при верификации зависят линейно от количества логических элементов дискретной функции. Вопрос об их модификации для фиксирования размера доказательств и количества билинейных спариваний остаётся открытым. Тем не менее в ряде

случаев применение протоколов zk-SNARK [22, 23] может оказаться целесообразным, например для дискретных функций с небольшим количеством логических элементов.

В таблице рассматриваются протоколы zk-SNARK для фиксированных дискретных функций с формированием CRS доверенной стороной с использованием секретных значений. В качестве сравнения с ними приведены характеристики производительности протокола zk-SNARK «Marlin» [36], который однократно строит универсальную и обновляемую CRS, впоследствии используемую для различных дискретных функций. По аналогичным соображениям приведены характеристики производительности протокола zk-SNARK «Dory» [21] доказательства корректности произведения полиномов степени d, который не требует предварительного формирования CRS доверенной стороной с использованием секретных значений.

Заключение

Рассмотрены основные способы повышения производительности, которые в зависимости от конструкции могут являться общими для разных протоколов zk-SNARK. Они основаны на применении хеш-функций, в том числе более адаптированных к рассматриваемым протоколам [2–6,] методах производительного деления [8, 9], умножения и возведения в степень [10–14] элементов множеств, разделении CRS на лаконичные компоненты [15, 17, 18] для доказывающих и верификаторов, а также на более эффективных билинейных спариваниях [16, 17]. Аналогично [18], доказательство протокола zk-SNARK [17] может сжиматься до двух элементов группы. Относительно применения хеш-функций существуют компромиссы между уменьшением размера доказательств и увеличением затрат на формирование доказательств/верификацию, уменьшением затрат на формирование доказательств и увеличением затрат на верификацию, фиксацией размера открытого входа и увеличением затрат на верификацию и др.

Зафиксирована возможность объединения цепочки промежуточных доказательств протоколов zk-SNARK в одно заключительное доказательство [64, 68, 71–73]. На примере протокола zk-SNARK [14] показаны структурные свойства полиномиального набора QAP, характеризующиеся наличием нулевых многочленов, что также сокращает трудозатраты. Представлен мощный инструмент пакетной верификации доказательств протоколов zk-SNARK [19], значительно сокращающий сложность вычислений за счёт более эффективной группировки элементов доказательств.

Сравнительный анализ протоколов zk-SNARK [7–9, 12–15, 17, 18, 20–36] показал, что конструкции [12–14, 17, 18, 32–35] формируют доказательства фиксированного размера и используют постоянное количество уравнений верификации, поэтому в большей степени подходят для практической реализации. Отдельно выделены работы [15, 30, 31] с фиксированными размерами доказательств, но со сложностью верификации, зависящей от входов, количества логических элементов и степеней полиномов выходов дискретных функций. Протоколы zk-SNARK [22, 23] выделяются простотой описания, однако в зависимости от количества логических элементов дискретных функций размеры доказательств и количество билинейных спариваний изменяются. Фиксирование размера доказательств и количества билинейных спариваний позволит [22, 23] приобрести более широкое распространение. Характерно, что многие рассмотренные протоколы имеют структуру, унаследованную или развитую от исторически основополагающих криптографических протоколов zk-SNARK [13, 15].

ЛИТЕРАТУРА

1. *Мартыненков И. В.* Краткие неинтерактивные аргументы с нулевым разглашением на основе наборов полиномов // Прикладная дискретная математика. 2023. № 59. С. 20–57.
2. *Grassi L., Khovratovich D., Rechberger C., et al.* Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. Cryptology ePrint Archive. Paper 2019/458. 2019. `https://eprint.iacr.org/2019/458`.
3. *Baylina J. and Bellés M.* 4-bit Window Pedersen Hash On The Baby Jubjub Elliptic Curve. 2019. `https://docs.iden3.io/publications/pdfs/Pedersen-Hash.pdf`. 8 p.
4. *Regev O.* New lattice based cryptographic constructions // Proc. 35th Ann. ACM Symp. on Theory of Computing. N.Y.: ACM, 2003. P. 407–416.
5. *Regev O.* New lattice-based cryptographic constructions // J. ACM. 2004. V. 51. Iss. 6. P. 899–942.
6. *Regev O.* On lattices, learning with errors, random linear codes, and cryptography // Proc. 37th Ann. ACM Symp. on Theory of Computing. N.Y.: ACM, 2005. P. 84–93.
7. *Zhang Y., Papamanthou C., and Katz J.* ALITHEIA: Towards practical verifiable graph processing // Proc. CCS'14. N.Y.: ACM, 2014. P. 856–867.
8. *Groth J.* Short pairing-based non-interactive zero-knowledge arguments // LNCS. 2010. V. 6477. P. 321–340.
9. *Lipmaa H.* Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments // LNCS. 2012. V. 7194. P. 169–189.
10. *Pippenger N.* On the evaluation of powers and related problems // 17th Ann. Symp. SFCS'1976. Houston, TX, USA, 1976. P. 258–263.
11. *Bos J. and Coster M.* Addition chain heuristics // LNCS. 1990. V. 435. P. 400–407.
12. *Ben-Sasson E., Chiesa A., Genkin D., et al.* SNARKs for C: Verifying program executions succinctly and in zero knowledge // LNCS. 2013. V. 8043. P. 90–108.
13. *Parno B., Howell J., Gentry C., and Raykova M.* Pinocchio: Nearly practical verifiable computation // Proc. 34th IEEE Symp. on Security and Privacy. Berkeley, CA, USA, 2013. P. 238–252.
14. *Ben-Sasson E., Chiesa A., Tromer E., and Virza M.* Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. Updated version, 2019. `https://eprint.iacr.org/2013/879.pdf`. 37 p.
15. *Gennaro R., Gentry C., Parno B., and Raykova M.* Quadratic span programs and succinct NIZKs without PCPs // LNCS. 2013. V. 7881. P. 626–645.
16. *Costello C.* Pairings for Beginners. `https://www.craigcostello.com.au/pairings/PairingsForBeginners.pdf`. 2012. 146 p.
17. *Groth J.* On the size of pairing-based non-interactive arguments // LNCS. 2016. V. 9666. P. 305–326.
18. *Danezis G., Fournet C., Groth J., and Kohlweiss M.* Square span programs with applications to succinct NIZK arguments // LNCS. 2014. V. 8873. P. 532–550.
19. *Lipmaa H.* Almost Optimal Short Adaptive Non-Interactive Zero Knowledge. Tech. Rep. 2014/396. International Association for Cryptologic Research, 2014. `http://eprint.iacr.org/2014/396`. 20 p.
20. *Gennaro R., Gentry C., and Parno B.* Non-interactive verifiable computing: Outsourcing computation to untrusted workers // LNCS. 2010. V. 6223. P. 465–482.
21. *Lee J.* Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments // LNCS. 2021. V. 13043. P. 1–34.
22. *Groth J., Ostrovsky R., and Sahai A.* Perfect non-interactive zero knowledge for NP // LNCS. 2006. V. 4004. P. 339–358.

23. *Groth J., Ostrovsky R., and Sahai A.* Non-interactive zaps and new techniques for NIZK // LNCS. 2006. V. 4117. P. 97–111.
24. *Abe M. and Fehr S.* Perfect NIZK with adaptive soundness // LNCS. 2007. V. 4392. P. 118–136.
25. *Gentry G.* Fully homomorphic encryption using ideal lattices // Proc. 41 Ann. ACM Symp. Theory of Computing. V. 9. N.Y.: ACM, 2009. P. 169–178.
26. *Groth J.* Linear algebra with sub-linear zero-knowledge arguments // LNCS. 2009. V. 5677. P. 192–208.
27. *Groth J.* Short non-interactive zero-knowledge proofs // LNCS. 2010. V. 6477. P. 341–358.
28. *Lipmaa H.* Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes // LNCS. 2013. V. 8269. P. 41–60.
29. *Fauzi P., Lipmaa H., and Zhang B.* Efficient modular NIZK arguments from shift and product // LNCS. 2013. V. 8257. P. 92–121.
30. *Kosba A.E., Papadopoulos D., Papamanthou C., et al.* TRUESET: Nearly Practical Verifiable Set Computations. Cryptology ePrint Archive. Report 2014/160. 2014. `http://eprint.iacr.org/2014/160`. 30 p.
31. *Lipmaa H.* Efficient NIZK arguments via parallel verification of benes networks // LNCS. 2014. V. 8642. P. 416–434.
32. *Ben-Sasson E., Chiesa A., Tromer E., and Virza M.* Scalable zero knowledge via cycles of elliptic curves // LNCS. 2014. V. 8617. P. 276–294.
33. *Costello C., Fournet C., Howell J., et al.* Geppetto: Versatile verifiable computation // Proc. IEEE Symp. SP'15. IEEE Computer Society, USA, 2015. P. 253–270.
34. *Backes M., Barbosa M., Fiore D., and Reischuk R. M.* ADSNARK: Nearly practical and privacy-preserving proofs on authenticated data // IEEE Symp. on Security and Privacy, San Jose, CA, USA, 2015. P. 271–286.
35. *Groth J. and Maller M.* Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs. IACR Cryptology ePrint Archive. 2017. 36 p.
36. *Chiesa A., Hu Y., Maller M., et al.* Marlin: Preprocessing zk-SNARKs with universal and updatable SRS // LNCS. 2020. V. 12105. P. 738–768.
37. *Applebaum B., Ishai Y., and Kushilevitz E.* From secrecy to soundness: Efficient verification via secure computation // LNCS. 2020. V. 6198. P. 152–163.
38. *Reitwiebner C.* zkSNARKs in a Nutshell. 2017. `https://blog.ethereum.org/2016/12/05/zksnarks-in-a-nutshell/`. 15 p.
39. *Bitansky N., Canetti R., Chiesa A., and Tromer E.* From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again // Proc. ITCS'12. N.Y.: ACM, 2012. P. 326–349.
40. *Goldwasser S., Lin H., and Rubinstein A.* Delegation of Computation without Rejection Problem from Designated Verifier CS-proofs. Cryptology ePrint Archive. Report 2011/456. 2011. 30 p.
41. *Ajtai M.* Generating hard instances of lattice problems (extended abstract) // Proc. STOC'96. N.Y.: ACM, 1996. P. 99–108.
42. *Papamanthou C., Shi E., Tamassia R., and Yi K.* Streaming authenticated data structures // LNCS. 2013. V. 7881. P. 353–370.
43. *Katz J., Kolesnikov V., and Wang X.* Improved non-interactive zero knowledge with applications to post-quantum signatures // Proc. ACM SIGSAC Conf. CCS'18. N.Y.: ACM, 2018. P. 525–537.

44. *Bünz B., Fisch B., and Szepieniec A.* Transparent SNARKs from DARK Compilers. Cryptology ePrint Archive. Paper 2019/1229. 2019. https://eprint.iacr.org/2019/1229. 59 p.
45. *Ames S., Hazay C., Ishai Y., and Venkitasubramaniam M.* Ligero: Lightweight sublinear arguments without a trusted setup // Proc. ACM SIGSAC Conf. CCS'17. N.Y.: ACM, 2017. P. 2087–2104.
46. *Fiat A. and Shamir S.* How to prove yourself: Practical solutions to identification and signature problems // LNCS. 1986. V. 263. P. 186–194.
47. *Aho A., Hopcroft J., and Ulman J.* The Design and Analysis of Computer Algorithms. Addison-Wesley, 1974. 470 p.
48. *Scott M.* Implementing cryptographic pairings // Proc. 1st First Intern. Conf. on Pairing-Based Cryptography. Berlin; Heidelberg: Springer Verlag, 2007. P. 177–196.
49. *Galbraith S. D., Harrison K., and Soldera D.* Implementing the Tate pairing // LNCS. 2002. V. 2369. P. 324–337.
50. *Barreto P. S. L. M., Lynn B., and Scott M.* Constructing elliptic curves with prescribed embedding degrees // LNCS. 2003. V. 2576. P. 257–267.
51. *Vercauteren F.* Optimal pairings // IEEE Trans. Inform. Theory. 2020. V. 56. No. 1. P. 455–461.
52. *Beuchat J. L., González-Díaz J. E., Mitsunari S., et al.* High-speed software implementation of the optimal ate pairing over Barreto — Naehrig curves // LNCS. 2010. V. 6487. P. 21–39.
53. *Scott M., Benger N., Charlemagne M., et al.* On the final exponentiation for calculating pairings on ordinary elliptic curves // LNCS. 2009. V. 5671. P. 78–88.
54. *Granger R. and Scott M.* Faster squaring in the cyclotomic subgroup of sixth degree extensions // LNCS. 2010. V. 6056. P. 209–223.
55. *Fuentes-Castañeda L., Knapp E., and Rodríguez-Henríquez F.* Faster hashing to $\mathbb{G}_2$ // LNCS. 2012. V. 7118. P. 412–430.
56. *Kim T., Kim S., and Cheon J. H.* On the final exponentiation in Tate pairing computations // IEEE Trans. Inform. Theory. 2013. V. 59. No. 6. P. 4033–4041.
57. *Solinas J. A.* ID-based Digital Signature Algorithms. 2003. http://cacr.uwaterloo.ca/conferences/2003/ecc2003/solinas.pdf. 32 p.
58. *Scott M.* Computing the Tate pairing // LNCS. 2005. V. 3376. P. 293–304.
59. *Granger R. and Smart N.* On Computing Products of Pairings. Cryptology ePrint Archive. Report 2006/172. 2006. 11 p.
60. *Arène C, Lange T., Naehrig M., and Ritzenthaler C.* Faster computation of the Tate pairing // J. Number Theory. 2022. V. 131. No. 5. P. 842–857.
61. *Frey G. and Rück H.-G.* A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves // Mathematics of Computation. 1994. V. 62. No. 206. P. 865–874.
62. *Frey G., Muller M., and Rück H.-G.* The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems // IEEE Trans. Inform. Theory. 2006. V. 45. No. 5. P. 1717–1719.
63. *Galbraith S. D., Paterson K. G., and Smart N. P.* Pairings for cryptographers // Discrete Appl. Math. 2008. V. 156. No. 16. P. 3113–3121.
64. *Bitansky N., Canetti R., Chiesa A., and Tromer E.* Recursive Composition and Bootstrapping for Snarks and Proof-Carrying Data. IACR Cryptology ePrint Archive. 2012. http://eprint.iacr.org/2012/095. 61 p.
65. *Valiant P.* Incrementally verifiable computation or proofs of knowledge imply time/space efficiency // LNCS. 2008. V. 4948. P. 1–18.
66. *Crescenzo G. D. and Lipmaa H.* Succinct NP proofs from an extractability assumption // LNCS. 2008. V. 5028. P. 175–185.

67. *Mei T.* Polylogarithmic two-round argument systems // J. Math. Cryptol. 2008. V. 2. No. 4. P. 343–363.
68. *Chiesa A., Ojha D., and Spooner N.* FRACTAL: Post-quantum and transparent recursive proofs from holography // LNCS. 2020. V. 12105. P. 769–793.
69. *Kattis A., Panarin K., and Vlasov A.* RedShift: Transparent SNARKs from List Polynomial Commitment IOPs. Cryptology ePrint Archive. Paper 2019/1400. 2019. `https://eprint.iacr.org/2019/1400`. 20 p.
70. *Ben-Sasson E., Bentov I., Horesh Y., and Riabzev M.* Fast Reed — Solomon interactive oracle proofs of proximity // 45th Intern. Colloquium ICALP'2018. V. 107. Prague, Czech Republic, 2018. P. 1–17. `https://doi.org/10.4230/LIPIcs.ICALP.2018.14`.
71. *Setty S.* Spartan: Efficient and general-purpose zkSNARKs without trusted setup // LNCS. 2020. V. 12172. P. 704–737.
72. *Bowe S., Grigg J., and Hopwood D.* Recursive Proof Composition without a Trusted Setup. Cryptology ePrint Archive. Paper 2019/1021. 2019. `https://eprint.iacr.org/2019/1021`. 31 p.
73. *Boneh D., Drake J., Fisch B., and Gabizon A.* Halo Infinite: Recursive zk-SNARKs from any Additive Polynomial Commitment Scheme. Cryptology ePrint Archive. Paper 2020/1536. 2020. `https://eprint.iacr.org/2020/1536`. 66 p.

REFERENCES

1. *Martynenkov I. V.* Kratkie neinteraktivnye argumenty s nulevym razglasheniem na osnove naborov polinomov [Zero-knowledge succinct non-interactive arguments of knowledge based on sets of polynomials]. Prikladnaya Diskretnaya Matematika, 2023, no. 59, pp. 20–57. (in Russian)
2. *Grassi L., Khovratovich D., Rechberger C., et al.* Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. Cryptology ePrint Archive, Paper 2019/458, 2019. `https://eprint.iacr.org/2019/458`.
3. *Baylina J. and Bellés M.* 4-bit Window Pedersen Hash On The Baby Jubjub Elliptic Curve. 2019. `https://docs.iden3.io/publications/pdfs/Pedersen-Hash.pdf`. 8 p.
4. *Regev O.* New lattice based cryptographic constructions. Proc. 35th Ann. ACM Symp. on Theory of Computing, N.Y., ACM, 2003, pp. 407–416.
5. *Regev O.* New lattice-based cryptographic constructions. J. ACM, 2004, vol. 51, iss. 6, pp. 899–942.
6. *Regev O.* On lattices, learning with errors, random linear codes, and cryptography. Proc. 37th Ann. ACM Symp. on Theory of Computing. N.Y., ACM, 2005, pp. 84–93.
7. *Zhang Y., Papamanthou C., and Katz J.* ALITHEIA: Towards practical verifiable graph processing. Proc. CCS'14. N.Y., ACM, 2014, pp. 856–867.
8. *Groth J.* Short pairing-based non-interactive zero-knowledge arguments. LNCS, 2010, vol. 6477, pp. 321–340.
9. *Lipmaa H.* Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. LNCS, 2012, vol. 7194, pp. 169–189.
10. *Pippenger N.* On the evaluation of powers and related problems. 17th Ann. Symp. SFCS'1976, Houston, TX, USA, 1976, pp. 258–263.
11. *Bos J. and Coster M.* Addition chain heuristicse LNCS, 1990, vol. 435, pp. 400–407.
12. *Ben-Sasson E., Chiesa A., Genkin D., et al.* SNARKs for C: Verifying program executions succinctly and in zero knowledge. LNCS, 2013, vol. 8043, pp. 90–108.

13. *Parno B., Howell J., Gentry C., and Raykova M.* Pinocchio: Nearly practical verifiable computation. Proc. 34th IEEE Symp. on Security and Privacy, Berkeley, CA, USA, 2013, pp. 238–252.
14. *Ben-Sasson E., Chiesa A., Tromer E., and Virza M.* Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture, Updated version, 2019. `https://eprint.iacr.org/2013/879.pdf`. 37 p.
15. *Gennaro R., Gentry C., Parno B., and Raykova M.* Quadratic span programs and succinct NIZKs without PCPs. LNCS, 2013, vol. 7881, pp. 626–645.
16. *Costello C.* Pairings for Beginners. `https://www.craigcostello.com.au/pairings/PairingsForBeginners.pdf`, 2012. 146 p.
17. *Groth J.* On the size of pairing-based non-interactive arguments. LNCS, 2016, vol. 9666, pp. 305–326.
18. *Danezis G., Fournet C., Groth J., and Kohlweiss M.* Square span programs with applications to succinct NIZK arguments. LNCS, 2014, vol. 8873, pp. 532–550.
19. *Lipmaa H.* Almost Optimal Short Adaptive Non-Interactive Zero Knowledge. Tech. Rep. 2014/396. International Association for Cryptologic Research, 2014. `http://eprint.iacr.org/2014/396`. 20 p.
20. *Gennaro R., Gentry C., and Parno B.* Non-interactive verifiable computing: Outsourcing computation to untrusted workers. LNCS, 2010, vol. 6223, pp. 465–482.
21. *Lee J.* Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. LNCS, 2021, vol. 13043, pp. 1–34.
22. *Groth J., Ostrovsky R., and Sahai A.* Perfect non-interactive zero knowledge for NP. LNCS, 2006, vol. 4004, pp. 339–358.
23. *Groth J., Ostrovsky R., and Sahai A.* Non-interactive zaps and new techniques for NIZK. LNCS, 2006, vol. 4117, pp. 97–111.
24. *Abe M. and Fehr S.* Perfect NIZK with adaptive soundness. LNCS, 2007, vol. 4392, pp. 118–136.
25. *Gentry G.* Fully homomorphic encryption using ideal lattices. Proc. 41 Ann. ACM Symp. Theory of Computing, vol. 9, N.Y., ACM, 2009, pp. 169–178.
26. *Groth J.* Linear algebra with sub-linear zero-knowledge arguments. LNCS, 2009, vol. 5677, pp. 192–208.
27. *Groth J.* Short non-interactive zero-knowledge proofs. LNCS, 2010, vol. 6477, pp. 341–358.
28. *Lipmaa H.* Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. LNCS, 2013, vol. 8269, pp. 41–60.
29. *Fauzi P., Lipmaa H., and Zhang B.* Efficient modular NIZK arguments from shift and product. LNCS, 2013, vol. 8257, pp. 92–121.
30. *Kosba A.E., Papadopoulos D., Papamanthou C., et al.* TRUESET: Nearly Practical Verifiable Set Computations. Cryptology ePrint Archive, Report 2014/160, 2014. `http://eprint.iacr.org/2014/160`. 30 p.
31. *Lipmaa H.* Efficient NIZK arguments via parallel verification of benes networks. LNCS, 2014, vol. 8642, pp. 416–434.
32. *Ben-Sasson E., Chiesa A., Tromer E., and Virza M.* Scalable zero knowledge via cycles of elliptic curves. LNCS, 2014, vol. 8617, pp. 276–294.
33. *Costello C., Fournet C., Howell J., et al.* Geppetto: Versatile verifiable computation. Proc. IEEE Symp. SP'15, IEEE Computer Society, USA, 2015, pp. 253–270.
34. *Backes M., Barbosa M., Fiore D., and Reischuk R. M.* ADSNARK: Nearly practical and privacy-preserving proofs on authenticated data. IEEE Symp. on Security and Privacy, San Jose, CA, USA, 2015, pp. 271–286.

35. *Groth J. and Maller M.* Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs. IACR Cryptology ePrint Archive, 2017. 36 p.
36. *Chiesa A., Hu Y., Maller M., et al.* Marlin: Preprocessing zk-SNARKs with universal and updatable SRS. LNCS, 2020, vol. 12105, pp. 738–768.
37. *Applebaum B., Ishai Y., and Kushilevitz E.* From secrecy to soundness: Efficient verification via secure computation. LNCS, 2020, vol. 6198, pp. 152–163.
38. *Reitwiebner C.* zkSNARKs in a Nutshell. 2017. `https://blog.ethereum.org/2016/12/05/zksnarks-in-a-nutshell/`. 15 p.
39. *Bitansky N., Canetti R., Chiesa A., and Tromer E.* From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. Proc. ITCS'12, N.Y., ACM, 2012, pp. 326–349.
40. *Goldwasser S., Lin H., and Rubinstein A.* Delegation of Computation without Rejection Problem from Designated Verifier CS-proofs. Cryptology ePrint Archive, Report 2011/456, 2011. 30 p.
41. *Ajtai M.* Generating hard instances of lattice problems (extended abstract). Proc. STOC'96, N.Y., ACM, 1996, pp. 99–108.
42. *Papamanthou C., Shi E., Tamassia R., and Yi K.* Streaming authenticated data structures. LNCS, 2013, vol. 7881, pp. 353–370.
43. *Katz J., Kolesnikov V., and Wang X.* Improved non-interactive zero knowledge with applications to post-quantum signatures. Proc. ACM SIGSAC Conf. CCS'18, N.Y., ACM, 2018, pp. 525–537.
44. *Bünz B., Fisch B., and Szepieniec A.* Transparent SNARKs from DARK Compilers. Cryptology ePrint Archive, Paper 2019/1229, 2019. `https://eprint.iacr.org/2019/1229`. 59 p.
45. *Ames S., Hazay C., Ishai Y., and Venkitasubramaniam M.* Ligero: Lightweight sublinear arguments without a trusted setup. Proc. ACM SIGSAC Conf. CCS'17, N.Y., ACM, 2017, pp. 2087–2104.
46. *Fiat A. and Shamir S.* How to prove yourself: Practical solutions to identification and signature problems. LNCS, 1986, vol. 263, pp. 186–194.
47. *Aho A., Hopcroft J., and Ulman J.* The Design and Analysis of Computer Algorithms. Addison-Wesley, 1974. 470 p.
48. *Scott M.* Implementing cryptographic pairings. Proc. 1st First Intern. Conf. on Pairing-Based Cryptography, Berlin; Heidelberg, Springer Verlag, 2007, pp. 177–196.
49. *Galbraith S. D., Harrison K., and Soldera D.* Implementing the Tate pairing. LNCS, 2002, vol. 2369, pp. 324–337.
50. *Barreto P. S. L. M., Lynn B., and Scott M.* Constructing elliptic curves with prescribed embedding degrees. LNCS, 2003, vol. 2576, pp. 257–267.
51. *Vercauteren F.* Optimal pairings. IEEE Trans. Inform. Theory, 2020, vol. 56, no. 1, pp. 455–461.
52. *Beuchat J. L., González-Díaz J. E., Mitsunari S., et al.* High-speed software implementation of the optimal ate pairing over Barreto — Naehrig curves. LNCS, 2010, vol. 6487, pp. 21–39.
53. *Scott M., Benger N., Charlemagne M., et al.* On the final exponentiation for calculating pairings on ordinary elliptic curves. LNCS, 2009, vol. 5671, pp. 78–88.
54. *Granger R. and Scott M.* Faster squaring in the cyclotomic subgroup of sixth degree extensions. LNCS, 2010, vol. 6056, pp. 209–223.
55. *Fuentes-Castañeda L., Knapp E., and Rodríguez-Henríquez F.* Faster hashing to $\mathbb{G}_2$. LNCS, 2012, vol. 7118, pp. 412–430.

56. *Kim T., Kim S., and Cheon J. H.* On the final exponentiation in Tate pairing computations. IEEE Trans. Inform. Theory, 2013, vol. 59, no. 6, pp. 4033–4041.
57. *Solinas J. A.* ID-based Digital Signature Algorithms. 2003. `http://cacr.uwaterloo.ca/conferences/2003/ecc2003/solinas.pdf`. 32 p.
58. *Scott M.* Computing the Tate pairing LNCS, 2005. vol. 3376, pp. 293–304.
59. *Granger R. and Smart N.* On Computing Products of Pairings. Cryptology ePrint Archive, Report 2006/172, 2006. 11 p.
60. *Arène C, Lange T., Naehrig M., and Ritzenthaler C.* Faster computation of the Tate pairing. J. Number Theory, 2022, vol. 131, no. 5, pp. 842–857.
61. *Frey G. and Rück H.-G.* A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Mathematics of Computation, 1994, vol. 62, no. 206, pp. 865–874.
62. *Frey G., Muller M., and Rück H.-G.* The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. IEEE Trans. Inform. Theory, 2006, vol. 45, no. 5, pp. 1717–1719.
63. *Galbraith S. D., Paterson K. G., and Smart N. P.* Pairings for cryptographers. Discrete Appl. Math., 2008, vol. 156, no. 16, pp. 3113–3121.
64. *Bitansky N., Canetti R., Chiesa A., and Tromer E.* Recursive Composition and Bootstrapping for Snarks and Proof-Carrying Data. IACR Cryptology ePrint Archive, 2012. `http://eprint.iacr.org/2012/095`. 61 p.
65. *Valiant P.* Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. LNCS, 2008, vol. 4948, pp. 1–18.
66. *Crescenzo G. D. and Lipmaa H.* Succinct NP proofs from an extractability assumption. LNCS, 2008, vol. 5028, pp. 175–185.
67. *Mei T.* Polylogarithmic two-round argument systems. J. Math. Cryptol., 2008, vol. 2, no. 4, pp. 343–363.
68. *Chiesa A., Ojha D., and Spooner N.* FRACTAL: Post-quantum and transparent recursive proofs from holography. LNCS, 2020. vol. 12105, pp. 769–793.
69. *Kattis A., Panarin K., and Vlasov A.* RedShift: Transparent SNARKs from List Polynomial Commitment IOPs. Cryptology ePrint Archive, Paper 2019/1400, 2019. `https://eprint.iacr.org/2019/1400`. 20 p.
70. *Ben-Sasson E., Bentov I., Horesh Y., and Riabzev M.* Fast Reed — Solomon interactive oracle proofs of proximity. 45th Intern. Colloquium ICALP'2018, vol. 107, Prague, Czech Republic, 2018, pp. 1–17. `https://doi.org/10.4230/LIPIcs.ICALP.2018.14`.
71. *Setty S.* Spartan: Efficient and general-purpose zkSNARKs without trusted setup. LNCS, 2020, vol. 12172, pp. 704–737.
72. *Bowe S., Grigg J., and Hopwood D.* Recursive Proof Composition without a Trusted Setup. Cryptology ePrint Archive, Paper 2019/1021, 2019. `https://eprint.iacr.org/2019/1021`. 31 p.
73. *Boneh D., Drake J., Fisch B., and Gabizon A.* Halo Infinite: Recursive zk-SNARKs from any Additive Polynomial Commitment Scheme. Cryptology ePrint Archive, Paper 2020/1536, 2020. `https://eprint.iacr.org/2020/1536`. 66 p.